Trabajo Fin de Máster

# Modelos estadísticos de riesgo de crédito y planificación estratégica

Daniel González Paz

**Máster en Técnicas Estadísticas**

**Curso 2023-2024**

II

# Propuesta de Trabajo Fin de Máster

| |
|---|
| **Título en galego:** Modelos estatísticos de risco de crédito e planificación estratéxica |
| **Título en español:** Modelos estadísticos de riesgo de crédito y planificación estratégica |
| **English title:** Statistical models of credit risk and strategic planning |
| **Modalidad:** Modalidad B |
| **Autor/a:** Daniel González Paz, Universidad de A Coruña |
| **Director/a:** Salvador Naya Fernández, Universidad de A Coruña; Javier Tarrío Saavedra, Universidad de A Coruña |
| **Breve resumen del trabajo:**<br>En este trabajo se presentan las bases conceptuales, acompañadas de un análisis comparativo, de distintos métodos: regresión logística, elastic net, árboles CART, random forest y XGBoost; con el objetivo de estudiar su rendimiento de cara a desarrollar un modelo de Scoring destinado a la admisión de préstamos al consumo. |
| **Recomendaciones:**<br>Capacidades analíticas y de resolución de problemas matemáticos y estadísticos. Se valorarán conocimientos de estadísticas y modelización (Python, R, Stata...) y técnicas de tratamiento de datos (SQL). Se valorarán conocimientos en Economías y Finanzas. Interés por desarrollarse en el campo de las finanzas cuantitativas y los modelos de riesgos, dos de los principales perfiles demandados en el sector financiero. Capacidad de trabajo en equipo y habilidades comunicativas. |
| **Otras observaciones:** |

Don Salvador Naya Fernández, Catedrático de la Universidad de A Coruña, don Javier Tarrío Saavedra, Profesor Titular de la Universidad de A Coruña y don Adrián Gutiérrez Barrio, Coordinador de Auditoría de Modelos de ABANCA, informan que el Trabajo Fin de Máster titulado

**Modelos estadísticos de riesgo de crédito y planificación estratégica**

fue realizado bajo su dirección por don Daniel González Paz 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 3 de Junio de 2024.

NAYA FERNANDEZ SALVADOR - DNI 76353731X
Firmado digitalmente por NAYA FERNANDEZ SALVADOR - DNI 76353731X
Fecha: 2024.06.03 14:05:14 +02'00'

El director:
Don Salvador Naya Fernández

TARRIO SAAVEDRA JAVIER - DNI 53164914T
Digitally signed by TARRIO SAAVEDRA JAVIER - DNI 53164914T
DN: c=ES, o=UNIVERSIDAD DE A CORUÑA, ou=CERTIFICADO ELECTRONICO DE EMPLEADO PUBLICO, ou=MATEMATICAS, ou=18373, serialNumber=IDCES-53164914T, sn=TARRIO SAAVEDRA, givenName=JAVIER, cn=TARRIO SAAVEDRA JAVIER - DNI 53164914T
Date: 2024.06.03 16:03:31 +02'00'

El director:
Don Javier Tarrío Saavedra

GONZALEZ PAZ DANIEL - 77460380Z
Firmado digitalmente por GONZALEZ PAZ DANIEL - 77460380Z
Fecha: 2024.06.03 16:25:56 +02'00'

El tutor:
Don Adrián Gutiérrez Barrio

El autor:
Don Daniel González Paz

---

**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.

# Índice

# Resumen

## Resumen en español

En este trabajo se presentan, por un lado, las bases teóricas necesarias para comprender los métodos que emplearemos en la construcción de un modelo de *Scoring*, aplicado a la admisión de préstamos al consumo de ABANCA y, por otro lado, la implementación, el análisis y posterior evaluación de los modelos.

Las técnicas estadísticas aplicadas para la resolución este problema específico, incluyen la regresión logística, los métodos de penalización (como el denominado *elastic net*), los árboles de decisión, *random forest* y métodos *boosting*, como es el caso del *XGBoost*. Para medir el desempeño de los modelos, utilizaremos varias métricas asociadas a la matriz de confusión (incluyendo las medidas de sensibilidad y especificidad) que son esenciales para evaluar la capacidad predictiva y la eficacia de los modelos. También emplearemos curvas *ROC* y *Precision-Recall* (especialmente recomendada para muestras no balanceadas), que nos permitirán visualizar y comparar el desempeño de los modelos para diferentes umbrales de decisión, facilitando así la identificación del modelo más adecuado para nuestro propósito.

Además, propondremos la aplicación de técnicas como los gráficos *PDP* o *ICE* y el método *LIME* que nos permitirán mejorar la interpretabilidad de los modelos del ámbito del *machine learning* empleados en el presente trabajo. De esta forma, seremos capaces de proporcionar una visión más clara sobre cómo los modelos toman decisiones en este contexto, confiriendo así una mayor confianza en su uso para la admisión de préstamos.

## English abstract

In this paper, we present, on one hand, the theoretical foundations necessary to understand the methods we will use in the construction of a Scoring model applied to the consumer loan admission process at ABANCA, and on the other hand, the implementation, analysis, and subsequent evaluation of the models.

The statistical techniques applied to solve this specific problem include logistic regression, pena-

lization methods (such as elastic net), decision trees, random forest, and boosting methods, such as XGBoost. To measure the performance of the models, we will use several metrics associated with the confusion matrix (including sensitivity and specificity measures), which are essential for evaluating the predictive capability and effectiveness of the models. We will also employ ROC and Precision-Recall curves (especially recommended for unbalanced samples), which will allow us to visualize and compare the performance of the models for different decision thresholds, thus facilitating the identification of the most suitable model for our purpose.

Additionally, we will propose the application of techniques such as PDP or ICE plots and the LIME method, which will enable us to improve the interpretability of the machine learning models employed in this study. In this way, we will be able to provide a clearer view of how the models make decisions in this context, thereby conferring greater confidence in their use for loan admission.

# Prefacio

Una de las principales preocupaciones en el sector bancario y financiero es el llamado riesgo de crédito. Este término se corresponde con la eventualidad de que una entidad financiera sufra pérdidas derivadas de que la contraparte no pueda asumir las obligaciones de pago contraídas antes de la fecha de vencimiento estipulada. En otras palabras, se trata de la posibilidad de que un deudor no reembolse el dinero prestado dentro del plazo acordado.

Así pues, con el objetivo de minimizar dicho riesgo, en la década de 1960 nace lo que se conoce como *Credit Scoring* (Siddiqi, 2012; Anderson, 2007). Este concepto se basa en emplear modelos estadísticos con el objetivo de obtener medidas numéricas con las que poder evaluar si un solicitante de crédito finalmente pagará la deuda contraída y, en función del resultado, tomar decisiones. En definitiva, pretende ordenar a la población según la probabilidad de que ocurra cierto evento, generalmente, el impago de una deuda.

En los últimos años, el uso de los modelos de *Scoring* ha aumentado considerablemente debido a diversos factores, entre los que se pueden enumerar la gran cantidad de datos disponibles, el aumento del poder computacional y la regulación bancaria, enfocada cada vez más en utilizar información objetiva y de calidad, de cara a la concesión de crédito (WBG and ICCR, 2019). Además, el término *Credit Scoring* se ha generalizado de tal forma que el mero uso de modelos estadísticos relacionados con la administración de crédito puede enmarcarse dentro de este concepto.

En el contexto de los modelos de *Scoring*, podemos distinguir dos casuísticas:

- Modelos de admisión, dirigidos a la evaluación de propuestas de crédito. Estos modelos son de tipo reactivo, es decir, se emplean una vez que el cliente realiza una solicitud de crédito y los datos utilizados son los que se conocen en el momento de la solicitud, ya sean los inherentes a la propia operación o asociados a características socioeconómicas del cliente.

- Modelos de comportamiento o comportamentales, enfocados hacia la evaluación continua de los clientes con el fin de, por ejemplo, modificar los límites de una línea de crédito con posterioridad a la contratación. Se aplica de forma proactiva bajo iniciativa de la entidad y cuenta con información de toda la operatoria realizada por el cliente desde que establece una relación con el banco.

La metodología más común empleada en la construcción de modelos de calificación crediticia es la regresión logística; sin embargo, con la llegada del *machine learning* entran en escena nuevos métodos, como el *random forest* o el *boosting*, cuyo uso, en ocasiones, proporciona un mayor poder predictivo.

La aplicación de estos modelos de *machine learning*, que buscan distinguir entre clientes que incumplen sus obligaciones financieras y aquellos que no, resulta en una potencial ganancia económica para la entidad, la cual puede ver maximizados sus beneficios al contar con modelos con un mayor desempeño y precisión, que clasifiquen mejor a los deudores.

Además, estos modelos también pueden tener un impacto positivo desde una perspectiva social, ya que permiten el uso de grandes volúmenes de datos, no únicamente limitados a los antecedentes financieros de los clientes, lo cual podría facilitar una mayor inclusión financiera en sectores de la población que suelen ser ignorados.

No obstante, ese mayor desempeño del modelo también lleva apareado nuevos retos desde el punto de vista del supervisor, ya que estos modelos de *machine learning* suelen ser menos interpretables y, por tanto, resulta más costoso garantizar que se cumplen todos los requisitos regulatorios (Alonso and Carbó, 2020, 2021; Dessain et al., 2023).

En el contexto específico de la empresa ABANCA, el uso de los modelos anteriormente mencionados es de especial relevancia, dedicando por ello numerosos y variados recursos a este fin. De hecho, la función de Auditoría de Modelos, enmarcada dentro del departamento de Auditoría Interna de la entidad, consiste en evaluar la precisión e idoneidad de los modelos estadísticos utilizados por el banco para tomar decisiones financieras y de riesgo que impactan de forma directa en la planificación estratégica del mismo. Además, los auditores del departamento también realizan la labor de verificar que los modelos estén alineados con las regulaciones pertinentes y las mejores prácticas de la industria, pudiendo proporcionar recomendaciones para mejorar la eficacia y la eficiencia de dichos modelos, por ejemplo, mediante la implementación de nuevas técnicas de modelado.

Este trabajo tiene como objetivo estudiar el comportamiento de algunos de los modelos utilizados en el ámbito del *Credit Scoring*, incluyendo entre ellos estas nuevas metodologías relacionadas con el *machine learning*. Para ello, en el Capítulo 1 presentaremos el problema concreto al que nos enfrentaremos, así como la base de datos de la que disponemos y los tratamientos iniciales a las que ha sido sometida (Sección 1.1 y 1.2). Además, también incluimos un apartado referido al tipo de métricas que emplearemos para evaluar los distintos modelos (Sección 1.3) y algunos métodos gráficos para mejorar la interpretabilidad de los modelos menos explicables (Sección 1.4).

Los modelos que analizaremos serán la regresión logística (Capítulo 2), *elastic net* o regresión logística penalizada (Capítulo 3), los árboles CART (Capítulo 4), el *random forest* (Capítulo 5) y el *XGBoost* (Capítulo 6). Los capítulos dedicados a cada modelo seguirán la misma estructura, con una primera sección dedicada a sentar las bases teóricas sobre los que se sustentan y otro apartado dedicado a su aplicación práctica, en el que prestaremos especial atención a su ajuste, análisis y evaluación.

Por último, en el Capítulo 7, incluimos una comparación final sobre los distintos modelos valorando cuál sería la elección más adecuada para el problema que queremos resolver. Adicionalmente, también

se enumeran las principales líneas de investigación futuras derivadas del presente trabajo y relacionadas con su ampliación y diversificación.

# Capítulo 1

# Modelo de Scoring

En este primer capítulo, introduciremos brevemente el modelo de *Scoring* que será el eje central de nuestro trabajo. Luego, profundizaremos en las variables utilizadas para construir el modelo y en los tratamientos iniciales a los que han sido sometidas, para finalmente, presentar diversas métricas de evaluación en las que nos basaremos para medir el rendimiento de los distintos métodos implementados Además, también incluiremos algunos enfoques gráficos que nos permitirán interpretar los modelos menos explicables.

## 1.1. Introducción al modelo

Como ya hemos adelantado en el Prefacio, el objetivo de este trabajo será analizar distintos métodos estadísticos en los que poder basar un modelo de *Scoring*, el uso específico para el que se desarrolla es la evaluación del segmento de Consumo No Vinculados con el que cuenta ABANCA. Más concretamente, se trata de un modelo de admisión para préstamos al consumo que son solicitados por clientes que no presentan una vinculación con el banco, por ejemplo, aquellos que no domicilian nómina, pensión o recibos; o cuyo saldo medio es inferior a cierta cantidad, entre otras casuísticas. La idea es que, para esa cartera, el modelo nos proporcione una probabilidad de mora en base a la cual se decidirá si interesa conceder o no el préstamo.

El punto de corte que se establece para decidir a partir de que probabilidad de mora se aprueba la concesión del préstamo, este se suele determinar de tal forma que se equilibre el número de falsos positivos y falsos negativos (ver Tabla 1.1). No obstante, se suele dar más peso a los verdaderos positivos, ya que el coste de no conceder un préstamo a un potencial cliente tiene un menor perjuicio sobre el banco que concedérselo a un cliente que vaya a resultar moroso. Cabe añadir que la decisión final se toma en base al apetito de riesgo que estipula la entidad en cada momento.

Además, hay dos opciones en cuanto a la concesión: se puede establecer un circuito binario en el que la decisión se reduce a aceptar o denegar el préstamo o, alternativamente, un circuito no binario en

el que además de las opciones contempladas, hay un tercer escenario de duda en el que un analista de crédito realiza una evaluación adicional de la operación para emitir un dictamen. A efectos ilustrativos, a lo largo del trabajo consideraremos un circuito binario donde el punto de corte está en el 10 % de probabilidad, siendo este un umbral habitual que toman como referencia las entidades financieras.

Es importante señalar que, después de obtener las probabilidades de mora a través de nuestros modelos de *Scoring*, deberían someterse a un proceso adicional de calibración a través del cual poder integrar las expectativas macroeconómicas en nuestras predicciones. Sin embargo, en lo que al trabajo respecta, nos centraremos únicamente en la primera fase del proceso, sin adentrarnos en la calibración correspondiente.

Hasta este momento, los modelos de *Scoring* desarrollados por la entidad se han fundamentado siempre en regresiones logísticas tradicionales. Con este trabajo se pretende, por un lado, sentar las bases para la adopción de otras posibles técnicas como la regresión logística penalizada o modelos de *machine learning* basados en árboles como el *random forest* o el *XGBoost* y, por otra parte, analizar si existe una mejora en el desempeño de estas nuevas alternativas con respecto a la regresión logística.

## 1.2. Construcción del modelo

Con el objetivo de efectuar la modelización, consideraremos que un cliente incurre en mora o se encuentra en *default* cuando se produce un impago por un periodo superior a 90 días (siguiendo las indicaciones de ABANCA), en otras palabras, en el momento en el que no se realiza el pago de una cuota y no se abona en los 3 meses posteriores.

Los datos con los que trabajaremos serán operaciones del histórico de la entidad que comprenden 7 años de muestra. Para estas operaciones, construiremos un indicador de mora que estará activo si el cliente entra en default durante el primer año después de haber solicitado el préstamo.

Una vez definido el perímetro de modelización, debemos considerar qué variables usaremos en el modelo. Como ya adelantamos, el tipo de variables a incluir serán referentes a la propia operación (cuota, plazo, importe, etc.) y a características socioeconómicas del cliente (edad, ingresos, profesión, entre otras).

En total contaremos con 38 variables, las cuales han sido codificadas de la siguiente forma, teniendo en cuenta las restricciones de confidencialidad que nos impiden proporcionar detalles específicos sobre ellas.

- `VAR_OP_CONT_i`: si es una variable continua referida a la operación, con $i = 1, ..., 10$.
- `VAR_OP_DISC_i`: si es una variable discreta referida a la operación, con $i = 1, 2$.
- `VAR_CLI_CONT_i`: si es una variable continua referida al cliente, con $i = 1, ..., 19$.
- `VAR_CLI_DISC_i`: si es una variable discreta referida al cliente, con $i = 1, ..., 7$.

Para poder construir la base de datos de la que se alimentará el modelo, hemos realizado un proceso de extracción de datos del *data mart* corporativo de la entidad (Pipe, 1997) a través de la herramienta SQL (Perez, 2011) con la que hemos obtenido y cruzado información de diferentes tablas.

Después de obtener la base de datos, procedemos a realizar un análisis exploratorio de las variables (los cuales no incluimos por temas de confidencialidad y por no ser el objetivo principal del trabajo) con el fin de identificar aquellas características para las cuales hay un número muy elevado de *missings*, es decir, variables que no están informadas para más de un 30 % de las operaciones, en cuyo caso se toma la decisión de descartar esas variables.

Para los campos en los que detectamos alguna desinformación de menor calibre, intentamos imputar ese *missing* en función del tipo de variable y la experiencia de la entidad en el tratamiento de esos datos. Por ejemplo, en el caso de que los ingresos de un cliente estuviesen desinformados, se podría llegar a asumir que esos ingresos son 0 debido a cómo se produce la alimentación de la variable.

Este primer análisis exploratorio también nos sirve para detectar ciertos *outliers* que resulta conveniente eliminar por poder ser causa potencial de un desajuste del modelo. Hay que tener en cuenta que un modelo de *Scoring* se basa en información previa para poder extraer conclusiones, por tanto, hay que ser cuidadosos a la hora de evaluar con ellos operaciones que puedan ser atípicas, por ejemplo, préstamos con un importe muy alto en relación al tipo de producto o segmento para el que está diseñado el modelo.

Después de haber analizado las variables y aplicado los filtros oportunos, nos quedamos con un total de XXXX operaciones de las cuales XXX tienen activado el indicador de mora, es decir, tenemos sobre un $X\,\%$ de operaciones morosas dentro de la muestra de modelización frente a un $XX\,\%$ de operaciones sanas, es decir, que no se encuentran en situación de mora.

Llegados a este punto, realizaremos un análisis de correlaciones entre las variables para evitar problemas de colinealidad. Nótese que antes de realizar el análisis, hemos llevado a cabo una transformación de las variables discretas en variables *Weight of Evidence* (*WoE*) tal y como se indicará y describirá en la Sección 2.2. Estas nuevas variables transformadas han sido codificadas como `VAR_CLI_DISC_WOE_i` o `VAR_OP_DISC_WOE_i`.

### 1.2.1. Análisis de correlaciones

Como tenemos un número considerable de variables, emplearemos la función `findCorrelation` del paquete de  , `caret` (Kuhn, 2008), para identificar las variables que presentan una correlación superior a 0.75. Debajo se muestra el bloque de código correspondiente junto con el resultado que proporciona.

```
indcorr_1 <- findCorrelation ( corr , cutoff =0.75 , names = T ,verbose=TRUE)
indcorr_1
```

```
Compare row 29  and column  6 with corr  0.958
Means:  0.201 vs 0.122 so flagging column 29
Compare row 11  and column  10 with corr  0.907
Means:  0.193 vs 0.118 so flagging column 11
Compare row 6  and column  17 with corr  0.83
Means:  0.167 vs 0.115 so flagging column 6
Compare row 13  and column  9 with corr  0.93
Means:  0.152 vs 0.112 so flagging column 13
Compare row 15  and column  27 with corr  0.775
Means:  0.144 vs 0.11 so flagging column 15
Compare row 9  and column  12 with corr  0.919
Means:  0.12 vs 0.108 so flagging column 9
Compare row 4  and column  3 with corr  0.923
Means:  0.146 vs 0.107 so flagging column 4
Compare row 2  and column  1 with corr  0.902
Means:  0.145 vs 0.104 so flagging column 2
Compare row 28  and column  14 with corr  0.797
Means:  0.133 vs 0.101 so flagging column 28
Compare row 33  and column  32 with corr  0.798
Means:  0.086 vs 0.101 so flagging column 32
```

```
[1] "VAR_CLI_CONT_19"    "VAR_CLI_CONT_8"     "VAR_CLI_CONT_6"     "VAR_CLI_CONT_10"
[5] "VAR_OP_CONT_4"      "VAR_OP_CONT_3"      "VAR_CLI_CONT_4"     "VAR_CLI_CONT_2"
[9] "VAR_CLI_CONT_18"    "VAR_CLI_DISC_WOE_2"
```

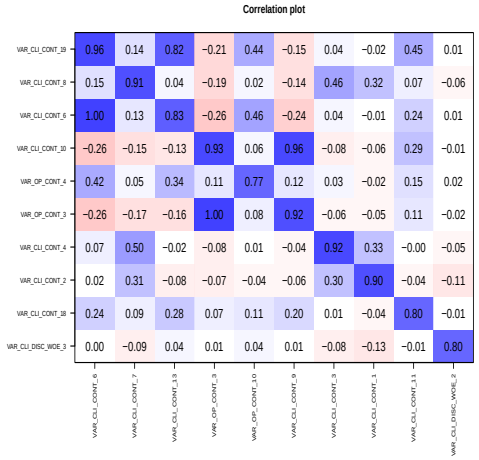



Figura 1.1: Análisis de correlaciones inicial.

Como podemos observar, se identifican 10 pares de variables que presentan una correlación de al menos 0.75. Con el fin de tener una idea intuitiva de esta estructura de correlación, en la Figura 1.1. hemos representado un gráfico de correlaciones en el que se incluyen los pares de variables con correlaciones elevadas para intentar identificar cuáles se deberían eliminar del análisis.

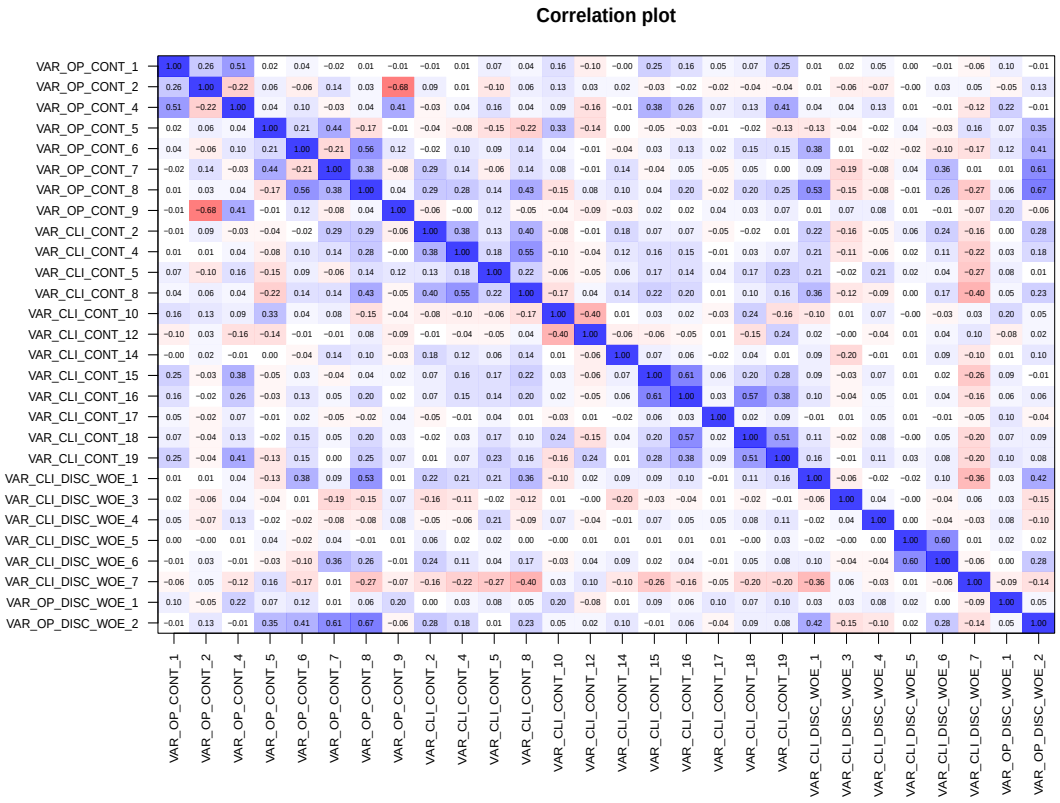



Figura 1.2: Análisis de correlaciones final.

En base a los resultados obtenidos en la Figura 1.1 y el sentido de negocio de cada variable, tomamos la decisión de eliminar las siguientes variables para evitar correlaciones elevadas: `VAR_OP_CONT_3`, `VAR_OP_CONT_10`, `VAR_CLI_CONT_1`, `VAR_CLI_CONT_3`, `VAR_CLI_CONT_6`, `VAR_CLI_CONT_7`, `VAR_CLI_CONT_9`, `VAR_CLI_CONT_11`, `VAR_CLI_CONT_13` y `VAR_CLI_DISC_WOE_2`; quedándonos finalmente con 28 variables. De esta forma, como podemos ver en la Figura 1.2, todos los pares de variables están caracterizados por una correlación menor de 0.75, con lo que así evitaremos posibles problemas de colinealidad futuros.

### 1.2.2. Muestra de entrenamiento y test

El procedimiento habitual en lo que a aprendizaje estadístico se refiere, en el cual se dispone de muestras relativamente grandes, consiste en separar el conjunto de datos en una muestra de entre-

namiento, con la que podremos ajustar el modelo, y una muestra de validación o test, sobre la que evaluar su desempeño.

En este contexto, es un procedimiento habitual la selección de manera aleatoria un 80 % de las observaciones para entrenamiento y un 20 % para test (Casal et al., 2021). Podemos dividir el conjunto de datos en muestra de entrenamiento y test mediante código de la siguiente forma, fijando una semilla para asegurar la reproducibilidad de los resultados (recogidos en la Figura 1.3).

```
set.seed(2311)
df=datos_filtrados_2
nobs <- nrow(df)
itrain <- sample(nobs, 0.8 * nobs)
train <- df[itrain, ]
test <- df[-itrain, ]
```

Como podemos observar en la Figura 1.3, la muestra está altamente desbalanceada en el sentido de que existen muchas más operaciones sanas que en *default*. Esto será relevante a la hora de evaluar el modelo, como veremos en la siguiente sección.



Figura 1.3: Frecuencias de las muestras de entrenamiento y validación separadas por la etiqueta de mora.

## 1.3. Evaluación del modelo

Una vez construidos los modelos que aplicaremos a nuestro problema, el siguiente paso será evaluar su desempeño en lo que a clasificación se refiere, es decir, estudiar si el modelo clasifica bien o no a las nuevas observaciones. Como hemos comentado en el apartado anterior, normalmente se selecciona un 20 % de observaciones de la muestra total que no intervienen en el ajuste y que se reservan para esta fase.

En esta sección, se mostrarán distintas métricas calculadas a partir de la llamada matriz de confusión o tabla de contingencia, para después definir la curva $ROC$ y la curva *Precision-Recall*. Para la elaboración de las siguientes secciones, se han tenido en cuenta los trabajos de Larner (2021) y McHugh (2012) para la Sección 1.3.1; Yang and Berdine (2017) y Nahm (2022) para la Sección 1.3.2; y Davis and Goadrich (2006), Saito and Rehmsmeier (2015) y Goadrich et al. (2004) para la Sección 1.3.3.

### 1.3.1. Matriz de confusión

Recordemos que estamos ante un problema de clasificación binaria donde las operaciones pueden estar sanas (0) o ser morosas (1). Si entendemos como observaciones "positivas" (en el sentido de que presentan la característica buscada) las que hemos marcado con 1 y "negativas" las marcadas con 0, tras la clasificación por parte del modelo del conjunto de test, se nos presentan las casuísticas recogidas en la Tabla 1.1.

| | | **Predicción** | |
|---|---|---|---|
| | | Negativo (N) | Positivo (P) |
| **Observación** | Negativo (N) | Verdaderos Negativos (TN) | Falsos Positivos (FP) |
| | Positivo (P) | Falsos Negativos (FN) | Verdaderos Positivos (TP) |

Tabla 1.1: Matriz de confusión para un problema de clasificación.

Idealmente querríamos tener un número elevado de TP y TN, mientras que buscaríamos que el FP y FN fuesen lo más reducidos posibles. Para determinar cómo de bien o mal estamos clasificando, a continuación definimos algunas de las métricas más utilizadas.

- *True Positive Rate* (sensibilidad o *recall*): se define como

$$TPR = \frac{TP}{TP + FN} = \frac{TP}{P},$$

que mide la tasa de verdaderos positivos sobre el total de positivos de la muestra.

- *True Negative Rate* (especificidad): se trata de una medida cuya expresión es

$$TNR = \frac{TN}{TN + FP} = \frac{TN}{N},$$

análoga al $TPR$ pero referida a la tasa de verdaderos negativos.

- *Accuracy* (precisión o *true rate*): esta medida que mide el desempeño del modelo de clasificación a través de la expresión

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{TP + TN}{P + N}.$$

  Esta métrica se corresponde con una medida global del rendimiento del modelo, no obstante hay que tener especial cuidado al interpretar sus valores, sobre todo cuando tenemos conjuntos de datos no balanceados.

Por ejemplo, en el caso extremo de un modelo que clasifique todas las observaciones de test como negativas, es decir, que asuma que todas están sanas considerando que en la muestra de entrenamiento tendremos unas XXXX operaciones sanas y XX operaciones morosas, obtendríamos una matriz de confusión como la que se muestra en la Tabla 1.2.



Tabla 1.2: Matriz de confusión para un modelo que únicamente clasifica en la categoría de sanos.

A partir de esta matriz de confusión, obtendríamos las distintas métricas asociadas que se encuentran recogidas en la Tabla 1.3.

| TPR | TNR | Accuracy |
|---|---|---|
| 0 | 1 | 0.9463 |

Tabla 1.3: Valores de las métricas para un modelo que clasifique a todas las observaciones de test como sanas.

Como podemos observar, obtenemos una precisión muy elevada, cuando realmente el modelo nunca acierta a la hora de predecir si una operación va a incurrir en mora. Con el fin de subsanar esta deficiencia, especialmente preocupante cuando las muestras están desbalanceadas, se recurre a la llamada "precisión balanceada" que se define como la media entre la sensibilidad y la especificidad

$$Balanced\ Accuracy = \frac{TPR + TNR}{2},$$

en nuestro ejemplo *Balanced Accuracy* $= \frac{0+1}{2} = 0.5$, que resulta bastante inferior a la precisión original y que captura de una manera más realista el desempeño del modelo.

