uc3m | Universidad Carlos III de Madrid

Máster en Ciencias Actuariales y Financieras

2023-2024

Trabajo Fin de Máster

Modelización del Riesgo de Crédito con Técnicas de Machine y Deep Learning

---

Raquel Martínez Quiroga

Tutor/es

Raquel Pérez Calderón

Madrid, mayo 2024

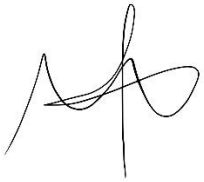
****

En caso de obtener una calificación igual o superior a 9.0 (Sobresaliente), autorizo la publicación de este trabajo en el centro de Documentación de la Fundación Mapfre.

Sí, ☒ autorizo a su publicación.

No, ☐ desestimo su publicación.

Firmado:



Raquel Martínez Quiroga.

Mayo 2024, Madrid.

****

Abstract

Las instituciones financieras están adoptando cada vez más tecnologías avanzadas, aprovechando métodos de aprendizaje automático y profundo en aplicaciones como la calificación crediticia, la detección de anomalías, los controles internos y el cumplimiento normativo. Es cuestión de tiempo que los analistas adopten estas técnicas para la predicción del riesgo en las entidades financieras, ya que son cada vez más accesibles para todo tipo de usuarios.

El propósito de este proyecto es desarrollar modelos de predicción de la probabilidad de incumplimiento sobre una cartera de clientes de una entidad bancaria española. Se exploran diversas técnicas, desde las más tradicionales; como la regresión logística, hasta métodos más avanzados como *Random Forest, XGBoost* o redes neuronales, basados en el aprendizaje automático y profundo. El objetivo principal es determinar si estas técnicas modernas ofrecen ventajas significativas en comparación con los métodos convencionales y cuantificar en qué medida esto ocurre para una muestra actual.

Con el fin de mejorar la precisión de los modelos, se aplican técnicas de balanceo de datos y validación cruzada. Para evaluar la efectividad de los modelos, se utiliza *backtesting* tanto en una muestra *in sample* como *out of sample*.

Por último, para abordar la poca interpretabilidad de los modelos de machine learning, se utilizan las técnicas de impureza de Gini y Shapley para medir la importancia de las variables en las predicciones de modelo final. En base a estos métodos, se desarrolla un perfil de riesgo del portfolio con las características más relevantes de los clientes que lo conforman.

Keywords: Inteligencia artificial, *machine learning*, *deep learning*, *random forest*, *extreme gradient boosting*, redes neuronales, probabilidad de *default*, perfil de riesgo.

CONTENIDO

1 Introducción 12

1.1. Estructura 13

1.2. Motivación 14

2 Técnicas de Estimación 17

2.1. Modelo Logit 17

2.2. Random Forest 20

2.3. XGBoost 22

2.4. Redes Neuronales 24

3 Exploración de la muestra 28

3.1. Data Availability (DA) 28

3.2. Revisión de las variables 30

3.3. Data Quality (DQ) 31

3.3.1. IV y WoE 36

3.3.2. Coeficientes de Correlación 38

4 Modelización 41

4.1. Flag de *Default* 41

4.2. Training y Testing Samples 42

4.3. Modelo Logit 44

4.4. Random Forest 49

4.5. XGBoost 54

4.6. Red Neuronal 57

4.6.1. Red neuronal de propagación resiliente *(Rprop+)* 57

4.6.2. Red neuronal recurrente con unidades GRU (CNN) 61

4.6.3. Comparativa de muestras balanceadas y sin balancear 64

5 Validación 66
5.1. Resultados sobre muestra de prueba *in sample* 66
5.1.1. Comparación Random Forest y XGBoost 68
5.2. Resultados sobre muestra *Out of Sample* 72
6 Explicabilidad del modelo RF y Perfil de Riesgo de la Cartera 74
6.1. Impureza de Gini 74
6.2. Valores Shapley 77
7 Conclusiones 80
7.1. Futuros campos de investigación 81
Bibliografía 83
Anexo A: Comparación predicciones sobre muestra sin balanceo y con balanceo 86
- Modelo Logit 86
- Random Forest 86
- Gradient Boosting 87
- Red neuronal (Rprop+) 87
- Red neuronal (CNN) 88
Anexo B: Tabla de Abreviaturas 89
Anexo C: Código 90

ÍNDICE DE FIGURAS

Ilustración 1 Fuente: Herramienta de IA GPT: Linear vs Logistic Regression..... 19
Ilustración 2 Fuente: Herramienta de IA GPT. Árbol de decisión simple 21
Ilustración 3 Elaboración propia: Distribución de la variable Score original. 33
Ilustración 4 Elaboración propia: Distribución de la variable Score tras eliminación de outliers. 34
Ilustración 5 Elaboración propia: Matriz de correlación entre las variables de la muestra. 39
Ilustración 6 Elaboración propia: Accuracy y Kappa en función de los hiperparámetros del modelo de random forest. 52
Ilustración 7 Elaboración propia: Tasa de verdaderos positivos y verdaderos negativos del modelo random forest sin / con balanceo. 54
Ilustración 8 Elaboración propia: Tasa de verdaderos positivos y verdaderos negativos del modelo XGBoost con / sin balanceo. 57
Ilustración 9: Fuente: Wikipedia. Función Sigmoide. 60
Ilustración 10 Elaboración propia: Tasa de verdaderos positivos y verdaderos negativos RProp+ con / sin balanceo 61
Ilustración 11 Elaboración propia: Pérdida y Accuracy durante el entrenamiento de la red neuronal CNN. 63
Ilustración 12 Elaboración propia: Tasa de verdaderos positivos y verdaderos negativos del modelo de red neuronal CNN. 64
Ilustración 13 Elaboración propia: Tasa de verdaderos positivos de todos los modelos en conjunto. 65
Ilustración 14 Elaboración propia: Tasa de verdaderos negativos de todos los modelos en conjunto. 65
Ilustración 15 Elaboración propia: Comparación de RF y XGBoost de especificidad, F1, precisión y sensibilidad. 68
Ilustración 16: Elaboración propia: Curva ROC XGBoost 69
Ilustración 17 Elaboración propia: Curva ROC Random Forest. 70
Ilustración 18 Elaboración propia: AUC diferenciando entre defaults y no defaults de XGBoost y Random Forest. 70

Ilustración 19 Elaboración propia: KS y Gini para XGBoost y Random Forest..... 71
Ilustración 20 Elaboración propia: Curva ROC de defaults y no defaults de XGBoost y Random Forest en la validación de la muestra out of sample. 72
Ilustración 21 Elaboración propia: Métricas de validación de la muestra out of sample de los modelos Random Forest y XGBoost. 73
Ilustración 22 Elaboración propia: Importancia de las variables en el modelo Random Forest según la impureza Gini. 75
Ilustración 23 Elaboración propia: Valores de Shapley del modelo Random Forest. 78

ÍNDICE DE TABLAS

Tabla 1 Elaboración propia: Análisis de muestra de entrenamiento total 28
Tabla 2 Elaboración propia: Cantidad de missings en la muestra train..... 32
Tabla 3 Elaboración propia: Análisis descriptivo de las variables. 35
Tabla 4 Elaboración propia: Information value de las variables. 37
Tabla 5 Elaboración propia: Análisis muestras train y test. 42
Tabla 6 Elaboración propia: Descripción de la muestra balanceada. 43
Tabla 7 Elaboración propia: VIF variables en el modelo logit..... 45
Tabla 8 Elaboración propia: Residuos primer modelo logit sin balanceo. 46
Tabla 9 Elaboración propia: AIC modelos logit sin eliminación de residuos extremos y con eliminación de residuos extremos. 46
Tabla 10 Elaboración propia: Modelo logit sin balanceo y sin eliminación de residuos extremos. 47
Tabla 11 Elaboración propia: Residuos del modelo con balanceo antes de la eliminación de residuos extremos. 48
Tabla 12 Elaboración propia: Modelo logit con datos balanceados y eliminación de residuos extremos. 48
Tabla 13 Elaboración propia: Matrices de confusión de las predicciones de los modelos logit sin balanceo y con balanceo. 49
Tabla 14 Elaboración propia: Parámetros probados en el entrenamiento del modelo Random Forest sin balanceo. 51
Tabla 15 Elaboración propia: Modelo final Random Forest. 53
Tabla 16 Elaboración propia: Resultados del modelo XGBoost. 56
Tabla 17 Elaboración propia: Código de creación red neuronal RProp+ 59
Tabla 18 Elaboración propia: Matrices de confusión de los modelos finales escogidos. 67
Tabla 19 Elaboración propia: Métricas de evaluación de los modelos. Precisión, sensibilidad, especificidad y F1 Score. 67
Tabla 20 Elaboración propia: Matrices de confusión de la validación sobre muestra out of sample de XGBoost y Random Forest. 72

ÍNDICE DE ECUACIONES

Ecuación 1 : Modelo logit 17
Ecuación 2: Máxima Verosimilitud..... 19
Ecuación 3: Fórmula Entropía..... 21
Ecuación 4: Tasa de aprendizaje de Gradient Boosting. 23
Ecuación 5: Perceptrón en Red Neuronal..... 25
Ecuación 6: Weight of Evidence. 36
Ecuación 7: Information Value 36
Ecuación 8: Coeficiente de correlación entre dos variables. 38
Ecuación 9:Generalized Linear Models. Link: logit. 44
Ecuación 10: Logit Probability..... 45
Ecuación 11: Kolmogorov Smirnov. 71
Ecuación 12: Gini. 71
Ecuación 13: Impureza de Gini 75
Ecuación 14: Valores de Shapley 77

1 INTRODUCCIÓN

El riesgo de crédito es uno de los más relevantes a tener en cuenta por las entidades financieras. Se trata de la probabilidad de que la contraparte no pueda hacer frente a sus obligaciones financieras en el futuro. Hay varias métricas dentro de este ámbito; se puede medir, por un lado, la probabilidad de impago (*default*). Por otro, la severidad (LGD), que mide la cantidad que no se espera recuperar en caso de haber incurrido en *default* o la exposición (EAD), que se refiere a la cantidad total expuesta en el momento que ocurre el default. Estos componentes críticos no solo son esenciales para comprender y gestionar el riesgo de crédito de manera efectiva, sino que también son pilares clave en la evaluación de la solidez financiera de instituciones y otros prestamistas.

A lo largo de los años, hemos presenciado avances significativos en los métodos de cálculo del riesgo de crédito. Los modelos y enfoques han evolucionado, impulsados por la creciente disponibilidad de datos, los avances en la tecnología y la sofisticación de las técnicas analíticas. Además, gracias a la inteligencia artificial, estas técnicas de estimación están cada vez más al alcance de los usuarios, ya sea por la sencillez de comprensión teórica que nos ofrece o por las mejoras en el costo computacional.
Sin embargo, a pesar de estos avances, la implementación total de estos métodos más avanzados aún no se ha generalizado en todas las entidades financieras. La complejidad de los nuevos modelos, junto con los desafíos en la recopilación y gestión de datos, así como las demandas regulatorias, han sido obstáculos importantes para su adopción generalizada. Por ello, las entidades financieras se enfrentan al desafío de equilibrar la precisión y la complejidad en sus modelos de gestión de riesgos. Si bien los métodos más avanzados pueden ofrecer una mejor comprensión del riesgo de crédito y, en última instancia, una gestión más efectiva del mismo, también pueden requerir recursos considerables en términos de tiempo, inversión y experiencia técnica.

Por tanto, aunque hemos hecho grandes avances en el cálculo de los parámetros clave del riesgo de crédito, aún queda trabajo por hacer para que estos avances se implementen en la práctica. Las entidades financieras continúan enfrentándose al desafío de encontrar el equilibrio adecuado entre la precisión y la complejidad en la gestión del riesgo, mientras se enfrentan también a medidas regulatorias y tecnológicas. Por esta razón, es de gran

importancia tratar de evaluar las nuevas metodologías de estimación de los parámetros y compararlas con otras más sencillas y usadas para valorar si ofrecen en la realidad mejoras significativas.

Los principales resultados obtenidos son: (i) el método de random forest es el que mejor se ajusta a la cartera estudiada, seguido de XGBoost, (ii) las técnicas de balanceo de datos arrojan resultados muy mejorados sobre las predicciones de los modelos, (iii) según la impureza de Gini y los valores Shapley, el número de refinanciaciones, el máximo número de días de impago y la media de líneas de activo del cliente son las variables que aportan mayor importancia al modelo final y las que mayor explicabilidad muestran.

1.1. Estructura

El presente trabajo se organiza en una serie de apartados diseñados para proporcionar una comprensión de los métodos utilizados a nivel teórico, el análisis de la base de datos, el proceso de entrenamiento de los modelos, su validación, una propuesta de perfil de riesgo de la cartera y las conclusiones.

1. En la primera parte, se explica en profundidad cada uno de los métodos de aprendizaje automático utilizados: regresión logística, random forest, extreme gradient boosting y redes neuronales (*RProp+ y CNN*). Se discuten las características y ventajas de cada método, especificando los parámetros que se deben establecer para poder llevar a cabo cada técnica.

2. La segunda parte, presenta un análisis detallado de la base de datos utilizada en el estudio, incluyendo una descripción de las variables, técnicas de preprocesamiento y estrategias de separación (entrenamiento y prueba) y balanceo de la muestra. Esta sección proporciona el fundamento necesario para comprender el contexto en el que se desarrollan los modelos de predicción del incumplimiento.

3. Posteriormente, se aborda el proceso de entrenamiento de los modelos, donde se aplican las técnicas previamente mencionadas utilizando los conjuntos de datos preparados en la sección anterior. Se describe el proceso paso a paso, desde la

preparación de los datos hasta la optimización de los modelos a través de la validación cruzada.

4. Una vez entrenados los modelos, se lleva a cabo su validación mediante backtesting, donde se evalúa su desempeño en la muestra de prueba formada a partir de la de desarrollo (*in sample*) y la muestra *out of sample*. Se analizan los resultados obtenidos y se discuten las implicaciones prácticas de los modelos en la predicción del riesgo de *default* en la cartera bancaria.

5. En la siguiente sección, para tratar de abordar el problema de interpretabilidad del modelo de machine learning, se realiza un perfil de riesgo de la cartera en base al método que mejores resultados presenta; Random Forest, a través de la impureza de Gini y los valores de Shapley. El objetivo de esto es presentar aquellas características o factores más relevantes de los clientes a la hora de tomar decisiones sobre el modelo.

6. Finalmente, se presentan las conclusiones del trabajo, basadas en los hallazgos del estudio y en la evaluación de los modelos. Se resumen las ventajas y limitaciones de cada método, se ofrecen recomendaciones para futuras investigaciones y se destacan las implicaciones prácticas de los resultados obtenidos.

1.2. Motivación

El propósito del estudio es comprender el potencial de las técnicas de aprendizaje automático y deep learning aplicadas al riesgo de crédito, para determinar si podrían ser beneficiosas en la predicción de la probabilidad de incumplimiento en una cartera bancaria actual. Estos modelos ofrecen una perspectiva innovadora al no imponer hipótesis estadísticas sobre la forma funcional de los datos, lo que permite capturar relaciones complejas que podrían pasar desapercibidas en enfoques tradicionales. Por este motivo, una de las principales ventajas de los modelos de aprendizaje automático es su capacidad para adaptarse a la estructura de los datos sin requerir suposiciones específicas sobre la distribución subyacente.

Es importante resaltar que el objetivo principal de este estudio no se limita únicamente a determinar cuál de las técnicas de predicción es la más eficaz para la cartera en cuestión, sino que también busca evaluar las técnicas de balanceo y remuestreo de la información. Este enfoque tiene como propósito entender, no solo la capacidad predictiva de los distintos modelos, sino también la influencia que tienen las estrategias de reprocesamiento de datos en la calidad de las predicciones. Se pretende explorar si estas técnicas adicionales pueden ofrecer mejoras significativas en la práctica, permitiendo así una comprensión más completa del proceso de modelado y predicción del riesgo.

Además, también se tiene como propósito explorar los diversos paquetes disponibles en RStudio para el desarrollo de estas técnicas de inteligencia artificial y aprendizaje profundo. Al analizar diferentes herramientas y bibliotecas de software, se busca identificar aquellas que sean más eficientes y adaptables a las necesidades específicas del estudio. Por lo tanto, el fin será desarrollar un modelo que asigne una probabilidad de incumplimiento a los clientes que constituyen la cartera que vamos a estudiar. Tras el cálculo de esta probabilidad utilizando los modelos, se podría establecer un seguimiento del riesgo de cada uno de los clientes y así evitar situaciones de insuficiencia financiera de nuestra compañía, además de cumplir con las políticas regulatorias. Esto nos ayuda a crear un perfil de riesgo extremadamente útil para la empresa.

Dicho perfil de riesgo se crea gracias a los métodos de impureza de Gini y Shapley que tratan de suplir la falta de interpretabilidad o explicabilidad de los modos de aprendizaje automático. Otro de los objetivos del estudio es, por tanto, tratar de implementar estás técnicas sobre la cartera de clientes

En la fase final de evaluación de nuestros modelos de aprendizaje automático, adoptaremos una serie de métricas estadísticas para cuantificar su desempeño. La matriz de confusión, la precisión, la sensibilidad o la robustez. Además, integraremos otras métricas relevantes como el valor F1, que ofrece un balance entre precisión y sensibilidad. Esto es otro de los objetivos de este estudio; comprender la importancia de cada uno de estos indicadores utilizados en la validación de modelos estadísticos.

Por último, cabe destacar que otra de las motivaciones del estudio es comprender la regulación actual en la que nos encontramos, IFRS9, a la hora de validar modelos y

calcular el riesgo de crédito por la entidad. Este pilar es fundamental para desarrollar modelos de estimación y perfiles de riesgo correctamente ya que establece las directrices a seguir en cada uno de los apartados.

2 TÉCNICAS DE ESTIMACIÓN

2.1. Modelo Logit

Un modelo logit, conocido, también como regresión logística, es un tipo de análisis de regresión utilizado para predecir el resultado de una variable categórica basada en una o más variables independientes. Lo que lo hace especial es que la variable dependiente que queremos predecir es binaria, es decir, solo tiene dos posibles resultados.

La función principal de un modelo logit es calcular la probabilidad de que ocurra un evento específico. Esto se hace a través de la función logística, que es una curva en forma de "S". Esta curva puede tomar cualquier valor numérico de entrada y mapearlo en un valor entre 0 y 1, lo cual es ideal para representar probabilidades.

La ecuación del modelo logit es la siguiente:

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k$$

Ecuación 1 : Modelo logit

Donde:

- p es la probabilidad de que ocurra el evento de interés.
- $\frac{p}{1-p}$ es el odds, o la razón de probabilidades, que compara la probabilidad de que ocurra el evento contra la probabilidad de que no ocurra.
- log es el logaritmo natural.
- β_0 es la intersección o sesgo.
- $\beta_1, \beta_2, \ldots \beta_k$ son los coeficientes de las variables.
- $X_1, X_2, \ldots X_k$ son las variables independientes.

La regresión logística es ampliamente utilizada en muchos campos de estudio porque, además de proporcionar estos coeficientes que informan sobre la relación entre las variables independientes y la probabilidad del evento, también ofrece la facilidad de interpretar estos coeficientes como *odds ratios* después de *exponentiarlos*, lo que puede ser muy informativo en términos de entender el impacto relativo de cada predictor.

En términos prácticos, el modelo logit es muy útil para tareas como la clasificación de crédito, diagnósticos médicos y cualquier otro ámbito donde se necesite clasificar resultados en dos grupos distintos basados en un conjunto de características observables. La sencillez del modelo logit es, de hecho, una de las razones principales por las que es el método preferido por muchas entidades para tareas de predicción, en campos como la banca, la medicina y la investigación social. Tal y como se menciona, esta preferencia se debe a varias características fundamentales que hacen al modelo logit particularmente atractivo:

1. Facilidad de interpretación: Los coeficientes del modelo logit pueden ser interpretados directamente como odds ratios después de una transformación exponencial, lo que permite una comprensión clara y directa de cómo las variables independientes afectan la probabilidad de un evento.
2. Modelado de probabilidades: A diferencia de otros modelos que pueden predecir valores numéricos sin restricciones, el modelo logit proporciona una salida en términos de probabilidad, garantizando que los resultados se mantengan dentro del rango de 0 a 1, lo cual es coherente con la naturaleza de las probabilidades.
3. Flexibilidad y robustez: A pesar de su simplicidad, el modelo logit es flexible en el manejo de relaciones no lineales y puede incluir interacciones entre variables. También es relativamente robusto a la variación en las mediciones de las variables independientes.
4. Eficacia en tamaños de muestra moderados: Los modelos logit son eficientes en términos de información y pueden proporcionar estimaciones fiables incluso cuando los tamaños de muestra no son extremadamente grandes.
5. Fundamento teórico sólido: La regresión logística tiene un fundamento teórico bien establecido en la teoría de la probabilidad y la estadística, lo que asegura su adecuación para el análisis inferencial.

En definitiva, a diferencia del método de modelización más conocido; la regresión lineal, este tipo de modelos nos dará resultados comprendidos entre cero y uno. Para el caso de querer predecir un evento de default es muy adecuado ya que debemos manejar la probabilidad de que ocurra (1) o no (0) el evento.