Otra medida de tipo global, que también tiene en cuenta la distribución de la muestra, es la denominada $\kappa$ de Cohen (Cook, 2005) que analiza la tasa de aciertos obtenida por el modelo frente a una clasificación al azar. Su definición es la siguiente,

$$\kappa = \frac{P_a - P_c}{1 - P_c},$$

donde $P_a$ se identifica con la tasa de de aciertos observada y $P_c$ con la proporción de aciertos que se daría por puro azar. También se puede expresar en términos de $TP, TN, FP$ y $FN$ como sigue

$$\kappa = \frac{2 \cdot (TP \cdot TN - FN \cdot FP)}{(TP + FP) \cdot (FP + TN) + (TP + FN) \cdot (FN + TN)}.$$

$\kappa$ puede tomar valores entre -1 y 1, siendo los umbrales habituales para su interpretación los siguientes: valores negativos indican que la tasa de aciertos es menor que la de una clasificación al azar, para $(0, 0.20)$ es ligeramente superior, en el intervalo $[0.20, 0.40)$ razonable, en $[0.40, 0.60)$ moderada, en el rango $[0.60, 0.80)$ sustancial y entre $[0.8, 1]$ casi perfecta.

En el caso ilustrativo que veníamos considerando en el cual todas las observaciones eran clasificadas como sanas, $\kappa = 0$ ya que $TP = 0$, por lo que equipararíamos el modelo a un clasificador aleatorio.

Además de las medidas globales que hemos estado discutiendo, en el caso de que el modelo no nos proporcione únicamente la categoría en la que han sido clasificadas las observaciones, sino también la probabilidad de asignar una observación a cierta categoría, podemos calcular otro tipo de métricas globales como el área bajo la curva $ROC$ o bajo la curva $PR$ que presentamos en las siguientes secciones.

### 1.3.2. Curva ROC (Receiver Operating Characteristic)

La curva $ROC$ (*Receiver Operating Characteristic*) se propone por primera vez en el campo de las telecomunicaciones con el objetivo de diferenciar entre una verdadera señal (resultado positivo) y simple ruido (resultado negativo) al analizar señales emitidas por radar; no obstante, rápidamente se fue extendiendo a otro tipo de aplicaciones como el aprendizaje estadístico.

Para representar dicha curva debemos situar en el eje $x$ lo que se denomina la tasa de falsos negativos ($FNR$) que viene dada por

$$FNR = 1 - TNR,$$

y en el eje $y$ la sensibilidad o $TPR$ que ya ha sido definida previamente.

De esta manera, para calcular la curva $ROC$, únicamente tendremos que representar las coordenadas dadas por el $TPR$ (sensibilidad) y el $FNR$ (1-especificidad) para distintos valores de los llamados puntos de corte. Estos se definen a partir de una malla de probabilidades en función de la cual consideramos clasificar una observación en positiva o en negativa. Por ejemplo, considerando como punto de corte

$c = 0.4$, diremos que si para una observación obtenemos una probabilidad superior a 0.4, será clasificada como positiva y si es inferior como negativa.

Así, a medida que modifiquemos el punto de corte iremos obteniendo diferentes valores de $TPR$ y $FNR$ según como sean las predicciones de probabilidad asociadas a nuestra muestra de test, e interpolando linealmente esos puntos, obtendremos la curva $ROC$ deseada. Podemos visualizar un ejemplo en la Figura 1.4.

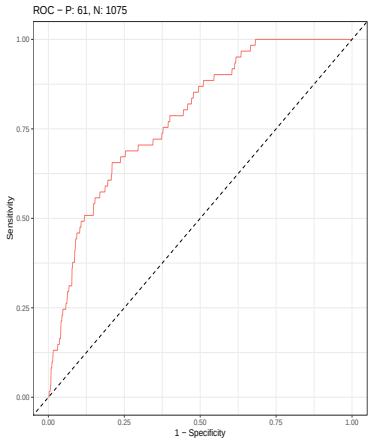



Figura 1.4: Ejemplo de curva $ROC$.

En base a la construcción de la curva $ROC$ que acabamos de describir, idealmente nos gustaría que la curva se acercase lo máximo posible a la esquina superior izquierda, lo que se correspondería con un $TPR = 1$ y un $FNR = 0$ (equivalentemente, un $TNR = 1$), mientras que la recta $y = x$ marcaría un desempeño propio de un clasificador aleatorio.

Una posible medida para analizar cómo de bueno es un método de clasificación en base a la curva $ROC$ es calcular el área bajo la curva ($AUC$), de tal forma que un $AUC = 0.5$ se correspondería con el clasificador aleatorio y un $AUC = 1$ con la máxima sensibilidad ($TPR$) y especificidad ($TNR$).

### 1.3.3. Curva PR (Precision-Recall)

Aunque el área bajo la curva $ROC$ es una medida global ampliamente utilizada en la comparación de modelos de clasificación, existen estudios (Davis and Goadrich, 2006; Saito and Rehmsmeier, 2015) en los cuales se aconseja utilizar otro tipo de curva cuando las muestras están desbalanceadas.

Estos autores defienden emplear la llamada curva *Precision-Recall* que consiste en representar sobre el eje $x$ la *recall* ($TPR$ o sensibilidad) y en el eje $y$ una medida que todavía no habíamos introducido, denominada valor predictivo positivo ($PPV$) o *precision* y que se define como

$$PPV = \frac{TP}{TP + FP},$$

cuantificando así la proporción de observaciones clasificadas como positivas que son realmente positivas.

Para representar nuestra curva $PR$, procederemos como en el caso anterior: calculando valores para las métricas involucradas ($TPR$ y $PPV$) en función de los puntos de corte y luego interpolando esos puntos; con la diferencia de que en este caso no se recurre a una interpolación lineal, para más detalle sobre cómo llevar a cabo la interpolación puede consultarse Goadrich et al. (2004). Podemos ver un ejemplo de la curva $PR$ en la Figura 1.5.

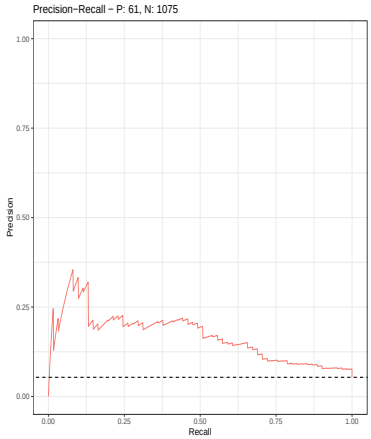



Figura 1.5: Ejemplo de curva $PR$.

También en este caso se puede considerar el área bajo la curva como medida de rendimiento del modelo. Mientras que en el caso de la curva $ROC$ teníamos que para un clasificador aleatorio obtendríamos un $AUC = 0.5$; para la curva $PR$, el $AUC$ de un clasificador aleatorio vendrá dado por $AUC = \frac{P}{P+N}$, siendo $P$ las observaciones positivas y $N$ las negativas.

## 1.4. Interpretación del modelo

Junto con la fase de evaluación de los modelos que acabamos de introducir, donde el objetivo es medir el rendimiento del modelo; también es de suma importancia ser capaces de dar una interpretación del mismo.

De acuerdo con Dessain et al. (2023), los modelos pueden ser clasificados en dos categorías: los llamados *cajas de cristal* que son intrínsecamente interpretables, lo que significa que podemos comprender la lógica subyacente y el razonamiento detrás de las predicciones que generan; y por otro lado, los modelos conocidos como *cajas negras*, que carecen de esa interpretabilidad global, y para los cuales necesitamos recurrir a técnicas que nos ayuden a entender las predicciones una vez estas ya se han realizado.

Como veremos en los siguientes capítulos, en los que entraremos a comprender como se formulan los distintos modelos utilizados, para la regresión logística, *elastic net* y los árboles de decisión podremos encontrar esa interpretabilidad intrínseca de la que hablábamos, ya que son modelos fácilmente

explicables.

Sin embargo, para el caso del *random forest* y del *XGBoost*, tendremos que recurrir a procedimientos adicionales que nos ayuden a entender su comportamiento. El objetivo de esta sección es introducir esas técnicas en las que nos apoyaremos para la comprensión de dichos modelos: los gráficos *PDP* (*Partial Dependence Plots*), *ICE* (*Individual Conditional Expectations*) o el método *LIME* (*Local Interpretable Model-Agnostic Explanations*).

### 1.4.1. Gráficos PDP

Los gráficos de dependencia parcial (*PDP*) fueron introducidos por Friedman (2001) y tienen como objetivo representar la relación entre el resultado de la predicción de cierto modelo y la evolución de un subconjunto de variables explicativas que el modelo emplea como predictores.

La idea para implementar el método sería considerar como $x_s$ el conjunto de variables de interés y como $x_c$ su complementario, de tal forma que los predictores al completo vendrían dados por $x = (x_s, x_c)$. Si denotamos por $\hat{f}(x)$ la predicción que proporciona el modelo para un cierto conjunto de variables, podemos definir la dependencia parcial de la respuesta con respecto a $x_s$ como

$$f_s(x_s) = \mathbb{E}\left[\hat{f}(x_s, x_c)|x_c\right], \tag{1.1}$$

que no es más que la esperanza de $\hat{f}(x_s, x_c)$ condicionada a $x_c$ y que podemos estimar de la siguiente forma

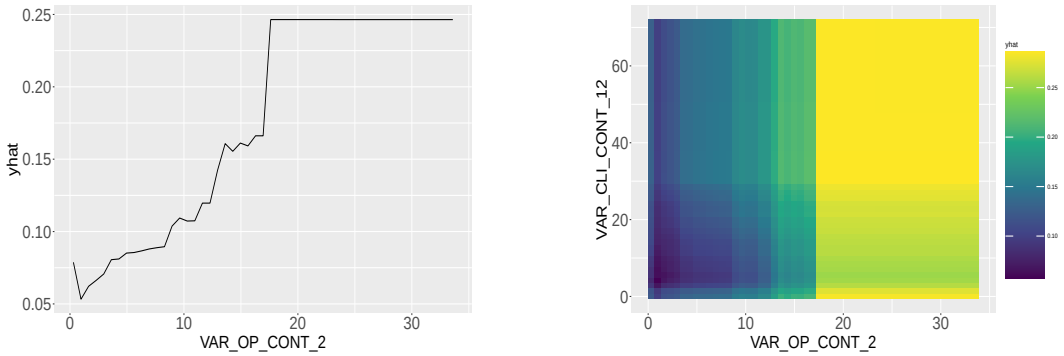
$$\bar{f}_s(x_s) = \frac{1}{n}\sum_{i=1}^{n} \hat{f}(x_s, x_{c,i}),$$

donde $x_{c,i}$ son los valores de $x_c$ que tenemos para cada observación $i = 1, ..., n$. De esta forma, lo que realmente estamos haciendo es promediar el efecto que tienen los predictores distintos de $x_s$ en el modelo.

Podemos implementar estos gráficos empleando la función `partial` del paquete `pdp` de R (Greenwell et al., 2017). A modo ilustrativo, si consideramos una única variable de interés $x_s = x_1$ y $\{x_{1,1}, ..., x_{1,k}\}$ cada uno de los valores que toma la variable en la muestra; el procedimiento a seguir sería, para cada $i = 1, ..., k$:

1. Reemplazar para cada una de las observaciones el valor de la variable $x_1$ por el valor $x_{1,i}$ que estamos considerando.
2. Calcular la predicción asociada para cada una de las observaciones.
3. Calcular la media de las predicciones obtenidas, $\bar{f}_1(x_{1,i})$.
4. Representar los puntos $\{x_{1,i}, \bar{f}_1(x_{1,i})\}$.

En la Figura 1.6, presentamos un ejemplo de un gráfico *PDP* unidimensional y otro bidimensional (en forma de mapa de calor) que analizaremos más adelante en la Sección 5.2.2.




(a) Gráfico *PDP* unidimensional.

(b) Gráfico *PDP* bidimensional.

Figura 1.6: Ejemplo de gráficos *PDP* unidimensional y bidimensional.

### 1.4.2. Gráficos ICE

Como hemos visto en el apartado anterior, los gráficos *PDP* buscan explicar la relación parcial promedio entre la predicción del modelo y un subconjunto de variables. No obstante, en presencia de interacción entre las variables, el comportamiento medio de las predicciones puede no capturar bien el efecto heterogéneo que ejerzan las variables explicativas sobre las predicciones.

Es por este motivo que Goldstein et al. (2015) desarrolla los gráficos de Efecto Individual Condicional o gráficos *ICE* que, al igual que los *PDP*, representan la evolución de las predicciones en función de una variable explicativa, pero en lugar de hacerlo de forma agregada mediante una media, lo realiza para cada observación. De manera formal, lo que haríamos sería representar el par $\{x_{s,i}, \hat{f}(x_{s,i}, x_{c,i})\}$ para cada una de las $i = 1, ..., n$ observaciones.

Ya hemos señalado cuál es la utilidad de los gráficos *ICE*, sin embargo, es habitual que las curvas representadas se solapen entre ellas y dificulten la interpretación. Por este motivo, se suele recurrir a los gráficos *ICE* centrados. La idea que subyace a estos gráficos es unir todas las curvas de predicción en cierto punto $x^*$ que suele identificarse con el mínimo o máximo del rango de valores de $x_s$, formalmente se definen como

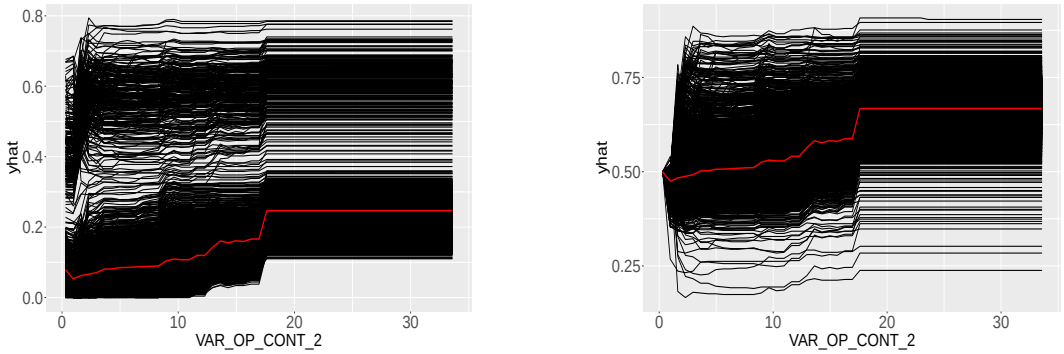
$$f_c^{(i)} = \hat{f}(x_{s,i}, x_{c,i}) - \mathbf{1}\hat{f}(x^*, x_{c,i}),$$

donde $\mathbf{1}$ es un vector de 1s de la dimensión requerida.

Así, el punto $\{x^*, \hat{f}(x^*, x_{c,i})\}$ puede considerarse como el caso base y estudiar como evolucionan las curvas de predicción a partir de ese punto. Por ejemplo, si consideramos el $x^*$ como el mínimo, tendríamos que todas las curvas comenzarían en 0, y en el valor máximo del eje de abscisas veríamos reflejado el efecto de $x_s$ sobre las predicciones en relación con el caso base.

A modo ilustrativo, en la Figura 1.7, mostramos tanto un gráfico *ICE* como su versión centrada, que serán analizados posteriormente en la Sección 5.2.2. Notar que en el gráfico centrado, las curvas de las predicciones no comienzan en 0 porque la función `pdp` con la que se representan realiza una

transformación a la hora de representar las probabilidades en el eje $y$.




(a) Gráfico *ICE*. (b) Gráfico *ICE* centrado.

Figura 1.7: Gráficos *ICE* e *ICE* centrado.

### 1.4.3. LIME

El método *LIME* aparece por primera vez en un artículo de Ribeiro et al. (2016) donde se pretende proporcionar una explicación para una predicción dada, independientemente del modelo considerado.

*LIME* se sustenta en la hipótesis de que cualquier modelo, por muy complejo que sea, puede verse como un modelo lineal a escala local, es decir, se espera que dos observaciones muy parecidas se comporten de manera similar a la hora de realizar una predicción sobre ellas.

Su objetivo final es suministrar de una manera visual, qué justificaciones hay a favor o en contra de una determinada predicción (en base a las variables empleadas) con el fin de que el usuario pueda determinar la razonabilidad de la misma.

La explicación proporcionada por *LIME* viene dada a partir de la siguiente expresión

$$\xi(x) = \arg\min_{g} \mathcal{L}(f, g, \pi_x) + \Omega(g),$$

donde $g$ denota el modelo interpretable que usaremos para explicar las predicciones a escala local con una complejidad asociada $\Omega(g)$. El modelo que querremos explicar será $f$ y $\pi_x(z)$ medirá la proximidad de $z$ con respecto a $x$. Así pues, nuestra intención es conseguir minimizar $\mathcal{L}(f, g, \pi_x)$ que lo que indica es cómo de poco fiable es aproximar $f$ a través de $g$ en una escala local (dada por $\pi_x$), sin que la complejidad $\Omega(g)$ se torne excesiva.

Para implementar el algoritmo, el primer paso consiste en construir nuevas observaciones $z$ fruto de perturbar algunos de los valores de las variables originales. Estas perturbaciones provendrán de obtener nuevos valores a partir de la distribución que siguen las variables.

El siguiente paso radica en convertir las variables perturbadas de cada observación en una *representación interpretable* que denotaremos por $z'$. En el caso de datos tabulares, esto se puede lograr

mediante la categorización de las variables y la creación de un vector de $0s$ y $1s$ que indica en qué categoría se encuentra cada observación.

A modo de ejemplo, si para una observación la variable correspondiente a ingresos toma el valor $x_1 = 1200$, una posible perturbación sería considerar $z_1 = 1300$ y para una categorización de ingresos que separa entre $[0, 1250), [1250, 3000)$ y $[3000, \infty)$, la representación interpretable de $z_1$ sería $z_1' = (0, 1, 0)$.

Teniendo esto en cuenta, la función $\mathcal{L}$ a minimizar vendrá dada por

$$\mathcal{L}(f, g, \pi_x) = \sum_{z,z'} \pi_x(z)(f(z) - g(z'))^2,$$

donde $f(z)$ es la predicción del modelo que queremos explicar y $g(z')$ la predicción dada por nuestro modelo simple para la representación interpretable de $z$. Cabe destacar, que la versión implementada en R (Hvitfeldt et al., 2022) considera para el modelo simplificado $g$ la regresión *ridge* (un modelo lineal penalizado que veremos en el Capítulo 3 para el caso de la regresión logística) y como $\pi_x$ un kernel exponencial sobre una cierta medida de distancia.

Con respecto a la complejidad asociada, seremos nosotros los que deberemos hacer una selección del número de características que queremos incorporar en el modelo *ridge*, y el algoritmo empleará un procedimiento de selección forward para decidir que variables incorporar. En la Figura 1.8 mostramos un ejemplo de explicación mediante *LIME* que será analizado en la Sección 6.2.2.

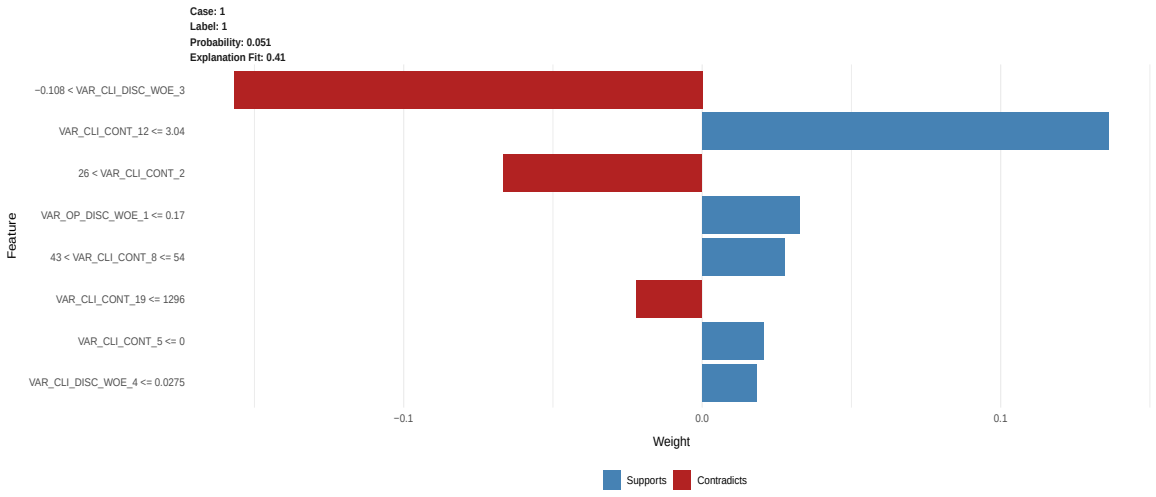



Figura 1.8: Ejemplo de *LIME*.

En el ámbito que nos ocupa, la utilización de *LIME* para explicar las predicciones que realizan los modelos de *machine learning* es especialmente útil por dos motivos fundamentales:

- En primer lugar, para los casos donde no está demasiado claro cómo debería clasificarse una observación. En base a las características seleccionadas por *LIME*, un analista podría decidir

en base a su experiencia y criterio cuáles tienen más peso. Por ejemplo, si *LIME* indica que la variable ingresos es menor a 100 y eso es indicativo de que la operación debería clasificarse como morosa, el analista en base a su criterio experto, puede considerar que ese es un factor de gran importancia en el que apoyarse para no conceder el préstamo.

- Por otra parte, la regulación obliga a las entidades a ser transparentes en los criterios de concesión de préstamos, no pudiendo emplear además ninguna característica discriminatoria. Haciendo uso de la metodología *LIME* podremos explicar a un cliente cuáles han sido los criterios elegidos para el dictamen que se realiza sobre el préstamo.

# Capítulo 2

# Regresión logística

En este capítulo se incluye una breve introducción de la regresión logística basándonos en las referencias Hosmer Jr et al. (2013), Kleinbaum et al. (2002) y Sheather (2009). Nos centraremos en dos conceptos intrínsecos a la misma: la Odds y la Odds Ratio, cuya interpretación analizaremos para las distintas casuísticas que nos podremos encontrar.

Una vez introducida esa breve introducción metodológica, pasaremos a construir un modelo logístico para nuestro conjunto de datos y analizaremos distintas métricas con el objetivo de evaluar su comportamiento.

## 2.1. Marco metodológico

El primer enfoque que adoptaremos será el de intentar modelizar la probabilidad de *default* a través de una regresión. Para ello querremos estudiar la relación existente entre la variable binaria $Y$ (que tomará el valor 1 si el contrato se encuentra en *default* o el valor 0 si está sano) y el resto de variables explicativas que hemos mencionado anteriormente.

Podríamos intentar emplear un modelo de regresión lineal para encontrar esa relación, de tal forma que

$$Y_i = x_i'\beta + \varepsilon_i \text{ con } i = 1, ..., n;$$

donde $Y_i$ es el valor de la variable respuesta para el individuo $i$-ésimo, $x_i' = (1, x_{i,1}, \ldots, x_{i,p-1})$ se corresponde con el vector fila asociado a las observaciones de las variables explicativas y $\beta = (\beta_0, \beta_1, \ldots, \beta_p)'$ es el vector de coeficientes. Sin embargo, el hecho de que la variable $Y$ únicamente pueda tomar dos valores genera una serie de circunstancias que quebrantan las suposiciones del modelo de regresión lineal:

- Linealidad: al querer modelizar la media de $Y$ condicionada a $X$ como una función lineal nos encontramos con el problema de que $\mathbb{E}(Y_i|X = x_i) = x_i'\beta$ podría darnos predicciones fuera del

intervalo $[0,1]$, lo cual no sería razonable ya que queremos estudiar el comportamiento de una probabilidad, puesto que $\mathbb{E}(Y_i|X=x_i)=\mathbb{P}(Y_i=1|X=x_i)$.

- Normalidad: en este caso nuestra variable $Y$, al presentar únicamente las categorías de fracaso y éxito, no sigue una distribución normal, sino que se identifica con una Bernoulli.
- Homocedasticidad: al seguir una distribución Bernoulli, la varianza de $Y$ viene dada por

$$Var(Y_i|X=x_i)=\mathbb{P}(Y_i=1|X=x_i)[1-\mathbb{P}(Y_i=1|X=x_i)],$$

y dado que $\mathbb{P}(Y=1|X=x)=\mathbb{E}(Y|X=x)$, tenemos que la varianza dependerá del valor de la variable explicativa, con lo cual estaremos ante un modelo heterocedástico.

Así pues, tendremos que recurrir a la regresión logística, que como ya hemos mencionado, es una herramienta ampliamente utilizada en el sector bancario y más concretamente en ABANCA, a la hora de predecir tasas de mora.

Para lograr nuestro objetivo, necesitaremos construir un modelo para la media condicionada de $Y$ con respecto a $X$, que en este caso se puede ver como una probabilidad de éxito (considerando como "éxito" una operación en *default*), condicionada al valor de la variable explicativa y que denotaremos de la siguiente forma

$$\pi(x)=\mathbb{E}(Y|X=x)=\mathbb{P}(Y=1|X=x).$$

Si ahora aplicamos una transformación a esta probabilidad condicionada, de tal forma que seamos capaces de convertir el intervalo $[0,1]$ en toda la recta real, estaremos en condiciones de emplear un modelo lineal. Para conseguirlo, utilizaremos la llamada función logit que se define como

$$g(y)=\log\left(\frac{y}{1-y}\right) \text{ para todo } y\in[0,1].$$

Así pues, podríamos representar la siguiente relación:

$$g(\pi(x))=\log\left(\frac{\pi(x)}{1-\pi(x)}\right)=\beta_0+\beta_1x_1+\cdots+\beta_px_p=x'\beta, \tag{2.1}$$

en la que $x'=(1,x_1,x_2,\ldots,x_p)$ estaría asociado al conjunto de $p$ variables independientes y $\beta=(\beta_0,\beta_1,\ldots,\beta_p)'$ representaría el vector de parámetros.

Cabe destacar que el cociente que interviene en la Ecuación (2.1) se conoce como Odds y no es más que la probabilidad de éxito entre la probabilidad de fracaso de la variable $Y$, es decir,

$$Odds(Y)=\frac{\mathbb{P}(Y=1)}{\mathbb{P}(Y=0)}=\frac{\pi(x)}{1-\pi(x)},$$

de tal forma que la Odds siempre será un número perteneciente al intervalo $[0,\infty)$, tomando valores mayores que 1 si la probabilidad de éxito es mayor que la de fracaso, y menores que 1 en caso contrario. Por tanto al aplicar el logaritmo conseguimos llegar a un valor entre $(-\infty,\infty)$, tal y como pretendíamos.

A partir de la Ecuación (2.1), podemos construir el modelo logístico sin más que aplicar la inversa de la función logit dada por

$$g^{-1}(z)=\frac{e^z}{1+e^z}, \text{ con } z\in(-\infty,\infty),$$

de tal forma que

$$\pi(x) = g^{-1}(x'\beta) = \frac{e^{x'\beta}}{1 + e^{x'\beta}}. \tag{2.2}$$

### 2.1.1. Estimación de los coeficientes

En este apartado supondremos que tenemos una muestra de $n$ observaciones independientes donde para cada observación contaremos con los valores correspondientes a las diferentes variables explicativas y a la variable respuesta: $(x_i, y_i)$ para $i = 1, \ldots, n$.

Con el fin de poder ajustar el modelo logístico, tendremos que proceder a la estimación de los coeficientes $\beta$ que son desconocidos. Para ello, podemos recurrir a técnicas de máxima verosimilitud teniendo en cuenta que cada $y_i$ sigue una distribución Bernoulli($\pi(x_i)$). Así pues, la estimación de los coeficientes, que denotaremos por $\hat{\beta}$, vendrá dada por aquellos valores que maximicen la función de verosimilitud

$$\mathcal{L}(\beta) = \prod_{i=1}^{n} \left[\pi(x_i)^{y_i}(1 - \pi(x_i))^{1-y_i}\right]. \tag{2.3}$$

No obstante, no existe una fórmula explícita para estos estimadores, que tendrán que ser calculados a través de métodos computacionales iterativos como Newton-Raphson o el método IRLS (*iteratively reweigted least squares*). Se puede encontrar más información al respecto en McCullagh (2019) o Venables and Ripley (2013).

Una vez calculados los estimadores, podremos realizar contrastes de hipótesis para comprobar si un coeficiente es o no significativo; o lo que es lo mismo, estudiar si cumple la hipótesis nula de que el coeficiente toma el valor 0. Para ello se empleará el llamado estadístico de Wald, que bajo la hipótesis nula sigue una $N(0, 1)$:

$$\frac{\hat{\beta}_j}{\hat{\sigma}(\hat{\beta}_j)} \sim N(0, 1),$$

siendo $\hat{\beta}_j$ la estimación de cierto coeficiente asociado a la variable $x_j$ y $\hat{\sigma}(\hat{\beta}_j)$ su error típico estimado. De este modo, si obtenemos que algún coeficiente no es significativo podremos eliminar la variable asociada del modelo y hacerlo más sencillo.

Otro método de selección de variables para simplificar el modelo consistiría en emplear un criterio global como el AIC, tal y como ocurría en los modelos de regresión lineal múltiple. Recordemos que

$$AIC = -2\ln(\mathcal{L}(\hat{\beta})) + 2p,$$

con $p$ el número de variables del modelo; por lo que en el caso de la regresión logística también podremos llevar a cabo métodos de selección forward o backward que minimicen el AIC, consiguiendo así un modelo con una verosimilitud grande y un número más reducido de parámetros, y por tanto más manejable.

### 2.1.2. Interpretación de los coeficientes

Para poder dotar de una interpretación a estos coeficientes, tenemos que introducir un nuevo concepto conocido como Odds Ratio, que no es más que el cociente de Odds de dos poblaciones, o lo que es lo mismo,

$$Odds\ Ratio = \frac{\pi(x=a)/(1-\pi(x=a))}{\pi(x=b)/(1-\pi(x=b))},$$

donde $a$ y $b$ representan dos poblaciones distintas.

Una vez hemos definido la Odds Ratio, estudiaremos su interpretación en función de la naturaleza de la variable explicativa. En primer lugar, supongamos que nuestra variable explicativa es una variable dicotómica que toma los valores $x=0$ y $x=1$, en este caso tendremos la siguiente expresión,

$$Odds\ Ratio = \frac{\pi(x=1)/(1-\pi(x=1))}{\pi(x=0)/(1-\pi(x=0))}, \tag{2.4}$$

en cuyo denominador está presente la Odds correspondiente al grupo que toma el valor $x=0$, al que denominaremos grupo de referencia. Desarrollando dicha Odds obtenemos que

$$Odds(Y|x=0) = \frac{\mathbb{P}(Y=1|x=0)}{\mathbb{P}(Y=0|x=0)} = \frac{\pi(x=0)}{1-\pi(x=0)} = e^{\beta_0}, \tag{2.5}$$

donde $\pi(x=0) = \dfrac{e^{\beta_0+\beta_1\cdot 0}}{1+e^{\beta_0+\beta_1\cdot 0}} = \dfrac{e^{\beta_0}}{1+e^{\beta_0}}$.

De esta forma, hemos llegado a que la Odds del grupo de referencia se corresponde con la exponencial del intercepto, lo que implica que para valores del intercepto negativos, la probabilidad de que $Y=1$ será menor que 0.5 y para valores de $\beta_0$ positivos, tendríamos que la probabilidad de $Y=1$ será mayor de 0.5.

Análogamente, si vemos lo que ocurre con el grupo dado por $x=1$ obtendremos que

$$Odds(Y|x=1) = \frac{\mathbb{P}(Y=1|x=1)}{\mathbb{P}(Y=0|x=1)} = \frac{\pi(x=1)}{1-\pi(x=1)} = e^{\beta_0+\beta_1}, \tag{2.6}$$

siendo $\pi(x=1) = \dfrac{e^{\beta_0+\beta_1\cdot 1}}{1+e^{\beta_0+\beta_1\cdot 1}} = \dfrac{e^{\beta_0+\beta_1}}{1+e^{\beta_0+\beta_1}}$.

Finalmente, sustituyendo las expresiones dadas en (2.5) y (2.6) en la Ecuación (2.4) obtenemos que

$$Odds\ Ratio = \frac{\pi(x=1)/(1-\pi(x=1))}{\pi(x=0)/(1-\pi(x=0))} = e^{\beta_1}.$$

En esta ocasión, para un coeficiente $\beta_1$ mayor que 0 tendremos una Odds Ratio mayor que 1, lo que indicaría un incremento en la Odds del grupo correspondiente a $x=1$ con respecto al grupo de referencia. Mientras que si $\beta_1$ fuese negativo, tendríamos una Odds Ratio menor que 1 y por tanto una disminución en la Odds.

En el caso de contar con una variable discreta con más de dos categorías, lo que se suele hacer es recodificarlas como variables *dummy*. Como regla general, para $k$ categorías necesitaremos $k-1$ variables *dummy*, ilustrémoslo con un ejemplo.

Supongamos que tenemos una variable categórica que toma 3 categorías, entonces construiríamos dos variables que se codificarían como figura en la Tabla 2.1, dependiendo de la categoría que queramos establecer como referencia (en este caso se correspondería con la Categoría 1).

| Variables | Cat 1 | Cat 2 | Cat 3 |
|---|---|---|---|
| $x_1$ | 0 | 1 | 0 |
| $x_2$ | 0 | 0 | 1 |

Tabla 2.1: Ejemplo de codificación de variables *dummy* para una variable categórica con 3 categorías.

Esta codificación nos permite fijar un grupo de referencia de tal forma que la exponencial de los coeficientes de la regresión logística asociados a cada variable *dummy* nos dará la Odds Ratio entre el grupo de referencia y el grupo para el que dicha variable tome el valor 1.

Siguiendo con el ejemplo, para el modelo de regresión logística que considera como predictor una variable categórica con 3 categorías obtendríamos

$$\log\left(\frac{\pi(x)}{1-\pi(x)}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2,$$

por lo que si quisiésemos calcular la Odds Ratio entre la Categoría 1 y la Categoría 3 llegaríamos a la expresión

$$Odds\ Ratio(Cat\ 1, Cat\ 3) = \frac{Odds(Y|x_1=0, x_2=1)}{Odds(Y|x_1=0, x_2=0)} = \frac{e^{\beta_0+\beta_1\cdot 0+\beta_2\cdot 1}}{e^{\beta_0+\beta_1\cdot 0+\beta_2\cdot 0}} = \frac{e^{\beta_0+\beta_2}}{e^{\beta_0}} = e^{\beta_2},$$

que como ya hemos comentado significa que la variación entre la Odds del grupo de referencia y la de la categoría 3 vendrá dada por $e^{\beta_2}$.

En el caso de contar con una variable explicativa continua, tendríamos que el modelo logístico seguiría la expresión

$$\log\left(\frac{\pi(x)}{1-\pi(x)}\right) = \beta_0 + \beta_1 x_1,$$

y llegaríamos a que $e^{\beta_1}$ se corresponde con la Odds Ratio resultante de haber incrementado en una unidad la variable explicativa, como podemos ver a continuación,

$$Odds\ Ratio(x_1 = x_0, x_1 = x_0 + 1) = \frac{Odds(Y|x_1 = x_0 + 1)}{Odds(Y|x_1 = x_0)} = \frac{e^{\beta_0+\beta_1(x_0+1)}}{e^{\beta_0+\beta_1 x_0}} = e^{\beta_1}.$$

Cabe observar, que para un modelo con más de una variable explicativa donde intervengan los distintos tipos de variables que hemos visto (dicotómica, categórica con varios grupos o continua), seguiremos pudiendo interpretar las exponenciales de los coeficientes como con las Odds Ratio asociadas al efecto de cada variable; pero eso sí, considerando que el resto de variables se mantienen constantes.

## 2.2. Weigth of Evidence (WoE)

Como ya hemos visto, para poder incluir variables explicativas categóricas en nuestro modelo, una técnica habitual es recurrir a variables *dummy*. Sin embargo, esto ocasiona que el número de variables se vea incrementado sustancialmente y lleva acarreado la necesidad de calcular un estimador para cada categoría.

Es por ello que otro enfoque para tratar las variables categóricas es aplicarles una transformación para obtener variables *WoE* (*Weight of Evidence*) para cada categoría que se calculan como sigue,

$$WOE_i = \log\left(\frac{\frac{N_i}{\sum_{i=1}^{k} N_i}}{\frac{P_i}{\sum_{i=1}^{k} P_i}}\right), \quad i = 1, ..., k; \tag{2.7}$$

siendo $k$ el número de categorías, $N_i$ el número de eventos negativos (en nuestro caso contratos sanos) y $P_i$ el número de eventos positivos (contratos morosos) para cada categoría.

Mediante esta transformación, obtenemos una variable discreta en el sentido de que solamente puede tomar unos pocos valores (tantos como categorías), pero que a efectos de modelización podemos tratar como si fuese una variable continua ya que solo necesitamos calcular un estimador por variable, lo que le otorga una mayor simplicidad al modelo.

Además, tendremos que la variable *WoE* tomará valores positivos para aquellas categorías donde la tasa de sanos sea superior a la tasa de morosos y valores negativos en caso contrario. Para una discusión más detallada se puede consultar Anderson (2007) y Smith et al. (2002).

Para transformar las variables discretas de nuestra base de datos en variables *WoEs* hemos empleado la función `woebin` de la librería `scorecard` de R (Xie, 2024).

```
woes=woebin(datos,x=var_discretas_2,y="MORA_12M",positive=0, breaks_list=categorias)
```

Para ello, tenemos que introducir la base de datos, las variables que queremos transformar en *WoEs*, los puntos de corte en los que dividimos las categorías, el indicador de malo y en este caso, dado que la función calcula los *WoEs* como $\log(Positivos/Negativos)$, para que concuerde con la Ecuación (2.7), tenemos que seleccionar `positive=0`.

En la Tabla 2.2, mostramos un ejemplo tomando la transformación de la variable `VAR_CLI_DISC_1`, donde podemos ver que la tasa de negativos es menor que la de positivos en los 2 primeros grupos mientras que en el tercero ocurre al contrario. De esta manera obtenemos una información que relaciona directamente a cada categoría con su comportamiento frente al *default*.

| | Variable | Categoría | WoE |
|---|---|---|---|
| 1 | VAR_CLI_DISC_1 | Grupo 1 | -0.36 |
| 2 | VAR_CLI_DISC_1 | Grupo 2 | -0.04 |
| 3 | VAR_CLI_DISC_1 | Grupo 3 | 0.72 |

Tabla 2.2: Ejemplo de transformación *WoE*.

## 2.3. Aplicación práctica

Una vez hemos sentado las bases en las que se fundamenta la regresión logística, se ha procedido a aplicar el método y al análisis de los resultados correspondientes, tal y como se indica en las siguientes secciones.

### 2.3.1. Ajuste del modelo

El primer paso es dividir el conjunto de datos en entrenamiento y test, tal y como veíamos en la Sección 1.2.2, realizando dicha división, para esta fase de ajuste contaremos con XXXX observaciones de las cuales XXXX se identifican como sanas y XXX como morosas.

Después de haber llevado a cabo esta separación, usaremos los datos de entrenamiento para ajustar una regresión logística a través de la función `glm`, donde emplearemos las 28 características de las que disponemos como variables explicativas. Posteriormente, mediante la aplicación de la función `summary` al modelo, analizaremos los estimadores de los coeficientes.

```
reg.log=glm(MORA_12M~.,family="binomial",data=train)
summary(reg.log)
```

```
Call:
glm(formula = MORA_12M ~ ., family = "binomial", data = train)

Coefficients:
                    Estimate  Std. Error z value Pr(>|z|)
(Intercept)       -4.085e+00  5.837e-01  -6.999 2.58e-12 ***
VAR_OP_CONT_1     -5.920e-05  1.284e-04  -0.461 0.644615
VAR_OP_CONT_2      1.278e-01  6.415e-02   1.991 0.046432 *
```

```
VAR_OP_CONT_4      -5.025e-06  9.172e-06  -0.548 0.583764
VAR_OP_CONT_5       9.531e-01  7.124e-01   1.338 0.180944
VAR_OP_CONT_6      -8.986e-01  7.647e-01  -1.175 0.239919
VAR_OP_CONT_7      -9.042e-01  7.504e-01  -1.205 0.228240
VAR_OP_CONT_8       1.048e+00  7.468e-01   1.403 0.160643
VAR_OP_CONT_9      -8.045e-04  4.102e-03  -0.196 0.844524
VAR_CLI_CONT_2     -3.259e-02  6.304e-03  -5.170 2.34e-07 ***
VAR_CLI_CONT_4      5.832e-03  4.493e-03   1.298 0.194304
VAR_CLI_CONT_5     -1.104e-02  9.760e-03  -1.131 0.257931
VAR_CLI_CONT_8      2.510e-02  6.770e-03   3.707 0.000210 ***
VAR_CLI_CONT_10    -2.934e-03  3.891e-03  -0.754 0.450780
VAR_CLI_CONT_12    -1.314e-03  1.868e-02  -0.070 0.943938
VAR_CLI_CONT_14     4.189e-02  2.108e-01   0.199 0.842499
VAR_CLI_CONT_15    -6.340e-07  1.695e-06  -0.374 0.708333
VAR_CLI_CONT_16    -5.376e-07  1.255e-06  -0.428 0.668426
VAR_CLI_CONT_17     1.165e-03  3.342e-03   0.349 0.727356
VAR_CLI_CONT_18     1.303e-04  1.814e-04   0.718 0.472606
VAR_CLI_CONT_19     1.286e-05  4.903e-05   0.262 0.793088
VAR_CLI_DISC_WOE_1 -5.836e-01  2.079e-01  -2.807 0.004994 **
VAR_CLI_DISC_WOE_3 -1.729e+00  2.883e-01  -5.998 2.00e-09 ***
VAR_CLI_DISC_WOE_4 -9.057e-01  4.511e-01  -2.008 0.044670 *
VAR_CLI_DISC_WOE_5  2.534e-01  5.885e-01   0.431 0.666800
VAR_CLI_DISC_WOE_6 -7.329e-01  3.103e-01  -2.362 0.018190 *
VAR_CLI_DISC_WOE_7  4.140e-01  3.000e-01   1.380 0.167548
VAR_OP_DISC_WOE_1  -8.408e-01  1.471e-01  -5.715 1.10e-08 ***
VAR_OP_DISC_WOE_2  -8.861e-01  2.413e-01  -3.672 0.000241 ***
---
Signif. codes:  0 "***" 0.001 "**" 0.01 "*" 0.05 "." 0.1 " " 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 1895.6  on 4539  degrees of freedom
Residual deviance: 1647.7  on 4511  degrees of freedom
AIC: 1705.7

Number of Fisher Scoring iterations: 8
```

Al utilizar todas las variables disponibles como predictoras, vemos que muchos de los coeficientes asociados no son estadísticamente significativos para un nivel de significación del 5 %. Para intentar subsanar este problema, podemos llevar a cabo un método de selección de variables mediante la función `step` que tiene como objetivo minimizar el AIC del modelo, que en este caso toma el valor de 1705.7.

```r
final.log=step(reg.log,trace = F)
summary(final.log)
```

```
Call:
glm(formula = MORA_12M ~ VAR_OP_CONT_2 + VAR_CLI_CONT_2 + VAR_CLI_CONT_8 +
VAR_CLI_CONT_16 + VAR_CLI_DISC_WOE_1 + VAR_CLI_DISC_WOE_3 +
VAR_CLI_DISC_WOE_4 + VAR_CLI_DISC_WOE_6 + VAR_CLI_DISC_WOE_7 +
VAR_OP_DISC_WOE_1 + VAR_OP_DISC_WOE_2, family = "binomial",
data = train)

Coefficients:
                    Estimate  Std. Error z value Pr(>|z|)
(Intercept)        -4.105e+00  2.884e-01 -14.236  < 2e-16 ***
VAR_OP_CONT_2       1.159e-01  3.622e-02   3.199  0.00138 **
VAR_CLI_CONT_2     -3.137e-02  5.959e-03  -5.264 1.41e-07 ***
VAR_CLI_CONT_8      2.986e-02  5.942e-03   5.024 5.06e-07 ***
VAR_CLI_CONT_16    -9.478e-07  6.531e-07  -1.451  0.14671
VAR_CLI_DISC_WOE_1 -5.611e-01  1.956e-01  -2.868  0.00413 **
VAR_CLI_DISC_WOE_3 -1.827e+00  2.747e-01  -6.652 2.90e-11 ***
VAR_CLI_DISC_WOE_4 -1.008e+00  4.390e-01  -2.297  0.02163 *
VAR_CLI_DISC_WOE_6 -6.300e-01  1.998e-01  -3.153  0.00161 **
VAR_CLI_DISC_WOE_7  4.709e-01  2.847e-01   1.654  0.09812 .
VAR_OP_DISC_WOE_1  -8.866e-01  1.397e-01  -6.345 2.22e-10 ***
VAR_OP_DISC_WOE_2  -7.499e-01  1.417e-01  -5.291 1.21e-07 ***
---
Signif. codes:  0 "***" 0.001 "**" 0.01 "*" 0.05 "." 0.1 " " 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 1895.6  on 4539  degrees of freedom
Residual deviance: 1656.6  on 4528  degrees of freedom
AIC: 1680.6

Number of Fisher Scoring iterations: 7
```

De esta forma hemos conseguido disminuir el número de variables explicativas implicadas llegando a un $AIC = 1680.6$. Pese a esta mejora, todavía hay coeficientes que no son significativos al 5 %, por lo que podemos intentar ajustar un nuevo modelo que no tenga en cuenta la variable asociada al coeficiente con un mayor $p$-valor y ver que nuevos resultados obtenemos.

Así pues, comenzamos eliminando la variable `VAR_CLI_CONT_16` que tiene asociado un $p$-valor de 0.14671, obteniendo el siguiente modelo.

```
reg.log.2=glm(MORA_12M~.,family="binomial",data=datos_logit)
summary(reg.log.2)
```

```
Call:
glm(formula = MORA_12M ~ ., family = "binomial", data = datos_logit)

Coefficients:
                    Estimate  Std. Error z value Pr(>|z|)
(Intercept)        -4.064590   0.287837 -14.121  < 2e-16 ***
VAR_OP_CONT_2       0.112698   0.036361   3.099  0.00194 **
VAR_CLI_CONT_2     -0.031184   0.005951  -5.240 1.61e-07 ***
VAR_CLI_CONT_8      0.027823   0.005835   4.768 1.86e-06 ***
VAR_CLI_DISC_WOE_1 -0.560968   0.196340  -2.857  0.00427 **
VAR_CLI_DISC_WOE_3 -1.804560   0.274328  -6.578 4.76e-11 ***
VAR_CLI_DISC_WOE_4 -1.048054   0.437650  -2.395  0.01663 *
VAR_CLI_DISC_WOE_6 -0.635989   0.199843  -3.182  0.00146 **
VAR_CLI_DISC_WOE_7  0.595989   0.273202   2.181  0.02915 *
VAR_OP_DISC_WOE_1  -0.898022   0.139179  -6.452 1.10e-10 ***
VAR_OP_DISC_WOE_2  -0.752234   0.141815  -5.304 1.13e-07 ***
---
Signif. codes:  0 "***" 0.001 "**" 0.01 "*" 0.05 "." 0.1 " " 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 1895.6  on 4539  degrees of freedom
Residual deviance: 1659.9  on 4529  degrees of freedom
AIC: 1681.9

Number of Fisher Scoring iterations: 7
```

Siguiendo esta metodología, se observa que se ha estimado un modelo para el que todos los coeficientes se vuelvan significativos para un nivel del 5 %. Además, hemos simplificando el modelo al contar con menos variables y únicamente hemos aumentado el AIC en 1.3 unidades, por lo que resulta razonable considerar este nuevo modelo en lugar del anterior. Cabe destacar que el modelo final cuenta con 10 variables de las cuales 7 hacen referencia a características socioeconómicas del cliente y tan solo 3 a las características propias de la operación.

### 2.3.2. Análisis del modelo

Como hemos visto en el apartado metodológico, el modelo de regresión logística es fácilmente interpretable ya que los coeficientes asociados a cada variable nos dan una información precisa de cómo se comporta el modelo de cara a predecir una observación.

Con el fin de analizar los coeficientes del modelo, comenzamos obteniendo las estimaciones de los mismos mediante el siguiente comando.

```
coef(reg.log.2)
```

```
(Intercept)      VAR_OP_CONT_2     VAR_CLI_CONT_2     VAR_CLI_CONT_8 VAR_CLI_DISC_WOE_1
-4.06458960         0.11269833        -0.03118354         0.02782328        -0.56096832
VAR_CLI_DISC_WOE_3 VAR_CLI_DISC_WOE_4 VAR_CLI_DISC_WOE_6 VAR_CLI_DISC_WOE_7  VAR_OP_DISC_WOE_1
-1.80455953        -1.04805359        -0.63598903         0.59598929        -0.89802222
VAR_OP_DISC_WOE_2
-0.75223353
```

Si calculamos las exponenciales de los coeficientes, obtenemos las Odds Ratio correspondientes al efecto que ejerce cada una de las variables, considerando que el resto se mantienen constantes.

```
exp(coef(reg.log.2))
```

```
(Intercept)      VAR_OP_CONT_2     VAR_CLI_CONT_2     VAR_CLI_CONT_8 VAR_CLI_DISC_WOE_1
0.01717003         1.11929422         0.96929765         1.02821396         0.57065622
VAR_CLI_DISC_WOE_3 VAR_CLI_DISC_WOE_4 VAR_CLI_DISC_WOE_6 VAR_CLI_DISC_WOE_7  VAR_OP_DISC_WOE_1
0.16454692         0.35061953         0.52941162         1.81482545         0.40737456
VAR_OP_DISC_WOE_2
0.47131269
```

Por ejemplo, en el caso de `VAR_OP_CONT_2`, obtenemos un coeficiente $\beta_1 = 0.112698$ (mayor que 0) y una Odds Ratio de 1.12 (mayor que 1), en consecuencia, la Odds de entrar en *default* se incrementará en un 12 % por cada unidad que aumente `VAR_OP_CONT_2`, siempre y cuando las demás variables permanezcan constantes.

Por otro lado, en el caso de la variable `VAR_CLI_CONT_2` presenta un coeficiente asociado negativo, $\beta_2 = -0.031184$ y por tanto una Odds Ratio de 0.97 (menor que 1), lo que implica que por cada unidad

que aumente `VAR_CLI_CONT_2` (manteniéndose el resto de variables constantes), la Odds de entrar en *default* se reducirá en un 3 %.

Por otro lado, la Odds se correspondería con $e^{\beta_0} = 0.017$, que al ser menor que 1 nos estaría indicando que, para el caso en el que todas las variables explicativas tomasen el valor 0, la probabilidad de que $Y = 1$, es decir, de que esa operación entre en *default*, es menor que 0.5. Cabe destacar, que esta información no es del todo relevante debido a la existencia de variables explicativas que no pueden tomar el valor 0.

### 2.3.3. Evaluación del modelo

Una vez que hemos ajustado el modelo con la muestra de entrenamiento, procederemos a evaluar su rendimiento con el conjunto de test, en este caso se compone de XXXX observaciones de las cuales XXXX están sanas y XX se encuentran en *default*. Para evaluar el modelo el mecanismo a seguir será calcular la predicción que arroja para cada observación y analizar a través de distintas métricas cómo de bien está prediciendo.

Como primer paso, en la Figura 2.1, podemos observar un histograma que representa la distribución de probabilidad de mora separada en función de si la observación se corresponde con una operación sana (0) o morosa (1).



Figura 2.1: Histograma de las probabilidades de mora estimadas en función de la categoría observada.

Como podemos ver en la Figura 2.1, en el caso de las operaciones sanas, las probabilidades se sitúan con valores bajos, que es lo que buscamos. En lo que se refiere a operaciones morosas, estas estimaciones también tienden a concentrarse en valores bajos de probabilidad (aunque estas tienden a ser más altas que en el caso de operaciones sanas); sin embargo, lo que nos gustaría es que las probabilidades se concentrasen en valores elevados. Esto es un indicativo de que el modelo tenderá a predecir mejor las operaciones sanas que las morosas, lo cual tiende a ocurrir cuando las clases están desbalanceadas.

A continuación, en la Tabla 2.3 representamos la matriz de confusión correspondiente a considerar morosas aquellas operaciones para las que el modelo arroja una probabilidad superior a 0.1 y que ha sido calculada a través del siguiente código.

```
pred <- factor(p.est > 0.1, labels = c("0", "1"))
table(obs,pred)
```



Tabla 2.3: Matriz de confusión para el modelo de regresión logística ajustado.

Con la función `confusionMatrix` de la librería `caret`, podemos obtener diversas métricas de evaluación vistas en la Sección 1.3.1, algunas de las cuales resumimos en la Tabla 2.4.

| TPR | TNR | Accuracy | Balanced Accuracy | $\kappa$ |
|---|---|---|---|---|
| 0.50820 | 0.87163 | 0.8521 | 0.68991 | 0.207 |

Tabla 2.4: Valores de las métricas para el modelo de regresión logística ajustado.

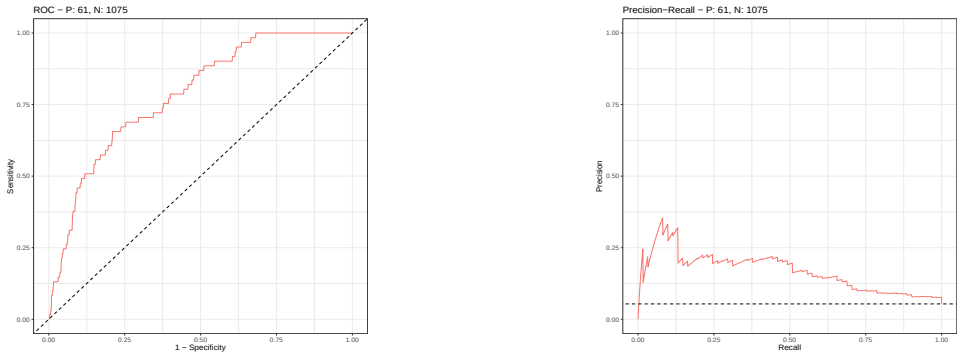
En base a estos resultados, podemos observar que el *TNR* es bastante superior al *TPR*, lo que quiere decir que el modelo acierta más a la hora de predecir operaciones sanas que operaciones morosas (claramente influido por tener una muestra con una mayor proporción de operaciones sanas). Por otra parte, los valores de *Balanced Accuracy* y $\kappa$ son aceptables.

A continuación, representaremos las curvas *ROC* y *Precision-Recall* en la Figura 2.2 y calcularemos sus *AUC* respectivos. Para ello emplearemos el paquete `precrec` (Saito and Rehmsmeier, 2017) en el que primero creamos un objeto con la función `evalmod` que toma como argumentos la probabilidad de mora y la marca de mora con la que están etiquetadas las observaciones de la muestra de test.

```
sscurves=evalmod(scores=p.est,labels=obs)
autoplot(sscurves, "ROC")
autoplot(sscurves, "PR")
aucs <- auc(sscurves)
```

Una vez construido este objeto, podemos representar la curva especificando si es del tipo $ROC$ o del tipo $PR$, y aplicando la función `auc` obtenemos las áreas bajo la curva correspondientes.

Hemos representado con una línea roja continua las respectivas curvas y en trazo negro discontinuo la curva que correspondería a un clasificador aleatorio, en el caso de la curva $ROC$ se corresponde con la recta $y = x$, y en el caso de la curva $PR$ se trata de una recta horizontal a la altura de $\frac{P}{P+N} = 0.05370$, en este caso.




(a) Curva $ROC$. (b) Curva $PR$.

Figura 2.2: Curvas $ROC$ y $PR$ para el modelo de regresión logística ajustado.

Como ya comentamos en la Sección 1.3.2 y 1.3.3, podemos medir el desempeño del modelo calculando el área bajo las dos curvas representadas, recogemos dichas medidas en la Tabla 2.5.

| **Curva** | ROC | PR |
|---|---|---|
| **AUC** | 0.7833 | 0.1666 |

Tabla 2.5: Áreas bajo las curvas $ROC$ y $PR$ del modelo de regresión logística ajustado.

Con respecto al $AUC$ de la curva $ROC$, recordemos que un valor de 0.5 se correspondía con una clasificación al azar y el 1 con una clasificación perfecta, por lo que obtenemos un valor bastante aceptable. Mientras que para el $AUC$ de la *Precision-Recall*, el valor correspondiente a un clasificador aleatorio sería de 0.05370, lo que significa que también estaríamos por encima.

De cara a realizar una comparación con el resto de métodos, dado que la selección al azar de las observaciones para la muestra de entrenamiento y test incorpora aleatoriedad en las métricas calculadas para evaluar el modelo, procederemos a generar 100 muestras de entrenamiento y test distintas sobre las que ajustar y evaluar el modelo.

Siguiendo este enfoque construiremos 100 modelos de regresión logística empleando la función `step` que ya hemos empleado, y cuyo objetivo es seleccionar las variables que entran en el modelo de tal

forma que se minimice el AIC.

```
df=datos_filtrados_2
B=100
auc.log=numeric(B)
pr.log=numeric(B)
set.seed(2311)

for (i in 1:B){
        nobs <- nrow(df)
        itrain <- sample(nobs, 0.8 * nobs)
        train <- df[itrain, ]
        test <- df[-itrain, ]
        reg.log=glm(MORA_12M~.,family="binomial",data=train)
        final.log=step(reg.log,trace = F)
        obs <- test$MORA_12M
        p.est <- predict(final.log, type = "response", newdata = test)
        sscurves=evalmod(scores=p.est,labels=obs)
        aucs <- auc(sscurves)
        auc.log[i]=aucs$aucs[1]
        pr.log[i]=aucs$aucs[2]
}

auc.log
summary(auc.log)
auc.log.medio=mean(auc.log)
auc.log.medio

pr.log
summary(pr.log)
pr.log.medio=mean(pr.log)
pr.log.medio
```

En la Tabla 2.6 resumimos el valor de el área bajo las curvas $ROC$ y $PR$ recogiendo el mínimo, el máximo y la media de esas métricas para los 100 modelos simulados. En el último capítulo, utilizaremos estas métricas para realizar una comparación entre los modelos.

| | Mínimo | Máximo | Media |
|---|---|---|---|
| **AUC ROC** | 0.6942 | 0.8283 | 0.7643 |
| **AUC PR** | 0.09811 | 0.25311 | 0.16693 |

Tabla 2.6: Mínimo, máximo y media de los *AUC ROC* y *AUC PR* relativos a los 100 modelos de regresión logística simulados.

# Capítulo 3

# Métodos de regularización

En este capítulo, en el que trataremos los llamados métodos de regularización o penalización, empezaremos proporcionando una visión general basada en su aplicación a modelos de regresión lineal (Friedman et al., 2010; Hastie et al., 2009; Berk et al., 2008), para después adaptarlos a la regresión logística (Pereira et al., 2016; James et al., 2013). Una vez introducidos los modelos de regularización, procederemos a detallar la regresión *ridge* (Hoerl and Kennard, 1970), *LASSO* (Tibshirani, 1996) y *elastic net* (Zou and Hastie, 2005), siendo esta última la que aplicaremos a nuestro conjunto de datos y posteriormente evaluaremos.