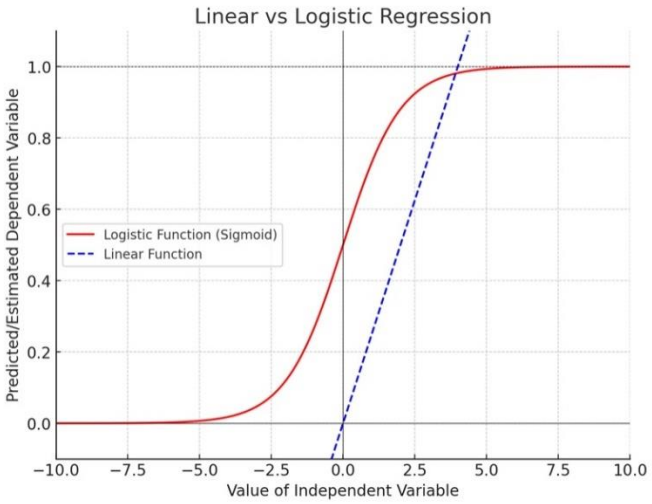


Ilustración 1 Fuente: Herramienta de IA GPT: Linear vs Logistic Regression

Como se puede ver en la Ilustración 1, la regresión logística sigue una curva en forma de S (sigmoidal) que es aproximadamente lineal en el centro, pero se curva hacia los extremos, asegurándose de que la salida se mantenga entre cero y uno, lo que es ideal para representar probabilidades. En contraste, la regresión lineal sigue una línea recta sin restricciones de valor, lo que puede llevar a predicciones que no se limitan al rango [0, 1].

La estimación de los coeficientes β del modelo logístico se realiza generalmente mediante el método de máxima verosimilitud. Este método busca encontrar los valores de los coeficientes que maximizan la función de verosimilitud, que es una medida de cuán probable es observar los datos dados los parámetros del modelo.

La función de verosimilitud L para un modelo de regresión logística con N observaciones es:

$$L(\beta|y) = \prod_{k=1}^{N} \frac{n_k!}{y_k!\,(n_k - y_k)!} P_k^{y_k} (1 - P_k)^{n_k - y_k}$$

Ecuación 2: Máxima Verosimilitud.

Los valores de los coeficientes β que maximizan la función de verosimilitud son conocidos como estimaciones de máxima verosimilitud. Estos valores se encuentran en los puntos críticos donde la primera derivada de la función se iguala a cero. Además, para que un punto crítico sea considerado un máximo, la segunda derivada en dicho punto debe ser negativa. Debido a la naturaleza compleja de las derivadas en estas funciones, el proceso de encontrar estas estimaciones puede ser bastante costoso computacionalmente.

Reducir la complejidad de la estimación de máxima verosimilitud en modelos como la regresión logística se logra comúnmente utilizando métodos iterativos. Estos métodos optimizan la función de verosimilitud paso a paso, evitando la necesidad de calcular derivadas de segundo orden completas, lo cual puede ser computacionalmente demandante, especialmente en modelos con muchas variables. Uno de los métodos más utilizados es el algoritmo de optimización iterativamente reponderada de cuadrados mínimos (IRLS), que se utiliza frecuentemente en la regresión logística.

2.2. Random Forest

El método de Random Forest es una técnica de aprendizaje de conjunto que construye y combina múltiples árboles de decisión para mejorar la precisión y la robustez de las predicciones. En el contexto de la predicción de una variable, sea esta continua (regresión) o categórica (clasificación), Random Forest aborda dos problemas principales de los árboles de decisión individuales: la varianza y el sobreajuste. Cada árbol se construye utilizando una muestra aleatoria con reemplazo de los datos de entrenamiento, conocido como *'bootstrap sample'*, y al hacer la predicción, los resultados de todos los árboles se agrupan para producir una única salida consolidada. La fuerza del Random Forest radica en la diversidad de sus árboles constituyentes. Durante la construcción de cada árbol, una submuestra aleatoria de las características es seleccionada en cada división, lo que aumenta la variabilidad entre los árboles y reduce la correlación entre ellos. Esto se traduce en un modelo global más generalizado y con mejor capacidad de generalización a datos no vistos.

Para evaluar la importancia de las variables en Random Forest, se pueden utilizar métricas como la disminución media de impureza, que mide la contribución de cada variable a la homogeneidad de los nodos y ramas en los árboles, o el permiso de características, que calcula el cambio en el rendimiento del modelo cuando los valores de una variable son aleatorizados. Estos métodos ayudan a comprender cuáles características influyen más en la predicción y ofrecen intuiciones sobre la relación entre las variables y la variable objetivo. La precisión de Random Forest se evalúa a través del "*out-of-bag*" (OOB) error, que es la media de los errores de predicción para cada observación calculada usando solo los árboles que no tuvieron esa observación en su muestra bootstrap. El error OOB es una

estimación interna de la validación cruzada y proporciona una métrica fiable de la precisión del modelo sin la necesidad de un conjunto de prueba separado.

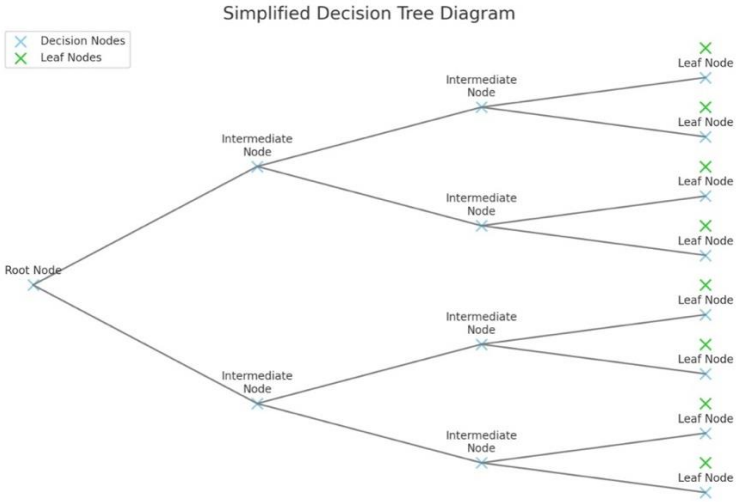


Ilustración 2 Fuente: Herramienta de IA GPT. Árbol de decisión simple

En la Ilustración 2, se puede ver cómo un árbol de decisión divide los datos en conjuntos cada vez más pequeños basándose en ciertas decisiones, comenzando en la raíz (el nodo superior) y moviéndose hacia abajo a través de los nodos intermedios hasta llegar a las hojas, donde se toma la decisión final. En un Random Forest, hay que imaginar que se tiene un conjunto de estos árboles que trabajan juntos, donde cada árbol tiene su propia muestra aleatoria de los datos y considera un subconjunto aleatorio de variables en cada división. La predicción final se hace tomando el promedio o la mayoría de los votos entre todos los árboles. Este enfoque colaborativo es lo que permite al Random Forest ser más preciso y menos propenso al sobreajuste que un solo árbol de decisión.

De forma simplificada, el proceso de creación de un árbol de decisión es el siguiente. En primer lugar, se divide la información en base a algunos criterios que maximizan la homogeneidad del resultado. En clasificación, un nodo puede ser dividido en base a la ganancia de información, el índice Gini, o la reducción de la impureza. En regresión, se utilizan métodos como la reducción de la varianza.

La ganancia de información mide el cambio en la entropía desde antes a después de la división de un conjunto de datos basado en un atributo dado. La entropía es una medida de la incertidumbre o impureza y se calcula:

$$H(S) = -\sum_{x \in X} p(x) \log_2 p(x)$$

Ecuación 3: Fórmula Entropía.

.

Donde $p(x)$ es la proporción de la clase x en el conjunto S.

En cuanto al índice Gini, este mide la impureza de un nodo. Un nodo es "puro" (Gini = 0) si todos los casos en el nodo pertenecen a una sola clase. Se calcula como $Gini(S) = 1 - \sum_{x \in X}[p(x)]^2$.

Para el caso de los árboles de regresión, se utiliza la reducción de la varianza. Esto consiste en seleccionar las divisiones que resultan en la mayor reducción posible en la varianza de los datos en los nodos resultantes.

La evaluación de un árbol de decisión se realiza generalmente usando métricas como la precisión, la matriz de confusión, el área bajo la curva ROC, entre otros. Además, los métodos de validación cruzada se utilizan para garantizar que el modelo generalice bien a datos no vistos.

Esto conduce al algoritmo en mayor escala de RF, que selecciona varias muestras por Bootstrap / remuestreo con reemplazamiento tal que $S_n^{\Theta_1}, \dots, S_n^k$ para implementar el algoritmo individual del árbol de decisión, obteniendo un número *K* de regresores tal que $\hat{h}\left(X, S_n^{\Theta_1}\right), \dots, \hat{h}(X, S_n^{\Theta_k})$. La colección de regresores produce *K* predicciones relativos a cada árbol, $\hat{Y}_1 = \hat{h}\left(X, S_n^{\Theta_1}\right), \dots, \hat{Y}_k = \hat{h}\left(X, S_n^{\Theta_k}\right)$. Por último, se lleva a cabo una media de los resultados de cada uno de los árboles obtenidos, resultando en una estimación de *Y* tal que $\hat{Y} = \frac{1}{K}\sum_{k=1}^{K} \hat{h}(X, S_n^k)$. Donde $\hat{Y}$ es la predicción de un bosque aleatorio de cantidad k-arboles.

La realización de un modelo de Random Forest puede ser computacionalmente intensa, especialmente en comparación con otros métodos de aprendizaje automático más simples como la regresión logística o los árboles de decisión individuales. La cantidad de esfuerzo computacional requerido depende del número de árboles, la profundidad de los mismos, el número de características y el tamaño del conjunto de datos.

2.3. XGBoost

El *Gradient Boosting* es una técnica de aprendizaje automático para problemas de regresión y clasificación que produce un modelo de predicción en forma de un ensamblaje de modelos predictivos débiles, típicamente árboles de decisión. Opera construyendo el modelo en etapas, y generaliza los modelos de *boosting* permitiendo la optimización de una función de pérdida diferenciable arbitraria.

De manera más sencilla y resumida, comienza con un modelo simple (un árbol de decisión, por ejemplo) y calcula sus predicciones. Después, se calculan los residuos (errores de predicción) del modelo actual y se construye un nuevo modelo que predice estos residuos. Este proceso se asemeja a un paso en el método de descenso del gradiente, donde el "gradiente" en este caso son los residuos del modelo actual. Este nuevo modelo se suma al modelo existente para mejorar las predicciones y el proceso se repite varias veces. Finalmente, cada nuevo modelo se agrega al ensamblaje con una tasa de aprendizaje (*learning rate*), que pondera la contribución de cada modelo y puede ayudar a evitar el sobreajuste.

El fundamento matemático del Gradient Boosting puede resumirse en la optimización iterativa de una función de pérdida. Supongamos que tenemos una función de pérdida tal que $L(y, f(x))$ que mide el error entre la predicción del modelo $f(x)$ y el verdadero valor de la salida y. El objetivo es encontrar un modelo $f(x)$ que minimice la función de pérdida para los datos de entrenamiento.

Se comienza con un modelo inicial $f_0(x)$, que puede ser tan simple como el promedio de salidas (en una regresión) o un modelo de log-odds de clasificación. Después, de entra en un ciclo donde se mejora el modelo inicial iterativamente. En el paso t, se calculan los residuos (gradientes negativos de la función de pérdida) para cada instancia de datos tal que $r_{ti} = \left[\frac{\delta L(y_i, f(x_i))}{\delta f(x_i)}\right]_{f(x)=f_{t-1}(x)}$. A continuación, se construye un árbol $h_t(x)$ para predecir residuos de la iteración actual. Para cada hoja del árbol, determinas el valor óptimo del multiplicador (también conocido como la tasa de aprendizaje) γ_{tj} que minimiza la función de perdida cuando se aplica a la *j-ésima* región de la hoja del árbol:

$$\gamma_{tj} = \arg\min_{\gamma} \sum_{x_i \in R_{tj}} L\left(y_i, f_{t-1}(x_i) + \gamma h_t(x_i)\right)$$

Ecuación 4: Tasa de aprendizaje de Gradient Boosting.

.

Se actualiza el modelo sumando este nuevo árbol ponderado por un factor de escala ϑ (learning rate): $f_t(x) = f_{t-1}(x) + \vartheta \sum_{j=1} \gamma_{tj} I\left(x \in R_{tj}\right)$. Donde I es la función indicadora que vale 1 si x cae en la región *j* de la hoja del árbol *t* y 0 en caso contrario. Finalmente, se repiten los pasos para un número predefinido de iteraciones o hasta que la mejora en la función de pérdida sea menor a un umbral determinado.

En este caso, la técnica que se va a llevar a cabo se conoce como XGBoost, abreviatura de "*Extreme Gradient Boosting*," es una implementación avanzada de la técnica de Gradient Boosting. XGBoost mejora este enfoque introduciendo técnicas de regularización para prevenir el sobreajuste, una gestión eficiente de memoria y capacidad de paralelización, así como algoritmos específicos para manejar datos faltantes y reducir la varianza, resultando en un rendimiento más rápido y preciso en tareas de clasificación y regresión.

2.4. Redes Neuronales

Una red neuronal, en el contexto de la IA, es una estructura computacional inspirada en las redes de neuronas biológicas de los cerebros animales, de ahí proviene su nombre. Está diseñada para simular la manera en que el cerebro humano procesa la información. Funciona interconectando un gran número de nodos, o "neuronas artificiales", en capas, donde cada nodo recibe y procesa señales y, luego, pasa su salida a otros nodos.
La red se compone generalmente de una capa de entrada, donde se reciben los datos; una o más capas ocultas, donde se realiza el procesamiento complejo; y una capa de salida, donde se concluye el resultado. Las neuronas en estas capas están conectadas por "pesos", que son parámetros ajustables que se calibran durante un proceso de entrenamiento.
Durante el entrenamiento, la red neuronal es alimentada con grandes cantidades de datos y las respuestas correctas (en un proceso supervisado), y la red ajusta los pesos de las conexiones para predecir la respuesta más acertadamente. A través de un proceso iterativo conocido como "propagación hacia atrás" y la optimización de una función de coste, la red mejora su capacidad para hacer predicciones o clasificaciones precisas.

Existen varios tipos de redes neuronales, cada una con su propia arquitectura y aplicaciones específicas. Algunos de los tipos más comunes son, por un lado, redes neuronales densas o completamente conectadas (*Fully Connected Neural Networks*); todas las neuronas en una capa están conectadas a todas las neuronas en la siguiente capa. Son utilizadas para patrones de datos generalizados donde no se requiere detectar relaciones espaciales o temporales. Por otro lado, redes neuronales convolucionales (Convolutional Neural Networks, CNNs); especializadas para procesar datos que tienen una estructura en cuadrícula, como imágenes. Utilizan una técnica matemática llamada convolución para detectar patrones y características espaciales. En tercer lugar, redes neuronales recurrentes (Recurrent Neural Networks, RNNs); diseñadas para trabajar con secuencias de datos, como el lenguaje natural o series temporales. Son capaces de mantener información en el tiempo gracias a sus conexiones en bucle, lo que les permite tener una especie de "memoria". Por último, unidades de memoria a corto plazo (Long Short-Term Memory, LSTM): son un tipo especializado de RNN capaz de aprender dependencias a largo plazo.

La red neuronal más sencilla es, a menudo, la red neuronal de una sola capa sin capas ocultas, conocida también como perceptrón. La matemática detrás de un perceptrón es relativamente simple y sirve como base para estructuras más complejas en el aprendizaje profundo. El funcionamiento de un perceptrón puede describirse:

$$y = f\left(\sum_{i=1}^{n}(w_i * x_i) + b\right)$$

Ecuación 5: Perceptrón en Red Neuronal.

Donde:

- x_i representa la entrada de la red neuronal.
- w_i son los pesos asignados a cada entrada de la red neuronal.
- b es el sesgo (*bias*), que proporciona un ajuste adicional a la salida.
- $\sum_{i=1}^{n}$ es el sumatorio que proporciona la suma ponderada de las entradas y sus pesos.
- f es la función de activación que se aplica a la suma ponderada para obtener la salida final y.

La función de activación f es lo que permite a las redes neuronales modelar relaciones no lineales. Para los perceptrones, una función de activación común es la función escalón, que devuelve 1 si la suma ponderada es mayor que un cierto umbral y 0 en caso contrario. Sin embargo, en prácticas modernas, se utilizan funciones de activación más sofisticadas como la función sigmoide, la tangente hiperbólica o la función de unidad lineal rectificada (ReLU), que permiten un aprendizaje más efectivo y son capaces de tratar con problemas de regresión y clasificación más complejos.

Para un conjunto de entradas X y una salida Y, el proceso de entrenamiento de la red busca encontrar los valores óptimos de los pesos W y el sesgo b que minimizan alguna función de pérdida, que mide el error entre las salidas predichas por la red y las salidas reales. Este proceso generalmente se realiza a través de un algoritmo llamado descenso del gradiente, que ajusta los pesos en la dirección que más reduce el error.

En redes neuronales más complejas con múltiples capas ocultas, este proceso se repite en cada capa, propagando las entradas a través de la red, lo cual le permite aprender características más abstractas a medida que se avanza en la profundidad de la misma. Cada capa captura un nivel diferente de abstracción del dato de entrada, lo que permite a la red resolver tareas complejas de clasificación y regresión. Es evidente que, a mayores capas, mayor esfuerzo computacional se requiere. Esto es una de las limitaciones de este tipo de procesos, además de la falta de interpretabilidad que presentan cuanto mayor es la complejidad.

En la primera etapa del estudio, se realiza el entrenamiento de una red neuronal mediante el método de propagación resiliente, específicamente *Rprop+* (*Resilient Propagation Plus*), utilizando el paquete *NeuralNet*. Este método de estimación es conocido por su eficacia en la adaptación de los pesos de la red, ajustándose solo por la dirección del gradiente de la función de error y no por su magnitud. Esto permite que *Rprop+* sea menos susceptible a los problemas derivados de gradientes pequeños o desproporcionadamente grandes, lo que resulta en una convergencia más rápida y estable en comparación con otros métodos de entrenamiento de redes neuronales.

En la segunda etapa, se implementa una red neuronal más compleja, específicamente una red neuronal convolucional (CNN) o una red de propagación recurrente con varias capas

GRU (Gated Recurrent Unit). Estas estructuras se diseñan y entrenan utilizando el paquete *Keras*, una biblioteca avanzada de aprendizaje automático que facilita la creación y el entrenamiento de modelos de aprendizaje profundo y que es una adaptación del lenguaje de programación Python. Las CNN son ampliamente utilizadas para el procesamiento de imágenes, donde pueden capturar eficazmente las jerarquías espaciales de características. Por otro lado, las redes GRU son variantes de las redes neuronales recurrentes (RNN) que mejoran la capacidad de la red para trabajar con secuencias de datos, al permitir que cada unidad tenga un mecanismo de actualización y reinicio, lo que ayuda a capturar dependencias temporales en los datos sin sufrir el problema del desvanecimiento del gradiente. Estos métodos serán explicados con mayor detalle más adelante en el trabajo, destacando su implementación mediante *Keras* y su aplicación práctica en el conjunto de datos específico que se está analizando.

3 EXPLORACIÓN DE LA MUESTRA

3.1. Data Availability (DA)

A la fase en la que se analiza la disponibilidad de los datos en el proceso de desarrollo y validación de un modelo de crédito se le denomina *Data Availability*, con las siglas DA, basándonos en la guía que cumple con la regulación IFRS9. El presente proyecto se sustenta en una base de datos que recopila información financiera y comportamental de clientes de una entidad bancaria específica. Este conjunto de datos abarca registros detallados de préstamos contratados por dichos clientes, con un enfoque en el historial de pagos y el seguimiento de incidentes de impago. El período cubierto por estos datos se extiende desde marzo de 2022 hasta junio de 2022, proporcionando una ventana temporal que refleja las dinámicas crediticias de la cartera de clientes durante ese cuatrimestre.
Asimismo, para evaluar la robustez del modelo y su capacidad predictiva, se dispone de una muestra de prueba correspondiente al período de enero a febrero del año 2023. Esta muestra de prueba es crítica para el estudio, ya que permite observar el comportamiento posterior de los clientes y determinar si han incurrido en default, es decir, si han fallado en el cumplimiento de sus obligaciones crediticias según los términos acordados. Gracias a esta información, podremos evaluar qué tan efectivos son los métodos estadísticos utilizados para la modelización y predicción.

Se procede a analizar en la Tabla 1 cómo se organiza en su totalidad la información sobre defaults del conjunto de entrenamiento del modelo:

	Observaciones
Total	1.994.833
Defaults	55.925
No Defaults	1.938.908
Ratio de Default	2,803%

Tabla 1 Elaboración propia: Análisis de muestra de entrenamiento total

No sorprende que exista una gran desproporción de casos de “No Default” que del contrario. Por la propia naturaleza de la variable, se espera que exista en la cartera un mayor número de clientes buenos que malos. Esto suele ser un problema a la hora de crear

modelos para poder predecir el impago y es este el motivo de que existan ciertas carteras conocidas como "*Low Default Portfolios*" que son estudiadas de manera aislada al resto. Las carteras de incumplimiento bajo (LDP) se refieren a carteras de préstamos con una incidencia de incumplimiento muy baja, que a menudo incluyen préstamos otorgados a grandes empresas y bancos globales. Estas carteras presentan desafíos, principalmente porque la escasez de datos internos sobre incumplimiento dificulta la construcción de modelos de probabilidad de incumplimiento (PD) precisos y predictivos. Dado que estos modelos generalmente se basan en datos históricos para predecir incumplimientos futuros, la falta de casos de incumplimiento pasados conduce a una baja confiabilidad estadística en las predicciones de los modelos.

La selección de variables para el modelo incluyó una amplia gama de características posibles, abarcando datos que van desde información sobre riesgos operativos basada en clasificaciones sistemáticas de contratos, registros de la Central de Información de Riesgos del BdE, comportamiento de cliente para crear un score (basado en score de admisión o de comportamiento), detalles de liquidez y garantías. También se consideró información financiera clave como balances, cuentas de pérdidas y ganancias, y la solvencia en el banco, así como datos relacionados con la facturación y movimientos transaccionales para obtener una visión del perfil de riesgo.