## 3.1. Marco metodológico

A la hora de ajustar un modelo de regresión, nos podemos encontrar con una serie de problemas relativos a las variables explicativas. Estos pueden ser tales como que exista una elevada dependencia entre las variables, que estas no aporten suficiente información (siendo irrelevantes para el modelo), que tengamos un gran número de predictores en comparación con el número de observaciones o que simplemente queramos reducir la cantidad de variables, con el objetivo de dotar de mayor interpretabilidad al modelo.

Un método para afrontar estas situaciones es recurrir a los llamados modelos penalizados o de regularización, cuyo enfoque se basa en añadir una penalización a la hora de estimar los coeficientes de las variables explicativas. Con esta práctica, se pretende forzar a que los parámetros de las variables irrelevantes, desde el punto de vista de la explicación de la variable respuesta, tomen valores iguales o cercanos a 0.

Incluir esta penalización tiene como consecuencia inmediata una reducción de la varianza del modelo, lo que puede conllevar una mejora en la precisión de la predicción; no obstante, a cambio, estamos aumentando el sesgo.

Una primera aplicación de estos métodos de regularización se lleva a cabo en el seno de los modelos de regresión lineal múltiple, donde la estimación de los coeficientes $\beta$ se obtiene a partir del método de mínimos cuadrados, es decir,

$$\hat{\beta} = \min_{\beta} \sum_{i=1}^{n} (y_i - x_i'\beta)^2 .$$

Así pues, el método de regularización consistiría en añadir una penalización a ese problema de minimización, de tal forma que pasaríamos a resolver

$$\hat{\beta} = \min_{\beta} \sum_{i=1}^{n} (y_i - x_i'\beta)^2 + \lambda P(\beta),$$

donde la elección de la penalización aplicada $P(\beta)$ variará dependiendo del método utilizado y $\lambda$ será un parámetro a determinar que cuantifica el peso de la penalización.

En nuestro caso, debido a los motivos expuestos en el anterior capítulo, nos resultará más interesante analizar los distintos modelos de regularización consistentes en penalizar una regresión logística, en lugar de una regresión lineal.

Recordemos que los coeficientes de la regresión logística podían obtenerse maximizando la Ecuación (2.3), o lo que es lo mismo, maximizar la log-verosimilitud que vendrá dada por

$$l(\beta) = \sum_{i=1}^{n} \left[ y_i \log(\pi(x_i)) + (1 - y_i) \log(1 - \pi(x_i)) \right].$$

Por consiguiente, empleando las propiedades de los logaritmos y las Ecuaciones (2.1) y (2.2) llegamos a que la log-verosimilitud se puede reescribir como

$$\begin{aligned} l(\beta) &= \sum_{i=1}^{n} \left[ y_i \log\left( \frac{\pi(x_i)}{1 - \pi(x_i)} \right) + \log(1 - \pi(x_i)) \right] \\ &= \sum_{i=1}^{n} \left[ y_i x_i'\beta - \log(1 + e^{x_i'\beta}) \right], \end{aligned}$$

de tal forma que podremos obtener la estimación de los coeficientes a partir de la siguiente expresión penalizada

$$\hat{\beta} = \max_{\beta} \sum_{i=1}^{n} \left[ y_i x_i'\beta - \log(1 + e^{x_i'\beta}) \right] - \lambda P(\beta). \tag{3.1}$$

Una vez obtenida la expresión general de la regresión logística penalizada, analizaremos los distintos métodos en función de cuál sea la definición de la penalización $P(\beta)$.

Por otra parte, teníamos pendiente determinar la selección del parámetro $\lambda$, esto puede realizarse mediante técnicas de validación cruzada. Las dos más extendidas son *Leave-one-out Cross Validation* (LOOCV) y *K-fold Cross Validation* sobre las que daremos una pequeña idea a continuación (James et al., 2013).

La técnica LOOCV consiste en particionar la muestra de entrenamiento de tal forma que dejemos una única observación para evaluar la predicción, y el resto para predecir. La idea sería realizar este

proceso para todas las observaciones y obtener medidas de error individuales que, promediándose, darían una medida de error global.

Uno de los mayores inconvenientes de esta metodología es el elevado tiempo de computación, dado que se tiene que realizar y evaluar la predicción para cada una de las $n$ observaciones. La *K-fold Cross Validation* disminuye la exigencia computacional al construir $k$ grupos de observaciones de los cuales se reserva uno para evaluar la predicción y se predice con los $k-1$ restantes; por tanto solamente habría que repetir el proceso $k$ veces ($k=5$ o $k=10$ habitualmente) en lugar de $n$.

Es importante destacar que para implementar las técnicas que trataremos a continuación suele ser conveniente estandarizar las variables explicativas ya que la penalización es sensible a los cambios de escala.

### 3.1.1. Regresión Ridge

La regresión *ridge* fue introducida por primera vez en Hoerl and Kennard (1970) y su objetivo radicaba en intentar solventar la inestabilidad presente en las estimaciones de los coeficientes ante la existencia de variables explicativas altamente correladas. Para ello, el método se fundamentaba en tomar como función de penalización la norma $L_2$ del vector de parámetros, sin considerar el $\beta_0$; esto es, $P(\beta) = \sum_{j=1}^{p} \beta_j^2$.

Al aplicar esta penalización a la regresión logística y sustituir esta expresión en la Ecuación (3.1), obtenemos

$$\hat{\beta} = \max_{\beta} \sum_{i=1}^{n} \left[ y_i x_i' \beta - \log(1 + e^{x_i' \beta}) \right] - \lambda \sum_{j=1}^{p} \beta_j^2, \tag{3.2}$$

que se corresponde con el siguiente problema de optimización

$$\begin{aligned} \max_{\beta} \quad & \sum_{i=1}^{n} \left[ y_i x_i' \beta - \log(1 + e^{x_i' \beta}) \right], \\ \textit{sujeto a} \quad & \sum_{j=1}^{p} \beta_j^2 \leq C, \end{aligned}$$

donde se puede ver de manera más explícita la restricción sobre los coeficientes limitados por una cierta constante $C$, que se relaciona con el parámetro $\lambda$ de la Ecuación (3.2).

En consecuencia, para los casos más extremos donde $\lambda = 0$ o $C \to \infty$, tendremos que la estimación de los coeficientes será la misma que la que obtenemos a partir de maximizar la verosimilitud. En cambio si $\lambda \to \infty$ o $C = 0$ los coeficientes únicamente podrán tomar el valor 0.

### 3.1.2. Regresión LASSO

Otro método de regularización presentado por Tibshirani (1996) es la regresión *LASSO* que intenta seleccionar el conjunto de variables más influyentes forzando el resto a 0 y aumentando así la

interpretabilidad del modelo. Para ello, en lugar de considerar como función de penalización la norma $L_2$ (como ocurría con la regresión *ridge*) toma la norma $L_1$, es decir, $P(\beta) = \sum_{j=1}^{p} |\beta_j|$. Con lo que obtendríamos la siguiente regresión logística penalizada,

$$\hat{\beta} = \max_{\beta} \sum_{i=1}^{n} \left[ y_i x_i' \beta - \log(1 + e^{x_i' \beta}) \right] - \lambda \sum_{j=1}^{p} |\beta_j|, \tag{3.3}$$

que también podemos expresar en forma de problema de optimización como

$$\begin{aligned} \max_{\beta} \quad & \sum_{i=1}^{n} \left[ y_i x_i' \beta - \log(1 + e^{x_i' \beta}) \right], \\ \textit{sujeto a} \quad & \sum_{j=1}^{p} |\beta_j| \leq C. \end{aligned}$$

### 3.1.3. Elastic Net

En los apartados anteriores hemos visto que la regresión *LASSO* fuerza coeficientes a 0 manteniendo únicamente los más relevantes, mientras que en el caso de la regresión *ridge* no se anulará ningún coeficiente, pero estos sí podrán tomar valores muy próximos a 0. Por otra parte, en el caso de la presencia de variables correladas, la regresión *LASSO* opta por quedarse con una única de esas variables mientras que *ridge* tiene tendencia a darles el mismo peso.

Para solventar estas deficiencias asociadas a cada uno de los métodos, nace *elastic net* de la mano de Zou and Hastie (2005), que es una nueva regresión penalizada a medio camino entre la *ridge* y la *LASSO*, donde la función de penalización viene dada por $P(\beta) = \alpha \sum_{j=1}^{p} |\beta_j| + (1-\alpha) \sum_{j=1}^{p} \beta_j^2$, de tal forma que para $\alpha = 0$ estaríamos ante una regresión *ridge* y con $\alpha = 1$ ante una *LASSO*.

Los coeficientes del *elastic net* se obtendrían como

$$\hat{\beta} = \max_{\beta} \sum_{i=1}^{n} \left[ y_i x_i' \beta - \log(1 + e^{x_i' \beta}) \right] - \lambda \left( \alpha \sum_{j=1}^{p} |\beta_j| + (1-\alpha) \sum_{j=1}^{p} \beta_j^2 \right).$$

cuyo problema de optimización asociado sería

$$\begin{aligned} \max_{\beta} \quad & \sum_{i=1}^{n} \left[ y_i x_i' \beta - \log(1 + e^{x_i' \beta}) \right], \\ \textit{sujeto a} \quad & \alpha \sum_{j=1}^{p} |\beta_j| + (1-\alpha) \sum_{j=1}^{p} \beta_j^2 \leq C. \end{aligned}$$

Obsérvese que en el artículo de Zou and Hastie (2005), aplican una corrección a la estimación de los coeficientes con el fin de mejorar el desempeño del método.

## 3.2. Aplicación práctica

Siguiendo el esquema del capítulo anterior, una vez presentado el marco teórico necesario para describir y comprender el modelo, lo ajustaremos a nuestros datos para poder extraer conclusiones al respecto.

### 3.2.1. Ajuste del modelo

Para llevar a cabo el ajuste de este modelo, así como los que introduciremos en capítulos venideros, hemos optado por hacer uso de la librería `caret` (Kuhn et al., 2020) debido a que incorpora entre otras funcionalidades la posibilidad de seleccionar los hiperparámetros óptimos de cada modelo antes de realizar el ajuste, mencionar que también nos hemos basado en el material de Casal et al. (2021) para llevar a cabo los ajustes con `caret`.

Los conjuntos de entrenamiento y test que usaremos serán los mismos que los empleados en el modelo de regresión logística analizado, dado que hemos mantenido la semilla para garantizar la reproducibilidad.

Por otro lado, como ya hemos comentado, para los métodos de regularización vistos en este capítulo, es recomendable estandarizar las variables antes de construir el modelo, y en este caso la librería `caret` también nos permite llevar a cabo este preprocesamiento de los datos, mediante el cual podremos centrar (`center`) y escalar (`scale`) las variables. Además, eliminará (si hubiese) variables con todos sus valores iguales (`zv`), ya que no aportarían información a la modelización.

Para el método *elastic net*, como ya hemos comentado en la sección anterior, contamos con 2 hiperparámetros: $\alpha$ (`alpha`) y $\lambda$ (`lambda`). Con respecto a como realizar su selección, optaremos por elegir de entre una malla de 5x5 (`tuneLength=5`) aquellos valores que optimicen la $\kappa$ de Cohen (`metric=Kappa`); también se podría optimizar la *Accuracy*, pero al contar con una muestra desbalanceada es más recomendable emplear $\kappa$ (ver Sección 1.3.1). Para ello haremos uso de técnicas de validación cruzada, concretamente la ya comentada *K-fold Cross Validation* con $k = 10$ (`trainControl(method = cv, number = 10, selectionFunction = oneSE)`).

Sin embargo, en lugar de seleccionar los hiperparámetros óptimos seguiremos la metodología propuesta por Breiman et al. (1984), denominada regla de un error estándar (*one standard error*) que consiste en seleccionar el hiperparámetro que haga más sencillo el modelo siempre que este se desvíe como mucho en un error estándar de la precisión obtenida al considerar el valor óptimo de los hiperparámetros. Llevando a cabo este procedimiento, se palia el efecto de la aleatoriedad inherente a la construcción del conjunto de entrenamiento y se considera que no se incurre en diferencias significativas en la precisión del modelo.

Además, como estamos ante un modelo de regresión logística penalizado, tendremos que especificar `family=binomial`. Así pues, el ajuste en  se realizaría según se indica en las siguientes líneas de código.

```
X=as.data.frame(train[,-1])
y=train[,1]$MORA_12M
caret.glmnet <- train(x = X, y = y, method = "glmnet",
preProc = c("zv", "center", "scale"),
family = "binomial",
metric="Kappa",
trControl = trainControl(method = "cv", number = 10, selectionFunction = "oneSE"),
tuneLength=5)
```

```
glmnet

4540 samples
28 predictor
2 classes: '0', '1'

Pre-processing: centered (28), scaled (28)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 4085, 4085, 4087, 4086, 4086, 4085, ...
Resampling results across tuning parameters:

alpha  lambda        Accuracy   Kappa
0.100  2.527553e-05  0.9458158  -0.0012750523
0.100  1.173186e-04  0.9458158  -0.0012750523
0.100  5.445447e-04  0.9458158  -0.0012750523
0.100  2.527553e-03  0.9462563  -0.0004256451
0.100  1.173186e-02  0.9464771   0.0000000000
0.325  2.527553e-05  0.9460365   0.0063333542
0.325  1.173186e-04  0.9460365   0.0063333542
0.325  5.445447e-04  0.9460361  -0.0008503487
0.325  2.527553e-03  0.9464771   0.0000000000
0.325  1.173186e-02  0.9464771   0.0000000000
0.550  2.527553e-05  0.9460365   0.0063333542
0.550  1.173186e-04  0.9460365   0.0063333542
0.550  5.445447e-04  0.9460361  -0.0008503487
0.550  2.527553e-03  0.9464771   0.0000000000
0.550  1.173186e-02  0.9464771   0.0000000000
0.775  2.527553e-05  0.9460365   0.0063333542
0.775  1.173186e-04  0.9460365   0.0063333542
0.775  5.445447e-04  0.9460361  -0.0008503487
0.775  2.527553e-03  0.9464771   0.0000000000
0.775  1.173186e-02  0.9464771   0.0000000000
```

```
1.000  2.527553e-05  0.9460365   0.0063333542
1.000  1.173186e-04  0.9460365   0.0063333542
1.000  5.445447e-04  0.9460361  -0.0008503487
1.000  2.527553e-03  0.9464771   0.0000000000
1.000  1.173186e-02  0.9464771   0.0000000000

Kappa was used to select the optimal model using  the one SE rule.
The final values used for the model were alpha = 0.1 and lambda = 0.01173186.
```

La Figura 3.1, muestra de forma más intuitiva que los valores seleccionados para el modelo son $\alpha = 0.1$ y $\lambda = 0.01173186$.

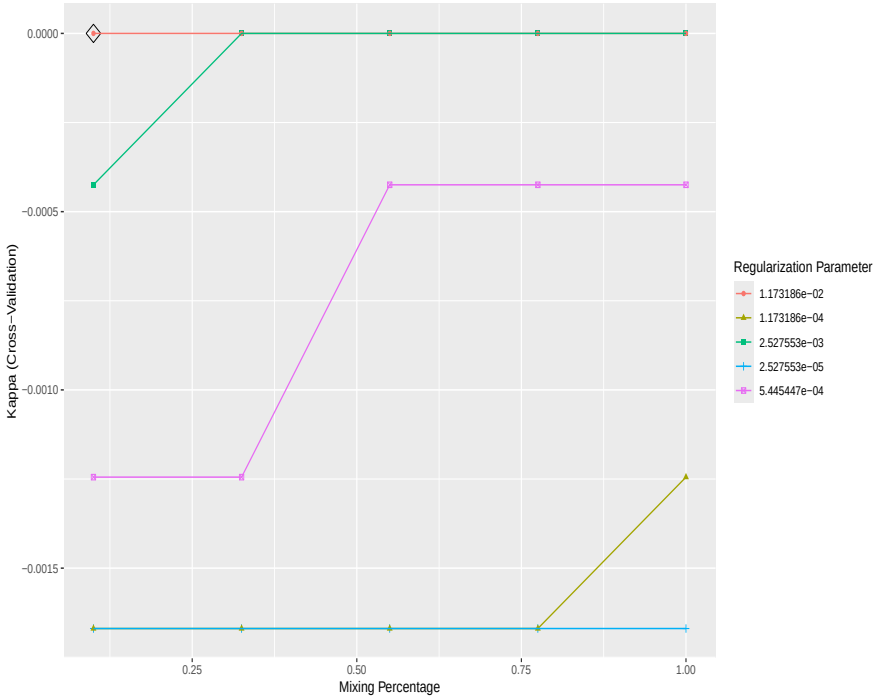



Figura 3.1: Gráfico de selección de hiperparámetros para el método glmnet.

### 3.2.2. Análisis del modelo

Tras obtener el valor de los hiperparámetros, podemos representar la evolución de los coeficientes asociados a las variables, en función del parámetro de penalización $\lambda$ que consideremos. Lo mostramos en la Figura 3.2.

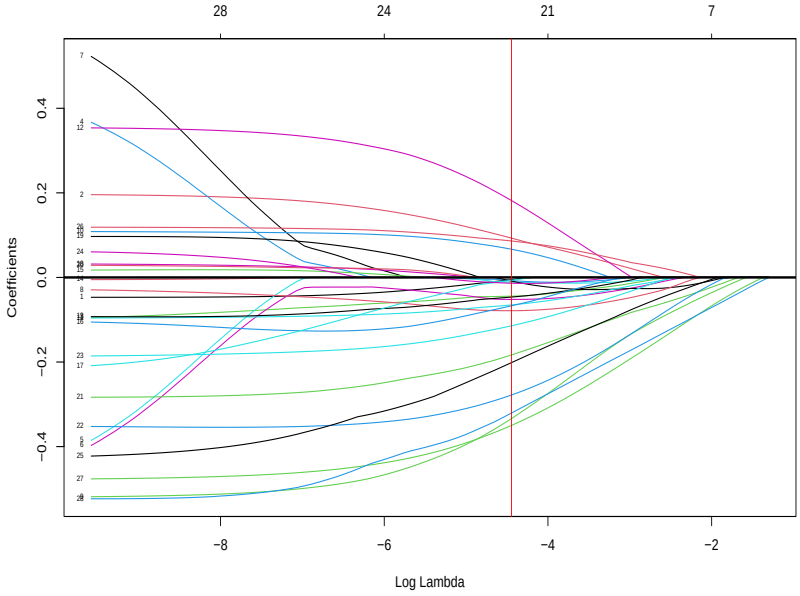



Figura 3.2: Gráfico de evolución de los coeficientes asociados al modelo en función del parámetro de penalización $\lambda$.

La propia Figura 3.2, muestra una línea vertical roja con el valor del logaritmo de $\lambda$, el cual ha sido obtenido mediante validación cruzada. Debido al elevado número de coeficientes, no se aprecia demasiado bien cuáles se hacen 0 para ese $\lambda$ de manera gráfica. No obstante, podemos observar que las variables 27 y 28 correspondientes a `VAR_OP_DISC_WOE_1` y `VAR_OP_DISC_WOE_2` son de las últimas en hacerse 0 por lo que les podemos atribuir una mayor importancia. A continuación, mostramos los coeficientes obtenidos para el modelo previamente ajustado.

```
coef(caret.glmnet$finalModel,s=caret.glmnet$bestTune$lambda,
                             alpha=caret.glmnet$bestTune$alpha)
```

```
                          s1
(Intercept)       -3.177025706
VAR_OP_CONT_1     -0.005118027
VAR_OP_CONT_2      0.093009741
VAR_OP_CONT_4     -0.044288827
VAR_OP_CONT_5     -0.007459992
VAR_OP_CONT_6     -0.011182639
VAR_OP_CONT_7     -0.051947317
VAR_OP_CONT_8     -0.008561011
VAR_OP_CONT_9     -0.078645663
VAR_CLI_CONT_2    -0.333753269
```

```
VAR_CLI_CONT_4       0.066700631
VAR_CLI_CONT_5      -0.064990450
VAR_CLI_CONT_8       0.181902928
VAR_CLI_CONT_10     -0.046411604
VAR_CLI_CONT_12      .
VAR_CLI_CONT_14      .
VAR_CLI_CONT_15     -0.067099265
VAR_CLI_CONT_16     -0.006656972
VAR_CLI_CONT_17      .
VAR_CLI_CONT_18      .
VAR_CLI_CONT_19      .
VAR_CLI_DISC_WOE_1 -0.183238801
VAR_CLI_DISC_WOE_3 -0.278004686
VAR_CLI_DISC_WOE_4 -0.114349076
VAR_CLI_DISC_WOE_5 -0.013625147
VAR_CLI_DISC_WOE_6 -0.201474431
VAR_CLI_DISC_WOE_7  0.086216659
VAR_OP_DISC_WOE_1  -0.349919304
VAR_OP_DISC_WOE_2  -0.320651706
```

De esta forma vemos más claramente que las variables que se hacen 0 (debido a la componente *LASSO* del modelo) son `VAR_CLI_CONT_12`, `VAR_CLI_CONT_14`, `VAR_CLI_CONT_17`, `VAR_CLI_CONT_18` y `VAR_CLI_CONT_19`. Asimismo, la componente *ridge* parece que también tiene su impacto ya que podemos ver coeficientes muy próximos a 0 como pueden ser `VAR_OP_CONT_1`, `VAR_CLI_CONT_16` o `VAR_OP_CONT_5`.

Por otro lado, contamos con 4 variables que tienen un efecto positivo, es decir, que a medida que aumentan su valor, la probabilidad de *default* se verá incrementada. Frente a estas 4, tendremos otras 19 con un efecto negativo, para las que mayores valores de la variable supondrán una menor tasa de mora.

Por tanto, un cliente que quiera que le concedan un préstamo debería intentar maximizar las variables que tienen un efecto negativo. Por ejemplo, en un caso hipotético, si el modelo considerase los ingresos como una variable con efecto negativo, una persona con unas mayores ganancias reduciría su estimación de probabilidad de impago, y por tanto tendría más opciones de conseguir el préstamo.

Por ilustrar un paradigma de una variable con efecto positivo, este podría ser el caso del tipo de interés: para un préstamo con intereses elevados puede resultar más difícil pagar las cuotas correspondientes, lo que elevaría la probabilidad de impago.

Además, podemos determinar las variables más influyentes basándonos en las que presenten un mayor valor absoluto de los coeficientes asociados. Las representamos en la Figura 3.3 mostradas por orden de importancia.

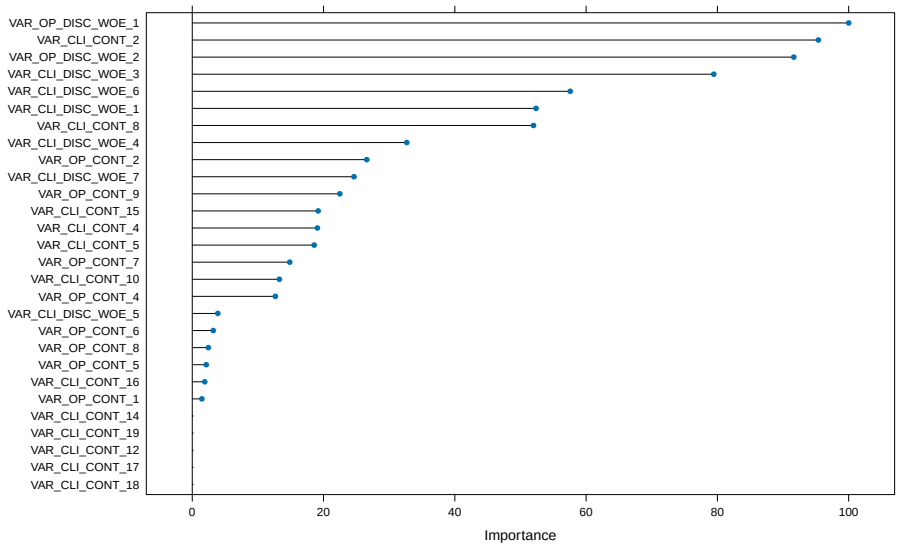



Figura 3.3: Variables del modelo *elastic net* ordenadas por importancia.

### 3.2.3. Evaluación del modelo

En este apartado, analizaremos el rendimiento del modelo en base a distintas métricas relacionadas con las predicciones realizadas sobre el conjunto de validación, formado por XXXX observaciones.

Comenzaremos representando un histograma en la Figura 3.4, donde podemos observar como se distribuyen las probabilidades de mora en función de si la observación de la muestra de test pertenecía a la categoría de operaciones morosas (1) o sanas (0).



Figura 3.4: Histograma de las probabilidades de mora estimadas en función de la categoría observada.

Como en el caso de la regresión logística, las operaciones sanas sí que obtienen valores de probabilidad de mora pequeños; sin embargo, para las operaciones marcadas como morosas, la probabilidad de mora apenas supera el 0.3, quedando por debajo de lo que obteníamos en la logística.

Al igual que para el modelo anterior, podemos apreciar que tendremos un rendimiento mejor a la hora de predecir operaciones sanas. En la Tabla 3.1, representamos la matriz de confusión asociada a considerar morosas aquellas operaciones para las que el modelo arroja una probabilidad superior a 0.1.



Tabla 3.1: Matriz de confusión para el modelo *elastic net* ajustado.

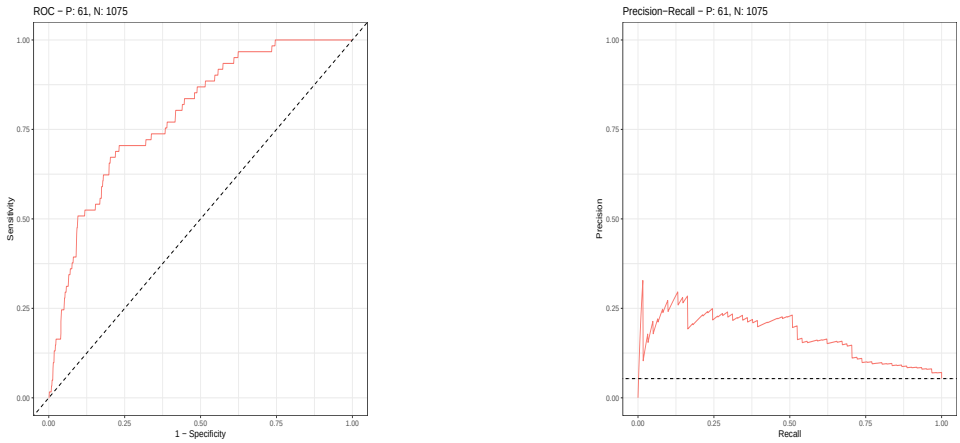
Podemos ver que, con respecto a la logística, el modelo *elastic net* clasifica de la misma forma a las observaciones marcadas como morosas para el punto de corte considerado. Sin embargo, clasifica mejor a las marcadas como sanas, ya que los falsos positivos son menores.

En consecuencia, en la Tabla 3.2 obtenemos un *TPR* mayor, por lo que también se incrementan el *Accuracy*, el *Balanced Accuracy* y $\kappa$.

| TPR | TNR | Accuracy | Balanced Accuracy | $\kappa$ |
|---|---|---|---|---|
| 0.50820 | 0.89953 | 0.8785 | 0.70387 | 0.2543 |

Tabla 3.2: Valores de las métricas para el modelo *elastic net* ajustado.

Acto seguido, representamos las curvas *ROC* y *PR* en la Figura 3.5, siguiendo el mismo procedimiento que en el caso anterior y siendo estas bastante similares a las vistas para la regresión logística.




(a) Curva *ROC*. (b) Curva *PR*.

Figura 3.5: Curvas *ROC* y *PR* para el modelo de *elastic net* ajustado.

Por último, calculamos el área bajo las dos curvas representadas. En la Tabla 3.3, podemos observar que para esta partición del conjunto de entrenamiento y test (la misma que en el anterior modelo) ambas métricas están sutilmente por encima de las que obteníamos en la regresión logística.

| **Curva** | ROC | PR |
|---|---|---|
| **AUC** | 0.7904 | 0.1718 |

Tabla 3.3: Áreas bajo las curvas *ROC* y *PR* del modelo *elastic net* ajustado.

En este capítulo también realizaremos la simulación de 100 conjuntos de entrenamiento para poder comparar los resultados entre modelos de una manera más fehaciente. A continuación, mostramos el código necesario para realizar la simulación.

```
df=datos_filtrados_2
B=100
auc.glmnet=numeric(B)
pr.glmnet=numeric(B)
set.seed(2311)

for (i in 1:B){
        nobs <- nrow(df)
        itrain <- sample(nobs, 0.8 * nobs)
        train <- df[itrain, ]
        test <- df[-itrain, ]
```

```
        X=as.data.frame(train[,-1])
        y=train[,1]$MORA_12M
        caret.glmnet <- train(x = X, y = y, method = "glmnet",
        preProc = c("zv", "center", "scale"),
        family = "binomial",
        metric="Kappa",
        trControl = trainControl(method = "cv", number = 10,
        selectionFunction = "oneSE"),
        tuneLength=5)
        obs <- test$MORA_12M
        p.est <- predict(caret.glmnet, type = "prob", newdata = test)
        sscurves=evalmod(scores=p.est[,2],labels=obs)
        aucs <- auc(sscurves)
        auc.glmnet[i]=aucs$aucs[1]
        pr.glmnet[i]=aucs$aucs[2]
}

auc.glmnet
summary(auc.glmnet)
auc.glmnet.medio=mean(auc.glmnet)
auc.glmnet.medio

pr.glmnet
summary(pr.glmnet)
pr.glmnet.medio=mean(pr.glmnet)
pr.glmnet.medio
```

En la Tabla 3.4, resumimos el valor de el área bajo las curvas $ROC$ y $PR$ recogiendo el mínimo, el máximo y la media de esas métricas para los 100 modelos simulados. Como podemos ver, el valor medio del $AUC\ ROC$ y del $AUC\ PR$, aunque se parecen bastante a los que obteníamos con la regresión logística, están ligeramente por debajo.

| | Mínimo | Máximo | Media |
|---|---|---|---|
| **AUC ROC** | 0.6909 | 0.8251 | 0.7608 |
| **AUC PR** | 0.09687 | 0.26863 | 0.16262 |

Tabla 3.4: Mínimo, máximo y media de los $AUC\ ROC$ y $AUC\ PR$ relativos a los 100 modelos de *elastic net* simulados.