Cabe mencionar que se dispone también de una muestra de prueba para realizar la validación de nuestros modelos a futuro. Esta abarca desde enero a febrero de 2023 y nos proporciona información a futuro de los mismos clientes que la muestra de entrenamiento. Basándonos en el ID del cliente, tenemos un *Flag de Default* de cada uno (formado por 0 / 1) y una PD asociada a cada uno de ellos. Esta muestra posterior al desarrollo la utilizaremos para validar el modelo final y crear el perfil de riesgo de la cartera.

Si no tuviéramos esta muestra de prueba, también seríamos capaces de poder realizar una validación de nuestros modelos separando la muestra con métodos como bootstrap, una técnica estadística de remuestreo con reemplazo sin necesidad de imposición de hipótesis sobre la distribución de los datos. Fue propuesto por Bradley Efron en 1979[1] y se utiliza ampliamente para estimar la varianza y construir intervalos de confianza para

[1] BRADLEY EFRON, Bootstrap confidence intervals for a class of parametric problems, Biometrika, Volume 72, Issue 1, April 1985, Pages 45–58, https://doi.org/10.1093/biomet/72.1.45

estimadores. Utilizaremos esta técnica para obtener una primera muestra de prueba, que será donde evaluaremos en primer lugar cada uno de los métodos para escoger el que mejores resultados presente y, después, probaremos este sobre una muestra un año posterior a la de desarrollo para ver si se ajusta a otro contexto económico al utilizado para el entrenamiento.

3.2. Revisión de las variables

Las variables que se incluyen en un modelo de predicción son la base más importante del mismo y de que este funcione bien y arroje resultados útiles para poder tomar decisiones. Este proceso de análisis implica evaluar detalladamente cada variable para discernir su influencia y relación con otras variables en el estudio. Realizar un análisis exhaustivo de las variables no solo ayuda a garantizar la precisión y relevancia de los modelos predictivos, sino que también facilita la identificación de correlaciones significativas y la eliminación de redundancias que podrían distorsionar los resultados de la investigación. El estudio adecuado de las variables asegura que los modelos desarrollados sean robustos y confiables. Además, permite a los analistas ajustar los modelos a medida que surgen nuevas tendencias o información, manteniendo su relevancia y efectividad.
En este apartado, se pretende explicar detalladamente cada uno de los factores que se ha decidido incluir en la base de datos utilizada como muestra final para el entrenamiento de los modelos. Las variables son las siguientes:

- ID: identificador de cliente.
- Flag_default: nuestra variable objetivo. Marca si el cliente ha incurrido o no en impago (variable construida a partir de ciertos criterios que se explican en el apartado de Flag_de_default).
- Fecha: fecha de la operación.
- Score: puntuación del cliente basado en admisión / comportamiento.
- Max_saldos_descubiertos_12m: máximo de saldos descubiertos en los últimos 12 meses.
- Max_saldos_descubiertos_3m: máximos saldos descubiertos en los últimos 3 meses.
- Max_recursos_totales: máximo de recursos totales en los últimos 12 meses.

- Recibos_devueltos_12m: cantidad de los recibos devueltos en los últimos 12 meses.
- Veces_default_12m: veces que el cliente alcanza los 90 días de retraso del pago en los últimos 12 meses.
- Cuota_amortizar: cantidad que le queda al cliente por amortizar del préstamo.
- Media_lineas_activo: promedio de líneas de activo en los últimos 12 meses.
- Saldo_medio_disponible_3m: saldo medio disponible de los últimos 3 meses.
- Liquidez_cliente: liquidez total del cliente.
- Credito_dispuesto: porcentaje del crédito total dispuesto por el cliente.
- Max_dias_incumplido_3m: máximo de días de default en los últimos 3 meses.
- Max_dias_incumplido_12m: máximo de días de default en los últimos 12 meses.
- Refinanciaciones: cantidad de refinanciaciones totales.

En base a toda esta información sobre los clientes, somos capaces de construir un mapa de probabilidades de default gracias a sus perfiles de riesgo para poder predecir el comportamiento de clientes con características similares. Por ello, el objetivo es concluir qué método de estimación es el más ajustado para nuestra cartera.

3.3. Data Quality (DQ)

La fase de análisis de la calidad de nuestros datos, la denominamos en el proceso de validación como *Data Quality* (DQ). La calidad de los datos debe evaluar que existen procesos y mecanismos de control establecidos para garantizar la calidad de los datos; que comprende su exactitud, coherencia, validez y trazabilidad: ¿Los datos son completos y precisos (es decir, los valores están presentes en los atributos que los requieren y los datos están sustancialmente libre de errores)? ¿Los datos son consistentes a través de los sistemas? ¿Los datos se basan en un sistema de clasificación adecuado y riguroso? ¿Existen controles de calidad de datos suficientes y adecuados para la información almacenada en las bases de datos?

El inconveniente más grande dentro de esta fase suele ser la evaluación de los valores faltantes o *missings* dentro de nuestra muestra. A continuación, podemos observar el número de faltantes de las variables seleccionadas en la muestra final para modelizar:

	Missings	% Missings
Código_persona	0	0,00%
Flag_Default	0	0,00%
Refinanciaciones	0	0,00%
Max_dias_incumplido_12m	914.627	45,85%
Max_dias_incumplido_3m	0	0,00%
Credito_dispuesto	0	0,00%
Liquidez_cliente	5.535	0,28%
Saldo_medio_disponible_3m	5.802	0,29%
Media_lineas_Activo	5.535	0,28%
Cuota_amortizar	693.340	34,76%
Veces_default_12m	0	0,00%
Recibos_devueltos_12m	0	0,00%
Max_recursos_totales	5.535	0,28%
Max_saldos_descubiertos_3m	5.535	0,28%
Max_saldos_descubiertos_12m	5.535	0,28%
Score	43.472	2,18%
Fecha	0	0,00%

Tabla 2 Elaboración propia: Cantidad de missings en la muestra train.

Las variables que más llaman la atención son, en este caso, el máximo días de incumplimiento a 1 año con un 45,85% de missings sobre el total de observaciones, la cuota a amortizar con un 34,76% y el Score con un 2,18%.

Los valores faltantes en la variable de máximo días de incumplimiento a 1 año se deben a que los clientes tienen menos de doce meses en la muestra y no han registrado días en default, lo que resulta en un registro incompleto de días durante el último año y, por tanto, se mantiene como un valor nulo. Por este motivo, se decide sustituir por cero aquellos valores faltantes de esta variable (no existe aún incumplimiento).

En cuanto a la variable de la cuota a amortizar, los valores faltantes corresponden a clientes que no poseen cuotas pendientes de amortización para el próximo mes y tampoco cuentan con registros de amortizaciones anteriores, lo que lleva a que el campo en la base de datos esté vacío y también se decide sustituir por ceros.

La variable de Score, los valores *missing* se atribuyen a clientes que aún no tienen una evaluación asociada por parte de la entidad y por ello se decide sustituir los valores faltantes por la media de scores, para que no afecte al resultado final.

Para las demás variables, la ausencia de datos se asocia a la falta de valores registrados en la base de datos del banco. Sin embargo, representan menos de un 1% del total de observaciones y por ello se decide sustituirlos por su media para cada caso.

Por otro lado, se ha realizado un análisis de *outliers* de las variables de la muestra. Los *outliers* o valores atípicos en una muestra son observaciones que se desvían tanto del resto de datos que levantan sospechas sobre su veracidad o precisión. Estos valores son significativamente diferentes y pueden estar influenciados por variabilidad en la medición o posibles errores experimentales, entre otras causas. En términos estadísticos, un *outlier* es una observación que se encuentra anormalmente lejos de otros valores en una distribución de datos.

Para el presente caso, destaca la variable score. Como podemos observar en la Ilustración 3, el histograma a continuación, hay una proporción de observaciones que presentan valores muy bajos y alejados del resto:

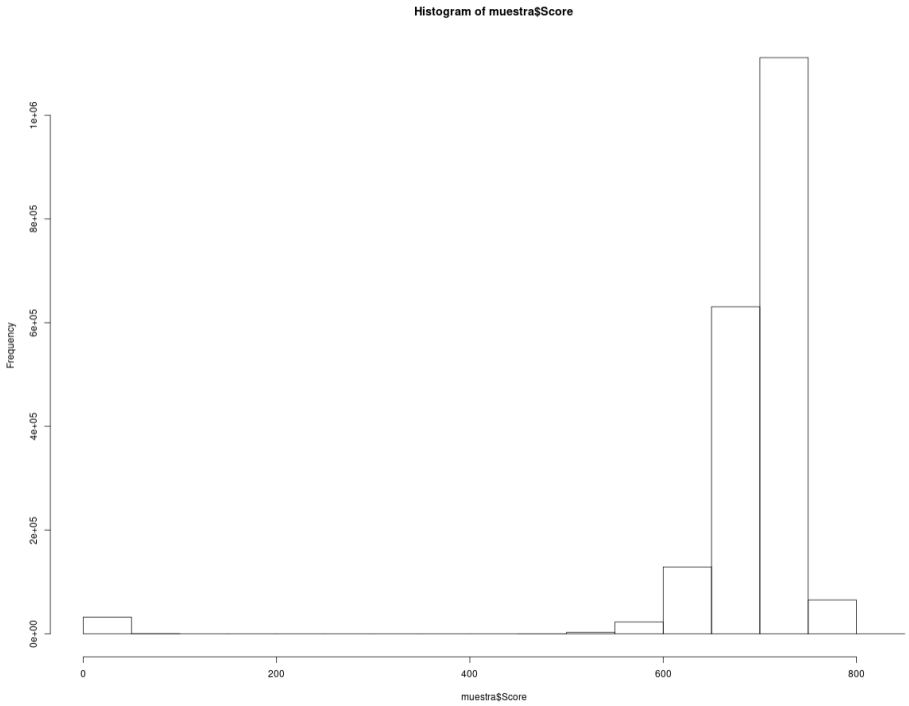


Ilustración 3 Elaboración propia: Distribución de la variable Score original.

Representan 32.800 observaciones de la variable, lo que supone un 1,64% del total. Al ser una proporción tan baja sobre la muestra completa, se decide eliminar estas entradas de la base de datos con el fin de que no alteren los resultados del modelo. Además, se ha observado una distribución de valores con una notable discontinuidad, donde los datos transitan abruptamente de valores inferiores a 50 a valores superiores a 400. Este fenómeno es inusual y sugiere la existencia de potenciales anomalías en los datos. Dicha discontinuidad en la secuencia de valores podría indicar varios escenarios potenciales:

errores en la captura o procesamiento de datos, la presencia de outliers debido a factores externos no controlados o una muestra sesgada que no refleja adecuadamente la población general. El histograma, en la Ilustración 4, tras la substracción de valores atípicos queda de la siguiente forma:

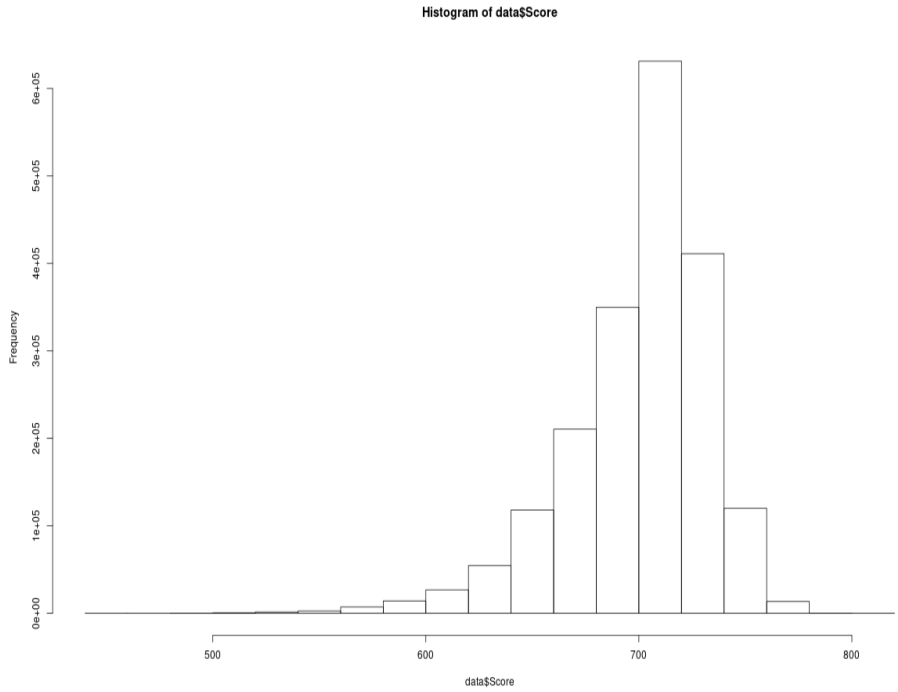


Ilustración 4 Elaboración propia: Distribución de la variable Score tras eliminación de outliers.

Para el resto de las variables, se analizan en la Tabla 3 algunas medidas descriptivas de las mismas.

	Mínimo	**1st Qu.**	**Median**	**Mean**	**3rd Qu.**	**Máximo**
Refinanciaciones	0	0	0	0,04	0	11
Max_dias_incumplido_12m	0	0	0	7,86	8	353
Max_dias_incumplido_3m	0	0	0	3,68	1.000	2.655
Credito_dispuesto	0	0	0	0,19	0,21	1,00
Liquidez_cliente	0	673	6.592	45.292	35.640	35.513.279
Saldo_medio_disponible_3m	0	0	1.536	11.582	7.194	245.975.806

Media_lineas_Activo	-592	6.357	27.378	66.999	76.643	234.003.125
Cuota_amortizar	0	101	390	389	390	139.674
Veces_default_12m	0	0	0	0,031	0	10
Recibos_devueltos_12m	0	0	0	2.113	7.814	1.468.792
Max_recursos_totales	-120.000.000	-11.980	13.620	317.000	260.800	407.300.000
Max_saldos_descubiertos_3m	-7.237.496	0	0	-205	0	5.056.626
Max_saldos_descubiertos_12m	-353.225	0	0	459	0	5.056.626
Score	454	684	708	700	723	813

Tabla 3 Elaboración propia: Análisis descriptivo de las variables.

En cuanto al porcentaje de crédito dispuesto, es interesante destacar que la media ronda el 20% del total y el tercer cuartil el alcanza el 21%. Sin embargo, el salto es notable hasta alcanzar el máximo, que es disponer del 100%. Ocurre algo parecido con otras de las variables: hay una gran diferencia entre el tercer cuartil y el máximo total. No obstante, presentan sentido económico y financiero dentro del contexto en el que nos encontramos.

Por último, dentro de este apartado, se debe mencionar que ha existido también eliminación completa de variables de la base de datos. A continuación, se muestran cuáles son y el motivo:

- Return_of_assets: se trata de una variable que explica el ROA. Este es un indicador financiero comúnmente utilizado para medir la rentabilidad de una empresa en relación con sus activos totales. ROA significa "Retorno sobre los Activos". El ROA es útil para comparar la rentabilidad de empresas en la misma industria, ajustando por diferencias en el tamaño de los activos. En este contexto, al ser una cartera a nivel cliente (cartera *retail*) y no corporate, no aporta información imprescindible. Además, presentaba una proporción de missings alrededor del 40% del total de observaciones.

- Gtos_finan: es una variable sobre información de gastos en los estados financieros. Mostraba características similares a las de la variable anterior.
- Score_enterprise: representa una puntuación o Score por parte del administrador empresarial. De nuevo, se trata de una variable más adecuada en caso de analizar una cartera *corporate* y no *retail*. Por otro lado, esta variable presentaba alta correlación con nuestra variable *Score* (incluida en el modelo) y es el principal motivo por el que se ha decidido prescindir de ella.

3.3.1. IV y WoE

Los criterios de Information Value y Weight of Evidence son métricas que ayudan a evaluar la relevancia predictiva de una variable en relación a otra variable objetivo, como es en nuestro caso la probabilidad de incumplimiento.

Por un lado, El WoE es una técnica utilizada para transformar una variable categórica o no lineal en una forma continua que es más adecuada para la modelización estadística, especialmente en modelos logísticos. Se calcula tomando el logaritmo natural de la división entre la proporción de buenos resultados y la proporción de malos resultados para cada categoría de la variable:

$$WoE = \ln\left(\frac{proporción\ \%\ de\ buenos}{proporción\ \%\ de\ malos}\right)$$

Ecuación 6: Weight of Evidence.

Por otra parte, el IV mide a fuerza predictiva de una variable independiente en relación con la variable dependiente. Se calcula como la suma del producto de la diferencia entre las proporciones de buenos y malos por el WoE de cada categoría:

$$IV = \sum(\%\ buenos - \%\ malos) * WoE$$

Ecuación 7: Information Value

Los valores de IV se interpretan, generalmente:

- IV < 0.02: poca predictividad.
- 0.02 < IV < 0.1: predictividad débil.
- 0.1 < IV < 0.3: predictividad media.
- 0.3 < IV < 0.5: fuerte predictividad.

- IV > 0.5: predictividad muy fuerte o sospechosamente alta (lo cual podría indicar que el modelo está demasiado ajustado a los datos de entrenamiento).

Los resultados de mayor a menor IV son los siguientes:

	IV
Media_lineas_Activo	2,770
Max_dias_incumplido_12m	1,613
Liquidez_cliente	1,535
Score	1,527
refinanciaciones	1,317
Max_dias_incumplido_3m	1,174
saldo_medio_disponible_3m	1,047
Cuota_amortizar	1,010
Recibos_devueltos_12m	0,918
Credito_dispuesto	0,706
Max_saldos_descubiertos_3m	0,523
Max_saldos_descubiertos_12m	0,369
Veces_default_12m	0,312
Max_recursos_totales	0,301

Tabla 4 Elaboración propia: Information value de las variables.

Media_lineas_Activo, Max_dias_incumplido_12m y Liquidez cliente tienen los valores más altos de IV, indicando que son extremadamente predictivas y críticas para el modelo. Esto sugiere que estas variables están muy relacionadas con la variable objetivo y proporcionan una fuerte discriminación entre los resultados. Score, Refinanciaciones y Max_dias_incumplido_3m también muestran una fuerte predictividad, haciendo significativas contribuciones al poder del modelo. Saldo_medio_disponible_3m, Cuota_amortizar, y Recibos_devueltos_12m caen en este rango, sugiriendo que tienen una buena capacidad para predecir el resultado, aunque no tan fuertemente como las variables en el rango más alto. Credito_dispuesto y Max_saldos_descubiertos_3m tienen alta predictividad, aunque menos que las anteriores también ya que se encuentran en el rango entre 0,5 – 0,7. Por último, Max_saldos_descubiertos_12m, Vecen_default_12m y Max_recursos_totales tienen una predictividad relativamente baja en comparación con otras variables. Esto no significa que estas variables sean inútiles, pero su capacidad para influir en la decisión del modelo es limitada en comparación con las más predictivas. En general, son resultados muy buenos que nos indican la necesidad de incluir todas las variables para las estimaciones que se van a realizar a continuación.

3.3.2. Coeficientes de Correlación

La correlación entre variables es una herramienta esencial para entender la fuerza y dirección de las relaciones lineales existentes en un conjunto de datos. Una medida ampliamente reconocida para evaluar la asociación lineal entre dos variables continuas es el coeficiente de correlación de Pearson, desarrollado por el matemático Karl Pearson en 1896, no 1986. Este coeficiente, denotado como ρ (rho), es la estandarización de la covarianza entre dos variables en relación con el producto de sus desviaciones estándar, y se calcula de la siguiente forma: $\rho = \frac{Cov(X,Y)}{\sigma_x \sigma_y}$. Donde $Cov(X,Y)$ es la covarianza entre las variables y σ_X y σ_Y son las desviaciones estándar de X e Y. Para el cálculo basado en muestras, el coeficiente r_{XY} se obtiene sustituyendo con las correspondientes estadísticas muestrales, y se expresa de la manera siguiente:

$$r_{xy} = \frac{\sum(X - \bar{X})(Y - \bar{Y})}{[\sum(X - \bar{X})^2 \ \sum(Y - \bar{Y})^2]^{1/2}}$$

Ecuación 8: Coeficiente de correlación entre dos variables.

El coeficiente resultante nos informa sobre la intensidad y la dirección de la relación lineal entre dos variables. Se interpreta dentro del rango de -1 a +1, donde valores cercanos a +1 o -1 denotan una asociación lineal fuerte y positiva o negativa, respectivamente, y un valor cercano a 0 indica una ausencia de relación lineal. También es importante destacar que, para la aplicación correcta del coeficiente de correlación de Pearson, se asume la normalidad bivariada de las variables involucradas y se debe prestar atención a los posibles efectos de los valores atípicos que pueden llevar a interpretaciones erróneas. Además, el coeficiente es sensible únicamente a las relaciones lineales, por lo que las asociaciones no lineales no se reflejarán en su valor.

La Ilustración 5 adjunta a continuación muestra un mapa de calor que representa la matriz de correlaciones entre las variables seleccionadas.