# Capítulo 4

# Árboles de decisión

Para elaborar la introducción metodológica de este capítulo relativa a los árboles de regresión y clasificación CART, nos hemos basado en los textos de Berk et al. (2008), James et al. (2013) y Hastie et al. (2009). Posteriormente, como ya es habitual, aplicaremos los árboles de clasificación a nuestra base de datos para estudiar su desempeño.

## 4.1. Marco metodológico

Los árboles de decisión son un método sencillo e interpretable que ha sido ampliamente utilizado para lograr predicciones en problemas tanto de regresión como de clasificación. Una metodología extendida en lo que a árboles de decisión se refiere es la denominada CART (*Classification and Regression Trees*), empezaremos viendo el caso de los problemas de regresión y posteriormente lo adaptaremos a clasificación.

### 4.1.1. Árboles de regresión CART

La idea principal de los árboles de decisión consiste en dividir el llamado espacio predictor, es decir, el conjunto de posibles valores correspondientes a las variables explicativas, en $J$ regiones no solapadas que denotaremos por $R_j$, con $j = 1, ..., J$ (consideraremos que estas regiones son rectángulos multidimensionales por sencillez).

Una vez delimitadas esas regiones, para cada nueva observación perteneciente a $R_j$, tomaremos como predicción común la media de la variable respuesta de las observaciones de entrenamiento dentro de ese $R_j$.

Después de describir el proceso de forma general, necesitamos establecer un criterio para dividir el espacio predictor en las regiones $R_j$ mencionadas. Pues bien, idealmente nos gustaría obtener divisiones

de tal forma que seamos capaces de minimizar la suma residual de cuadrados ($RSS$) definida como

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

donde hemos denotado por $\hat{y}_{R_j}$ a la media de la variable respuesta de las observaciones de entrenamiento contenidas en la región $R_j$.

No obstante, este enfoque es infactible debido al coste computacional; es por ello que se opta por un algoritmo *greedy* por etapas, donde en cada etapa se selecciona una variable explicativa y un punto de corte asociado con el objetivo de dividir el espacio predictor en dos semiplanos, preocupándose únicamente de que la división en esa etapa sea óptima, sin considerar divisiones que pudiesen ser mejores teniendo en cuenta pasos posteriores.

La metodología consistiría pues, en partir de la muestra de entrenamiento total y encontrar una variable $X_j$ con su correspondiente punto de corte $s$ con la que dividir el espacio predictor en

$$R_1(j, s) = \{X | X_j \leq s\} \text{ y } R_2(j, s) = \{X | X_j > s\},$$

seleccionando tanto la variable como el punto de corte de tal forma que se cumpla el siguiente criterio

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

que ya es más asumible de evaluar, puesto que bastaría con recorrer el conjunto de valores que pueden tomar las distintas variables para seleccionar el punto de corte óptimo, y analizando todas las variables se llegaría al par $(j, s)$ buscado. Una vez realizada esta partición se repetiría el proceso de forma iterativa para subdividir las regiones resultantes.

Hay que tener en cuenta que el tamaño del árbol es un hiperparámetro que determinará la complejidad del modelo y que deberemos regular con el fin de controlar que no se produzca un sobreajuste como resultado de considerar un árbol demasiado grande. Así pues, el siguiente asunto a tratar será en qué momento debemos detener el algoritmo.

Una posible alternativa sería considerar como criterio de parada un umbral de mejora del $RSS$ por debajo del cual no haríamos crecer más el árbol. No obstante, podría darse el caso de que una mejora poco significativa en una etapa fuese necesaria para obtener un mejor modelo a partir de etapas posteriores.

Por tanto, será preferible detener el algoritmo únicamente cuando se alcance un mínimo de observaciones por nodo, y una vez obtenido ese árbol, podarlo para obtener como resultado un subárbol que provendrá de colapsar una cierta cantidad de nodos internos del árbol original.

Idealmente, estaríamos interesados en que este subárbol tenga asociado un error lo más bajo posible sobre la muestra de validación, pero no es viable evaluar todos los subárboles por ser demasiado costoso. La estrategia a seguir será entonces fijar un parámetro de complejidad $\alpha$ con el que poder seleccionar el tamaño del árbol y obtener así un subárbol óptimo.

Este subárbol será el resultado de minimizar la siguiente expresión de la *RSS* penalizada

$$RSS = \sum_{j=1}^{T} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2 + \alpha T, \tag{4.1}$$

donde $T$ denota el tamaño del subárbol, es decir, su número de nodos terminales. Este hiperparámetro $\alpha$ se puede determinar empleando técnicas de validación cruzada como las ya expuestas en el capítulo anterior.

Cabe observar que para cada valor de $\alpha$ únicamente existirá un árbol que minimice la RSS penalizada. Para $\alpha = 0$, el árbol que minimiza la Ecuación (4.1) es el árbol completo y a medida que tomemos valores mayores, se penalizarán los subárboles con muchos nodos terminales, de tal manera que estos se irán colapsando de forma consecutiva hasta obtener una sucesión finita de subárboles en la cual se encontrará el subárbol óptimo.

### 4.1.2. Árboles de clasificación CART

Atendiendo a nuestro propósito, nos resultará más interesante la aplicación de los árboles de clasificación en lugar de los de regresión. En este caso, a diferencia de lo que ocurría en los árboles de regresión, donde usábamos la media de las observaciones para predecir; identificaremos cuál es la categoría más frecuente para las observaciones de entrenamiento en cada nodo terminal, con el objetivo de poder clasificar a las nuevas observaciones en esa categoría modal (que variará dependiendo de la región donde se encuentre la nueva observación).

Otra distinción relevante es que en este caso no podemos tomar el RSS como criterio de parada para realizar las distintas divisiones del árbol, por lo que necesitaremos tener en cuenta la proporción de observaciones de entrenamiento de cada categoría, que denotaremos por $\hat{p}_k$ siendo $k = 1, ..., K$, las distintas categorías. En base a esta proporción se presentan las siguientes métricas:

- Tasa de error de clasificación: se define como

  $$1 - \max_k(\hat{p}_k)$$

  y denota el ratio de observaciones que no pertenecen a la categoría mayoritaria.

- Índice de Gini: su expresión se corresponde con

  $$\sum_{k=1}^{K} \hat{p}_k(1 - \hat{p}_k),$$

  en este caso se tienen en cuenta las proporciones de todas las clases, y en base a su propia definición, obtendremos valores pequeños de esta métrica cuando la mayoría de observaciones pertenezcan a una única clase, lo que indicaría que ese nodo es *puro*.

- Entropía: viene dada por

  $$-\sum_{k=1}^{K} \hat{p}_k \log(\hat{p}_k),$$

y en esta ocasión, también obtendremos valores pequeños si las observaciones se concentran en una única clase.

A la hora de evaluar las divisiones en el árbol, suele emplearse el índice de Gini o la entropía por ser más sensibles a la pureza de los nodos; mientras que para realizar la poda del árbol es más aconsejable emplear la tasa de error de clasificación si lo que queremos es maximizar la precisión de la predicción del árbol final.

Como veremos a continuación, una gran ventaja de los árboles CART es su interpretabilidad, ya que pueden ser representados gráficamente y su clasificación consiste en ir aplicando reglas de decisión. Sin embargo, estos métodos también tienen sus desventajas, debido a que normalmente su poder de predicción es bajo en comparación con otras alternativas y además, no son especialmente robustos, puesto que pequeños cambios en los datos pueden dar lugar a árboles completamente distintos.

En los próximos capítulos veremos cómo los árboles de decisión pueden servir de base para otros métodos más potentes como el *random forest* o el *boosting*.

## 4.2. Aplicación práctica

Como venimos haciendo a lo largo de este trabajo, comenzaremos ajustando el modelo de los árboles CART sobre nuestro conjunto de entrenamiento para después de analizar los resultados obtenidos, medir su desempeño con la muestra de validación.

### 4.2.1. Ajuste del modelo

Como ya mencionamos anteriormente, emplearemos la librería `caret` para realizar el ajuste. En este caso también hemos empleado un enfoque *K-fold Cross Validation* con $k = 10$ para seleccionar un parámetro de complejidad $\alpha$ de entre una rejilla de 25 valores, de tal forma que optimice la $\kappa$ de Cohen mediante la regla de un error estándar de Breiman.

```
caret.rpart <- train(MORA_12M ~ ., method = "rpart", data = train,
metric="Kappa",
trControl = trainControl(method = "cv", number = 10, selectionFunction = "oneSE"),
tuneLength = 25)
caret.rpart
```

```
CART

4540 samples
28 predictor
```

```
2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 4086, 4086, 4086, 4087, 4086, 4085, ...
Resampling results across tuning parameters:

cp            Accuracy   Kappa
0.0000000000  0.9354615  0.08336939
0.0002572016  0.9354615  0.08336939
0.0005144033  0.9354615  0.08336939
0.0007716049  0.9356818  0.08445677
0.0010288066  0.9356818  0.08445677
0.0012860082  0.9356818  0.08445677
0.0015432099  0.9374439  0.08407426
0.0018004115  0.9374439  0.08407426
0.0020576132  0.9376642  0.08437775
0.0023148148  0.9385457  0.08057582
0.0025720165  0.9385457  0.08057582
0.0028292181  0.9387660  0.08108318
0.0030864198  0.9396475  0.08255862
0.0033436214  0.9396475  0.08255862
0.0036008230  0.9407484  0.08520237
0.0038580247  0.9407484  0.08520237
0.0041152263  0.9414091  0.07497599
0.0043724280  0.9411889  0.04674231
0.0046296296  0.9429510  0.03131584
0.0048868313  0.9429510  0.03131584
0.0051440329  0.9429510  0.03131584
0.0054012346  0.9425105  0.01109220
0.0056584362  0.9425105  0.01109220
0.0059156379  0.9427307  0.01141538
0.0061728395  0.9438321  0.01337431

Kappa was used to select the optimal model using  the one SE rule.
The final value used for the model was cp = 0.004115226.
```

En la Figura 4.1 comprobamos que el valor del parámetro de complejidad que maximiza $\kappa$ según el criterio de un error estándar de Breiman es $\alpha = 0.004115226$, tomando este valor de $\alpha$ el árbol resultante es el que recogemos en la Figura 4.2 y que hemos representado empleando la librería `rpart.plot`.

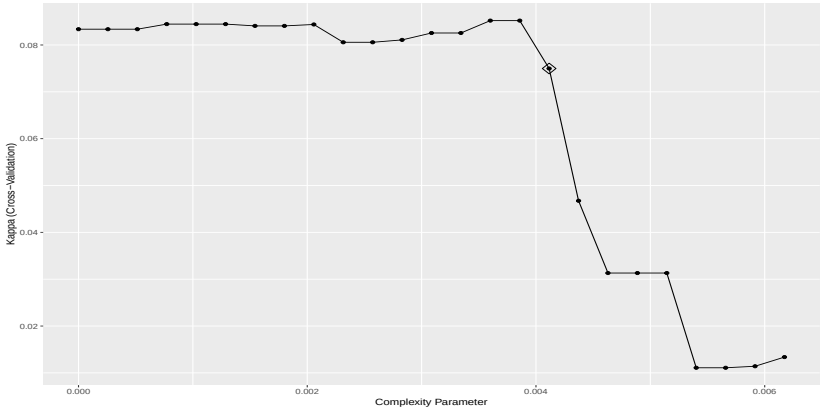



Figura 4.1: Gráfico de selección de hiperparámetros para el modelo basado en un árbol CART.

### 4.2.2. Análisis del modelo

La información que tendremos en cada nodo del árbol consiste en: la clasificación (0 o 1), la probabilidad de sano o de mora respectivamente y el porcentaje de la muestra de entrenamiento que está contenida en cada nodo.

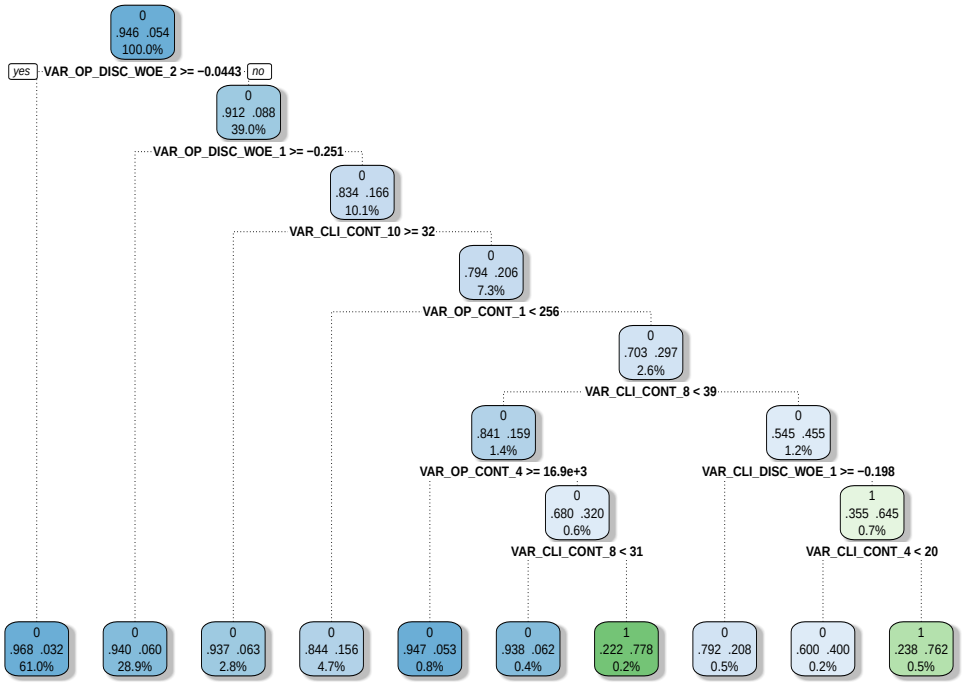



Figura 4.2: Árbol obtenido para el parámetro de complejidad seleccionado ($\alpha = 0.004115226$).

Podemos observar en los nodos del árbol de la Figura 4.2, que el modelo CART clasifica a un 97.3 % de las observaciones como operaciones sanas, mientras que tan solo un 0.7 % se clasifica como morosa. A continuación, interpretaremos algunos de los nodos terminales, teniendo en cuenta que tomar la rama de la izquierda implica que se cumple la condición expuesta y seguir el camino de la rama de la derecha significa que no se cumple.

En el caso de que la variable VAR_OP_DISC_WOE_2 $\geq -0.0443$, el árbol clasificará a esas observaciones como sanas. Esto ocurre debido a que en ese nodo se agrupan un 61 % de la muestra de entrenamiento, es decir, 2770 operaciones, de las cuales únicamente 88 están etiquetadas como morosas, por lo que el modelo asignará una probabilidad de mora del 3.2 % para operaciones que caigan en ese nodo.

Analicemos ahora el séptimo nodo que es uno de los que clasifica las operaciones como morosas. Para acabar en este nodo una operación tiene que cumplir los siguientes criterios:

- VAR_OP_DISC_WOE_2 $< -0.0443$.
- VAR_OP_DISC_WOE_1 $< -0.251$.
- VAR_CLI_CONT_10 $< 32$.
- VAR_OP_CONT_1 $\geq 256$.
- VAR_CLI_CONT_8 $\in [31, 39)$.
- VAR_OP_CONT_4 $< 16900$.

En nuestro conjunto de entrenamiento tenemos un total de 9 operaciones que cumplen las anteriores características de las cuales 7 tienen marca de *default*, por ello a las operaciones de ese nodo se les asignará una tasa de mora del 77.8 %.

El único otro nodo que clasifica operaciones como morosas es el último, y para pertenecer a él se tienen que verificar los siguientes requisitos:

- VAR_OP_DISC_WOE_2 $< -0.0443$.
- VAR_OP_DISC_WOE_1 $< -0.251$.
- VAR_CLI_CONT_10 $< 32$.
- VAR_OP_CONT_1 $\geq 256$.
- VAR_CLI_CONT_8 $\geq 39$.
- VAR_CLI_DISC_WOE_1 $< -0.198$.
- VAR_CLI_CONT_4 $\geq 20$.

Para este nodo tendremos un 0.5 % de las observaciones totales, es decir, 21 operaciones de las cuales 5 se encuentran en *default* por lo que el modelo asignará una probabilidad de mora del 76.2 %.

Como hemos podido comprobar, el modelo es bastante sencillo de interpretar si vamos siguiendo las reglas de decisión que nos marca. Para poder consultar exactamente cuales son las reglas empleadas podemos emplear el siguiente comando.

```
caret.rpart$finalModel
```

```
n= 4540

node), split, n, loss, yval, (yprob)
* denotes terminal node

1) root 4540 243 0 (0.94647577 0.05352423)
2) VAR_OP_DISC_WOE_2>=-0.04430989 2770  88 0 (0.96823105 0.03176895) *
3) VAR_OP_DISC_WOE_2< -0.04430989 1770 155 0 (0.91242938 0.08757062)
6) VAR_OP_DISC_WOE_1>=-0.2510259 1313  79 0 (0.93983244 0.06016756) *
7) VAR_OP_DISC_WOE_1< -0.2510259 457  76 0 (0.83369803 0.16630197)
14) VAR_CLI_CONT_10>=32.01022 127   8 0 (0.93700787 0.06299213) *
15) VAR_CLI_CONT_10< 32.01022 330  68 0 (0.79393939 0.20606061)
30) VAR_OP_CONT_1< 256.315 212  33 0 (0.84433962 0.15566038) *
31) VAR_OP_CONT_1>=256.315 118  35 0 (0.70338983 0.29661017)
62) VAR_CLI_CONT_8< 38.5 63  10 0 (0.84126984 0.15873016)
124) VAR_OP_CONT_4>=16850 38   2 0 (0.94736842 0.05263158) *
125) VAR_OP_CONT_4< 16850 25   8 0 (0.68000000 0.32000000)
250) VAR_CLI_CONT_8< 30.5 16   1 0 (0.93750000 0.06250000) *
251) VAR_CLI_CONT_8>=30.5 9   2 1 (0.22222222 0.77777778) *
63) VAR_CLI_CONT_8>=38.5 55  25 0 (0.54545455 0.45454545)
126) VAR_CLI_DISC_WOE_1>=-0.1979281 24   5 0 (0.79166667 0.20833333) *
127) VAR_CLI_DISC_WOE_1< -0.1979281 31  11 1 (0.35483871 0.64516129)
254) VAR_CLI_CONT_4< 19.5 10   4 0 (0.60000000 0.40000000) *
255) VAR_CLI_CONT_4>=19.5 21   5 1 (0.23809524 0.76190476) *
```

Al igual que hicimos en el modelo *elastic net*, podemos obtener la importancia de las variables que intervienen en el árbol ajustado. En este caso, la función `varImp` de `caret` calcula la reducción en la función de pérdida (por defecto la función empleada utiliza el índice de Gini para realizar las divisiones) que consigue cada variable al realizar una división y se devuelve la suma. Además, dado que puede haber variables candidatas que son importantes pero que finalmente no son seleccionadas para ramificar un nodo, también se tienen en consideración las principales variables competidoras en cada división, es por ello que en la Figura 4.3 aparecen más variables de las que intervienen en el árbol.

Cabe destacar que las variables `VAR_CLI_CON_2` y `VAR_OP_DISC_WOE_1` están en el top 5 de variables más importantes tanto en este método basado en árboles, como en el método *elastic net*.

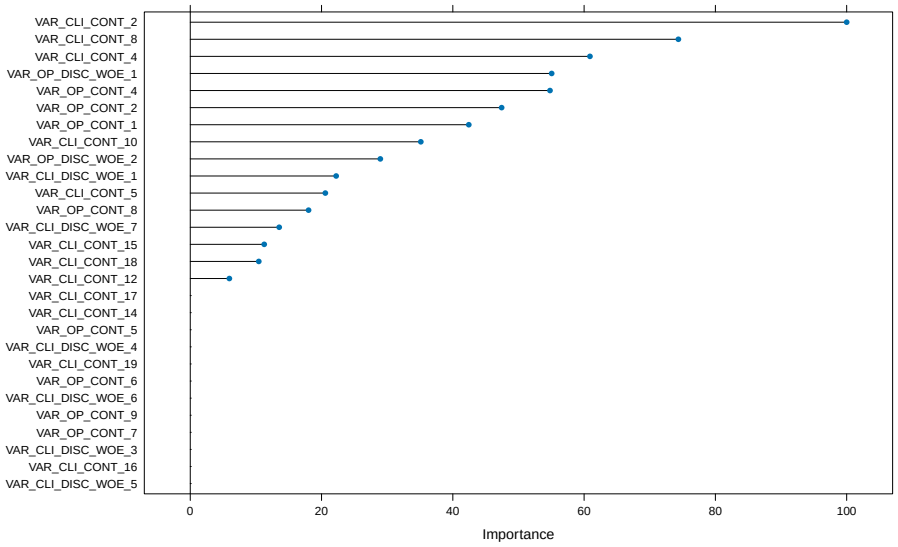



Figura 4.3: Variables del modelo CART ordenadas por importancia.

### 4.2.3. Evaluación del modelo

Como es habitual, empezaremos este apartado de evaluación del modelo con la Figura 4.4, donde ilustramos un histograma en el que podemos apreciar cómo se distribuyen las probabilidades de mora en función de si la observación de la muestra de test pertenece a la categoría de operaciones morosas (1) o sanas (0).



Figura 4.4: Histograma de las probabilidades de mora estimadas en función de la categoría observada.

Podemos ver que las probabilidades de mora representadas en el histograma son precisamente las que obtuvimos en los nodos terminales, en este caso, separadas en función de la marca de mora. Para este modelo, la probabilidad de mora asociada a las operaciones sanas está altamente concentrada en valores cercanos a 0 y ocurre algo similar para las operaciones morosas, aunque es cierto que obtenemos más observaciones con probabilidades elevadas que en el caso de sanos.

En la Tabla 4.1, representamos la matriz de confusión obtenida como resultado de considerar morosas aquellas operaciones para las que el modelo arroja una probabilidad superior a 0.1.



Tabla 4.1: Matriz de confusión para el modelo basado en árboles CART.

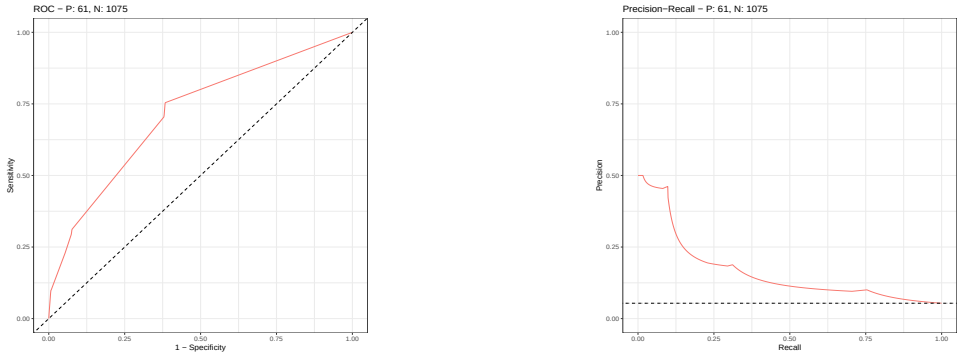
Una primer análisis que podemos realizar es que el número de verdaderos positivos, con respecto a los otros métodos, decae prácticamente a la mitad, y algo parecido ocurre con el número de falsos positivos. Esto quiere decir que el modelo está prediciendo un número más elevado de observaciones como sanas, lo que elevará el *TNR* y disminuirá el *TPR*; lo vemos en la Tabla 4.2.

| TPR | TNR | Accuracy | Balanced Accuracy | $\kappa$ |
|---|---|---|---|---|
| 0.22951 | 0.94605 | 0.9076 | 0.58778 | 0.1618 |

Tabla 4.2: Valores de las métricas para el modelo basado en árboles CART.

Este aumento en la especificidad (*TNR*), desemboca en una *Accuracy* más elevada; sin embargo, tanto la *Balanced Accuracy* como $\kappa$ disminuyen al tener en cuenta que las clases están desbalanceadas.

A continuación, representamos las curvas *ROC* y *PR* en la Figura 4.5 para obtener más información sobre el desempeño del modelo.




(a) Curva *ROC*. (b) Curva *PR*.

Figura 4.5: Curvas *ROC* y *PR* para el modelo basado en árboles CART.

Echando un vistazo a la representación de la curva *ROC*, podemos intuir que el área bajo la curva será algo menor en esta ocasión, ya que parece que la curva se aleja bastante del ideal de situarse cerca de la esquina superior izquierda. Comprobamos esta intuición mediante la Tabla 4.3, donde recogemos los valores de los *AUCs* correspondientes y en la que podemos apreciar que el área bajo la curva *Precision-Recall* también disminuye, aunque en menor medida.

| **Curva** | ROC | PR |
|---|---|---|
| **AUC** | 0.7117 | 0.1642 |

Tabla 4.3: Áreas bajo las curvas *ROC* y *PR* del modelo basado en árboles CART.

Una vez más, incluimos el código necesario para realizar el proceso de simulación con el que obtendremos un resumen de las métricas obtenidas para cada uno de los modelos basados en árboles CART ajustados y en base a las cuales podremos extraer conclusiones.

```
df=datos_filtrados_2
B=100
auc.tree=numeric(B)
pr.tree=numeric(B)
ref.tree=numeric(B)
set.seed(2311)

for (i in 1:B){
        print(i)
        nobs <- nrow(df)
        itrain <- sample(nobs, 0.8 * nobs)
```

```
        train <- df[itrain, ]
        test <- df[-itrain, ]
        caret.rpart <- train(MORA_12M ~ ., method = "rpart", data = train,
        tuneLength = 25,
        metric="Kappa",
        trControl = trainControl(method = "cv", number = 10,
        selectionFunction = "oneSE"))

        obs <- test$MORA_12M
        p.est <- predict(caret.rpart, type = "prob", newdata = test)
        sscurves=evalmod(scores=p.est[,2],labels=obs)
        aucs <- auc(sscurves)
        auc.tree[i]=aucs$aucs[1]
        print(auc.tree[i])
        pr.tree[i]=aucs$aucs[2]
        print(pr.tree[i])
}

auc.tree
summary(auc.tree)
auc.tree.medio=mean(auc.tree)
auc.tree.medio

pr.tree
summary(pr.tree)
pr.tree.medio=mean(pr.tree)
pr.tree.medio
```

En la Tabla 4.4 resumimos el valor de el área bajo las curvas $ROC$ y $PR$ recogiendo el mínimo, el máximo y la media de esas métricas para los 100 modelos simulados.

| | Mínimo | Máximo | Media |
|---|---|---|---|
| **AUC ROC** | 0.5000 | 0.7762 | 0.6472 |
| **AUC PR** | 0.0423 | 0.1224 | 0.1249 |

Tabla 4.4: Mínimo, máximo y media de los $AUC\ ROC$ y $AUC\ PR$ relativos a los 100 modelos de árboles CART simulados.

Como podemos apreciar, tanto la media del *AUC ROC* como del *AUC PR* están sustancialmente por debajo de los otros modelos considerados. Además, podemos ver que para algún conjunto de datos el árbol resultante ha obtenido un *AUC ROC* de 0.5, es decir, la clasificación realizada por ese árbol es comparable a la de un clasificador aleatorio. Señalar que probablemente ese árbol ajustado no contenía ninguna división, por tanto constaba de un único nodo que clasificaba a todas las operaciones como sanas.

# Capítulo 5

# Random Forest

En este capítulo, siguiendo nuestra estructura previa, comenzaremos con un marco teórico elaborado en base a las referencias de James et al. (2013), Hastie et al. (2009) y Kuhn et al. (2013) para poder entender los fundamentos del denominado *random forest* o bosques aleatorios. Posteriormente, implementaremos los métodos vistos en la Sección 1.4 para poder comprender mejor el comportamiento del modelo, y por último evaluaremos el desempeño del *random forest* sobre nuestro caso de aplicación.

## 5.1. Marco metodológico

Para adquirir unas nociones precisas del funcionamiento de *random forest*, es crucial dedicar tiempo a entender el método subyacente en el que se fundamenta: el *bagging*. Este será el eje principal que trataremos en la siguiente sección.

### 5.1.1. Bagging

En el capítulo anterior, señalamos que una de las desventajas de los árboles de decisión residía en su sensibilidad a pequeñas variaciones en los datos, lo que podía generar árboles notablemente diferentes entre sí. Este comportamiento produce que estos métodos cuenten con una elevada varianza.

Una posible forma de subsanar este problema es recurrir al *Bootstrap aggregation*, también conocido como *bagging*, que fue propuesto por Breiman (1996), y resulta especialmente útil en el ámbito de los árboles de decisión.

La idea sobre la que se fundamenta el método consiste en el hecho de que promediar un conjunto de observaciones logra hacer disminuir su varianza. Así pues, podríamos construir modelos para muchos conjuntos de entrenamiento, a partir de los cuales obtendríamos predicciones asociadas a una observación determinada, que luego se promediarían para dar lugar a una predicción final. En otras palabras,

dados $b = 1, ..., B$ conjuntos de entrenamiento la predicción del *bagging* se expresaría de la siguiente forma

$$\hat{f}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^b(x),$$

donde $\hat{f}^b(x)$ es la predicción generada por el modelo para el conjunto de entrenamiento $b$ y la observación $x$.

No obstante, en la práctica no tenemos a disposición distintas muestras de entrenamiento, por lo que una técnica habitual es recurrir al bootstrap para poder generarlas. En el caso concreto de utilizar árboles de decisión como modelo de predicción (el *bagging* permite emplear otras alternativas), el procedimiento consistiría en construir árboles de máxima profundidad sin podar; estos árboles tendrán una varianza elevada y poco sesgo, de forma que al promediar sus predicciones, contrarrestaremos esa desventaja reduciendo también su varianza.

El esquema a seguir que hemos descrito se adapta perfectamente en el caso de árboles de predicción; no obstante, para el caso de querer realizar una clasificación, tendríamos que efectuar una pequeña modificación: dada una observación, tomaríamos la categoría que arroja cada uno de los $B$ árboles construidos y clasificaríamos a la observación de interés en la clase que más se repita.

Otro punto a favor del *bagging* es que, debido al propio comportamiento del bootstrap, aproximadamente 1/3 de las observaciones del conjunto de entrenamiento no se utilizan en la remuestra bootstrap que se emplea para ajustar cada árbol, estas son las denominadas observaciones *out-of-bag* (OOB). Esto es interesante debido a que podemos realizar predicciones para estas observaciones utilizando los árboles donde las observaciones OOB no se han tenido en cuenta para el ajuste, y generar así una única predicción para cada observación. Si repetimos este mismo proceso para cada una de las observaciones de nuestro conjunto de datos, podremos obtener medidas de error globales de forma directa, sin necesidad de recurrir a una muestra de test.

Con respecto a cuál es el número $B$ de árboles deberíamos tomar, el enfoque habitual es estudiar como se comporta la convergencia de manera gráfica. Si obtenemos dicha convergencia para un cierto $B$, el nivel de error no experimentará unas variaciones notables para un número de árboles superior. En consecuencia, incrementar mucho el número de árboles no resulta en una mejora significativa de las predicciones, pero tampoco supondrá un riesgo de sobreajuste; en lo que sí debemos tener cuidado es en el coste computacional asociado.

Por otro lado, el *bagging* también tiene un inconveniente adicional con respecto a los árboles: a costa de mejorar las predicciones obtenidas, la interpretabilidad se reduce de manera drástica, siendo considerado por ello, un método de caja negra.

A continuación, veremos cómo el *random forest* incluye una modificación sobre el *bagging* que le permitirá disminuir la varianza en mayor medida.

### 5.1.2. Random Forest

Como ya hemos adelantado, el *random forest* es una mejora del *bagging* que se aplica a los árboles de decisión. La diferencia principal radica en qué el *random forest* busca reducir la correlación entre los diferentes árboles que usamos para predecir.

La cuestión a resolver es la siguiente: cuando un predictor adquiere gran importancia, es probable que se posicione en la parte superior del árbol, lo que conduce a la generación de árboles similares y por tanto, a una correlación entre ellos, que implicaría a su vez una correlación entre los predictores y las predicciones. Es bajo estas circunstancias, cuando el *bagging* se vuelve menos efectivo, pues promediar cantidades que están correladas no supone una reducción de la varianza tan importante como cuando no lo están.

Para evitar esta problemática, Breiman (2001) propone seleccionar, sobre el número total de variables, un subconjunto de variables candidatas para dividir el árbol en cada corte, es decir, seleccionar $m \leq p$ variables de forma aleatoria que serán las únicas que se evalúen a la hora de realizar una división. Cabe notar que para el caso de que $m = p$, el método de *random forest* coincidiría con el *bagging*.

Por tanto, el hiperparámetro asociado al *random forest* será $m$, es decir, el número de variables a seleccionar aleatoriamente en cada corte del árbol. Habitualmente se logran buenos resultados considerando $m = \sqrt{p}$ para problemas de clasificación y $m = p/3$ para regresión.

Otra ventaja del algoritmo de los bosques aleatorios sobre el *bagging* es que son computacionalmente más eficientes, ya que a pesar de necesitar un mayor número de árboles de cara a garantizar la convergencia, al tener que evaluar únicamente un subconjunto de variables en cada corte adquiere una mayor rapidez.

## 5.2. Aplicación práctica

### 5.2.1. Ajuste del modelo

Comenzamos ajustando el modelo mediante el método `rf`, siguiendo el procedimiento realizado en los capítulos anteriores; con la diferencia de que en esta ocasión fijaremos una malla para la selección del hiperparámetro. Como acabamos de ver en la sección anterior, para el caso de clasificación se recomienda tomar $m = \sqrt{p}$, por lo que vamos a evaluar el modelo para los valores $(\frac{\sqrt{p}}{2}, \sqrt{p}, \frac{3\sqrt{p}}{2}, 2\sqrt{p})$, que redondearemos para obtener un número entero (`floor`) .

```
p=dim(train)[2]-1
rf.caret <- train(MORA_12M ~ ., data = train, method = "rf",
metric="Kappa",
trControl = trainControl(method = "cv", number = 10, selectionFunction = "oneSE"),
```

```
tuneGrid=data.frame(mtry=floor(seq(0.5,2,0.5)*sqrt(p)))
rf.caret
```

```
Random Forest

4540 samples
28 predictor
2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 4086, 4086, 4086, 4087, 4086, 4085, ...
Resampling results across tuning parameters:

mtry  Accuracy   Kappa
2     0.9464762   0.0000000000
5     0.9462559  -0.0004247036
7     0.9462559   0.0065107252
10    0.9464762   0.0205620778

Kappa was used to select the optimal model using  the one SE rule.
The final value used for the model was mtry = 7.
```

Como podemos observar en la salida, el valor obtenido por validación cruzada empleando la regla de un error estándar de Breiman ha sido $m = 7$. Representamos los resultados para los distintos valores de $m$ de una forma más visual en la Figura 5.1.

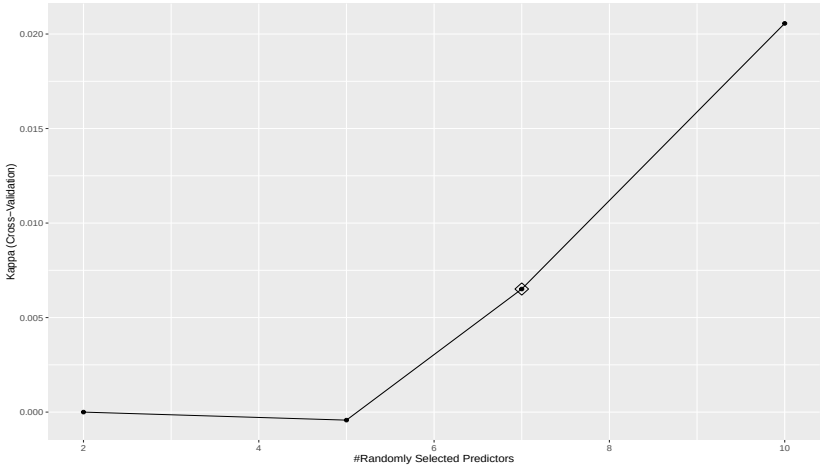



Figura 5.1: Gráfico de selección de hiperparámetros para el modelo basado en *random forest*.

Otra comprobación que podemos realizar consiste en estudiar la convergencia del error de las observaciones out-of-bag en función de los árboles considerados. Por defecto, el método considera 500 árboles; sin embargo, en la Figura 5.2, podemos apreciar que a partir de los 100 árboles ya podemos hablar de convergencia, por tanto podríamos haber ajustado el modelo con menos árboles para ahorrar coste computacional.

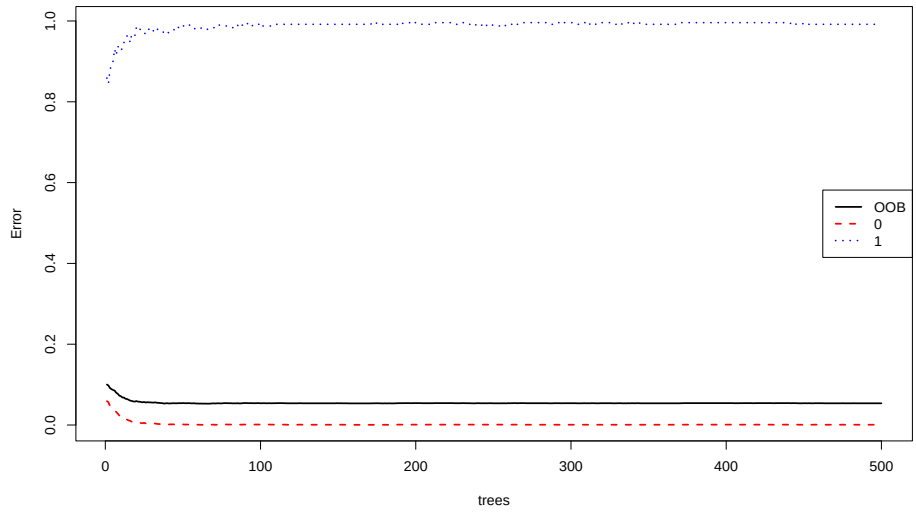



Figura 5.2: Convergencia del error OOB en función del número de árboles considerados.

Una consideración adicional que debemos hacer a la hora de estudiar la convergencia es analizar si existe una dependencia entre los distintos árboles que construye el modelo, para ello podemos indagar en cuál es la variable que se selecciona para realizar el primer corte de cada árbol mediante el siguiente código.

```
split_var_1 <- sapply(seq_len(rf.caret$finalModel$ntree),
+          function(i) getTree(rf.caret$finalModel, i, labelVar=TRUE)[1,"split var"])
table(split_var_1)
split_var_1
```

```
VAR_CLI_CONT_10    VAR_CLI_CONT_12    VAR_CLI_CONT_14    VAR_CLI_CONT_15
3                  1                  0                  2
VAR_CLI_CONT_16    VAR_CLI_CONT_17    VAR_CLI_CONT_18    VAR_CLI_CONT_19
6                  1                  2                  3
VAR_CLI_CONT_2     VAR_CLI_CONT_4     VAR_CLI_CONT_5     VAR_CLI_CONT_8
85                  0                 12                  0
```

```
VAR_CLI_DISC_WOE_1 VAR_CLI_DISC_WOE_3 VAR_CLI_DISC_WOE_4 VAR_CLI_DISC_WOE_7
28                  9                  0                 11
VAR_OP_CONT_1      VAR_OP_CONT_2      VAR_OP_CONT_4      VAR_OP_CONT_5
0                 53                 17                 6
VAR_OP_CONT_6      VAR_OP_CONT_7      VAR_OP_CONT_8      VAR_OP_CONT_9
2                  9                 50                 6
VAR_OP_DISC_WOE_1  VAR_OP_DISC_WOE_2 VAR_CLI_DISC_WOE_6 VAR_CLI_DISC_WOE_5
86                104                  4                  0
```

En base a los resultados bastante heterogéneos que obtenemos, podemos deducir que no hay una dependencia excesiva entre los árboles que distorsione las conclusiones sobre la convergencia que hemos realizado.

Por otro lado, cabe observar que el error OOB para las observaciones clasificadas como morosas es bastante superior al de las observaciones sanas. Aunque en la Sección 5.2.3 analizaremos el comportamiento del modelo sobre el conjunto de validación, podemos profundizar un poco más en lo que ocurre con la muestra OOB.

El modelo toma por defecto $c = 0.5$ como punto de corte para clasificar las observaciones, obteniendo así la matriz de confusión mostrada en la Tabla 5.1. En consecuencia, la tasa de error OOB que hemos obtenido viene dada por

$$1 - Accuracy = \frac{FN + FP}{P + N} = 0.0537,$$

mientras que las de las observaciones negativas y positivas se corresponderán con

$$1 - TNR = 0.000698,$$

$$1 - TPR = 1 - \frac{2}{2 + 241} = 0.991769.$$

Como ya habíamos apreciado en la Figura 5.2, la tasa de error de las observaciones positivas es muy superior a la de las negativas, estando la primera situada casi en 1, y la segunda en 0.



Tabla 5.1: Matriz de confusión para el modelo *random forest* empleando la muestra OOB.

### 5.2.2. Análisis del modelo

Una vez obtenidos los hiperparámetros del modelo y habiendo ajustado el mismo, vamos a realizar un análisis en profundidad para intentar comprender mejor el *random forest* resultante. En primer lugar, podemos hacer uso de la función `varImp` para medir la importancia de las variables, en el caso de los bosques aleatorios, la medida resultante proviene de medir la reducción en la función de pérdida empleando el índice de Gini (como ocurría con los árboles) y promediar esta reducción para todos los árboles construidos. Procediendo de esta forma, obtenemos los resultados de la Figura 5.3.

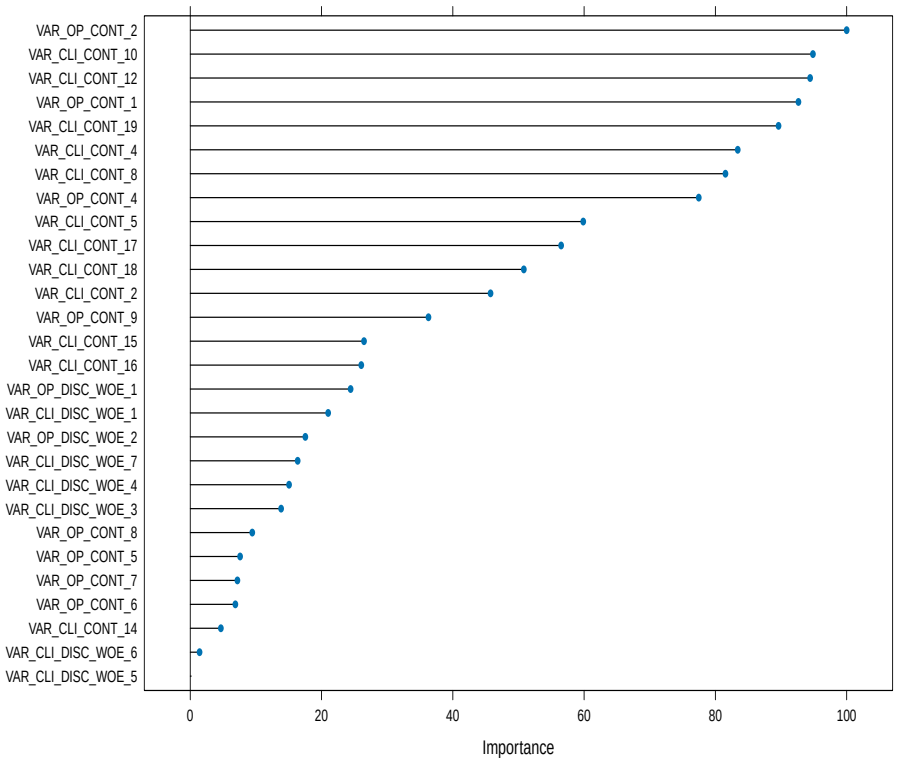
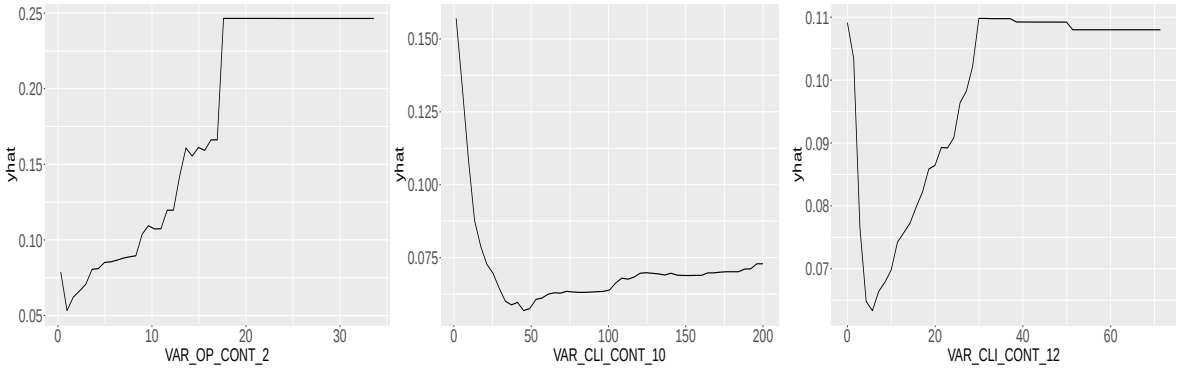



Figura 5.3: Variables del *random forest* ordenadas por importancia.

En este caso, tenemos que las 5 primeras variables correspondientes a `VAR_OP_CONT_2`, `VAR_CLI_CONT_10`, `VAR_CLI_CONT_12`, `VAR_OP_CONT_1` y `VAR_CLI_CONT_19` tienen una importancia similar.

Analizaremos las 3 primeras variables más en detalle a partir de los mecanismos introducidos en la Sección 1.4. En primer lugar, representaremos los gráficos *PDP* asociados a cada variable a través de la librería `pdp`, donde hemos especificado que las predicciones se muestren en la escala de la probabilidad de mora, lo ilustramos en la siguiente línea de código y representamos los 3 gráficos en la Figura 5.4.

```
pdp1 <- partial(rf.caret, "VAR_OP_CONT_2",prob=TRUE, which.class = 2)
autoplot(pdp1)
```




(a) `VAR_OP_CONT_2`. (b) `VAR_CLI_CONT_10`. (c) `VAR_CLI_CONT_12`.

Figura 5.4: Gráficos *PDP* para las 3 variables más importantes del modelo *random forest*.

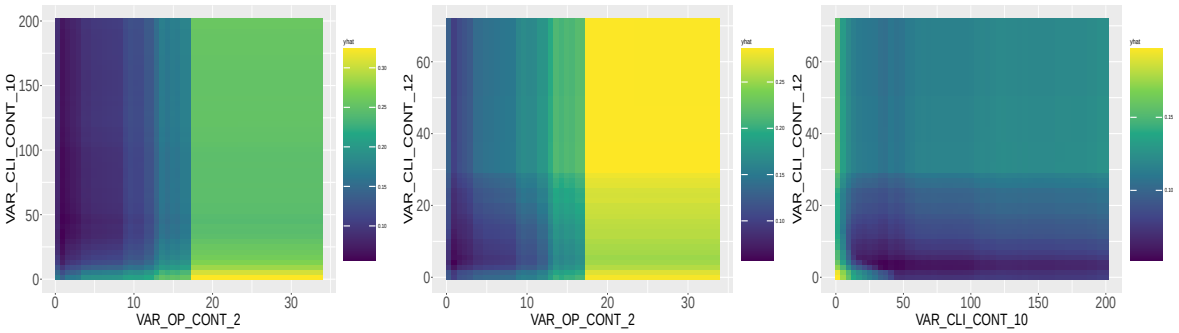
En función de los gráficos obtenidos podemos obtener las siguientes conclusiones:

- Para la variable `VAR_OP_CONT_2`, excepto en el primer tramo (hasta 2.5) donde se produce una disminución de la probabilidad de mora, nos encontramos con que para los valores de `VAR_OP_CONT_2` comprendidos entre 2.5 y 17.5 (aproximadamente), se produce un incremento progresivo de la probabilidad de *default* que asignamos a una operación, siendo a partir de 17.5, el efecto de esta variable constante.

- Para la variable `VAR_CLI_CONT_10` tenemos una disminución importante de la predicción de probabilidad que efectúa el modelo hasta el valor 50, comenzando a partir de este punto a experimentar un ligero crecimiento.

- En lo que respecta a la variable `VAR_CLI_CONT_12` tiene un comportamiento similar a la anterior, en tanto que se produce una disminución muy considerable de la predicción en los primeros valores. Para `VAR_CLI_CONT_12` $> 5$ vemos justamente lo contrario, un fuerte incremento, hasta que la variable pierde una influencia considerable a partir de valores superiores a 30.

A continuación, en la Figura 5.5 representaremos otra serie de gráficos *PDP*, en este caso bidimensionales, que tendrán en cuenta la interacción 2 a 2 entre las variables consideradas.



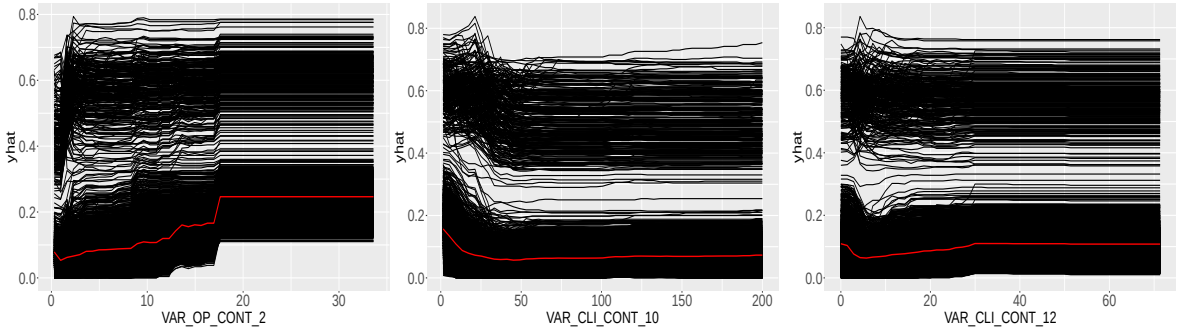

(a) `OP_CONT_2` y `CLI_CONT_10`. (b) `OP_CONT_2` y `CLI_CONT_12`. (c) `CLI_CONT_10` y `CLI_CONT_12`.

Figura 5.5: Gráficos *PDP* con la interacción 2 a 2 de las 3 variables más importantes del modelo *random forest*.

El análisis que podríamos extraer sería el siguiente:

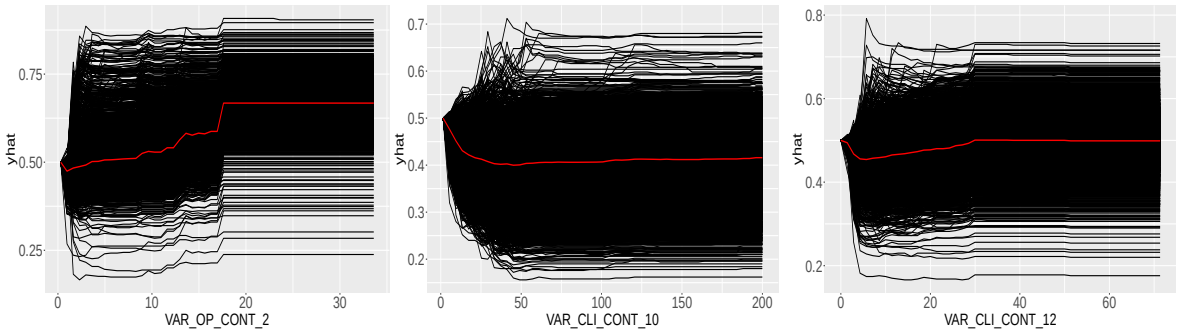
- En el primer gráfico vemos que una mayor probabilidad vendrá dada para valores de `VAR_OP_CONT_2` superiores a 17.5 y esta se verá maximizada cuando además `VAR_CLI_CONT_10` tome valores inferiores a 25.
- Para la segunda representación, vemos la influencia de `VAR_OP_CONT_2` produce que para valores superiores a 17.5 la probabilidad de mora aumente, siendo mayor para valores de `VAR_CLI_CONT_12` superiores a 30 o inferiores a 5.
- En la última gráfica, podemos observar que la probabilidad de mora máxima está concentrada para valores de `VAR_CLI_CONT_12` inferiores a 5 y de `VAR_CLI_CONT_10` menores de 12.5.

A continuación, representamos tanto los gráficos *ICE* como los *ICE* centrados (véase Sección 1.4.2) para las mismas variables que venimos tratando, con el objetivo de estudiar el comportamiento de las predicciones individuales.




(a) `VAR_OP_CONT_2`. (b) `VAR_CLI_CONT_10`. (c) `VAR_CLI_CONT_12`.

Figura 5.6: Gráficos *ICE* para las 3 variables más importantes del modelo *random forest*.




(a) `VAR_OP_CONT_2`. (b) `VAR_CLI_CONT_10`. (c) `VAR_CLI_CONT_12`.

Figura 5.7: Gráficos *ICE* centrados para las 3 variables más importantes del modelo *random forest*.

En base a las Figuras 5.6 y 5.7, observamos que existen algunas observaciones que tienen un comportamiento distinto con respecto a lo que mostraban los gráficos *PDP*. No obstante, aunque no se aprecia del todo bien debido a la gran cantidad de registros, parece que las discrepancias se centran en los primeros valores de las variables, estabilizándose a medida que avanzamos a lo largo del eje $x$, por lo que el enfoque promedio de los gráficos *PDP* puede considerarse adecuado.

Por último, emplearemos el método *LIME* para estudiar cuál es el comportamiento del modelo a la hora de predecir algunas observaciones. Consideraremos 3 observaciones diferentes: el Caso 1 será una observación asociada a la mínima probabilidad que asignó el modelo, el Caso 3 será la que ostenta la probabilidad máxima y el Caso 2 se corresponderá con una observación con una probabilidad media.

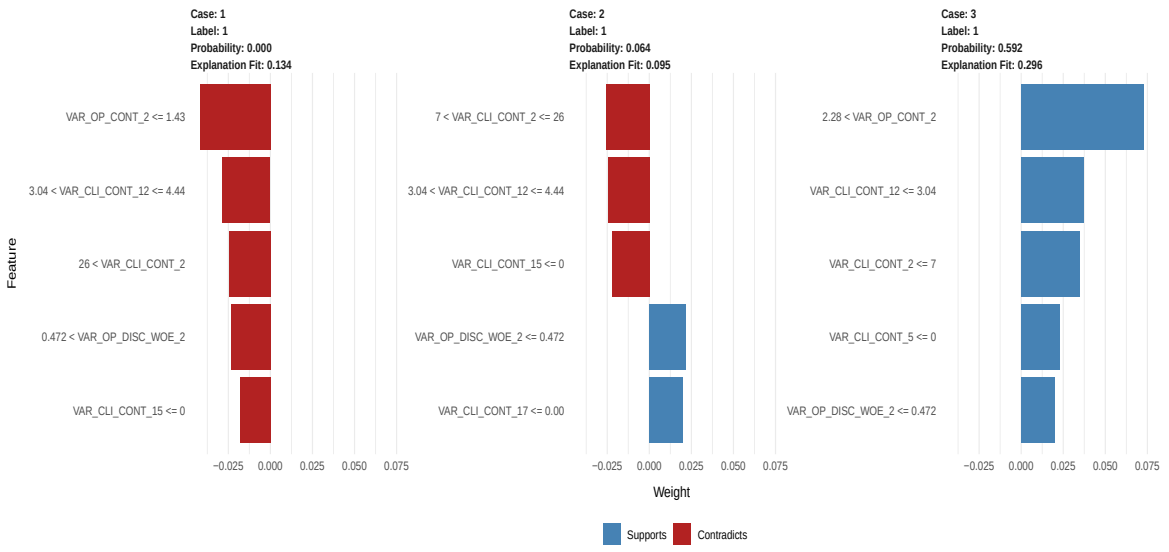



Figura 5.8: Predicciones realizadas por *random forest* asociadas a 3 observaciones explicadas mediante *LIME*.

En la Figura 5.8, para cada observación contamos con los siguientes elementos:

- La etiqueta asociada a una operación morosa.
- La predicción de probabilidad de mora que le asignó el modelo.
- El $R^2$ asignado al modelo lineal local (regresión *ridge*) usado para proporcionar las explicaciones.

Cabe resaltar que al incrementar el número de variables utilizadas (en este caso 5), aumentaremos el $R^2$ del modelo, pero la interpretación podría volverse más complicada.

Además, contamos con 5 características empleadas en el modelo original y que nos indican, en función del valor de la variable presente en cada observación, si esa característica respaldaría o no clasificar dicha observación como una operación en *default*. Es importante notar que las variables aparecen ordenadas en función de la magnitud del coeficiente asociado al modelo lineal local utilizado.

En el caso de la primera observación, para la cual el *random forest* asignó una probabilidad de mora de 0, *LIME* muestra que las 5 variables seleccionadas para explicar la predicción están en contra de la suposición de que esa observación se debería clasificar como morosa y por tanto, ratifica la predicción que realiza el bosque aleatorio.

Para el Caso 3, al que el modelo asigna una probabilidad de 0.592, ocurre exactamente lo contrario, las 5 variables seleccionadas por *LIME* respaldan que la observación se clasifique como morosa. Podemos observar que mientras que obtener un `VAR_OP_CONT_2` $\leq 1.43$, (que recordemos que estaba entre las variables más importantes) se considera una justificación para presumir que esa observación no va a incurrir en mora, que el valor sea `VAR_OP_CONT_2` $> 2.28$, tiene la connotación opuesta.

Finalmente, para el caso intermedio al que el modelo da una probabilidad de 0.064, *LIME* encuentra argumentos a favor y en contra de que la operación vaya a incurrir en mora. Más concretamente, tenemos que los valores de `VAR_CLI_CONT_2`, `VAR_CLI_CONT_12` y `VAR_CLI_CONT_15` para esta observación contribuyen a que se pueda clasificar como sana (según *LIME*), mientras que los valores de `VAR_CLI_CONT_17` y `VAR_OP_DISC_WOE_2` inclinan a pensar que se debería clasificar como morosa.

### 5.2.3. Evaluación del modelo

En este apartado, evaluaremos el modelo sobre la muestra de test siguiendo el esquema de los anteriores capítulos. A través de la Figura 5.9, realizamos una primera exploración visual para comprender las probabilidades que el modelo está prediciendo para cada una de las categorías observadas.

Como para el resto de modelos, las predicciones de las observaciones marcadas como sanas se sitúan en su mayoría cercanas a 0, mientras que las asociadas a operaciones morosas están más repartidas a lo largo del rango de valores, llegando a probabilidades de hasta 0.6. En la Tabla 5.2 representamos la matriz de confusión dada después de considerar como punto de corte 0.1 para clasificar a una operación como morosa.

Figura 5.9: Histograma de las probabilidades de mora estimadas en función de la categoría observada.