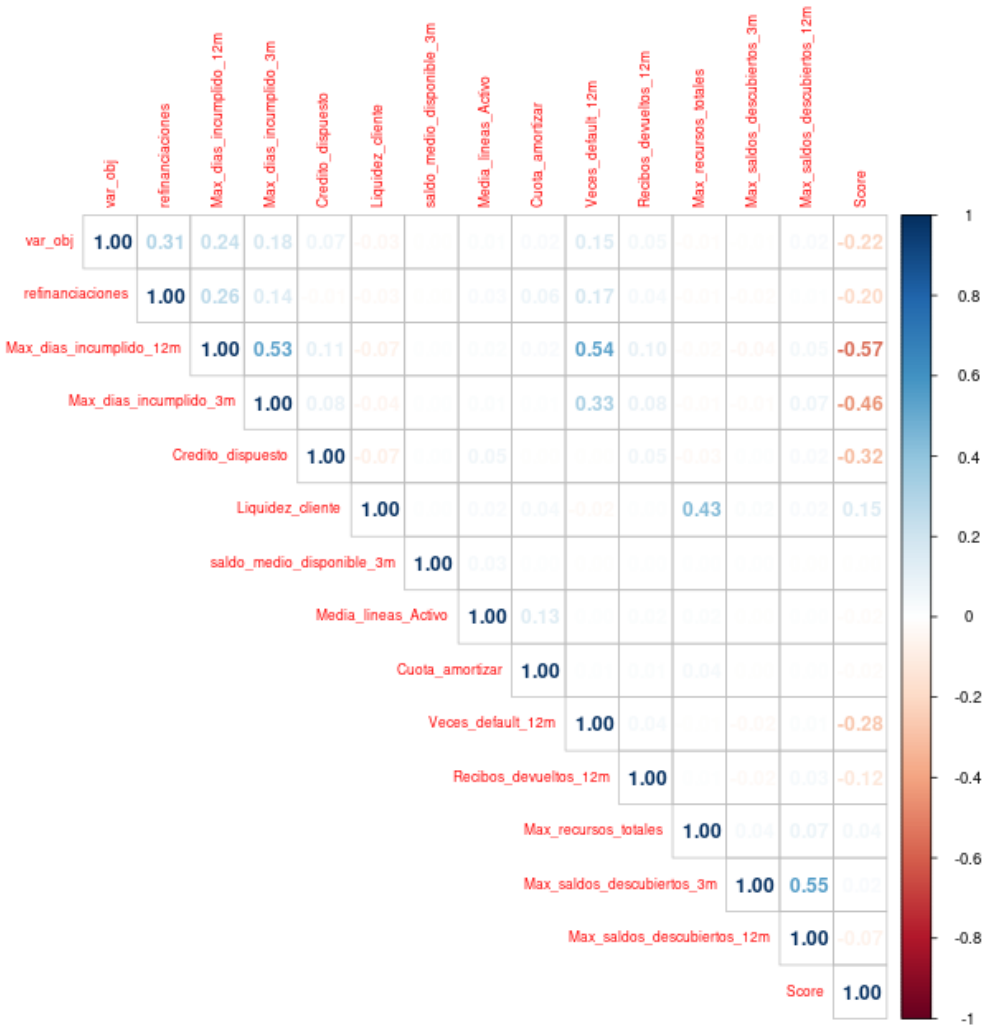


Ilustración 5 Elaboración propia: Matriz de correlación entre las variables de la muestra.

Cada celda del mapa de calor denota el coeficiente de correlación para cada par de variables, con valores de 1 en la diagonal que reflejan la perfecta autocorrelación. Los tonos más oscuros indican una correlación más fuerte, ya sea positiva (azul oscuro) o negativa (rojo oscuro), mientras que los tonos más claros sugieren una asociación más débil. Se observa lo siguiente:

1. Max_dias_incumplido_12m y Max_dias_incumplido_3m: Esta correlación parece ser fuerte y positiva (0.53), lo cual es lógico ya que el número máximo de días de incumplimiento en un período de 3 meses seguramente estará relacionado con el número máximo de días de incumplimiento en un período de 12 meses; si un cliente tiene una historia de incumplimiento en el corto plazo, es probable que esto también afecte el conteo en el largo plazo.

2. Max_saldos_descubiertos_3m y Max_saldos_descubiertos_12m: Otra correlación fuerte y positiva (0.55) indica que los saldos descubiertos en una ventana de tiempo de 3 meses están estrechamente relacionados con los de una ventana de tiempo de 12 meses. Esto también tiene sentido intuitivamente; los problemas de saldo descubierto no se limitan generalmente a un corto período de tiempo.
3. *Score*: El *Score* parece tener correlaciones negativas fuertes con algunas variables como Max_dias_incumplido_12m / Max_dias_incumplido_3m (-0.57 y -0.46), Crédito_dispuesto (-0.32) y Veces_default_12m (-0.28). Esto indica que a mayor puntuación (*Score*), hay menos días de incumplimiento, menos crédito dispuesto y menos incidencias de default en un año. Sugiere que el *Score* está calculado de tal manera que refleje la probabilidad de impagar en el futuro.

La presencia de correlaciones significativas dentro de un conjunto de datos puede sugerir varias posibilidades. Por un lado, podríamos estar observando una causalidad potencial, donde una variable influye en otra; sin embargo, es crucial recordar que correlación no implica causalidad y que factores externos pueden estar influyendo en ambas variables. Asimismo, una alta correlación puede señalar redundancia, indicando que dos variables reflejan información similar y, por tanto, una de ellas podría ser prescindible en un modelo analítico. Estos hallazgos también ofrecen *insights* valiosos para la toma de decisiones estratégicas, como en el sector financiero, donde correlaciones robustas pueden mejorar los modelos de predicción de incumplimiento de pagos. No obstante, las correlaciones son solo indicativas y requieren de un análisis más profundo, incluyendo la evaluación de la multicolinealidad y la realización de pruebas estadísticas, para conocer la verdadera naturaleza de las interacciones.

4 MODELIZACIÓN

4.1. Flag de *Default*

Según la normativa IFRS 9 y las directrices de la Autoridad Bancaria Europea (EBA), un cliente se considera en "*default*" cuando tiene atrasos de pago de más de 90 días. Esta definición es crucial porque afecta a cómo las instituciones financieras deben reconocer y manejar el crédito riesgoso.

Cabe destacar que, siguiendo las directrices de la EBA, se ha establecido una nueva definición de incumplimiento, aplicable a través de la actualización "EBA *NDoD* en IFRS9". Se introduce un estado de prueba para los contratos que salen del estado de incumplimiento. Durante un mínimo de tres meses, estos contratos se considerarán en la Etapa 2 con una PD del 100%. Por otro lado, se establece umbrales tanto relativos como absolutos para determinar la morosidad, siendo estos el 1% de la exposición total. También, se considera el concepto *UtP* (*Unlikely to pay*). Se aplica a deudores que probablemente no cumplirán sus obligaciones crediticias, incluyendo situaciones como procesos judiciales o sobreendeudamiento. Además, si más del 20% de la exposición total de un cliente está en mora, esto puede afectar al resto de sus exposiciones. Existe también la asignación de incumplimiento irreversible. Los contratos irreversibles se definen usando tanto datos nuevos como antiguos, y se consideran irreversibles aquellos contratos sin fecha de regreso a un estado sano o sin transición documentada. Asimismo, aquellas operaciones que cuenten con más de 4 refinanciaciones se consideran en estado de impago manteniendo el *flag de default* igual a uno.

Aunque el marco regulatorio y las directrices actualizadas sobre la definición de incumplimiento son complejos e incluyen varios factores y estados detallados, la creación de la flag de default en la base de datos se mantiene relativamente simple y directa. En la práctica, el establecimiento de esta flag se basa principalmente en el número de días en mora de un préstamo o crédito. Aunque las nuevas regulaciones incluyen varios estados y umbrales para la morosidad, la norma de los 90 días sigue siendo un estándar reconocido por la EBA e IFRS9 para identificar de manera inequívoca un incumplimiento. Además, se considera que un cliente está en default si en los cuatro meses después del acuerdo del

contrato, realiza financiaciones. Este enfoque asegura que la operativa diaria sea manejable y que se pueda monitorizar y reaccionar eficazmente ante los cambios en el estado de los créditos de los clientes.

4.2. Training y Testing Samples

Para realizar las estimaciones de default en clientes, se ha seleccionado una muestra de entrenamiento que abarca el período de marzo a junio de 2022. Esta muestra permite observar el comportamiento de la cartera de clientes y el desarrollo de sus contratos, tomando en cuenta diversas variables descritas en el apartado de Revisión de las variables. La división de esta muestra para fines de modelado se ha hecho en una proporción de 70-30%, destinando el 70% para el entrenamiento del modelo y el 30% restante para su validación. Esta segmentación ayuda a garantizar que el modelo pueda ser ajustado y afinado con suficientes datos, mientras que se reserva un segmento significativo para evaluar su rendimiento en condiciones similares a las operacionales. Es importante mencionar que la tasa de default de ambas muestras se ha mantenido constante en la división con el fin de obtener resultados más precisos de los modelos.

	N	Ratio de default
Train	1373422	2,57895%
Test	588611	2,57890%

Tabla 5 Elaboración propia: Análisis muestras train y test.

Cabe mencionar que la ratio de default no es el mismo que el presentado previamente debido a la eliminación de variables que se ha explicado detalladamente en el apartado de Exploración de la muestra.

Además, para abordar el desbalance entre las clases de clientes en default y no default, se ha estimado con ambos conjuntos de datos: la muestra original de entrenamiento y una muestra balanceada. Esta técnica se muestra en el Código adjuntado en el Anexo B.

- La técnica de balanceo de datos descrita combina submuestreo ("*under-sampling*") y sobremuestreo ("*over-sampling*") para ajustar la proporción de clases en un conjunto de datos desequilibrado. En primer lugar, se realiza un submuestreo de la clase mayoritaria ("no defaults"), es decir; operaciones sanas, seleccionando aleatoriamente el 50% de sus observaciones sin reemplazo, reduciendo así su tamaño y mitigando su dominio en el conjunto de datos. Esto ayuda a minimizar el sesgo introducido por la gran cantidad de datos de la clase mayoritaria, permitiendo que el modelo preste más atención a la clase minoritaria.

- Posteriormente, se lleva a cabo un sobremuestreo de la clase minoritaria ("*defaults*") utilizando *bootstraping*, donde se seleccionan muestras con reemplazo de los datos originales. Este proceso se repite iterativamente, generando un 10% adicional de la clase minoritaria en cada iteración, hasta que su tamaño alcanza aproximadamente el 85% del total inicial de dicha clase. Combinando estas técnicas, se obtiene un conjunto de datos más equilibrado, mejorando la capacidad del modelo para aprender patrones de ambas clases y reduciendo el riesgo de sesgo hacia la clase mayoritaria.

- La muestra final presenta las características de la Tabla 6, donde podemos observar que la tasa de defaults ha aumentado de 2,6% a 10,4% a causa del sobremuestreo de impagos y submuestreo de operaciones sanas con las técnicas de balanceo de datos descritas previamente.

	Muestra balanceada
N	746.924
Defaults	77.923
No defaults	669.001
Ratio de defaults	10,433%

Tabla 6 Elaboración propia: Descripción de la muestra balanceada.

En lo que respecta a la evaluación del modelo, se lleva a cabo inicialmente en la muestra de prueba, que se deriva directamente de la muestra de desarrollo original. Esta muestra de prueba ha sido configurada para mantener constante la tasa de default y comprende el 30% de la muestra de desarrollo, no utilizada en la fase de entrenamiento. Este paso es

crucial para testear cómo el modelo maneja datos nuevos y para verificar su capacidad de generalización y robustez en condiciones no observadas durante la fase de entrenamiento.

A su vez, se realiza una evaluación adicional del modelo final utilizando una muestra que abarca un periodo de tiempo posterior a la de desarrollo. A esta muestra se le denomina "Muestra *Out of Sample*" o "Fuera de la muestra" en contexto de backtesting. En concreto, abarca un periodo de tiempo de la cartera de clientes un año después, aproximadamente (enero – febrero 2023). Esta muestra es particularmente valiosa por ser posterior a la muestra utilizada para el desarrollo del modelo, ofreciendo una oportunidad única para validar la capacidad predictiva del modelo bajo condiciones de mercado actualizadas y potencialmente diferentes. Este tipo de evaluación es esencial para entender cómo el modelo se comportará en escenarios futuros y en condiciones dinámicas del mercado.

4.3. Modelo Logit

Como se ha explicado previamente, los modelos logit son ampliamente utilizados en la práctica para realizar estimaciones, debido a su solidez y la amplia documentación de su base matemática que les confiere una alta fiabilidad. Sin embargo, estos modelos no están exentos de limitaciones. Una de las principales es que asumen linealidad en el logaritmo de las probabilidades, lo cual puede no ser adecuado para describir relaciones más complejas entre las variables. Además, requieren de un tamaño de muestra suficientemente grande para garantizar la precisión y la estabilidad. Otra limitación es su sensibilidad a los valores atípicos, ya que estos pueden afectar significativamente la estimación de los parámetros del modelo, llevando a resultados potencialmente sesgados. Para el desarrollo del modelo logístico sobre nuestra muestra, se ha utilizado la función *glm*(), que se refiere a *generalized linear models*. Se utiliza la siguiente expresión explicada más en detalle en el apartado de Técnicas de estimación.

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k$$

Ecuación 9:Generalized Linear Models. Link: logit.

Donde la probabilidad es:

$$p = \frac{e^{\beta_0+\beta_1 x_0+\beta_2 x_2+\cdots+\beta_k x_k}}{1 + e^{\beta_0+\beta_1 x_0+\beta_2 x_2+\cdots+\beta_k x_k}}$$

Ecuación 10: Logit Probability.

Dentro de la función, debemos especificar el argumento de familia: una descripción de la distribución de errores y la función de enlace que se utilizará en el modelo. En este caso, será binomial por la naturaleza de nuestra variable *default* y se especifica *link = "logit"*.

En primer lugar, se estima el modelo con la muestra de entrenamiento sin balancear con el fin de evaluar en cuál de las muestras funciona mejor y si el sobremuestreo de la variable default ayuda a que el modelo prediga mejor. Cabe destacar que se han eliminado del modelo las variables "Max_saldos_descubiertos_3m" y "Max_activo_total" por no ser significativas. Esto indica que no tenemos suficiente evidencia estadística para rechazar que el coeficiente de estas variables sea igual a cero (H0: hipótesis nula) para un nivel de significancia de 0.05.

Se ha analizado el Factor de Inflación de la Varianza (VIF), que mide cuánto se infla la varianza de un coeficiente estimado debido a la correlación lineal con otras variables predictoras en el modelo. Se calcula como $VIF_i = \frac{1}{1-R_i^2}$ donde R_i^2 es el coeficiente de determinación de una regresión de la variable i-ésima sobre todas las otras variables predictoras.

	VIF
Refinanciaciones	1,097
Max_dias_incumplido_12m	1,083
Max_dias_incumplido_3m	1,815
Credito_dispuesto	1,088
Liquidez_cliente	1,083
Saldo_medio_disponible_3m	1,042
Media_lineas_Activo	1,002
Cuota_amortizar	1,009
Veces_default_12m	1,524
Recibos_devueltos_12m	1,015
Max_recursos_totales	1,041
Max_saldos_descubiertos_12m	1,033
Score	1,875

Tabla 7 Elaboración propia: VIF variables en el modelo logit.

Estos valores indican que la multicolinealidad no parece ser un problema significativo para las variables predictoras del modelo.

Para el primer caso (muestra sin balanceo de datos), se estimó el modelo usando la muestra de entrenamiento, tal y como se ha indicado. Sin embargo, se observaron valores de los residuos extremadamente altos por encima del cuantil 99% y bajos por debajo del cuantil 1%. Se muestran a continuación:

	Residuos modelo
Min	-271.600
1st Qu.	-1,000
Median	-1,000
Mean	9,18,E+09
3rd Qu.	-1,000
Max	4,50,E+15
Q10%	-1,042
Q25%	-1,020
Q50%	-1,011
Q75%	-1,006
Q90%	-1,002
Q95%	-1,000
Q99%	19,67

Tabla 8 Elaboración propia: Residuos primer modelo logit sin balanceo.

Por este motivo, se decide aplicar un algoritmo de eliminación de las observaciones asociadas a aquellas predicciones con residuos extremos, en este caso por encima del cuantil 99% (como observamos en la tabla superior, por encima de 19,67), así como aquellas observaciones por debajo del cuantil 1% que alcanzan hasta un valor de 2,71E+5. Se eliminan en total 27.332 observaciones, lo cual no representa ni un 2% del total. Obtenemos un modelo, con un AIC significativamente mejorado en la Tabla 8.

	AIC
Modelo 1	251.209
Modelo 2	88.066

Tabla 9 Elaboración propia: AIC modelos logit sin eliminación de residuos extremos y con eliminación de residuos extremos.

El primer modelo estimado con la muestra sin balancear queda como se muestra en la Tabla 9:

	Estimates	Std, Error	z value	Pr(>|z|)	
(Intercept):	1,6E+01	0,103	52,275	2,00E-16	***
Refinanciaciones:	4,0E+00	0,008771	164,735	2,00E-16	***
Max_dias_incumplido_12m:	2,8E-02	0,0002192	59,854	2,00E-16	***
Max_dias_incumplido_3m:	1,7E-02	0,0003063	19,842	2,00E-16	***
Credito_dispuesto:	1,9E+00	0,01208	59,9	2,00E-16	***
Liquidez_cliente:	-3,5E-05	0,000251	-47,031	2,00E-16	***
Saldo_medio_disponible_3m:	-1,3E-05	0,000304	-11,969	2,00E-16	***
Media_lineas_Activo:	-1,8E-07	0,0000049	3,623	0,0156	*
Cuota_amortizar:	3,7E-05	0,002864	3,737	1,06E-09	***
Veces_default_12m:	-2,5E-01	0,01003	-17,123	2,00E-16	***
Recibos_devueltos_12m:	7,2E-06	0,0001844	17,043	2,00E-16	***
Max_saldos_descubiertos_12m:	4,8E-06	0,0003971	5,73	1,99E-08	***
Score:	-3,3E-02	0,0001489	-93,393	2,00E-16	***

Tabla 10 Elaboración propia: Modelo logit sin balanceo y sin eliminación de residuos extremos.

Todas las variables muestran sentido económico y financiero para explicar nuestra variable objetivo (*default*).

Dado que el AIC ha mejorado significativamente, los residuos ya no presentan valores extremos que puedas distorsionar las estimaciones y que el VIF es adecuado para todas las variables, entre otros motivos, este será el modelo que vamos a comparar con el de la muestra balanceada

Al modelo realizado con la muestra balanceada se le han aplicado las mismas modificaciones que al previo, dado que también se observaban valores entremos en los residuos. En este caso, obteníamos unos residuos con las siguientes métricas mostradas en la Tabla 10.

	Residuos modelo balanceado
Min	-4,5E+15
1st Qu.	-1,000
Median	-1,000
Mean	1,8,E+10
3rd Qu.	-1,000
Max	4,50,E+15
Q1%	-17.531,75
Q10%	-1,14
Q25%	-1,06
Q50%	-1,03
Q75%	-1,01

Q90%	1,03
Q95%	3,61
Q99%	18,44

Tabla 11 Elaboración propia: Residuos del modelo con balanceo antes de la eliminación de residuos extremos.

Tras la eliminación de observaciones que mostraban residuos extremos a la muestra obtenemos el siguiente modelo mostrado en la Tabla 11 utilizando datos balanceados.

	Estimates	**Std. Error**	**z value**	**Pr(>|z|)**	
(Intercept):	8,088	0,1319	61,33	2,0E-16	***
Refinanciaciones:	3,623	0,01597	226,824	2,0E-16	***
Max_dias_incumplido_12m:	0,02414	0,0003253	74,204	2,0E-16	***
Max_dias_incumplido_3m:	0,02325	0,0004305	54,012	2,0E-16	***
Credito_dispuesto:	0,9667	0,01363	70,924	2,0E-16	***
Liquidez_cliente:	-2,378E-05	0,0003583	66,358	2,0E-16	***
Saldo_medio_disponible_3m:	-6,425E-06	0,0003633	17,685	2,0E-16	***
Media_lineas_Activo:	9,758E-07	0,00004278	22,809	2,0E-16	***
Cuota_amortizar:	0,00001291	0,004193	3,079	2,1E-03	***
Veces_default_12m:	-0,3643	0,01309	27,823	2,0E-16	***
Recibos_devueltos_12m:	5,158E-06	0,0002725	18,929	2,0E-16	***
Max_saldos_descubiertos_12m:	2,607E-06	0,000592	4,404	1,1E-05	***
Score:	-0,017	0,0001904	-89,281	2,0E-16	***

Tabla 12 Elaboración propia: Modelo logit con datos balanceados y eliminación de residuos extremos.

Ambos modelos mostrados en este apartado se evalúan sobre la muestra de prueba (*test sample*) para evaluar cuál de los dos funciona mejor para nuestros datos. En el Anexo A, se adjunta una comparación de los resultados para justificar cuál se elige finalmente.

En este caso, se tomará el modelo de la muestra balanceada porque es el que presenta mejor tasa de verdaderos positivos. Esta decisión se toma debido a que vamos a preferir que nuestro modelo sea más conservador (es decir, preferimos que haya falsas predicciones de default, cuando en la realidad no se ha incurrido en default, que lo contrario). Utilizamos la matriz de confusión, en la Tabla 12, para observar estas métricas.

Matriz de confusión sin balanceo

Predicción \ Valor Real	0	1
0	567.566	5.865
1	11.560	3.620

Matriz de confusión con balanceo

Predicción \ Valor Real	0	1
0	558.781	9.650
1	8.733	11.447

Tabla 13 Elaboración propia: Matrices de confusión de las predicciones de los modelos logit sin balanceo y con balanceo.

Podemos observar que la tasa de verdaderos positivo, que se calcula como $TVP = VP / (VP + FP)$ es de 54,26% para la muestra con balanceo y de 38,16% para la muestra sin balanceo. En cuanto a la tasa de verdaderos negativos, también se obtiene un mejor ajuste en la métrica balanceada. Concretamente, obtenemos una TVN de 98,46% para la muestra con balanceo y de 98,00% para la muestra sin balancear.