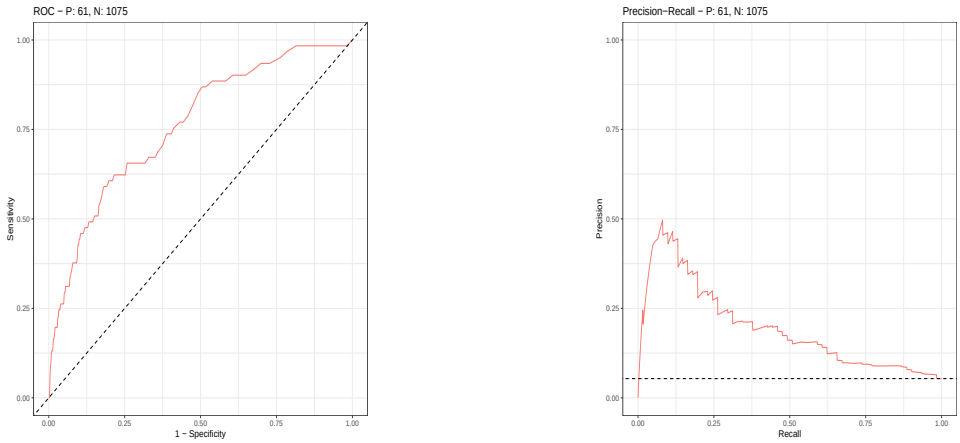
Tabla 5.2: Matriz de confusión para el modelo basado en *random forest*.

Este modelo presenta un mayor número de verdaderos positivos y una disminución de falsos negativos, eso sí, a cambio también se incrementa considerablemente la proporción de falsos positivos. Esto ocasionará un aumento de la sensibilidad y una disminución de la especificidad. Además, también se consigue aumentar también la *Balanced Accuracy* y $\kappa$ como podemos ver en la Tabla 5.3.

| TPR | TNR | Accuracy | Balanced Accuracy | $\kappa$ |
|---|---|---|---|---|
| 0.59016 | 0.81209 | 0.8002 | 0.70113 | 0.1698 |

Tabla 5.3: Valores de las métricas para el modelo basado en *random forest*.

A continuación, representamos las curvas *ROC* y *PR* en la Figura 5.10 para obtener una foto más completa sobre el rendimiento del modelo.




(a) Curva *ROC*. (b) Curva *PR*.

Figura 5.10: Curvas *ROC* y *PR* para el modelo basado en *random forest*.

Con lo que respecta a la curva *ROC*, esta es semejante a las que venimos observando hasta ahora; sin embargo, la curva *PR* parece que toma valores más elevados al comienzo, lo que podría desencadenar un aumento en el *AUC* correspondiente. En la Tabla 5.4 vemos que el *AUC ROC* está un poco por debajo de lo que obteníamos con las regresiones logística y *elastic net*, pero en cambio el *AUC PR* aumenta bastante.

| **Curva** | ROC | PR |
|---|---|---|
| **AUC** | 0.7607 | 0.1937 |

Tabla 5.4: Áreas bajo las curvas *ROC* y *PR* del modelo basado en *random forest*.

Por último, llevaremos a cabo la simulación para los distintos conjuntos de entrenamiento y poder obtener así medidas de *AUC* más robustas de cara a la comparación final.

```
df=datos_filtrados_2
B=100
auc.rf=numeric(B)
pr.rf=numeric(B)
ref.rf=numeric(B)
set.seed(2311)

for (i in 1:B){
        nobs <- nrow(df)
        itrain <- sample(nobs, 0.8 * nobs)
```

```
        train <- df[itrain, ]
        test <- df[-itrain, ]
        p=dim(train)[2]-1
        rf.caret <- train(MORA_12M ~ ., data = train, method = "rf",
        metric="Kappa",
        trControl = trainControl(method = "cv", number = 10,
        selectionFunction = "oneSE"),
        tuneGrid=data.frame(mtry=floor(seq(0.5,2,0.5)*sqrt(p))))
        obs <- test$MORA_12M
        p.est <- predict(rf.caret, type = "prob", newdata = test)
        sscurves=evalmod(scores=p.est[,2],labels=obs)
        aucs <- auc(sscurves)
        auc.rf[i]=aucs$aucs[1]
        pr.rf[i]=aucs$aucs[2]
}

auc.rf
summary(auc.rf)
auc.rf.medio=mean(auc.rf)
auc.rf.medio

pr.rf
summary(pr.rf)
pr.rf.medio=mean(pr.rf)
pr.rf.medio
```

En la Tabla 5.5, resumimos el valor de el área bajo las curvas *ROC* y *PR* recogiendo el mínimo, el máximo y la media de esas métricas para los 100 modelos simulados. En una primera aproximación, parece que los valores obtenidos son peores que los de la regresión logística y el *elastic net*, eso sí, mejorando las métricas de los árboles CART.

| | Mínimo | Máximo | Media |
|---|---|---|---|
| **AUC ROC** | 0.6657 | 0.8005 | 0.7360 |
| **AUC PR** | 0.09212 | 0.25155 | 0.16940 |

Tabla 5.5: Mínimo, máximo y media de los *AUC ROC* y *AUC PR* relativos a los 100 modelos de *random forest* simulados.

# Capítulo 6

# Boosting

En este capítulo, exploraremos tres técnicas clave enmarcadas dentro del denominado *boosting*. Comenzaremos introduciendo el algoritmo *AdaBoost* (Freund and Schapire, 1995) para cuyo desarrollo hemos empleado las referencias Schapire (2003) y Berk et al. (2008). Seguidamente, examinaremos la alternativa del *Stochastic Gradient Boosting* desarrollada por Friedman (2001) y para el que nos hemos apoyado en Hastie et al. (2009) y Friedman (2002). Finalmente daremos unas nociones sobre el *XGBoost*, una implementación altamente eficiente publicada por Chen and Guestrin (2016) que complementaremos con la perspectiva de Bentéjac et al. (2021).

Además, como ya es habitual, después de tratar los fundamentos teóricos, profundizaremos en la implementación del *XGBoost* para nuestro problema específico de *Credit Scoring* para después evaluar su desempeño.

## 6.1. Marco metodológico

El *boosting* se fundamenta en la premisa de que resulta más sencillo encontrar muchas reglas de clasificación y/o predicción simples, que una única regla compleja con una elevada precisión. Así pues, la estrategia a seguir consiste en un aprendizaje lento mediante la construcción de reglas de predicción iniciales que, aunque débiles, se refinan a lo largo de iteraciones sucesivas hasta fusionarse en una sola regla de predicción, que resulta más precisa que el uso de las reglas individuales por separado.

La metodología propuesta implicaría que cada nueva regla clasifique un conjunto de observaciones, priorizando aquellas que fueron incorrectamente clasificadas por la regla anterior. Este proceso se repetiría sucesivamente, permitiendo así una mejora continua en la precisión de la clasificación. En este sentido, el uso de árboles como algoritmos de aprendizaje base es una buena elección, ya que que proporcionan reglas de clasificación sencillas y su coste computacional es mínimo (no obstante el *boosting* también admite otros métodos distintos a los árboles).

Una de las primeras versiones de *boosting* que cosechó gran éxito en los problemas de clasificación fue el algoritmo *AdaBoost* (Freund and Schapire, 1995) que será el tema central del siguiente apartado.

### 6.1.1. AdaBoost

Para explicar este algoritmo consideraremos un problema de clasificación binaria en el que las observaciones estarán codificadas como 1 o $-1$ y los predictores débiles (predictores marginalmente superiores a un procedimiento aleatorio) que usaremos serán árboles de clasificación con poca profundidad. Así pues, el método se podría resumir en los siguientes pasos:

1. Seleccionar el número de iteraciones (árboles) que utilizaremos en el modelo y que denotaremos por $T$.

2. Inicializar los pesos correspondientes a cada observación como $w_i = 1/n$, para $i = 1, ...n$; es decir, comenzamos asignando el mismo peso a todas las observaciones.

3. Iterar sobre $t = 1, ..., T$ a través del siguiente proceso:

   - Construir un árbol para las observaciones de entrenamiento ponderadas por los pesos iniciales $w_i$.
   - Calcular la tasa del error de clasificación (definida en la Sección 4.1.2), que denotaremos por $e_t$. Es importante notar que a la hora de calcular la proporción $\hat{p}_k$, hemos de tener en cuenta las ponderaciones de las observaciones.
   - Calcular un parámetro $\alpha_t$ dado por $\alpha_t = \log\left(\frac{1-e_t}{e_t}\right)$, necesario para actualizar los pesos de las observaciones.
   - Para las observaciones que han sido mal clasificadas actualizar su anterior peso como $w_i = w_i e^{\alpha_t}$, mientras que para las observaciones bien clasificadas el peso no se altera.

4. Si denotamos por $\hat{y}_i^t$ la clasificación obtenida por cada árbol $t$ relativa a una observación $i$, la asignación final de esa observación vendrá dada por

$$\hat{y}_i = signo\left(\sum_{t=1}^{T} \alpha_t \hat{y}_i^t\right),$$

   es decir, si el resultado de la suma es positivo, entonces la observación se clasificará como 1, y si es negativa como $-1$.

Como podemos observar, al igual que ocurría con el método de *random forest*, el *boosting* también se ayuda de los árboles de decisión para construir una predicción. La principal diferencia radica en que los árboles utilizados en el *boosting* no son independientes, puesto que parten de un único conjunto de entrenamiento cuyas observaciones se van ponderando en función de la clasificación anterior.

### 6.1.2. Stochastic Gradient Boosting (SGB)

Años después de la creación del *AdaBoost*, Friedman (2001) propuso un nuevo método denominado *Stochastic Gradient Boosting*, con el que quiso adaptar la idea que hay detrás método de la dirección de máximo descenso para resolver problemas de optimización. Existen numerosas variantes del *SGB* dependiendo de los clasificadores que se usen como base, pero nosotros nos vamos a centrar en la que emplea árboles CART como predictores débiles.

Comenzaremos dando una idea del procedimiento que lleva a cabo el *Gradient Boosting* sin incorporar por el momento la componente aleatoria. Así pues, el primer paso en el *Gradient Boosting* supone ajustar un árbol de regresión a la muestra de entrenamiento con el que obtendremos unas primeras predicciones, que posteriormente usaremos para calcular ciertos residuos definidos en la Ecuación (6.2).

Utilizando los residuos mencionados, procederemos a ajustar un nuevo árbol con el que obtener reestimaciones que sumaremos a la anterior predicción, dando lugar a un valor estimado actualizado que denotaremos por $\hat{y}_i$. Ante un problema de clasificación binaria donde las variables están codificadas como $1s$ y $0s$, se suele asignar un 1 a aquellas observaciones con $\hat{y}_i > 0.5$ y un 0 a las que cumplen con $\hat{y}_i \leq 0.5$.

Una vez presentada la idea del método, daremos unos detalles más formales al respecto. Para representar un árbol CART, podemos emplear la siguiente expresión

$$m(x) = \sum_{j=1}^{J} \gamma_j I(x \in R_j),$$

donde $x$ representa el conjunto de variables explicativas de la observación que queremos predecir, $\gamma_j$ denota la predicción que realiza el árbol para las observaciones pertenecientes a la región $R_j$ e $I$ es una variable indicadora que toma el valor 1 si $x \in R_j$ y 0 en caso contrario.

Si denotamos por $t = 1, ..., T$ a los distintos árboles que ajusta el método, lo que buscamos en cada etapa es encontrar un árbol que minimice la siguiente expresión

$$\sum_{i=1}^{n} L(y_i, f_{t-1}(x_i) + m(x_i)), \tag{6.1}$$

donde $f_{t-1}(x_i)$ representa las predicciones del árbol ajustado en la etapa anterior y $L$ es una función de pérdida definida de antemano; por ejemplo, la RSS para el caso de regresión (pueden consultarse otras funciones de pérdida en Hastie et al. (2009), página 360).

Una forma de minimizar la Expresión (6.1) es ajustar el árbol de la siguiente etapa sobre los residuos

$$r_i^t = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{t-1}}. \tag{6.2}$$

Aquí es donde el método de máximo descenso cobra relevancia, ya que ajustar un árbol para esos residuos es equivalente a actualizar nuestras predicciones con el objetivo de minimizar al máximo la pérdida producida.

A continuación, presentamos un esquema del *Stochastic Gradient Boosting*, donde la única diferencia con el *Gradient Boosting* reside en que a la hora de ajustar los árboles en cada etapa no empleamos todas las observaciones, sino que en el Paso 2.ii) realizamos una selección aleatoria.

1. Inicializamos el algoritmo con un valor dado por $f_0(x) = \arg\min_\kappa \sum_{i=1}^N L(y_i, \kappa)$.

2. Para $t = 1, ..., T$:

   i) Calculamos los residuos dados por

   $$r_i^t = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{t-1}}, \text{ para } i = 1, \ldots, n.$$

   ii) Hacemos una selección aleatoria de $m < n$ observaciones (residuos).

   iii) Ajustamos un árbol para esos residuos obteniendo regiones $R_j^t$ para cada uno de los nodos terminales $j = 1, ..., J_t$.

   iv) Para cada nodo $j = 1, 2, ..., J_t$, calculamos la predicción correspondiente

   $$\gamma_j^t = \arg\min_\gamma \sum_{x_i \in R_j^t} L(y_i, f_{t-1}(x_i) + \gamma).$$

   v) Calculamos

   $$f_t(x) = f_{t-1}(x) + \rho \sum_{j=1}^{J_t} \gamma_j^t I(x \in R_j^t).$$

   donde $\rho$ es un parámetro de penalización que determina lo que denominamos "tasa de aprendizaje".

3. $\hat{f}(x) = f_T(x)$.

El motivo de incluir un parámetro de penalización $\rho$ en el Paso 2.v) se debe a que, a diferencia de lo que ocurría con el *random forest*, este método puede tener problemas de sobreajuste si empleamos demasiadas iteraciones en el proceso. Es por ello que se selecciona una tasa de aprendizaje para que el método no vaya demasiado deprisa, lo habitual es seleccionar $\rho \in [0.001, 0.1]$. Además, también podemos controlar el sobreajuste determinando la profundidad de los árboles que usaremos como predictores débiles, y que deberían ser solo marginalmente superiores a un clasificador aleatorio.

Como hemos precisado a lo largo de la sección, en el caso de este método, los hiperparámetros a determinar (que podremos seleccionar mediante validación cruzada) serían: el número de iteraciones $T$, el número de observaciones aleatorias $m$ a considerar, la profundidad de los árboles que usaremos y la tasa de aprendizaje $\rho$.

### 6.1.3. XGBoost

El *XGBoost* (Chen and Guestrin, 2016) es un método relativamente reciente, que se basa en los fundamentos del *Gradient Boosting*, pero que resulta ser más eficiente. Esta mejora en el rendimiento

se consigue minimizando en cada iteración, la siguiente función de pérdida penalizada

$$\sum_{i=1}^{n} L(y_i, f_{t-1}(x_i) + m(x_i)) + \Omega(m(x)), \tag{6.3}$$

$$\Omega(m(x)) = \alpha M + \frac{1}{2}\nu||\omega||^2, \tag{6.4}$$

donde $M$ representa el tamaño del árbol, $\omega$ es el vector de las predicciones realizadas para cada una de las hojas, el valor $\alpha$ controla el crecimiento del árbol de tal forma que para valores más elevados menos nodos tendremos y $\nu$ es un parámetro de regularización adicional, análogo a la tasa de aprendizaje del *Gradient Boosting*.

Además, la función de pérdida $L$ de la Ecuación (6.3) se puede representar como un desarrollo de Taylor de segundo orden en el que intervienen las derivadas primera y segunda de la propia función. A partir de esta circunstancia, y empleando la hessiana, también se pueden limitar las divisiones adicionales que se realicen para un nodo del árbol.

En el caso del *XGBoost*, también podemos incorporar un componente aleatorio que ayude a reducir el sobreajuste de dos formas distintas: o bien realizando una selección aleatoria de las observaciones a ajustar (filas) como ocurría con el *Stochastic Gradient Boosting*, o tomar un subconjunto aleatorio de las características de la muestra (columnas).

Por último, otro elemento de valor del *XGBoost* es que logra paralelizar varios procesos para obtener una mayor velocidad de computación a la hora de entrenar los árboles empleados, y de esta forma, convertirse en una alternativa atractiva frente al *SGB*.

## 6.2. Aplicación práctica

### 6.2.1. Ajuste del modelo

Una vez sentadas las bases del *XGBoost*, aplicaremos el modelo a nuestro conjunto de datos y estudiaremos los resultados obtenidos. Para llevar a cabo el ajuste, nos apoyaremos en la librería `caret` que realizará una selección de los hiperparámetros de tal forma que se optimice la $\kappa$ de Cohen mediante validación cruzada. En el caso del *XGBoost*, los hiperparámetros a seleccionar son:

- `eta`: se corresponde con la tasa de aprendizaje $\nu$ de la Ecuación (6.4), para la que consideraremos los valores 0.001, 0.01 y 0.1, con el objetivo de forzar un aprendizaje lento para disminuir el riesgo de sobreajuste.
- `nrounds`: será el número de iteraciones realizadas. Como las posibles tasas de aprendizaje que emplearemos como candidatas son relativamente pequeñas, seleccionaremos el número de iteraciones entre 100, 150 y 200.

- `max_depth`: se corresponde con la profundidad máxima que pueden alcanzar los árboles. Dado que buscamos predictores débiles, limitaremos la profundidad del árbol a 1 o 2 divisiones en aras de evitar el sobreajuste.
- `gamma`: se corresponde con el valor $\alpha$ de la Ecuación (6.4), el valor por defecto que tiene asociado es 0, y como ya hemos prefijado la profundidad máxima, vamos a mantener ese valor para no ser demasiado conservadores.
- `min_child_weight`: esta es la mínima suma del peso asignado por la hessiana para permitir una nueva partición en un nodo del árbol. Puede ser cualquier valor positivo, y cuanto mayor sea su magnitud, más conservador será el algoritmo. Por defecto, utilizaremos el valor predeterminado de 1.
- `colsample_bytree`: hace referencia al porcentaje de características seleccionadas que se considerarán para hacer crecer los árboles. Consideraremos el 80 %, 90 % y 100 %.
- `subsample`: se refiere al número de observaciones que se seleccionarán aleatoriamente en cada iteración. También consideraremos el 80 %, 90 % y 100 %.

Así pues, el ajuste se realizaría a través del siguiente código:

```
set.seed(2311)
caret.xgb <- train(MORA_12M ~ ., method = "xgbTree", data = train, metric="Kappa",
trControl = trainControl(method = "cv", number = 10, selectionFunction = "oneSE"),
tuneGrid=expand.grid("eta"=c(0.001,0.01,0.1),
"nrounds"=c(100,150,200),
"max_depth"=c(1,2),
"gamma"=0,
"min_child_weight"=1,
"colsample_bytree"=c(0.8,0.9,1),
"subsample"=c(0.8,0.9,1)),
verbosity = 0)
```

En esta ocasión, debido a que la malla de hiperparámetros generada contiene 162 combinaciones, no realizaremos una representación gráfica y mostraremos solo algunas de ellas incluyendo, eso sí, la combinación seleccionada que recogemos en la Tabla 6.1.

```
eXtreme Gradient Boosting

4540 samples
28 predictor
2 classes: '0', '1'
```

```
No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 4086, 4086, 4086, 4087, 4086, 4085, ...
Resampling results across tuning parameters:

eta    max_depth  colsample_bytree  subsample  nrounds  Accuracy   Kappa
0.100  2          0.8               0.8        100      0.9464762   0.0000000000
0.100  2          0.8               0.8        150      0.9469162   0.0149097820
0.100  2          0.8               0.8        200      0.9462550   0.0136340462
0.100  2          0.8               0.9        100      0.9464762   0.0000000000
0.100  2          0.8               0.9        150      0.9464762   0.0000000000
0.100  2          0.8               0.9        200      0.9464762   0.0071845691
0.100  2          0.8               1.0        100      0.9464762   0.0000000000
[ reached getOption("max.print") -- omitted 20 rows ]

Tuning parameter 'gamma' was held constant at a value of 0
Tuning parameter 'min_child_weight' was held constant at a
value of 1
Kappa was used to select the optimal model using  the one SE rule.
The final values used for the model were nrounds = 150, max_depth = 2, eta = 0.1, gamma = 0,
colsample_bytree = 0.8, min_child_weight = 1 and subsample = 0.8.
```

| eta | nrounds | max_depth | gamma | min_child_weight | colsample_bytree | subsample |
|---|---|---|---|---|---|---|
| 0.1 | 150 | 2 | 0 | 1 | 0.8 | 0.8 |

Tabla 6.1: Valores de los hiperparámetros del *XGBoost* seleccionados para el conjunto de entrenamiento considerado.

### 6.2.2. Análisis del modelo

Después de haber ajustado el modelo, veremos cuáles son las variables más importantes que intervienen a la hora de realizar las predicciones empleando *XGBoost*. En este caso, la metodología que emplea la función `varImp` mantiene el mismo enfoque que tenía para los árboles CART, con la diferencia de que se suman las importancias obtenidas en cada iteración del *boosting*, llegando así al siguiente ranking.

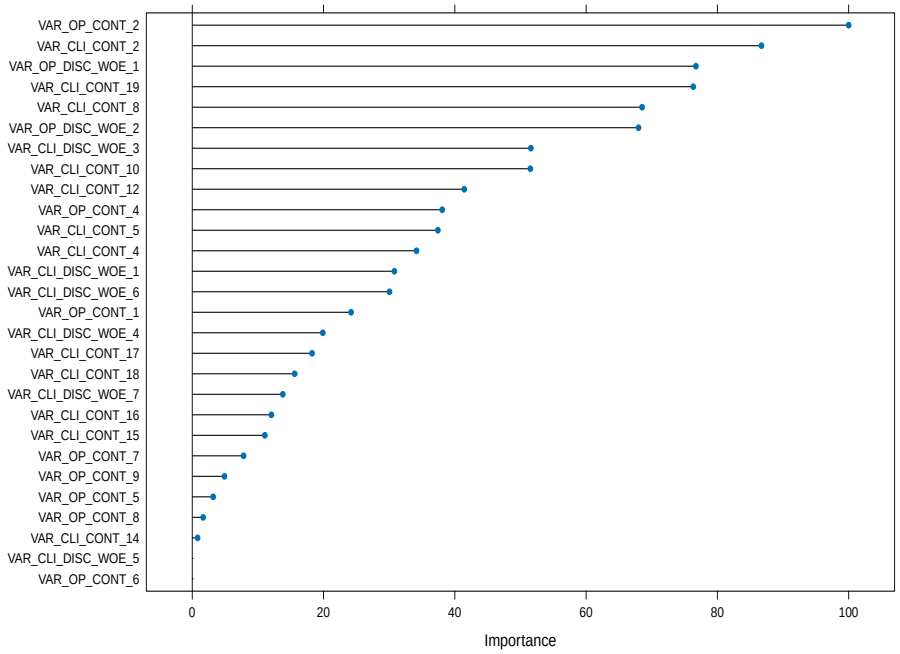


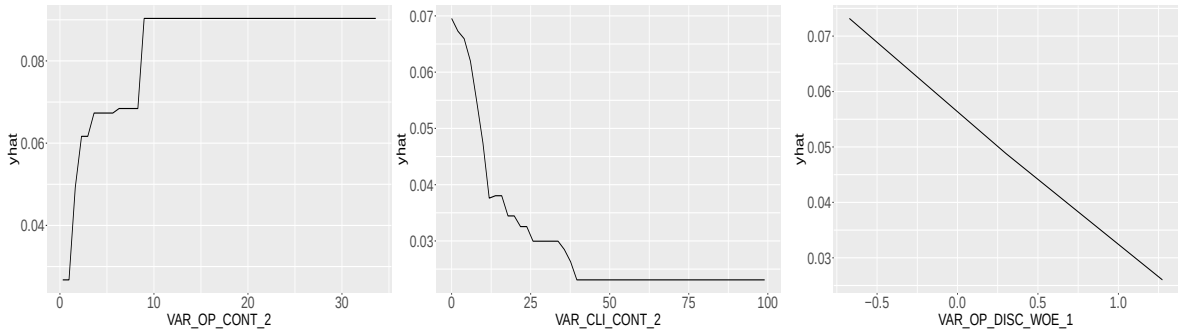

Figura 6.1: Variables del *XGBoost* ordenadas por importancia.

Como podemos ver en la Figura 6.1, las 3 variables que mayor influencia tienen en el modelo son `VAR_OP_CONT_2`, `VAR_CLI_CONT_2` y `VAR_OP_DISC_WOE_1`. Es interesante observar que dentro de las 5 variables más importantes aparecen `VAR_OP_CONT_2` y `VAR_CLI_CONT_19` que también estaban dentro del top 5 de los bosques aleatorios o `VAR_CLI_CONT_8` que ocupaba una posición alta en el caso de los árboles CART.

Al igual que lo que ocurría con el método *random forest*, *XGBoost* es uno de los denominados métodos de caja negra, por lo que no podemos precisar cuál es el comportamiento global que seguirá el modelo. No obstante, emplearemos las 3 variables más importantes seleccionadas para representar su efecto sobre las predicciones mediante gráficos *PDP*, *ICE* y el método *LIME*.

Comenzaremos analizando los gráficos *PDP* representados en la Figura 6.2. Podemos ver que las variables `VAR_OP_CONT_2` y `VAR_CLI_CONT_2` tienen comportamientos completamente opuestos: en el primer caso la probabilidad de mora va creciendo a medida que aumenta la variable hasta que llega a un punto en el que deja de tener influencia, mientras que en el segundo las predicciones tienden a ser menores a medida que `VAR_CLI_CONT_2` se hace más grande hasta que se estabilizan superado cierto umbral.

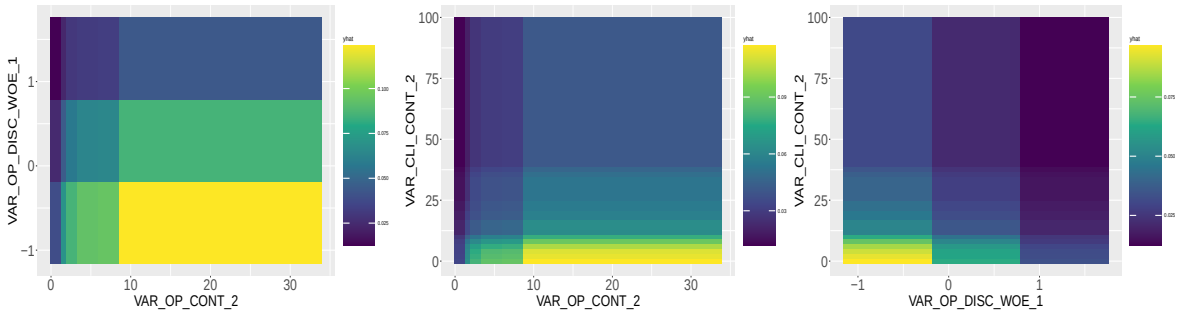
En el caso de la variable `VAR_OP_DISC_WOE_1`, observamos un claro comportamiento decreciente, y además, lineal. Esto se explica debido a que estamos ante una variable *WoE* que no toma demasiados valores.




(a) `VAR_OP_CONT_2`. (b) `VAR_CLI_CONT_2`. (c) `VAR_OP_DISC_WOE_1`.

Figura 6.2: Gráficos *PDP* para las 3 variables más importantes del modelo *XGBoost*.

A continuación, en la Figura 6.3 representaremos los gráficos *PDP* bidimensionales que se corresponden con unos mapas de calor generados a partir de representar en el eje $x$ y en el eje $y$ las combinaciones de los pares de variables que hemos tratado antes.




(a) `OP_CONT_2` y `OP_DISC_WOE_1` . (b) `OP_CONT_2` y `CLI_CONT_2`. (c) `OP_DISC_WOE_1` y `CLI_CONT_2`.

Figura 6.3: Gráficos *PDP* con la interacción 2 a 2 de las 3 variables más importantes del modelo *XGBoost*.

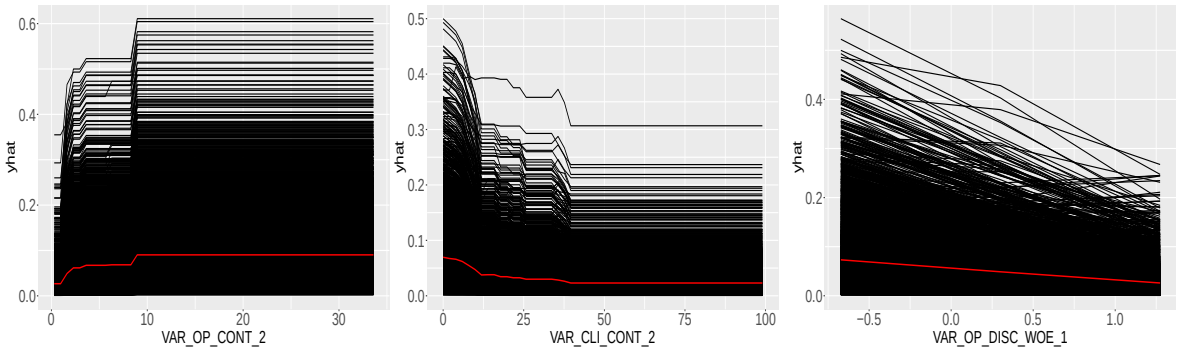
Para la primera representación, podemos observar que para valores de `VAR_OP_CONT_2` entre 0 y 10 la probabilidad de mora va aumentando, hasta que a partir de ese punto la variable deja de tener importancia dejando únicamente la influencia de `VAR_OP_DISC_WOE_1`, que presenta mayores predicciones cuanto menor es su valor.

Con respecto al segundo gráfico, en consonancia con lo que veíamos en los *PDP* individuales, las probabilidades más altas se concentran para valores elevados de `VAR_OP_CONT_2` y valores bajos de `VAR_CLI_CONT_2`, mientras que las menores probabilidades se dan exactamente en las condiciones contrarias.