4.4. Random Forest

En el proceso de configuración de un modelo de Random Forest para la predicción de una variable, es crucial especificar adecuadamente los parámetros y controles del entrenamiento para optimizar el rendimiento y asegurar la generalización del modelo. A continuación, se describe teóricamente cómo se fundamenta este procedimiento.

El primer paso involucra la implementación de una técnica de validación robusta, en este caso, la validación cruzada repetida. Esta metodología no solo evalúa la eficacia del modelo en diferentes subconjuntos de datos, sino que también repite el proceso varias veces para minimizar las variaciones en la estimación del rendimiento del modelo debido a la aleatoriedad en la selección de los subconjuntos. Al usar repetidamente la validación cruzada, se proporciona una evaluación más confiable y estable de cómo el modelo puede esperarse que actúe en datos no vistos.
Este enfoque se utiliza también en el siguiente método, *Gradient Boosting*, que explicaremos a continuación. Para este caso, se establece un número de pliegues de 10 (*number = 10*). Esto indica que dividimos el conjunto de datos en 10 subconjuntos de partes iguales para llevar a cabo la validación. Además, se repite la validación un total de 3 veces (*repeats = 3*) para obtener una estimación más robusta del rendimeinto del modelo.

En segundo lugar, es esencial definir una red de hiperparámetros a explorar. Esto incluye parámetros como el número de predictores a considerar en cada división (*mtry*), el tamaño mínimo de los nodos al final de cada árbol (*min.node.size*), y la regla de división basada en la pureza de los nodos (*splitrule*). La optimización de estos hiperparámetros es

fundamental porque cada configuración puede tener un impacto significativo en la capacidad del modelo para aprender de los datos sin sobre-ajustarse. Para optimizar todavía más el modelo se hace uso de una malla de hiperparámetros que permite realizar un barrido exhaustivo de posibles combinaciones de parámetros, buscando aquellos que maximicen la precisión del modelo:

- *mtry*: Los valores establecidos para este parámetro son 3, 4, 5 y 7. Esto significa que durante la construcción de cada árbol en el bosque, se considerarán aleatoriamente 3, 4, 5 o 7 características como candidatas para la división en cada nodo. Este rango de valores permite explorar diferentes niveles de complejidad del modelo y controlar el sobreajuste.
- *min.node.size*: Los valores establecidos son 5, 15, 20 y 30. Esto indica que el tamaño mínimo permitido para los nodos terminales (hojas) en los árboles del bosque será de al menos 5, 15, 20 o 30 observaciones.
- *splitrule*: El valor establecido es "*gini*", lo que significa que se utilizará la métrica de impureza de Gini[2] para decidir cómo dividir los nodos en los árboles del bosque.
- *Num_trees*: se prueban 100, 300 y 500 árboles en una malla y se almacenan en un objeto llamado *modellist* con las estimaciones.

El paralelismo es otra consideración importante en la configuración del entrenamiento, habilitando el uso de múltiples procesadores para acelerar el proceso. Esto es especialmente útil cuando se manejan grandes volúmenes de datos o cuando se realizan múltiples simulaciones como parte de la validación cruzada repetida. Permitir el procesamiento paralelo puede reducir significativamente el tiempo requerido para encontrar el mejor modelo. Si bien es cierto, el costo computacional de realizar un algoritmo que evalúe grandes posibilidades de combinaciones de parámetros es enorme. Por ello, el abanico de oportunidades es limitado para el estudio y podría ser mejorado utilizando dispositivos más potentes y con mayor capacidad de servidor.

[2] La impureza de Gini es una medida de cuán mezcladas están las clases en un nodo y se utiliza para encontrar la mejor manera de dividir el nodo en subgrupos más homogéneos en términos de la variable objetivo. Esto proporciona una guía sobre cómo construir árboles que maximicen la homogeneidad de las clases en los nodos terminales. [IBM Cognos Analytics, 11.1.0, “Medida de impureza de Gini”, 24-02-29].

A continuación, se muestra la evaluación de los distintos parámetros probados para el modelo final.

mtry	min.node.size	splitrule	Accuracy	Kappa	AccuracySD	KappaSD
3	5	gini	98,79%	69,62%	0,020%	0,66%
3	15	gini	98,51%	59,85%	0,022%	0,84%
3	20	gini	98,41%	56,04%	0,023%	0,91%
3	30	gini	98,27%	50,45%	0,022%	0,91%
4	5	gini	98,85%	71,65%	0,020%	0,63%
4	15	gini	98,58%	62,42%	0,023%	0,82%
4	20	gini	98,48%	58,89%	0,023%	0,85%
4	30	gini	98,33%	53,16%	0,022%	0,84%
5	5	gini	98,89%	72,92%	0,019%	0,59%
5	15	gini	98,63%	64,16%	0,021%	0,75%
5	20	gini	98,52%	60,42%	0,023%	0,82%
5	30	gini	98,38%	54,99%	0,023%	0,90%
7	5	gini	98,93%	74,15%	0,021%	0,63%
7	15	gini	98,68%	66,22%	0,022%	0,69%
7	20	gini	98,58%	62,79%	0,024%	0,83%
7	30	gini	98,43%	57,24%	0,022%	0,82%

Tabla 14 Elaboración propia: Parámetros probados en el entrenamiento del modelo Random Forest sin balanceo.

Según la Tabla 13, la medida de *Accuracy*[3] parece ser bastante alta en todas las configuraciones, alrededor del 98.7% a 98.9%. Esto sugiere que el modelo generalmente predice correctamente, independientemente de la configuración de *mtry* y *min.node.size*. El valor de *Kappa*[4], aunque más variable, sigue siendo relativamente alto, lo cual es bueno ya que indica un buen nivel de acuerdo más allá de lo que se esperaría por casualidad. Sin embargo, hay variaciones dependiendo de los hiperparámetros.

Tanto en la Tabla 13 como en la Ilustración 6 observamos que a medida que el valor de mtry aumenta de 3 a 7 y el *min.node.size* varía de 5 a 30, no se observan cambios drásticos en la precisión, pero hay pequeñas variaciones en los valores de Kappa. Por ejemplo, el modelo con *mtry* = 7 y *min.node.size* = 5 tiene un Kappa especialmente alto (74,15%) comparado con otras configuraciones. Esto podría indicar una mejor capacidad del

[3] Accuracy mide la proporción de predicciones correctas del modelo sobre el total de predicciones realizadas. [BS ISO 5725-1: "Accuracy (trueness and precision) of measurement methods and results - Part 1: General principles and definitions.", p.1 (1994)].

[4] El valor kappa, o índice de Kappa de Cohen, mide la concordancia entre las predicciones del modelo y las etiquetas verdaderas, ajustando por la concordancia que ocurriría por azar. En el contexto de un random forest, un valor de kappa más alto indica un mejor rendimiento del modelo. Así, seleccionar el modelo con el valor de kappa más alto implica elegir el modelo cuyos hiperparámetros proporcionan la mayor precisión ajustada para la clasificación. [Carletta, Jean. (1996) Assessing agreement on classification tasks: The kappa statistic. Computational Linguistics, 22(2), pp. 249–254].

modelo para generalizar en esta configuración particular bajo el uso de esta métrica específica.

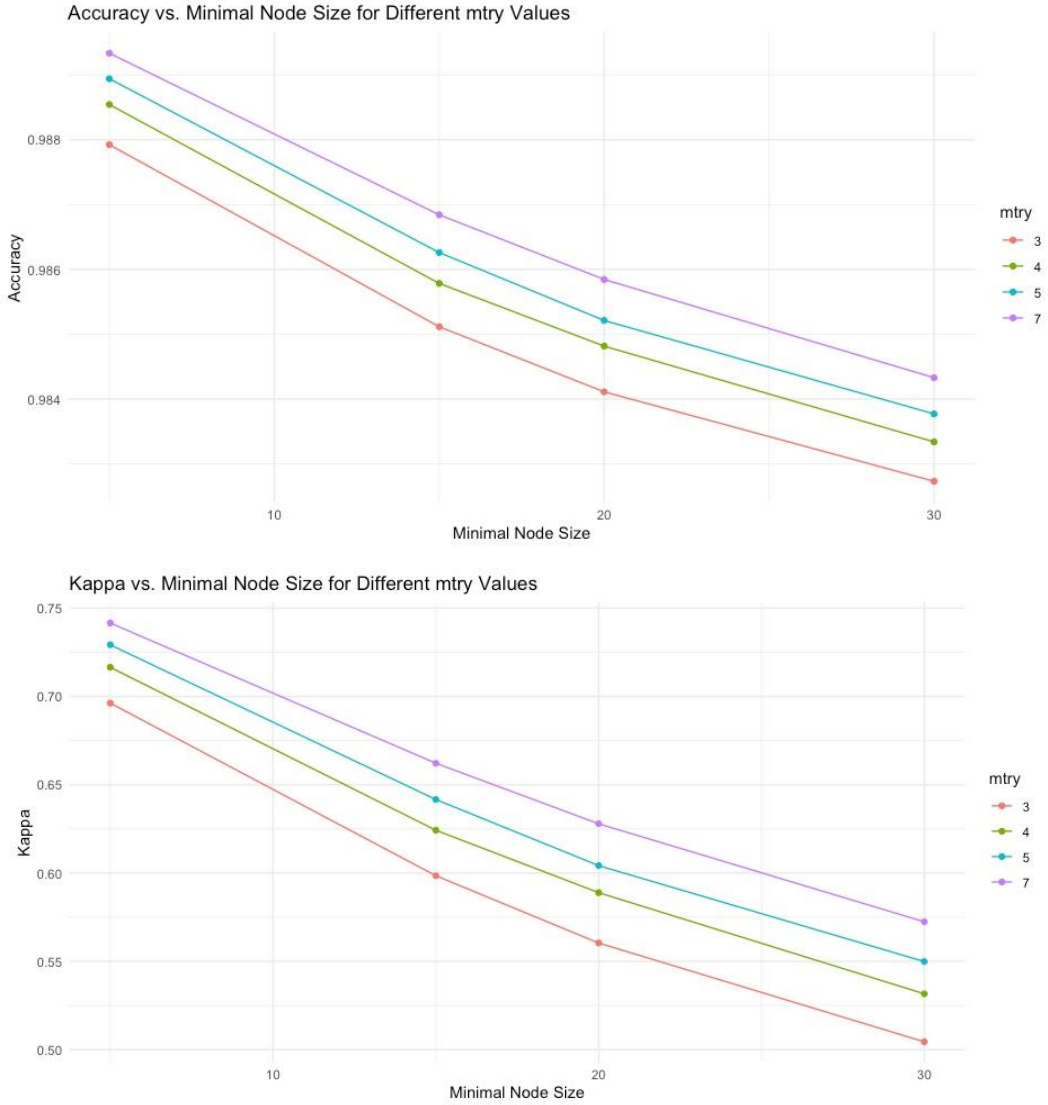


Ilustración 6 Elaboración propia: Accuracy y Kappa en función de los hiperparámetros del modelo de random forest.

El modelo final escogido queda entonces como se muestra en la Tabla 14.

Ranger result
Call: ranger::ranger(dependent.variable.name = ".outcome", data = x, mtry = min(param$mtry, ncol(x)), min.node.size = param$min.node.size, splitrule = as.character(param$splitrule), write.forest = TRUE, probability = classProbs, ...)
Type: Probability estimation

Number of trees: 100
Sample size: 1,373,422
Number of independent variables: 15
Mtry: 7
Target node size: 5
Splitrule: Gini
OOB prediction error (Brier s.): 0.007762101

Tabla 15 Elaboración propia: Modelo final Random Forest.

- Type: "Probability estimation", lo cual indica que el modelo está configurado para estimar probabilidades de las clases de la variable objetivo.
- Number of trees: 100.
- Sample size: 1,373,422. El número de muestras utilizadas para entrenar el modelo.
- Number of independent variables: 15. El total de variables independientes disponibles para hacer las divisiones en los árboles.
- Mtry: 7. Número de variables probadas al realizar cada división en el árbol.
- Target node size: 5. El tamaño objetivo para los nodos de los árboles.
- OOB prediction error (Brier s.): 0.007762101. El error de predicción fuera de bolsa medido mediante el score de Brier para estimaciones de probabilidad. Un valor más bajo indica un mejor rendimiento del modelo. El score de Brier es una medida de la precisión de las probabilidades predichas y es especialmente útil en modelos de clasificación probabilísticos.

Por último, en cuanto a la comparativa de los modelos sobre la muestra balanceada y sin balancear, se puede consultar en el Anexo A los resultados de las matrices de confusión sobre los resultados de las predicciones. En este caso, ambos modelos se ajustan prácticamente igual y dan resultados correctos. Además, los parámetros finales obtenidos son los mismos en ambos casos. Sin embargo, sobre la muestra balanceada obtenemos ligeramente mejores resultados según la Ilustración 7.

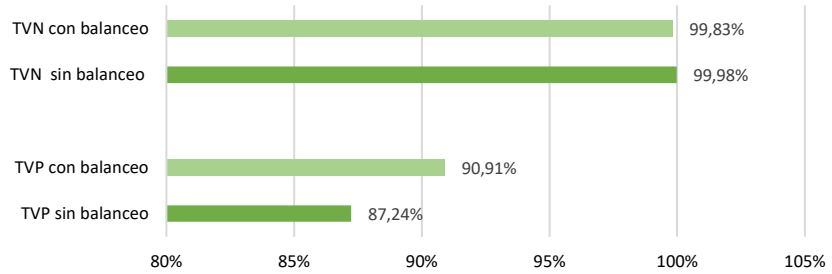


Ilustración 7 Elaboración propia: Tasa de verdaderos positivos y verdaderos negativos del modelo random forest sin / con balanceo.

Aunque la muestra sin balanceo ajuste mejor los no defaults, se procede a utilizar el modelo con la muestra balaceada porque nos interesa que prediga mejor los casos de clientes malos que de clientes buenos. Cuantos más clientes malos seamos capaces de predecir, menor pérdida económica final y menor riesgo crediticio enfrentaremos.

4.5. XGBoost

Este método de aprendizaje automático, tal y como explicábamos previamente, es una técnica que construye un modelo predictivo en forma de un conjunto de modelos débiles, típicamente árboles de decisión, de manera secuencial. Cada árbol se ajusta a los errores residuales del modelo anterior, lo que permite mejorar la predicción en cada iteración. La base matemática del *Gradient Boosting* se centra en minimizar una función de pérdida que cuantifica la discrepancia entre las predicciones del modelo y los valores observados. Para esto, se utiliza un algoritmo de optimización basado en gradientes que ajusta los parámetros del modelo en la dirección que reduce la pérdida.

El algoritmo utilizado para entrenar el modelo se conoce como XGB o Extreme Gradient Boosting, que es una extensión del método tradicional, pero con mejoras a nivel de optimización y prevención del sobreajuste.

Uno de los inconvenientes de este método de estimación es la cantidad de parámetros que hay que establecer previamente. Esto hace que poder encontrar el valor óptimo de cada uno de ellos sea complicado y costoso en términos computacionales. No obstante, se ha optado por establecer una malla de hiperparámetros para que el algoritmo establezca aquel que mejor se ajusta a nuestros datos y resulte en un modelo más acertado.

Algunos de los parámetros más importantes que se pueden ajustar para mejorar la predicción son:

- **Número de iteraciones (*nrounds*):** especifica cuántos árboles de decisión se agregarán al conjunto en el proceso de entrenamiento. Un mayor número de iteraciones puede mejorar la precisión del modelo, pero también aumenta el riesgo de sobreajuste. Se prueba una malla de 50, 100 y 150.
- **Profundidad máxima del árbol (*max_depth*):** Controla la complejidad de los árboles de decisión individuales en el conjunto. Aumentar la profundidad máxima puede permitir que los árboles capturen relaciones más complejas en los datos, pero también puede aumentar el riesgo de sobreajuste. Se establece una malla de 10, 20 y 30.
- **Tasa de aprendizaje (*eta*):** Controla la contribución de cada árbol al modelo final. Una tasa de aprendizaje más baja requiere más iteraciones para alcanzar la misma precisión, pero puede mejorar la generalización del modelo. En este caso, se prueba con los valores de 0,1 y 0,01.
- **Muestreo de columnas (*colsample_bytree*):** Especifica la fracción de características que se deben seleccionar al construir cada árbol de decisión. Esto puede ayudar a reducir el sobreajuste al introducir variabilidad en el proceso de entrenamiento. Aquí se establece un valor de 0,5 y 1.
- **Parámetros de regularización (*gamma, min_child_weight*):** Controlan la complejidad del modelo al penalizar la inclusión de nodos adicionales en los árboles de decisión. Ajustar estos parámetros puede ayudar a evitar el sobreajuste al limitar la complejidad del modelo. Para el caso de *gamma* se prueban γ=1 y γ=10. Para *min_child_weight* se establece en 1.

Al igual que en el método de *Random Forest*, se hace uso de la validación cruzada para la estimación sobre la muestra de entrenamiento. De nuevo, esto consiste en dividir el conjunto de datos total en 10 partes (*k folds*), entrenar el modelo utilizando cada uno de estos subconjuntos y evaluándolo sobre el siguiente. Esto se repite 3 veces en total, de forma que cada subconjunto se utiliza como muestra de prueba para evaluar el rendimiento una vez. Se calcula el rendimiento de cada subconjunto en cada repetición de la validación cruzada y, finalmente, se promedian dichas métricas obtenidas para conocer una estimación general del rendimiento del modelo. El objetivo de este método es reducir el sesgo y variabilidad en la evaluación. Sin embargo, es útil únicamente cuando la muestra suficientemente grande, lo cual se cumple para el presente estudio.

Los resultados del modelo final se ven en la Tabla 15.

eXtreme Gradient Boosting

746924 samples
15 predictor
2 classes: 'X0', 'X1'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 709577, 709578, 709578, 709577, 709577, 709578, ...
Resampling results across tuning parameters:

max_depth	colsample_bytree	Accuracy	Kappa
10	0.5	0.9407677	0.6248423
10	1.0	0.9411225	0.6325797
20	0.5	0.9880050	0.9337092
20	1.0	0.9886280	0.9380834
30	0.5	0.9909643	0.9505538
30	1.0	0.9906318	0.9493062

Tuning parameter 'nrounds' was held constant at a value of 50
Tuning parameter 'eta' was held constant at a value of 0.1
Tuning parameter
'gamma' was held constant at a value of 1
Tuning parameter 'min_child_weight' was held constant at a value of 1
Tuning parameter 'subsample'
was held constant at a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were nrounds = 50, max_depth = 30, eta = 0.1, gamma = 1, colsample_bytree = 0.5, min_child_weight = 1
and subsample = 1.

Tabla 16 Elaboración propia: Resultados del modelo XGBoost.

En este modelo, hemos utilizado la muestra sin balanceo ya que presentaba mejores resultados que la muestra sin balancear. Se puede ver la comparativa en el Anexo A. En la Ilustración 8 se muestran los resultados de la tasa de verdaderos positivos y verdaderos negativos en ambos modelos.

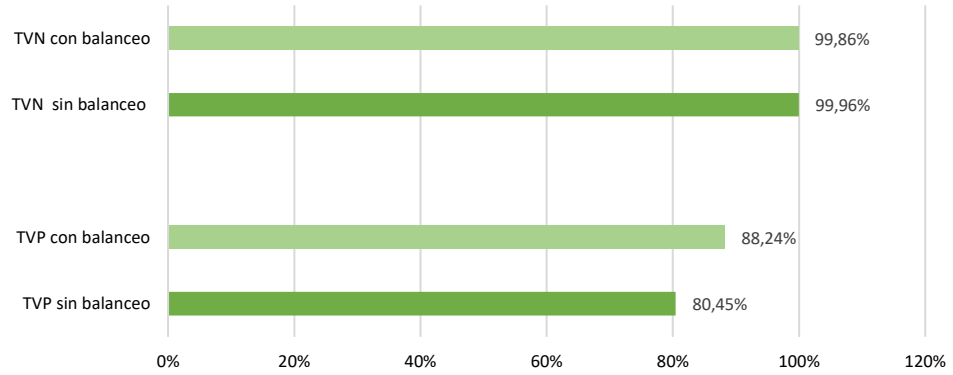


Ilustración 8 Elaboración propia: Tasa de verdaderos positivos y verdaderos negativos del modelo XGBoost con / sin balanceo.

Ocurre algo parecido al caso de Random Forest, ambos modelos muestran métricas y resultados similares. Pero, como se ha explicado, escogemos el modelo con datos sin balanceo por mostrar ligeramente mejores resultados.

4.6. Red Neuronal

Tal y como se ha explicado, las redes neuronales son técnicas que utilizan aprendizaje profundo y que imitan la manera en que el cerebro humano procesa información, lo que permite modelar patrones complejos y no lineales en los datos. Esta capacidad las hace especialmente aptas para lidiar con la naturaleza de datos financieros, donde las interacciones entre variables pueden ser sutiles y a la vez altamente significativas.

Para el desarrollo de este modelo se incluyen todas las variables en el entrenamiento, excluyendo lógicamente aquellas que no son numéricas o factores (fecha y código de persona). Se llevan a cabo dos tipos de redes neuronales; por un lado, una red de propagación resiliente y, por otro, una red neuronal recurrente con unidades GRU (*Gated Recurrent Unit*).

4.6.1. Red neuronal de propagación resiliente *(Rprop+)*

En el primer modelo, se submuestrea aleatoriamente el conjunto de datos de entrenamiento seleccionando en cien mil observaciones para entrenar la red neuronal.

Submuestrear datos implica seleccionar aleatoriamente una porción más pequeña de observaciones de un conjunto de datos más grande. Esto se hace por varias razones. En primer lugar, para reducir el tiempo y los recursos necesarios. Al trabajar con una muestra más pequeña, el proceso de entrenamiento se vuelve más eficiente y menos exigente computacionalmente. En segundo lugar, puede ayudar a prevenir el sobreajuste, ya que reduce la complejidad del modelo y la probabilidad de que este se adapte demasiado a las particularidades de los datos de entrenamiento.

Después de submuestrear los datos, se lleva a cabo la normalización de características. Se escalan las características seleccionadas utilizando la función *scale*(), que estandariza cada columna de datos restando la media y dividiendo por la desviación estándar. La normalización de características implica ajustar las escalas de las diferentes características en un conjunto de datos para que tengan una escala comparable. Esto es importante porque las características con escalas muy diferentes pueden afectar negativamente la convergencia del algoritmo de entrenamiento y el rendimiento del modelo resultante. Por ejemplo, una característica que abarca un rango mucho más grande de valores puede dominar el proceso de optimización durante el entrenamiento, lo que lleva a un sesgo en el modelo hacia esa característica en particular. Normalizar las características asegura que todas tengan una contribución equitativa al proceso de aprendizaje, lo que puede mejorar la convergencia del modelo y su capacidad para generalizar a nuevos datos de manera efectiva.

Para llevar a cabo esta técnica se utiliza el paquete de R “neuralnet”. El algoritmo utilizado para entrenar la red es *Rprop+* o Propagación Resiliente, es un algoritmo diseñado para superar algunos de los problemas asociados con los métodos tradicionales de retropropagación, particularmente en lo que respecta al ajuste de los pesos durante el entrenamiento de redes neuronales. El objetivo principal de *Rprop* es realizar ajustes de pesos más efectivos sin ser afectado por la magnitud del gradiente. Este enfoque ayuda a evitar los problemas comunes de los gradientes muy grandes o pequeños que pueden llevar a pasos de aprendizaje ineficaces.

A diferencia de la retropropagación tradicional, donde el tamaño del paso de actualización de los pesos depende de la magnitud del gradiente, *Rprop* utiliza sólo el signo del gradiente. Esto significa que la actualización de cada peso se realiza en función de la dirección en la que se debe realizar el ajuste (incrementar o disminuir el peso), independientemente de cuán grande sea el gradiente.

Cada peso tiene su propio tamaño de paso individual que se ajusta según la evolución del cambio en el signo de su gradiente parcial a través de las iteraciones. Si el gradiente de un peso cambia de signo, lo que indica un posible sobreajuste más allá de un mínimo local, el tamaño del paso para ese peso se reduce. Si el signo del gradiente permanece constante, el tamaño del paso puede aumentarse hasta un máximo predefinido, facilitando así una convergencia más rápida.
Por otro lado, si el cambio en el signo del gradiente sugiere que el último paso fue demasiado grande (lo que normalmente resulta en un aumento del error total), *Rprop+* no solo reduce el tamaño del paso, sino que también revierte el último cambio de peso. Esta capacidad para deshacer un paso basado en su efectividad mejora la estabilidad del algoritmo y ayuda a prevenir oscilaciones y divergencias.

La red neuronal entrenada y la explicación de sus parámetros se muestra en la Tabla 16 y se explica a continuación.

```
formula <- as.formula(paste("var_obj ~ ."))
red_neuronal <- neuralnet(formula,
                          data = datos_muestra_scaled,
                          hidden = c(5, 10),
                          stepmax = 1e8,
                          algorithm = "rprop+",
                          rep = 3,
                          threshold = 0.01)
```

Tabla 17 Elaboración propia: Código de creación red neuronal RProp+

- *Hidden* = (5, 10). Esto indica el número de capas ocultas de la red. En este caso hay dos capas ocultas, la primera con 5 neuronas y la segunda con 10 neuronas.
- *Stepmax* = 1e+8. Esto es el número máximo de pasos o iteraciones que el algoritmo puede tomar durante el proceso de entrenamiento
- *Algorithm: rprop+*. Algoritmo previamente explicado de propagación resiliente.
- *Rep = 3*. Número de veces que se entrena la red neuronal completa.
- *Threshold* = 0,01.

Cabe destacar que la función de activación por defecto en esta red neuronal es sigmoidal, que es especialmente útil en problemas de regresión y clasificación binaria. Esta función de activación se define como: $\sigma(x) = \frac{1}{1+e^{-x}}$

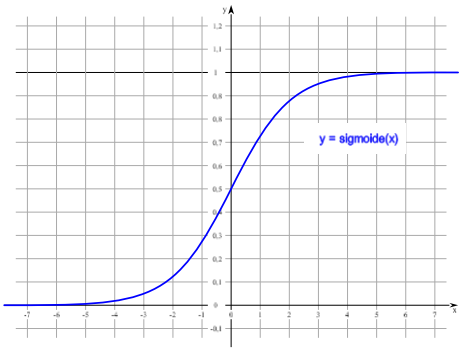


Ilustración 9: Fuente: Wikipedia. Función Sigmoide.

Donde x es el valor de la entrada de la función y e es la base del logaritmo natural. El rango de salida de la función se encuentra entre 0 y 1. Visualmente, la función tiene forma de "S" (de ahí su nombre). Esto significa que, para valores de entrada muy altos o bajos, los cambios en la salida son muy pequeños, mientras que para valores cercanos a 0, pequeños cambios en la entrada pueden resultar en grandes cambios en la salida.
La función es suave y continua, y tiene una derivada fácil de calcular, que es esencial para el algoritmo de retropropagación utilizado para entrenar redes neuronales. La derivada de la función sigmoidal es $\sigma(x) * \big(1 - \sigma(x)\big)$, que también es suave y continua. Cada neurona en la red recibe ciertas entradas, que son combinadas linealmente según los pesos y sesgos de la red. Es decir, se calcula un valor z como una suma ponderada de las entradas. Este valor z se pasa a través de la función sigmoidal para obtener la salida de la neurona, σ(z), que estará entre 0 y 1.

Por último, se ha escogido el modelo entrenado sobre datos balanceado por presentar mejores predicciones sobre la muestra de prueba. La diferencia es bastante significativa según las TVP y TVN, como podemos observar en la Ilustración 10. Los resultados de las matrices de confusión se muestran en el Anexo A, al igual que para el resto de los métodos de estimación.

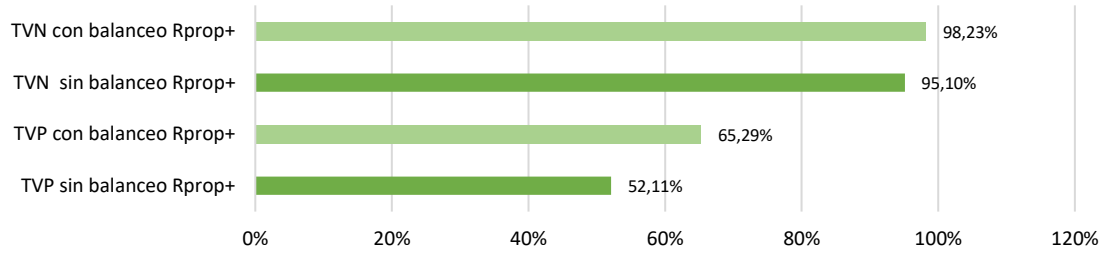


Ilustración 10 Elaboración propia: Tasa de verdaderos positivos y verdaderos negativos RProp+ con / sin balanceo

4.6.2. Red neuronal recurrente con unidades GRU (CNN)

Este modelo utiliza múltiples capas de GRU (*Gated Recurrent Unit*) junto con capas de normalización por lotes y capas de *dropout*. Para desarrollar este modelo, se ha utilizado el paquete "*keras*". Se trata una interfaz a la biblioteca *Keras* de Python. Esto significa que el paquete *Keras* de R permite a los usuarios de R utilizar la biblioteca *Keras*, que está escrita en Python, directamente desde R. Se logra mediante el uso del paquete *reticulate*, que proporciona una interfaz completa entre R y Python. Por otro lado, se hace uso de TensorFlow, que es una biblioteca de código abierto para la computación numérica que facilita el desarrollo de modelos de aprendizaje automático. *Keras* es una API de alto nivel para la construcción y el entrenamiento de modelos de aprendizaje profundo que es más fácil de usar en comparación con TensorFlow, que proporciona un control granular y es más flexible. *Keras* puede correr sobre TensorFlow, lo que significa que usa TensorFlow como su backend para realizar operaciones complejas de aprendizaje automático más fácilmente. Esto combina la simplicidad de *Keras* con la robustez y la flexibilidad de TensorFlow.

A continuación, se explica qué hace cada tipo de capa de la red neuronal CNN y su propósito en el modelo en base al código utilizado:

1. **Capas GRU (*Gated Recurrent Unit*)**

layer_gru(units = 64, input_shape = c(ncol(x_train), 1), return_sequences = TRUE):

- Units: 64 unidades en la capa GRU, que representa el número de neuronas en esa capa.

- Input Shape: Define la forma de los datos de entrada esperados por la capa. Aquí, ncol(x_train) representa el número de características por muestra y 1 indica que se esperan datos unidimensionales por característica.
- Return Sequences: Cuando está configurado en TRUE, la capa devuelve la secuencia completa de salidas para cada muestra (necesario para apilar GRU u otras capas recurrentes).

Estas capas son capaces de mantener un estado interno que les ayuda a procesar secuencias de datos, capturando información temporal.

2. Capas de Normalización por Lotes (Batch Normalization)

layer_batch_normalization( ):
Normaliza las activaciones de la capa anterior en cada lote, es decir, aplica una transformación que mantiene la salida media cercana a 0 y la desviación estándar de la salida cercana a 1. Esto ayuda a mejorar la velocidad y estabilidad del entrenamiento.

3. Capas de Dropout

layer_dropout(rate = 0.2):
Aleatoriamente "apaga" un porcentaje de las unidades (neuronas) durante el entrenamiento, en este caso el 20% (rate = 0.2), lo que ayuda a prevenir el sobreajuste al hacer que el modelo sea menos sensible a la variabilidad específica de los datos de entrenamiento.

4. Capa Densa (Dense Layer)

layer_dense(units = 1, activation = 'sigmoid'):
Una capa densa con una sola unidad.
Activation: Utiliza la función de activación *sigmoid*, previamente explicada, que es común en problemas de clasificación binaria ya que comprime las salidas a un rango entre 0 y 1.

5. Compilación y Entrenamiento del Modelo

compile( ) configura el modelo para el entrenamiento:

- Loss: 'binary_crossentropy' es una función de pérdida común para problemas de clasificación binaria.
- Optimizer: 'adam' es un optimizador que ajusta automáticamente la tasa de aprendizaje y es muy efectivo en la mayoría de las situaciones.
- Metrics: 'accuracy' para medir la precisión del modelo durante el entrenamiento y la validación.

fit( ) entrena el modelo:

- Epochs: Número de veces que el modelo trabajará a través del conjunto completo de datos.
- Batch Size: Número de muestras por actualización del gradiente.
- Validation Data: Datos utilizados para evaluar la pérdida y cualquier métrica del modelo al final de cada época. En este algoritmo, se incluye la muestra de prueba y se realiza *backtest* a la vez que se entrena el modelo.

A continuación, se muestra una gráfica de la evolución de la pérdida y precisión del modelo sobre la muestra de entrenamiento y validación:

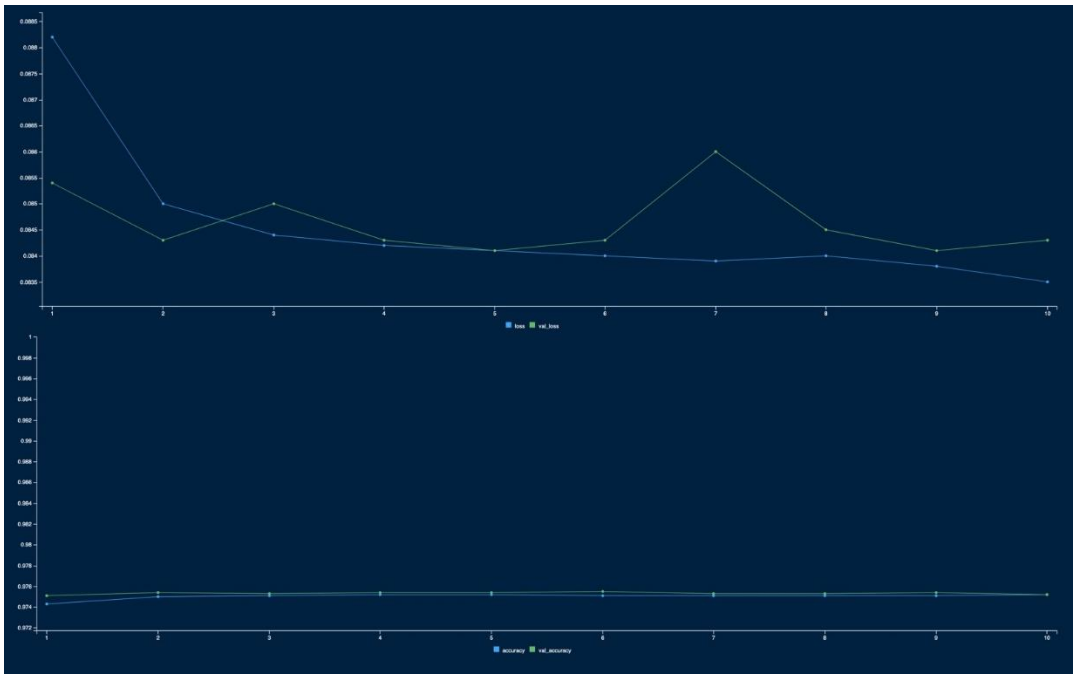


Ilustración 11 Elaboración propia: Pérdida y Accuracy durante el entrenamiento de la red neuronal CNN.

Podemos ver en la Ilustración 11 cómo, al inicio, hay gran diferencia en la pérdida para cada uno de los conjuntos. Pero, a medida que se entrena el modelo, estas métricas se van estabilizando y ajustando a los datos. En la gráfica solo se muestran los 10 primeros entrenamientos del modelo para que se aprecie la diferencia. En la etapa siete, se puede ver que el conjunto de validación no se ajusta del todo a la de entrenamiento, lo cual es normal que ocurra a lo largo de la estimación. Por otra parte, podemos ver que la precisión del modelo es bastante ajustada durante estas diez etapas.

En cuanto al entrenamiento sobre la muestra con y sin balanceo, se obtiene de nuevo mejores resultados sobre la muestra balanceada. Tal y como podemos observar en la Ilustración 12, dónde se compara la Tasa de verdaderos positivos y verdaderos negativos de la matriz de confusión proveniente de la validación que se muestra en el Anexo A.

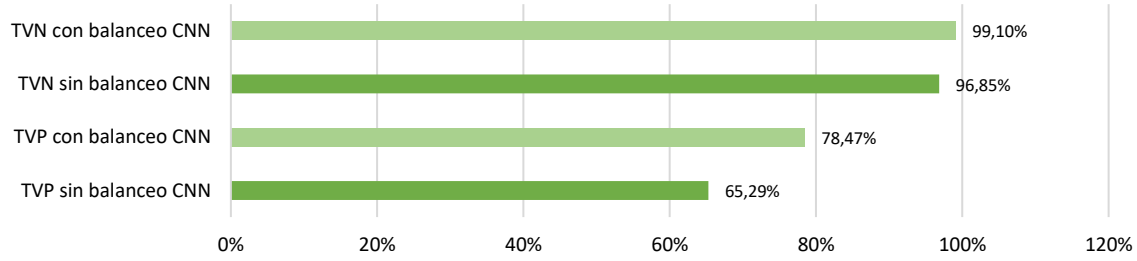


Ilustración 12 Elaboración propia: Tasa de verdaderos positivos y verdaderos negativos del modelo de red neuronal CNN.

4.6.3. Comparativa de muestras balanceadas y sin balancear

Este apartado es meramente ilustrativo. Aunque ya se han presentado las conclusiones finales sobre la elección de los modelos con y sin balanceo, se procede a mostrar en la Ilustración 13 una gráfica de barras de la tasa de verdaderos positivos y en la Ilustración 14 de la tasa de verdaderos negativos donde se comparan los resultados de todos los modelos en conjunto.

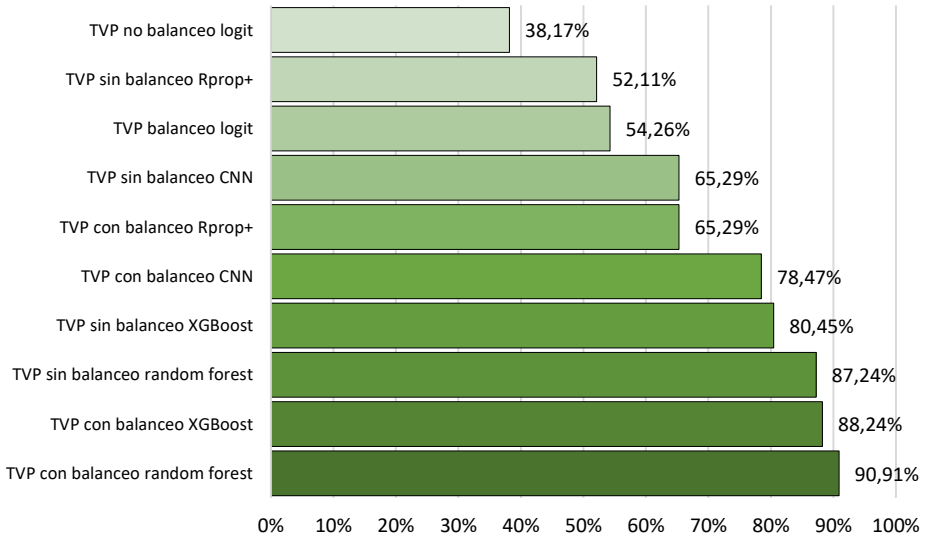


Ilustración 13 Elaboración propia: Tasa de verdaderos positivos de todos los modelos en conjunto.

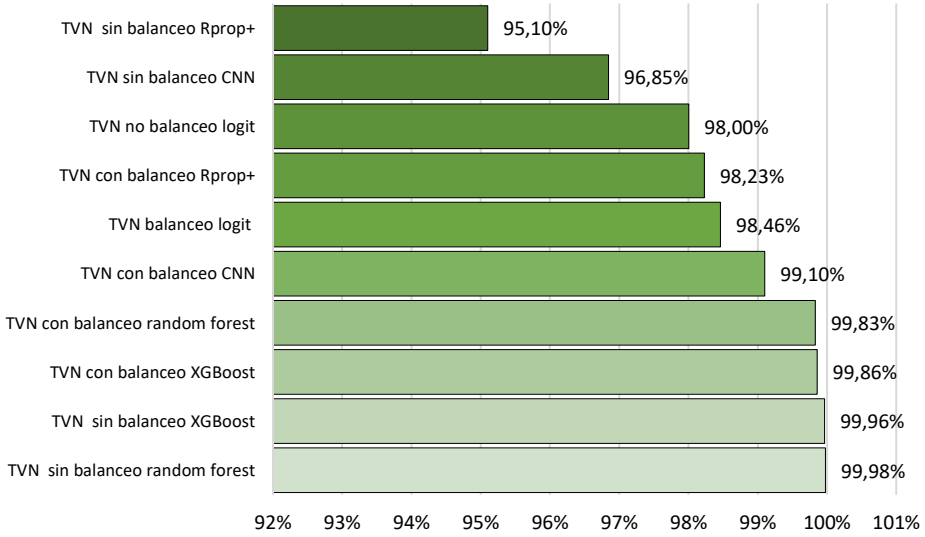


Ilustración 14 Elaboración propia: Tasa de verdaderos negativos de todos los modelos en conjunto.

En línea con las conclusiones que se explicarán en el apartado de validación, los modelos con la muestra balanceada funcionan mejor que sin balaceo debido al bajo ratio de default de la muestra original. Por otro lado, el modelo que mejor funciona es Random Forest con balanceo, seguido de XGBoost con balanceo para la TVP y Random Forest sin balanceo y XGBoost sin balanceo para la TVN.

5 VALIDACIÓN

5.1. Resultados sobre muestra de prueba *in sample*

En este apartado se mostrarán los resultados de las predicciones usando cada uno de los métodos con el fin de evaluar cuál funciona mejor. Se compararán los resultados sobre la muestra de prueba obtenida a partir de la de desarrollo tras la división, como se explica en el apartado Training y Testing samples, que presenta un total de 588.611 observaciones. Todas las predicciones se realizan a partir de la función *predict*.

La matriz de confusión es una herramienta para evaluar el rendimiento de un modelo de clasificación, proporcionando una tabla que compara las predicciones del modelo con los valores reales. En una configuración binaria, incluye Verdaderos Positivos, Falsos Positivos, Verdaderos Negativos y Falsos Negativos, permitiendo calcular otras métricas detalladas como precisión, sensibilidad, especificidad y F1-score. Estas métricas ofrecen una percepción más detallada sobre cómo el modelo maneja diferentes tipos de clasificaciones, siendo particularmente útiles en situaciones donde las clases están desbalanceadas o donde los costos de los errores varían significativamente.

Logit

	No default (0)	Default (1)
No default (0)	558.781	9.650
Default (1)	8.733	11.447

Random Forest

	No default (0)	Default (1)
No default (0)	572.461	1.380
Default (1)	970	13.800

XGBoost

	No default (0)	Default (1)
No default (0)	572.614	1.785
Default (1)	817	13.395

Neural Net (Rprop+)

	No default (0)	Default (1)
No default (0)	563.278	5.269
Default (1)	10.153	9.911

Neural Net (CNN)		
	No default (0)	Default (1)
No default (0)	568.278	3.269
Default (1)	5.153	11.911

Tabla 18 Elaboración propia: Matrices de confusión de los modelos finales escogidos.