Por último, para el *PDP* donde intervienen `VAR_OP_DISC_WOE_1` y `VAR_CLI_CONT_2`, al tener las dos variables un comportamiento decreciente, obtenemos que a medida que nos acercamos a la esquina superior derecha las predicciones son menores, concentrándose las probabilidades más elevadas cuando

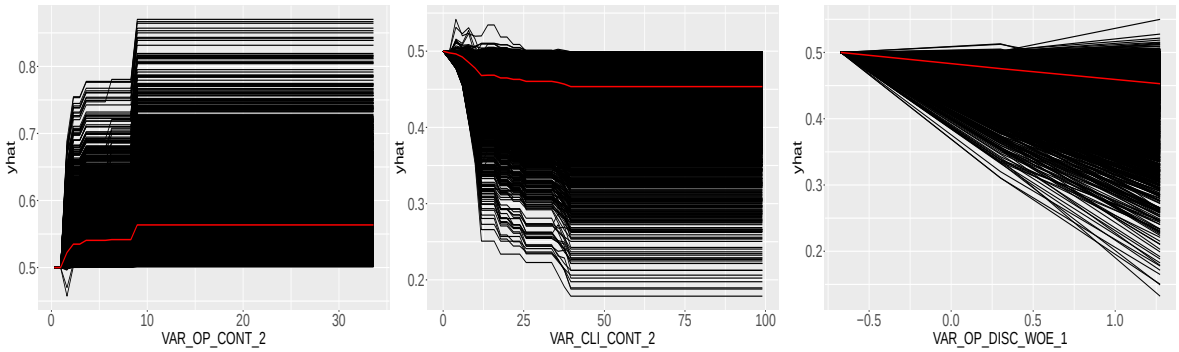
las dos variables toman valores pequeños.

Seguidamente, representamos los gráficos *ICE* e *ICE* centrados en las Figuras 6.4 y 6.5 respectivamente, donde podremos analizar si el enfoque promedio de los gráficos *PDP* representa de manera fidedigna el comportamiento de las predicciones individuales de las observaciones.




(a) `VAR_OP_CONT_2`. (b) `VAR_CLI_CONT_2`. (c) `VAR_OP_DISC_WOE_1`.

Figura 6.4: Gráficos *ICE* para las 3 variables más importantes del modelo *XGBoost*.




(a) `VAR_OP_CONT_2`. (b) `VAR_CLI_CONT_2`. (c) `VAR_OP_DISC_WOE_1`.

Figura 6.5: Gráficos *ICE* centrados para las 3 variables más importantes del modelo *XGBoost*.

Con respecto a la representación referente a `VAR_OP_CONT_2`, parece que las predicciones tienen un comportamiento similar a la media, excepto algunas de ellas durante el primer tramo en el que disminuye la probabilidad cuando esta debería aumentar.

En el segundo gráfico, podemos ver que existen un mayor número de discrepancias también en los primeros valores de la variable, donde parece que la probabilidad tiende a crecer para algunas observaciones. No obstante, podríamos decir que la tendencia mayoritaria es la marcada por la media.

Finalmente, en el caso de la variable *WoE*, observamos que la tendencia lineal decreciente es la que siguen la mayor parte de las observaciones.

Una vez estudiado el comportamiento del modelo de una forma más general, intentaremos dar explicación a predicciones individuales asociadas a observaciones concretas. Tomaremos 3 observaciones con distintas probabilidades sobre las que emplearemos el método *LIME* y cuyo resultado final aparece recogido en la Figura 6.6.

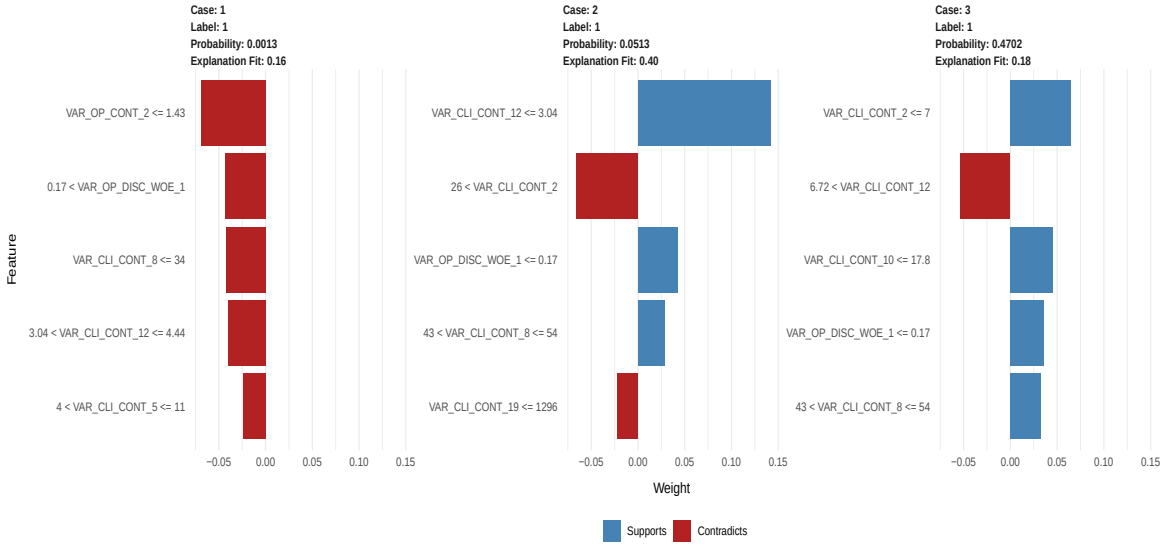



Figura 6.6: Predicciones realizadas por *XGBoost* asociadas a 3 observaciones explicadas mediante *LIME*.

Para el Caso 1, los valores de las 5 variables representadas asociadas a la observación que estamos analizando, contradicen la suposición de clasificar esa observación como morosa y es por ello que la probabilidad de mora dada por el *XGBoost* es muy cercana a 0.

Por lo que podemos ver en el Caso 3, obtenemos una probabilidad de 0.47, bastante elevada en base a las predicciones que hemos estado viendo. Casi todas las características favorecen una probabilidad de mora elevada, excepto por `VAR_CLI_CONT_12` $>$ 6.72, que sería un factor en contra de que la observación vaya a incurrir en mora, a pesar de que el cómputo global de las 5 variables nos indicaría que sí entraría en *default*.

En el Caso 2, vemos que hay variables tanto a favor como en contra de que la operación vaya a incurrir en mora. Puede parecernos algo sorprendente que la observación no tenga asociada una predicción de probabilidad de mora mayor, teniendo en cuenta el hecho de que la `VAR_CLI_CONT_12` $\leq$ 3.04 tiene un peso bastante significativo y apoya la suposición de morosidad.

Sin embargo, hay que valorar que (además de basarse en una escala local) el método *LIME* está aportando esa explicación basándose únicamente en 5 variables. En la Figura 6.7, podemos ver que ampliando la explicación a 8 variables, se incluye `VAR_CLI_DISC_WOE_3` que tiene una aportación negativa bastante importante.

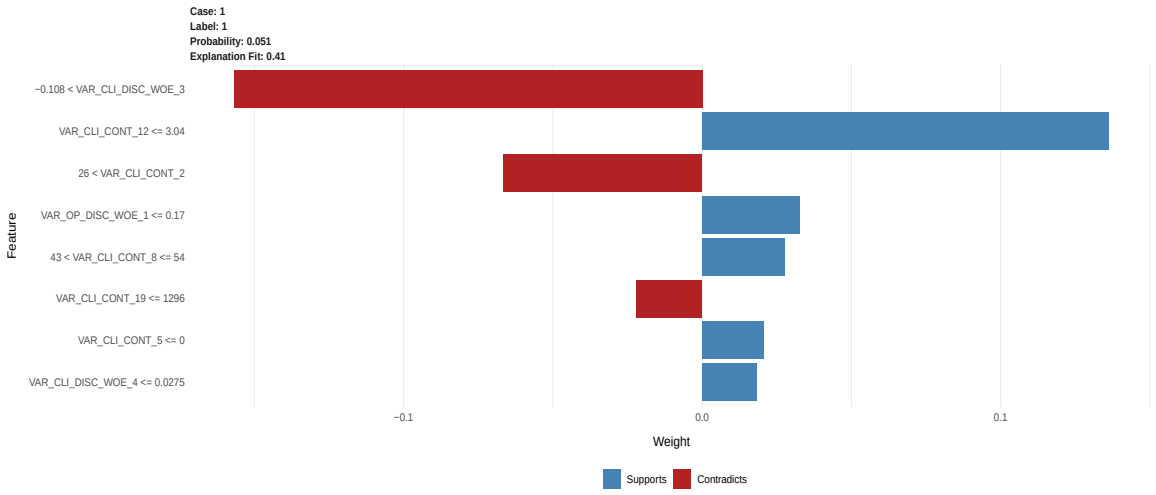



Figura 6.7: Explicación mediante *LIME* del Caso 2 empleando 8 variables.

### 6.2.3. Evaluación del modelo

Dedicaremos esta sección a evaluar el rendimiento del último de los modelos presentados, el *XGBoost*, haciendo uso de la muestra de validación.

El primer paso será echar un vistazo a las probabilidades que arroja el modelo en función de si estamos ante la clase de operaciones sanas o en *default*, lo vemos en la Figura 6.8.



Figura 6.8: Histograma de las probabilidades de mora estimadas en función de la categoría observada.

Lo cierto es que a simple vista, obtenemos unos resultados bastante similares a los dados por el *random forest*, siendo en esta ocasión las probabilidades más altas que obtenemos del orden de 0.5.

Podemos también analizar distintas métricas a partir de la matriz de confusión representada en la Tabla 6.2 y que ha sido calculada considerando 0.1 como punto de corte para clasificar una operación como morosa.



Tabla 6.2: Matriz de confusión para el modelo basado en *XGBoost*.

Con respecto al modelo de *random forest*, *XGBoost* clasifica más operaciones en sanas que en morosas, lo que conlleva a un aumento de los verdaderos y falsos negativos y a un descenso de los verdaderos y falsos positivos.

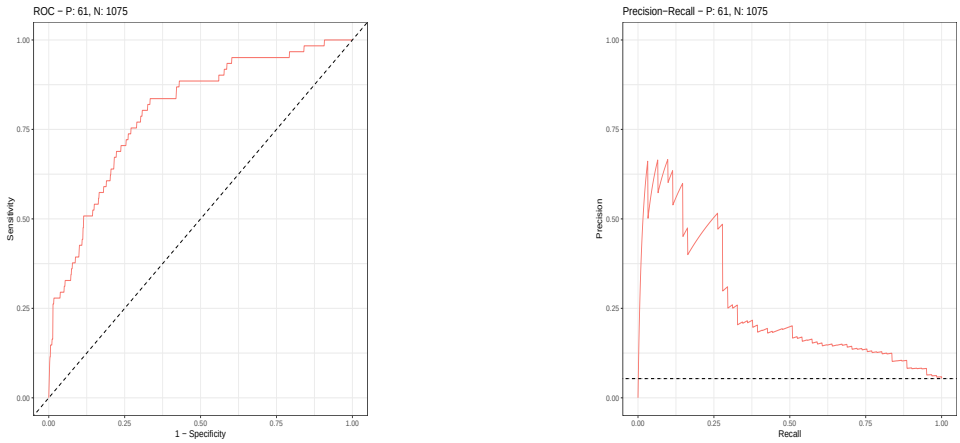
Es interesante destacar que obtenemos los mismos FN (30) y TP (31) que con la regresión logística y *elastic net*. En consecuencia obtendremos la misma sensibilidad (*TPR*), como podemos ver en la Tabla 6.3, que resulta bastante inferior a la que conseguíamos con el *random forest*.

Además, la *Accuracy* y $\kappa$ se ven beneficiados ante el aumento del *TNR*, mientras que la *Balanced Accuracy* resulta mermada frente a la obtenida por el *random forest*.

| *TPR* | TNR | Accuracy | Balanced Accuracy | $\kappa$ |
|---|---|---|---|---|
| 0.50820 | 0.87814 | 0.8583 | 0.69317 | 0.2169 |

Tabla 6.3: Valores de las métricas para el modelo basado en *XGBoost*.

En la Figura 6.9, hemos representado las curvas *ROC* y *PR*. Con respecto a la primera, es bastante similar a las obtenidas para el resto de modelos, sin embargo, la curva *Precision-Recall* parece que consigue llegar a puntos más alejados en el eje vertical, lo que podría desencadenar un aumento en el *AUC* correspondiente. En la Tabla 5.4 vemos que el *AUC ROC* está un poco por debajo de lo que obteníamos con las regresiones logística y *elastic net*, pero en cambio el *AUC PR* aumenta bastante.




(a) Curva *ROC*. (b) Curva *PR*.

Figura 6.9: Curvas *ROC* y *PR* para el modelo basado en *XGBoost*.

En la Tabla 6.4, donde podemos ver el área bajo cada curva, obtenemos dos resultados bastante buenos: 0.8012 para la curva *ROC* y 0.2541 para la curva *PR*. No obstante, tendremos que esperar a realizar una simulación con distintos conjuntos de entrenamiento para poder extraer conclusiones más fiables.

| **Curva** | ROC | PR |
|---|---|---|
| **AUC** | 0.8012 | 0.2541 |

Tabla 6.4: Áreas bajo las curvas *ROC* y *PR* del modelo basado en *XGBoost*.

Para finalizar este capítulo, incluimos la simulación de los 100 conjuntos de entrenamiento sobre los que lanzaremos el *XGBoost* para analizar cuál es su rendimiento global.

```
df=datos_filtrados_2
B=100
auc.xgb=numeric(B)
pr.xgb=numeric(B)
ref.xgb=numeric(B)
set.seed(2311)
for (i in 1:B){
        nobs <- nrow(df)
        itrain <- sample(nobs, 0.8 * nobs)
        train <- df[itrain, ]
        test <- df[-itrain, ]
```

```
        caret.xgb <- train(MORA_12M ~ ., method = "xgbTree", data = train,
        metric="Kappa",
        trControl = trainControl(method = "cv", number = 10,
        selectionFunction = "oneSE"),
        tuneGrid=expand.grid("eta"=c(0.001,0.01,0.1),
        "nrounds"=c(100,150,200),"max_depth"=1:2,
        "gamma"=0,"colsample_bytree"=c(0.8,0.9,1),
        "min_child_weight"=1,"subsample"=c(0.8,0.9,1)),
        verbosity = 0)
        obs <- test$MORA_12M
        p.est <- predict(caret.xgb, type = "prob", newdata = test)
        sscurves=evalmod(scores=p.est[,2],labels=obs)
        data_info <- attr(sscurves, "data_info")
        aucs <- auc(sscurves)
        auc.xgb[i]=aucs$aucs[1]
        pr.xgb[i]=aucs$aucs[2]
}
auc.xgb
summary(auc.xgb)
auc.xgb.medio=mean(auc.xgb)
auc.xgb.medio
pr.xgb
summary(pr.xgb)
pr.xgb.medio=mean(pr.xgb)
pr.xgb.medio
```

Asimismo, en la Tabla 6.5 resumimos el valor de el área bajo las curvas *ROC* y *PR* recogiendo el mínimo, el máximo y la media para los 100 modelos simulados. Podemos observar que las medias obtenidas para los *AUC ROC* y *PR* son peores que las del conjunto de entrenamiento empleado en la sección anterior. En el último capítulo compararemos las métricas de las simulaciones de los distintos modelos.

| | Mínimo | Máximo | Media |
|---|---|---|---|
| **AUC ROC** | 0.6579 | 0.8186 | 0.7450 |
| **AUC PR** | 0.09304 | 0.28625 | 0.17799 |

Tabla 6.5: Mínimo, máximo y media de los *AUC ROC* y *AUC PR* relativos a los 100 modelos *XGBoost* simulados.

# Capítulo 7

# Conclusiones y líneas futuras

Después de haber analizado cada uno de los modelos de *Scoring* propuestos de forma individual, en este capítulo realizaremos una comparación entre los resultados que hemos obtenido con el fin de intentar determinar cuál será el modelo más adecuado para nuestro propósito. Asimismo, indicaremos algunas posibles nuevas vías de investigación con la que se podría enriquecer el trabajo en un futuro.

## 7.1. Comparación entre modelos

A lo largo del trabajo hemos recopilado una serie de métricas de evaluación que nos servirán para decidir el método más apropiado para nuestra tarea de encontrar un modelo de *Scoring* destinado al segmento de Consumo Admisión de ABANCA.

En la Tabla 7.1, mostramos el $AUC$ correspondiente a la Curva $ROC$ y a la Curva *Precision-Recall* resultantes de realizar la simulación de 100 conjuntos de entrenamientos distintos sobre los que se ajustaron los diferentes modelos. Como podemos observar, la regresión logística es la que obtiene un mayor $AUC\ ROC$ y también el segundo mejor $AUC\ PR$, únicamente superada por el *XGBoost*.

| | Logística | Elastic Net | Árbol | Random Forest | XGBoost |
|---|---|---|---|---|---|
| **AUC ROC** | 0.7643 | 0.7608 | 0.6472 | 0.7360 | 0.7450 |
| **AUC PR** | 0.1669 | 0.1626 | 0.1249 | 0.1694 | 0.1780 |

Tabla 7.1: Áreas bajo la curva $ROC$ y $PR$ de los modelos implementados.

Como era previsible, analizando las métricas obtenidas, la utilización de los árboles de decisión

como modelo de clasificación en sí mismo no produce resultados demasiado competitivos, lo que nos lleva a descartar su uso en solitario. Sin embargo, cuando se emplean como base para métodos como el *bagging* o el *boosting*, adquieren una mayor relevancia.

Por otro lado, la regresión logística presenta un mejor *AUC ROC* y *AUC PR* que la regresión logística penalizada *elastic net* y el *random forest*. Además, su simplicidad e interpretabilidad nos proporcionan argumentos sólidos para preferirla sobre estos otros modelos. Así pues, la duda plausible estaría entre la utilización de la regresión logística o emplear el *XGBoost*, que a pesar de tener un 0.0143 menos de *AUC ROC*, obtiene un 0.0111 adicional de *AUC PR*.

Es importante recordar que la curva *ROC* representa los valores de la tasa de verdaderos positivos (*TPR*) y la tasa de falsos negativos (*FNR*) para distintos puntos de corte, de tal forma que para muestras desbalanceadas donde haya muchas más observaciones negativas que positivas los resultados podrían estar sesgados.

En cambio, la curva *PR* se construye a partir de la tasa de verdaderos positivos y el valor predictivo positivo que mide la cantidad de predicciones positivas que realmente lo son. De esta forma se minimiza el efecto que pueda tener el elevado número de observaciones negativas sobre las positivas.

Por tanto, en base a estas consideraciones, seríamos más partidarios de utilizar el *XGBoost*. Para obtener más información al respecto, hemos decidido calcular los *TPR* y *TNR* medios de cada método, como resultado de haber realizado las 100 simulaciones para los distintos conjuntos de entrenamiento.

| | Regresión Logística | XGBoost |
|---|---|---|
| **True Negative Rate** | 0.8591 | 0.7197 |
| **True Positive Rate** | 0.4868 | 0.5380 |

Tabla 7.2: *TPR* y *TNR* medios obtenidos a partir de las 100 simulaciones de la regresión logística y el *XGBoost*.

A partir de los resultados recogidos en la Tabla 7.2, podemos ver que la regresión logística proporciona una mayor tasa de verdaderos negativos (+0.1394), mientras que el *XGBoost* tiene una tasa de verdaderos positivos mas elevada (+0.0512), para el umbral especificado del 10 %. En el caso de modificar el umbral, tendríamos que realizar una actualización de los puntos de corte para ver cuál sería el rendimiento esperado; no obstante, teniendo en cuenta el 10 % y el hecho de que detectar a un deudor moroso antes de que impague aporta más beneficios al banco que perder a un potencial cliente, nos mantendríamos en nuestra decisión de seleccionar el *XGBoost*.

A pesar de ello, y aunque hemos introducido diversas técnicas para fomentar la interpretabilidad del *XGBoost*, como los gráficos *PDP*, *ICE* o el método *LIME*; se observa que la regresión logística es

un modelo más fácilmente explicable y comprensible, lo que también implica que resulta más auditable desde el punto de vista del regulador y del propio departamento de Auditoría Interna de la entidad.

Por tanto, en este sentido, tampoco sería descabellado seguir empleando la regresión logística en aras de la interpretabilidad, puesto que tampoco podemos decir que el rendimiento de este modelo sea mucho peor que el del *XGBoost*, sobre todo teniendo en cuenta que presenta algunas métricas donde su desempeño es mejor.

No obstante, cabe destacar que el conjunto de datos que hemos empleado del segmento de Consumo Admisión es relativamente pequeño, puesto que tiene poco más de 5.500 operaciones. Este "pequeño" tamaño muestral puede favorecer a que la regresión logística encuentre relaciones lineales de manera más sencilla. Mientras que para carteras con muchas más observaciones, probablemente el *XGBoost* obtuviese una ventaja competitiva importante al poder extraer relaciones entre las variables de una forma no lineal.

En conclusión, basándonos en la capacidad predictiva y en la importancia de detectar deudores potencialmente morosos, el modelo a seleccionar debería ser el *XGBoost*, considerando además que lo podemos dotar de interpretabilidad empleando los mecanismos que hemos presentado. Si bien es cierto que, al ser pequeñas las diferencias existentes con el modelo logístico, para el conjunto de datos tratado, también podríamos optar por esta alternativa que resulta más explicable, pero es posible que al aplicar este modelo a un conjunto de datos más complejo, con muchas más operaciones, el rendimiento del modelo logístico sea inferior que el del *XGBoost*.

## 7.2. Líneas futuras

Como próximos pasos a seguir para continuar la senda iniciada en este trabajo, planteamos analizar los métodos implementados en la aplicación a otras carteras. Por ejemplo, emplearlos en el segmento de Hipotecas donde contamos con un número sustancialmente mayor de operaciones sobre las que previsiblemente los modelos tendrán comportamientos distintos.

Además, también incorporaremos nuevos métodos al estudio comparativo, como pueden ser las redes neuronales (Kvamme et al., 2018) que, enmarcándose dentro del *machine learning*, permiten resolver problemas con relaciones complejas, pero también presentan una interpretación muy complicada. Otra opción a incluir serían los modelos aditivos generalizados o GAM (Liu and Cela, 2007; Liu et al., 2009), que nos permitirían dotar de una mayor flexibilidad a la regresión logística al incorporar relaciones no lineales entre las variables explicativas y la variable de interés.

En lo que respecta a las métricas empleadas para medir el rendimiento de los modelos, hemos optado por un enfoque *out of sample*, es decir, hemos separado el conjunto de datos en muestra de entrenamiento y validación, y sobre esta última, hemos evaluado el comportamiento del modelo. No obstante, cuando se generen suficientes datos, también podremos emplear un procedimiento *out of time* consistente en utilizar las operaciones que se formalizan con posterioridad a la implantación del modelo como la muestra sobre la que realizar la validación.

Por otra parte, en lo que a interpretación de los métodos de *machine learning* se refiere, podríamos estudiar el valor de Shapley (Molnar, 2022) que es un concepto propio de la teoría de juegos cooperativos que se basa en la ecuanimidad, como alternativa al método *LIME*. En nuestro ámbito de aplicación, el enfoque sería asignar, de una forma equilibrada, a cada variable el peso que ha tenido en la predicción de una cierta observación.

Finalmente, resaltar que tal y como hemos mencionado en la introducción, el uso de estos modelos de *Scoring* está en auge, no limitándose únicamente a la concesión de crédito de forma reactiva. Un ejemplo interesante, sería el envío de propuestas de financiación a clientes que no tienen a ABANCA como su entidad principal, pero que han activado el servicio de Open Banking. Esto permite al banco acceder a la información financiera de los clientes en otras entidades a través de su banca electrónica, lo que a su vez posibilita la realización de ofertas atractivas para conseguir una migración de estos potenciales usuarios hacia ABANCA y aumentar así su vinculación.

# Bibliografía


Alonso, A. and Carbó, J. M. (2020). Machine learning in credit risk: measuring the dilemma between prediction and supervisory cost. *Documentos de Trabajo del Banco de España*.

Alonso, A. and Carbó, J. M. (2021). Understanding the performance of machine learning models to predict credit default: a novel approach for supervisory evaluation. *Documentos de Trabajo del Banco de España*.

Anderson, R. (2007). *The credit scoring toolkit: theory and practice for retail credit risk management and decision automation*. Oxford University Press.

Bentéjac, C., Csörgő, A., and Martínez-Muñoz, G. (2021). A comparative analysis of gradient boosting algorithms. *Artificial Intelligence Review*, 54:1937–1967.

Berk, R. A. et al. (2008). *Statistical learning from a regression perspective*, volume 14. Springer.

Breiman, L. (1996). Bagging predictors. *Machine learning*, 24:123–140.

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

Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984). Cart. *Classification and regression trees*.

Casal, R. F., Costa, J., and Oviedo, M. (2021). Aprendizaje estadístico. https://rubenfcasal.github.io/aprendizaje_estadistico/. Accedido el 5 de Marzo de 2024.

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

Cook, R. J. (2005). Kappa. *Encyclopedia of biostatistics*, 4.

Davis, J. and Goadrich, M. (2006). The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240.

Dessain, J., Bentaleb, N., and Vinas, F. (2023). Cost of explainability in ai: An example with credit scoring models. In *World Conference on Explainable Artificial Intelligence*, pages 498–516. Springer.

Freund, Y. and Schapire, R. E. (1995). A decision-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*, pages 23–37. Springer.

Friedman, J., Hastie, T., and Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1.

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

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational statistics & data analysis*, 38(4):367–378.

Goadrich, M., Oliphant, L., and Shavlik, J. (2004). Learning ensembles of first-order clauses for recall-precision curves: A case study in biomedical information extraction. In *International Conference on Inductive Logic Programming*, pages 98–115. Springer.

Goldstein, A., Kapelner, A., Bleich, J., and Pitkin, E. (2015). Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. *journal of Computational and Graphical Statistics*, 24(1):44–65.

Greenwell, B. M. et al. (2017). PDP: An R package for constructing partial dependence plots. *R J.*, 9(1):421.

Hastie, T., Tibshirani, R., Friedman, J. H., and Friedman, J. H. (2009). *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer.

Hoerl, A. E. and Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67.

Hosmer Jr, D. W., Lemeshow, S., and Sturdivant, R. X. (2013). *Applied logistic regression*, volume 398. John Wiley & Sons.

Hvitfeldt, E., Pedersen, T. L., and Benesty, M. (2022). *lime: Local Interpretable Model-Agnostic Explanations*. R package version 0.5.3.

James, G., Witten, D., Hastie, T., Tibshirani, R., et al. (2013). *An introduction to statistical learning*, volume 112. Springer.

Kleinbaum, D. G., Klein, M., and Pryor, E. R. (2002). *Logistic regression: a self-learning text*, volume 94. Springer.

Kuhn, M. (2008). Building predictive models in R using the caret package. *Journal of statistical software*, 28:1–26.

Kuhn, M., Johnson, K., et al. (2013). *Applied predictive modeling*, volume 26. Springer.

Kuhn, M., Wing, J., Weston, S., Williams, A., Keefer, C., Engelhardt, A., Cooper, T., Mayer, Z., Kenkel, B., Team, R. C., et al. (2020). Package caret. *The R Journal*, 223(7).

Kvamme, H., Sellereite, N., Aas, K., and Sjursen, S. (2018). Predicting mortgage default using convolutional neural networks. *Expert Systems with Applications*, 102:207–217.

Larner, A. (2021). *The 2x2 matrix: contingency, confusion and the metrics of binary classification*. Springer Nature.

Liu, W. and Cela, J. (2007). Improving credit scoring by generalized additive model. In *SAS global forum*.

Liu, W., Chase, J. M., Vu, C., Cela, J., et al. (2009). Generalizations of generalized additive model (gam): A case of credit risk modeling.

McCullagh, P. (2019). *Generalized linear models*. Routledge.

McHugh, M. L. (2012). Interrater reliability: the kappa statistic. *Biochemia medica*, 22(3):276–282.

Molnar, C. (2022). Interpretable machine learning. https://christophm.github.io/interpretable-ml-book. Accedido el 8 de Mayo de 2024.

Nahm, F. S. (2022). Receiver operating characteristic curve: overview and practical use for clinicians. *Korean journal of anesthesiology*, 75(1):25.

Pereira, J. M., Basto, M., and Da Silva, A. F. (2016). The logistic lasso and ridge regression in predicting corporate failure. *Procedia Economics and Finance*, 39:634–641.

Perez, M. (2011). *Microsoft SQL Server 2008 R2. Motor de base de datos y administración*. RC Libros.

Pipe, P. (1997). The data mart: A new approach to data warehousing. *International Review of Law, Computers & Technology*, 11(2):251–262.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "Why should I trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144.

Saito, T. and Rehmsmeier, M. (2015). The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, 10(3):e0118432.

Saito, T. and Rehmsmeier, M. (2017). Precrec: fast and accurate precision-recall and roc curve calculations in r. *Bioinformatics*, 33 (1):145–147.

Schapire, R. E. (2003). The boosting approach to machine learning: An overview. In *Nonlinear estimation and classification*, pages 149–171. Springer.

Sheather, S. (2009). *A modern approach to regression with R*. Springer Science & Business Media.

Siddiqi, N. (2012). *Credit risk scorecards: developing and implementing intelligent credit scoring*, volume 3. John Wiley & Sons.

Smith, E. P., Lipkovich, I., and Ye, K. (2002). Weight-of-evidence (woe): quantitative estimation of probability of impairment for individual and multiple lines of evidence. *Human and Ecological Risk Assessment*, 8(7):1585–1596.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 58(1):267–288.

Venables, W. N. and Ripley, B. D. (2013). *Modern applied statistics with S-PLUS*. Springer Science & Business Media.

World Bank Group and International Committee on Credit Reporting (2019). Credit scoring approaches guidelines. https://thedocs.worldbank.org/en/doc/935891585869698451-0130022020/original/CREDITSCORINGAPPROACHESGUIDELINESFINALWEB.pdf. Accedido el 15 de Abril de 2024.

Xie, S. (2024). scorecard: Credit risk scorecard. https://github.com/ShichenXie/scorecard.

Yang, S. and Berdine, G. (2017). The receiver operating characteristic (roc) curve. *The Southwest Respiratory and Critical Care Chronicles*, 5(19):34–36.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 67(2):301–320.