Otras medidas:

	Precisión	Sensibilidad	Especificidad	F1
Logit	96,87%	54,26%	98,46%	69,55%
Random Forest	99,60%	90,90%	99,83%	95,06%
XGBoost	99,55%	88,24%	99,86%	93,56%
Neural Net (Rprop+)	97,37%	65,29%	98,22%	78,17%
Neural Net (CNN)	98,57%	78,46%	99,10%	87,37%

Tabla 19 Elaboración propia: Métricas de evaluación de los modelos. Precisión, sensibilidad, especificidad y F1 Score.

Los resultados en la Tabla 19 muestran una clara superioridad de los modelos de Random Forest y XGBoost en términos de rendimiento global, especialmente en las métricas de precisión y F1[5], con valores que alcanzan el 99.60% y 95,06% para Random Forest, y 99,55% y 93,56% para XGBoost, respectivamente. Estos modelos también destacan por su alta especificidad, superando el 99%, lo que indica una excelente capacidad para identificar correctamente las instancias negativas. Por otro lado, los modelos de Redes Neuronales, tanto el *Rprop+* como el CNN, muestran un buen equilibrio entre sensibilidad y especificidad, con el CNN alcanzando una precisión del 98,57% y una F1 de 87.37%, lo que demuestra su eficacia en la clasificación general. Sin embargo, el modelo Logit presenta una menor precisión y sensibilidad, con valores del 96,87% y 54,26%, respectivamente, a pesar de su alta especificidad del 98.46%. Esto sugiere que, aunque es eficaz para identificar correctamente las instancias negativas, su capacidad para detectar correctamente las instancias positivas es limitada. En conjunto, estos resultados indican que los modelos basados en árboles de decisión como Random Forest y XGBoost son los más adecuados para aplicaciones donde la precisión y la capacidad de detección son críticas, mientras que los modelos de redes neuronales pueden ofrecer un buen rendimiento equilibrado en diversas situaciones. Los modelos que utilizan machine learning y deep learning presentan resultados altamente acertados para los datos que se

[5] F1 Score: medida armónica de la precisión y sensibilidad del modelo. $F1 = 2\ x\ \frac{Precisión\ x\ sensibilidad}{Precisión+sensibilidad}$. Fuente: [V7labs.com/blog/f1-score-guide, "*F1 Score in Machine Learning*", Rohit Kundu, 2022].

están manejando. Para Random Forest, XG Boost y Redes Neuronales, el modelo final escogido es sobre la muestra con balanceo de datos, como se ha explicado en el apartado 4.6.3.

A pesar de que los métodos modernos requieren un mayor costo computacional en comparación con los enfoques tradicionales, los beneficios que ofrecen justifican esta inversión. La notable mejora en la precisión de las estimaciones de incumplimiento que proporcionan estos modelos avanzados puede traducirse en decisiones más informadas y eficientes en la gestión de riesgos crediticios. Por lo tanto, aunque el aumento en el costo computacional y la necesidad de experiencia técnica especializada pueden representar desafíos iniciales, los resultados obtenidos en este estudio refuerzan la importancia y el valor añadido de integrar técnicas más modernas en los procesos analíticos para la evaluación de riesgos.

5.1.1. Comparación Random Forest y XGBoost

En este apartado se va a llevar a cabo una comparación de los modelos que mejores resultados han presentado: Random Forest y XGBoost. En este caso, se pretende analizar adicionalmente las medidas de la curva AUC-ROC, el índice Gini y la métrica de Kolmogorov Smirnov (KS), además de las mostradas en el apartado anterior, pero, de forma individual.

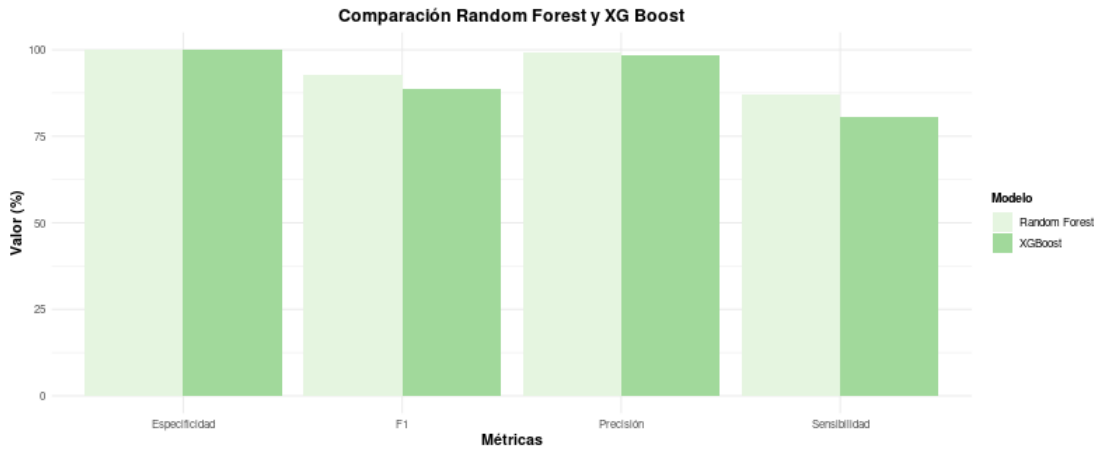


Ilustración 15 Elaboración propia: Comparación de RF y XGBoost de especificidad, F1, precisión y sensibilidad.

Según la Ilustración 15, el modelo de Random Forest demuestra un rendimiento ligeramente superior al de XGBoost en términos de F1 Score, Precisión y Sensibilidad. Ambos modelos tienen una especificidad muy alta, lo que indica que ambos son excelentes para evitar falsos positivos. Pero, con un F1 Score más alto, Random Forest muestra un mejor equilibrio entre precisión y sensibilidad, haciendo que sea una opción ligeramente mejor para aplicaciones donde ambos aspectos son cruciales.
Random Forest sería en este caso más recomendado para predecir la probabilidad de default debido a su mejor desempeño. Esto significa que ofrece un mejor balance entre captar la mayor cantidad de defaults (sensibilidad) y asegurar que las predicciones positivas (clientes que se predicen como impagos en el futuro) sean correctas (precisión). Además, también minimiza el riesgo de perder verdaderos casos de default y clasificar erróneamente a buenos clientes.

Por otro lado, analizamos la curva AUC-ROC. La AUC (Área Bajo la Curva) mide el desempeño del modelo, donde un valor más cercano a uno indica una mejor capacidad predictiva. La curva ROC muestra la relación entre la tasa de verdaderos positivos y la tasa de falsos positivos, ayudando a identificar cuál modelo predice con mayor precisión los riesgos de incumplimiento.

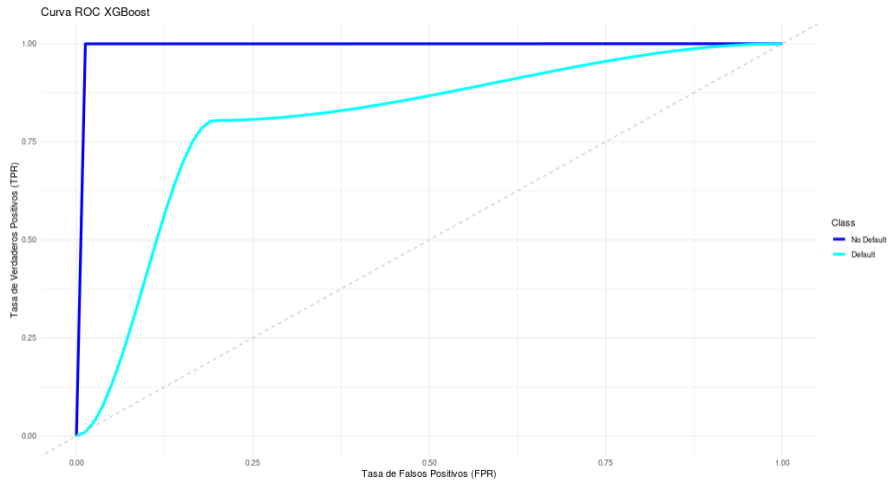


Ilustración 16: Elaboración propia: Curva ROC XGBoost

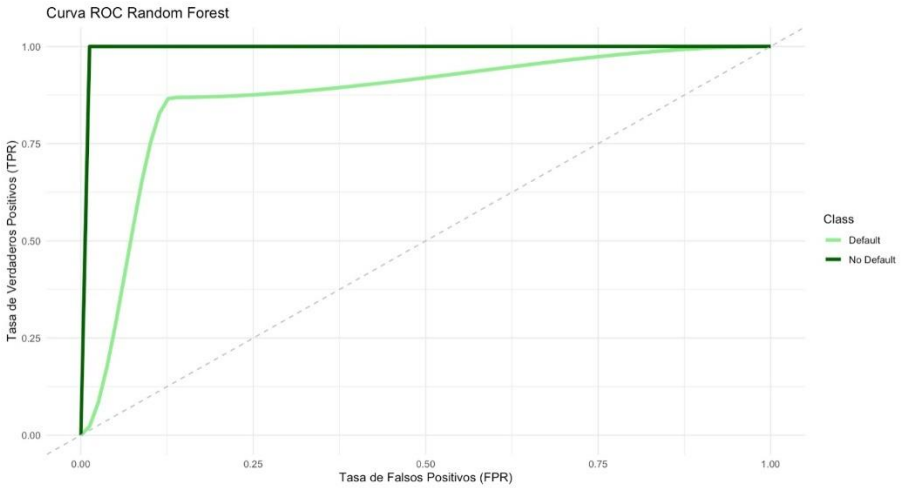


Ilustración 17 Elaboración propia: Curva ROC Random Forest.

Las Ilustraciones 16 y 17 muestran la curva ROC para el modelo XGBoost (azul) y Random Forest (verde). En la Ilustración 18, para el modelo XGBoost obtenemos un AUC para casos no default de 99,96% y para casos de default de 80,44%. Para Random Forest obtenemos un AUC en no defaults de 99,98% y en defaults de 86,89%.

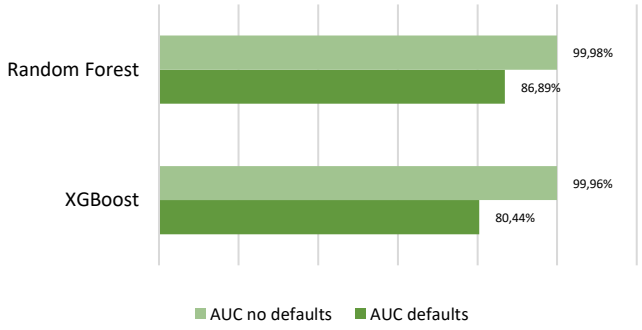


Ilustración 18 Elaboración propia: AUC diferenciando entre defaults y no defaults de XGBoost y Random Forest.

Era de esperar que ambos modelos ajustaras mejor los casos de no incumplimiento debido a la dominancia de esta clase en la muestra total. Sin embargo, el ajuste para los casos de default es también muy bueno en los dos.

En cuanto a la métrica de KS, esta es una herramienta estadística que mide la diferencia entre dos distribuciones de probabilidad. Se utiliza para comparar una muestra con una distribución teórica o para comparar dos muestras entre sí. Específicamente, se basa en la mayor distancia vertical entre las funciones de distribución acumulada (FDC) de las dos distribuciones que se están comparando. Sirve para evaluar si una muestra sigue una distribución específica o para determinar si dos muestras provienen de la misma

distribución. En el contexto en el que nos encontramos, se usa para comparar ambos modelos en la predicción de incumplimientos porque mide qué tan bien cada modelo distingue entre eventos de *default* y no *default*. Al evaluar la mayor diferencia entre las distribuciones acumuladas de estas probabilidades, un valor KS más alto indica que el modelo tiene mejor capacidad discriminativa. Así, comparando los valores KS de ambos modelos, podemos determinar cuál predice con mayor precisión los riesgos de incumplimiento. La forma de calcularlo, entonces, es la siguiente:

$$KS = Max\left[\sum_i P(score = i|buenos) - \sum_i P(score = i|malos)\right]$$

Ecuación 11: Kolmogorov Smirnov.

Por último, también evaluaremos el índice Gini. Este índice se basa en el valor del área bajo la curva. Mide la capacidad de un modelo para diferenciar entre eventos de default y no default. Un índice Gini más alto indica que el modelo tiene una mayor capacidad para distinguir entre estas dos clases. Se calcula:

$$Gini = 2 * AUC - 1$$

Ecuación 12: Gini.

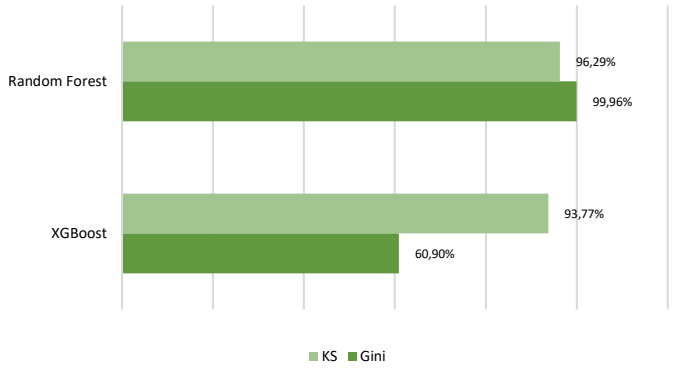


Ilustración 19 Elaboración propia: KS y Gini para XGBoost y Random Forest.

Comparar distintos índices como el valor KS, el AUC y el índice Gini es crucial para tomar decisiones informadas sobre la selección del modelo final, ya que cada métrica ofrece una perspectiva diferente sobre el rendimiento del modelo. En este caso, al analizar dichas medidas en la Ilustración 19, el modelo de Random Forest ha demostrado tener

mejores resultados de evaluación finales en comparación con el modelo XGBoost, lo que indica una mayor capacidad discriminativa y precisión en la predicción de defaults.

5.2. Resultados sobre muestra *Out of Sample*

En este segundo apartado de la fase de validación, se procede a evaluar los dos modelos que mejor han funcionado sobre la muestra de desarrollo para la muestra *out of sample*. Como se ha explicado en el apartado de Exploración de la muestra, esta base de datos *out of sample* pertenece a un periodo posterior a la de desarrollo. Concretamente, pertenece a enero y febrero de 2023 y contiene 1.255.464 observaciones. En ella podemos ver el comportamiento de los clientes de la muestra de desarrollo un año después, para evaluar si realmente han incurrido en default o no. Es de esperar que los modelos funcionen peor sobre esta segunda muestra que sobre la original, ya que son distintos periodos en el tiempo. Pero, nuestro objetivo es que se ajuste lo máximo posible.
Las predicciones arrojan los resultados de la Tabla 20.

XG Boost OOS		
	No default (0)	Default (1)
No default (0)	1.206.770	16.046
Default (1)	16.209	16.439

Random Forest OOS		
	No default (0)	Default (1)
No default (0)	1.202.586	15.125
Default (1)	20.393	17.360

Tabla 20 Elaboración propia: Matrices de confusión de la validación sobre muestra out of sample de XGBoost y Random Forest.

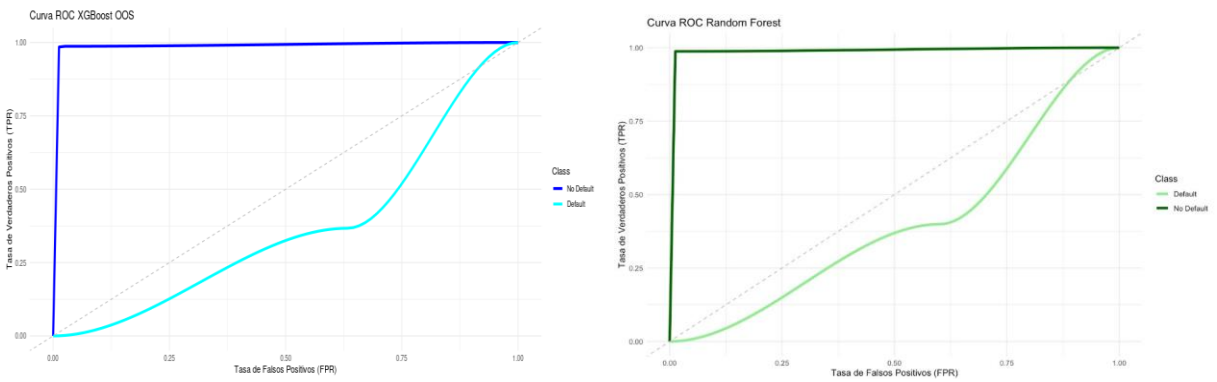


Ilustración 20 Elaboración propia: Curva ROC de defaults y no defaults de XGBoost y Random Forest en la validación de la muestra out of sample.

En base a la Ilustración 20, ambos modelos funcionan excepcionalmente bien a la hora de predecir clientes buenos, pero, en el caso contrario, la situación es diferente y empeora en comparación con el backtest realizado sobre la muestra de desarrollo.
Los resultados de las métricas se muestran en la Ilustración 21.

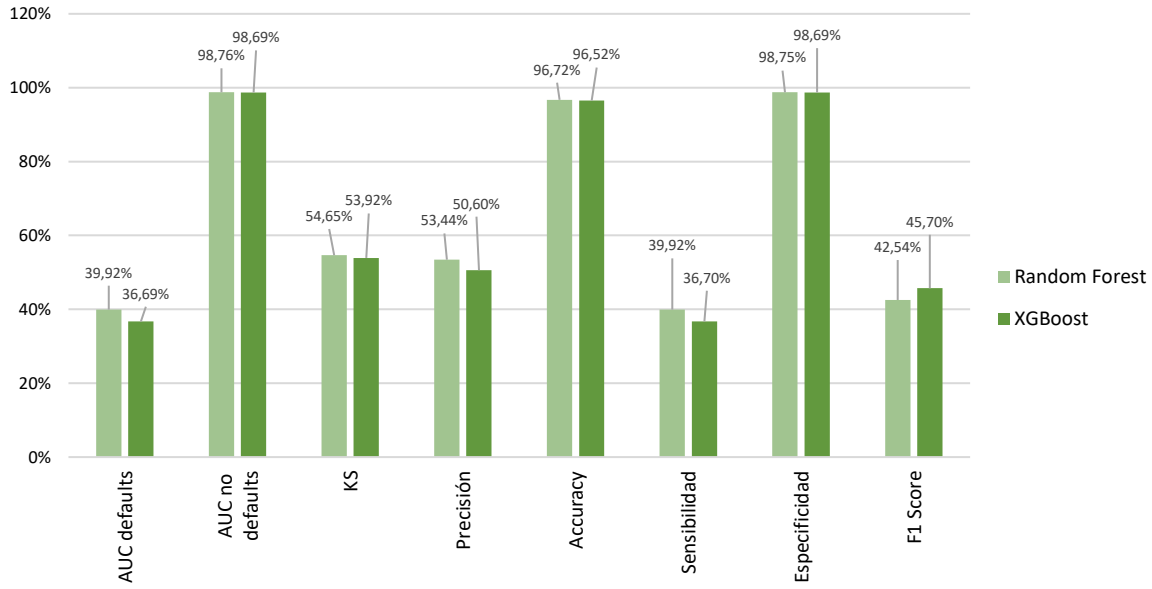


Ilustración 21 Elaboración propia: Métricas de validación de la muestra out of sample de los modelos Random Forest y XGBoost.

Definitivamente, aunque ambos modelos presentan muy buenos resultados, se optaría por utilizar el modelo Random Forest para predecir la PD, ya que demuestra una mejor capacidad para identificar casos de default en comparación con XGBoost, que, aunque también funciona bastante bien, no alcanza la misma precisión en la detección de incumplimientos. Dependiendo de la finalidad que se quieran alcanzar, se podría elegir XGBoost si el objetivo es una buena separación entre clases y un equilibrio entre precisión y sensibilidad. Por otro lado, si el enfoque está en la detección de defaults, de debería elegir Random Forest. Sería ideal desarrollar un algoritmo híbrido que combine ambos modelos para aprovechar las fortalezas de cada uno, logrando así predicciones más ajustadas a la realidad.

Es importante destacar que los resultados obtenidos son bastante sólidos, especialmente considerando que la muestra utilizada para la evaluación no fue la misma empleada para

el entrenamiento del modelo en este caso, lo que generalmente resulta en un rendimiento inferior.

En base a estas conclusiones, junto con los resultados obtenidos del backtesting de la muestra de desarrollo, se ha decidido escoger el modelo de Random Forest para elaborar un perfil de riesgo general de la cartera que estamos estudiando. Esta decisión se fundamenta en la superior capacidad del Random Forest para predecir los casos de default y su alta precisión en la detección de incumplimientos, lo que lo convierte en la opción más adecuada para calcular el riesgo de crédito y la probabilidad de impago en esta cartera.

6 EXPLICABILIDAD DEL MODELO RF Y PERFIL DE RIESGO DE LA CARTERA

Con el fin de afrontar la poca interpretabilidad de los modelos de machine, en esta sección, se aborda la explicabilidad del modelo final de Random Forest y se realiza un perfil de riesgo de la cartera, utilizando dos técnicas clave: la impureza de Gini y los valores de Shapley. La impureza de Gini nos permitirá identificar la importancia de cada variable en la construcción del modelo, proporcionando una visión clara de cómo cada característica contribuye a la toma de decisiones. Por otro lado, los valores de Shapley ofrecerán una descomposición precisa de las predicciones del modelo, ayudándonos a entender el impacto específico de cada variable en las decisiones individuales. Estas herramientas nos permitirán no solo mejorar la transparencia del modelo de Machine Learning, sino también explicar de manera efectiva los factores que influyen en el perfil de riesgo de nuestra cartera. Por tanto, se analizarán cuáles son las variables de mayor importancia en base al modelo final escogido, aquel que devolvía mejores resultados: random forest.

6.1. Impureza de Gini

La impureza de Gini es una medida utilizada para evaluar la calidad de una partición en los algoritmos de clasificación, como los árboles de decisión. Teóricamente, la impureza de Gini para un nodo *t* se define como

$$G(t) = 1 - \sum_{i=1}^{C} p_i^2$$

Ecuación 13: Impureza de Gini

, donde *C* es el número de clases y p_i es la proporción de observaciones de la clase *i* en el nodo *t*. La impureza de Gini alcanza su valor mínimo (0) cuando todas las observaciones del nodo pertenecen a una sola clase, y su valor máximo $1 - \frac{1}{C}$ cuando las observaciones se distribuyen uniformemente entre las clases. Esta métrica se utiliza para elegir la partición óptima en cada paso del crecimiento del árbol, favoreciendo aquellas que resultan en nodos más puros.

A continuación, se muestran en la Ilustración 22 las variables más relevantes utilizando el método de impureza de Gini a la hora de entrenar el modelo para hacer predicciones de comportamiento. Esto se realiza insertando en la función de entrenamiento el parámetro " importance = '*impurity*' ".

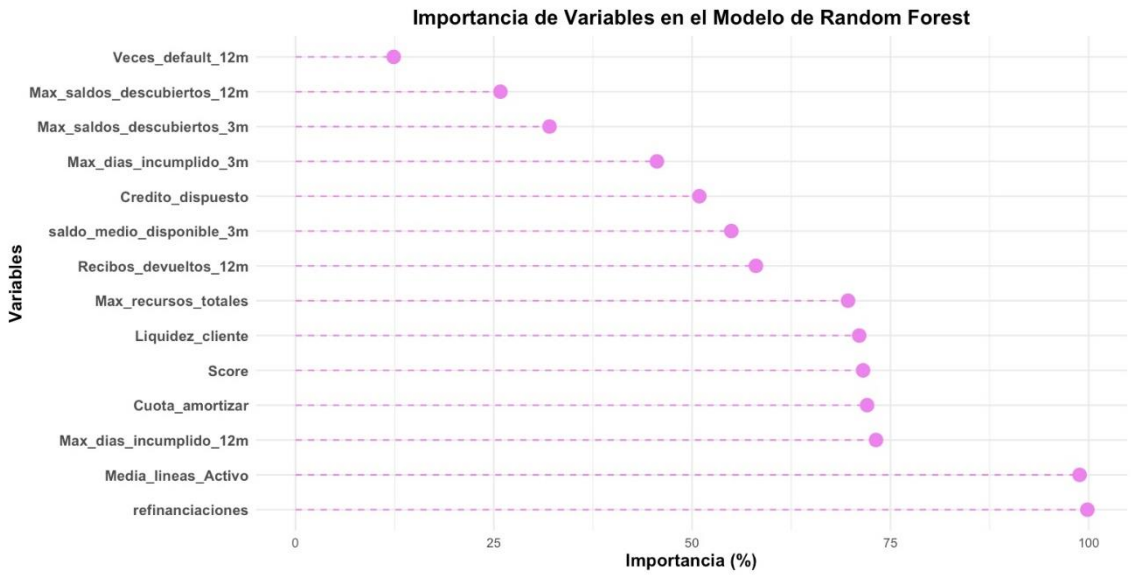


Ilustración 22 Elaboración propia: Importancia de las variables en el modelo Random Forest según la impureza Gini.

- Refinanciaciones: un alto número de refinanciaciones puede sugerir problemas recurrentes en la capacidad del cliente para cumplir con los términos originales de su deuda, lo que es un indicativo de mayor riesgo crediticio.

- Media de líneas de activo: el promedio de líneas de crédito activas que posee el cliente. Un mayor promedio puede señalar una buena aceptación por parte de otras instituciones financieras, pero también implica una mayor exposición a la deuda, lo cual puede ser un factor de riesgo si no se maneja adecuadamente.
- Máximos días de incumplido en 12 meses: un mayor número de días de incumplimiento sugiere dificultades en la gestión de sus obligaciones financieras.
- Cuota a amortizar: cuotas altas pueden ser preocupantes si no están adecuadamente respaldadas por ingresos suficientes, lo que puede aumentar el riesgo de incumplimiento.
- Score: representa una puntuación general del historial crediticio del cliente. Un score elevado generalmente indica un buen historial crediticio, lo que reduce el riesgo percibido. Es imprescindible que la entidad tenga una unidad ocupada en desarrollar modelos de scoring de clientes para llevar a cabo predicciones de probabilidad de incumplimiento, así como de otros parámetros; LGD o EAD, tanto en modelos de provisiones como de capital.
- Liquidez del cliente una mayor liquidez es un indicador positivo, ya que sugiere que el cliente tiene suficiente efectivo o activos fácilmente convertibles en efectivo para cubrir sus deudas inmediatas.
- Máximos recursos totales: clientes con mayores recursos totales tienen una mayor capacidad para hacer frente a sus deudas, lo que disminuye el riesgo de crédito.
- Recibos devueltos en 12 meses: un mayor número de recibos devueltos puede ser un fuerte indicativo de problemas de flujo de caja y aumenta la percepción de riesgo.
- Saldo medio disponible en 3 meses: un saldo medio alto sugiere que el cliente maneja adecuadamente sus fondos y tiene una buena capacidad para cumplir con sus obligaciones financieras.
- Crédito dispuesto: un mayor crédito disponible puede reflejar una buena capacidad de endeudamiento y gestión financiera, reduciendo el riesgo percibido.
- Máximos días de incumplido en 3 meses: un mayor número de días de incumplimiento reciente sugiere problemas financieros actuales, incrementando el riesgo de crédito a corto plazo.
- Máximos saldos descubiertos en 3 meses: saldos descubiertos altos pueden indicar problemas de gestión de fondos y mayor riesgo financiero.

- Máximos saldos descubiertos en 12 Meses: al igual que la anterior, saldos descubiertos elevados en un periodo prolongado también sugieren una gestión financiera deficiente, aumentando el riesgo crediticio. Es cierto que, para nuestro modelo, es más relevante la cantidad de saldo descubierto a menor plazo que a largo.
- Veces en default en 12 meses: a mayor número de veces en default, mayor probabilidad de incumplimiento. Esta variable aparece como la menos significativa, pero esto se debe a la cantidad de missings que presenta. Como se ha explicado previamente, los missings en esta variable son a causa de clientes que no tienen una antigüedad mayor a un año en la base de datos, por lo tanto, no tienen registros de default en esta variable.

6.2. Valores Shapley

Por otro lado, se calculan los valores de Shapley. Dichos valores, basados en la teoría de juegos, cuantifican la contribución de cada característica a una predicción específica. Teóricamente, el valor de Shapley para una característica *i* en una predicción se calcula como la media ponderada de las diferencias en la predicción cuando la característica *i* se incluye y se excluye de todos los posibles subconjuntos de características. La fórmula para calcular los valores phi se muestra en la Ecuación 14.

$$\phi_i = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!\,(|N| - |S| - 1)!}{|N|!} [f(S \cup \{i\}) - f(S)]$$

Ecuación 14: Valores de Shapley

Observamos que *S* es un subconjunto de características, *N* es el conjunto total de características, y $f(S)$ es la predicción del modelo con el subconjunto *S*. En R, utilizando el paquete iml, los valores de Shapley se calculan con la función *Shapley$new(predictor, x.interest),* donde predictor es el modelo y *x.interest* es la observación para la cual se calculan los valores Shapley.

Ambos métodos son técnicas de explicabilidad en modelos de ML, pero difieren en su enfoque y aplicación. La impureza de Gini mide la calidad de una partición en los nodos

de un árbol de decisión, evaluando la homogeneidad de las clases en cada nodo, y se usa principalmente para seleccionar las divisiones óptimas durante el entrenamiento del modelo. Por otro lado, los valores de Shapley, basados en la teoría de juegos, cuantifican la contribución individual de cada característica a una predicción específica, proporcionando una explicación detallada del impacto de cada variable en las decisiones del modelo.

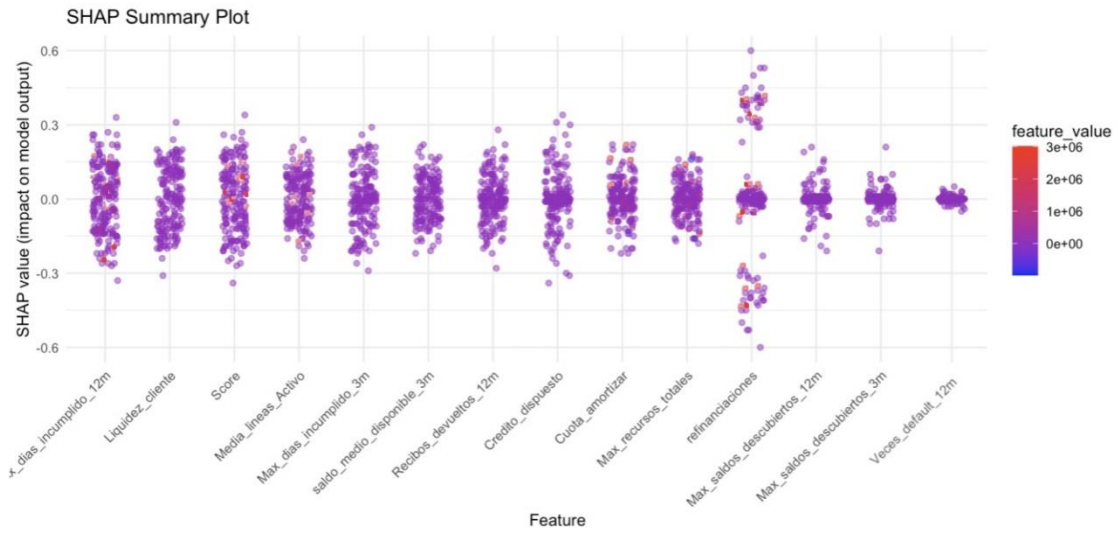


Ilustración 23 Elaboración propia: Valores de Shapley del modelo Random Forest.

En la Ilustración 23 podemos ver la distribución de los valores de Shapley de modelo sobre una muestra de tres mil datos. Las variables mas influyentes son Refinanciaciones, Max_dias_incumplido_12m, Score, Media_lineas_Activo y Cuota_amortizar. Un color rojo indica un valor alto de la característica, mientras que un color azul indica un valor bajo. En dichas variables, los puntos rojos indican que los valores altos de estas características tienen un impacto significativo en la predicción del modelo, ya sea aumentando o disminuyendo el valor predicho dependiendo de la dispersión vertical de los puntos. La predominancia del color morado en los puntos de la gráfica sugiere que muchos de los valores de las características están en un rango medio, ni extremadamente altos ni extremadamente bajos. Esto muestra que la mayoría de las observaciones en el conjunto de datos tienen valores intermedios para esas características, y estos valores intermedios tienen un impacto significativo en la predicción de la variable objetivo en el modelo.

Por otro lado, la variable "Refinanciaciones" muestra valores extremos en la gráfica SHAP porque su impacto indiviaul en la predicción del modelo es significativo y varía considerablemente. Los valores extremos indican que, dependiendo de las circunstancias específicas, las refinanciaciones pueden aumentar o disminuir drásticamente la predicción. Esta variabilidad se debe a que las refinanciaciones están altamente correlacionadas con el riesgo de crédito, impactando fuertemente la PD.

Para muchas variables, como "Score" y "Liquidez_cliente", los valores SHAP están distribuidos de manera bastante simétrica alrededor de cero, lo que sugiere que estas variables tienen un impacto tanto positivo como negativo en la predicción del modelo dependiendo de su valor específico.

Este apartado de interpretabilidad y perfil de riesgo es muy relevante tanto para la propia entidad como para otras entidades financieras en el momento de estructurar y almacenar información sobre clientes en sus bases de datos. Al identificar qué variables son más significativas para explicar el comportamiento crediticio de un cliente, pueden optimizar sus procesos de recopilación de datos, enfocándose en los indicadores más reveladores y pertinentes. Además, este es un método para afrontar la poca interpretabilidad de los modelos de machine learning, de forma que nos ayuda a entender el sustento de las decisiones que se toman durante del entrenamiento de estos. Esto reafirma que la importancia de tener bases de datos bien elaboradas y con información relevante es crucial para el desarrollo de modelos que reflejen fielmente la situación financiera de una empresa. Bases de datos precisas y completas permiten a los modelos de evaluación de riesgo capturar con mayor exactitud los factores críticos que influyen en el comportamiento financiero de los clientes. Esto no solo mejora la precisión en la evaluación del riesgo crediticio, sino que también facilita la toma de decisiones estratégicas, promoviendo una gestión de riesgos más eficaz y una mejor alineación de los productos financieros con las necesidades y perfiles de los clientes.

7 CONCLUSIONES

- Los resultados del análisis indican que el método Random Forest sobre la muestra balanceada es el que mejor se ajusta a la cartera de clientes, ofreciendo las estimaciones más precisas según las métricas evaluadas, incluyendo precisión, sensibilidad, especificidad, F1 score, AUC-ROC, Gini y KS. Le sigue de cerca el método de Extreme Gradient Boosting, que también muestra un desempeño notablemente alto y resultados bien ajustados para la muestra balanceada.

- En contraste, el modelo logístico, que requiere también de un balanceo previo de los datos para mejorar su rendimiento, es el que presenta los resultados menos favorables. Estas diferencias subrayan la superioridad de los modelos de aprendizaje automático avanzado en la tarea de predecir la probabilidad de incumplimiento.

- El método de balanceo de datos, que combina submuestreo de la clase mayoritaria y sobremuestreo de la clase minoritaria, mejora significativamente las predicciones en modelos de machine learning, deep learning y el modelo logit tradicional al abordar el problema del desequilibrio de clases. Este enfoque equilibra la representación de ambas clases en el conjunto de datos, lo que permite al modelo aprender patrones relevantes de la clase minoritaria sin ser dominado por la clase mayoritaria.

- Además, es crucial destacar la importancia de las técnicas de validación cruzada en la optimización del desempeño de los modelos de Machine Learning. Esta metodología no solo ha demostrado ser eficaz para reducir el costo computacional, sino que también mejora la calidad de las estimaciones al permitir que los modelos sean probados en múltiples subconjuntos de datos antes de realizar predicciones finales. Implementar validación cruzada asegura que los modelos sean robustos y generalizables, optimizando así su capacidad para predecir incumplimientos en carteras de clientes con mayor precisión y eficiencia.

- Aunque está bien documentado que los métodos basados en inteligencia artificial superan en rendimiento a los métodos convencionales como las regresiones logísticas o lineales, presentan el desafío de la "caja negra" ("*black box*"), es decir, una menor interpretabilidad. Esta falta de transparencia es una preocupación significativa para los reguladores, quienes son reticentes a aceptar estos métodos para realizar estimaciones que impacten directamente en el balance final. Por lo tanto, es esencial abordar este problema para facilitar una mayor integración de las técnicas de aprendizaje automático en prácticas financieras, lo cual deriva en técnicas como las aplicadas en el presente estudio: valores de Shapley e impureza de Gini.

- Por último, a la hora de analizar la interpretabilidad de los modelos utilizando la impureza de Gini y valores Shapley, concluimos que las características más relevantes en el modelo final de random forest son el número de refinanciaciones y la media de líneas de activo del cliente. Estás variables son clave a la hora de entrenar el modelo. De cerca le siguen el número de días de incumplimiento, la cuota total a amortizar, el score o puntuación, la liquidez del cliente y el máximo de recursos totales. Esto es interesante tanto para la entidad financiera propietaria de la muestra que se ha estudiado como para otras entidades ya que ayuda a conocer qué variables son relevantes para almacenar en las bases de datos de sus clientes y, con ellas, desarrollar modelos con mayor capacidad predictiva.

7.1. Futuros campos de investigación

- Sería valioso realizar en el futuro un estudio para explorar qué tipos de redes neuronales resultarían más efectivos para estimar la probabilidad de incumplimiento de una cartera de crédito. En el trabajo actual, se ha intentado ajustar los parámetros de las redes neuronales, pero no se han logrado resultados significativamente superiores en comparación con otros métodos de estimación. Identificar la arquitectura neural óptima podría proporcionar avances en la precisión de los modelos de riesgo de crédito. Además, con los últimos avances

tecnológicos, se están desarrollando muchos paquetes que facilitan el uso de estas técnicas para cualquier usuario.

- Se podrían volver a probar los métodos sobre las muestras balanceadas de otra manera, es decir; sobre-muestrear en menor medida para perder menos precisión de la información o sub-muestrear menos con el mismo objetivo. De tal manera que se vuelvan a comparar los resultados y se obtenga una métrica exacta en forma de intervalo para saber en qué medida esto puede ser favorable a los resultados finales del modelo. Esto es de gran importancia porque la técnica del balanceo de la muestra se utiliza ampliamente, en especial en el campo de la investigación, y puede afectar significativamente a los resultados finales.

- Sería relevante realizar un estudio sobre el cálculo de la probabilidad de incumplimiento de una cartera similar a la utilizada en este trabajo, pero incorporando una dimensión temporal que permita predecir, no solo si un cliente hará default, sino también, el momento específico en que esto podría ocurrir. Este enfoque permitiría un análisis más detallado basado en las características individuales de cada cliente, optimizando así las estrategias de intervención temprana y la gestión del riesgo de crédito en el tiempo, lo que podría traducirse en una mitigación más eficaz del riesgo para las entidades.

BIBLIOGRAFÍA

[1] L. Breiman, “Random Forests,” *Machine Learning* , vol. 45, pp. 5-32, 2001.

[2] T. Hastie, J. Friedman and R. Tibshirani, “Random Forests,” in *The Elements of Statistical Learning*, New York, Springer, 2009, pp. 587-604.

[3] R. L. H. J. Noriega JP, «Machine Learning for Credit Risk Prediction: A Systematic Literature Review.,» *MDPI*, p. 169, 2023.

[4] Cayan Atreio Portela Bárcena Saavedra, «Comparing lifetime estimates of probability of default for refinancing operations with survival analysis and ensemble methods,» *Taylor & Francis*, vol. 10, pp. 1-26, 2024.

[5] A. N. Y. Kyriakos Georgiou, «DEEP NEURAL NETWORKS FOR PROBABILITY,» *Journal of Industrial and Management Optimization*, 2023.

[6] M. Addo, «Credit risk analysis using machine and deep learning models,» *Risks*, p. 38, 2028.

[7] D. Beerbaum, «Significant increase in credit risk according to ifrs 9: Implications for financial institutions,» *International Journal of Economics & Management Sciences*, pp. 1-3, 2015.

[8] I. B. a. J. Bilbao, «Overfitting problem and the over-training in the era of data: Particularly for artificial neural networks,» de *Eighth International Conference on Intelligent Computing and Information Systems*, 2017.

[9] N. L. Fabio Sigrist, «Machine learning for corporate default risk: Multi-period prediction, frailty correlation, loan portfolios, and tail probabilities,» *European*

Journal of Operational Research, Vols. %1 de %2Volume 305, Issue 3, pp. 1390-1406, 2023.

[10] Junliang Wang, «Research on finance Credit Risk Quantification Model Based on Machine Learning Algorithm,» *Academic Journal of Science and Technology,* vol. 10 , nº 1, 2024.

[11] Alessandro Bitetto, «Machine learning and credit risk: Empirical evidence from small- and mid-sized businesses,» *Socio-Economic Planning Sciences,* vol. 90, 2023.

[12] Sarder Abdulla Al Shiam, «Credit Risk Prediction Using Explainable AI,» *Journal of Business and Management Studies,* vol. 6, nº 2, p. 61–66, 2024.

[13] B. Efron, «Bootstrap confidence intervals for a class of parametric problems,» *Biometrika,* vol. Volume 72, nº Issue 1, p. Pages 45–58, 1979.

[14] E. R. J. Romero, «Modelo de Machine Learning basado en la Maquina Potenciadora de Gradiente de Luz para predecir la probabilidad de impago en clientes de la cartera de Tarjeta de Crédito,» *Universidad Nacional Mayor de San Marcos, Lima, Perú: Revista de investigación de sistemas e informática* , pp. 155-168, 2023.

[15] A. B. &. S. K. B. &. Ardhendu, «Credit risk evaluation: a comprehensive study,» *Springer Science+Business Media, LLC, part of Springer Nature 2022,* 17 February 2022.

[16] Christopher A. Ramezan, «Effects of Training Set Size on Supervised Machine-Learning Land-Cover Classification of Large-Area High-Resolution Remotely Sensed Data,» *Remote Sens,* vol. 13, p. 368, 2021.

[17] Paolo Giudici, «Artificial Intelligence risk measurement,» *Expert Systems with Applications. Elsevier Ltd. University of Pavia,* nº 121 220, p. 235, August 2023.

[18] Ayoub El-Qadi 1, «Credit Risk Scoring Forecasting Using a Time Series Approach,» *Physical Sciences Forum,* pp. 18-22, July 2022.

[19] J. Chase, «Estimating Parameters Required for Credit Risk Modeling,» *Elsevier: Risk Management,* nº Charpter 11, pp. 235-252, New York, NY. 2006.

[20] Branka Hadji Misheva, «EXPLAINABLE AI IN CREDIT RISK MANAGEMENT,» *Swiss National Science Foundation within the project "Mathematics and Fintech",* nº 2103.00949, March 2, 2021.

[21] D. Mhlanga, «Financial Inclusion in Emerging Economies: The Application of Machine Learning and Artificial Intelligence in Credit Risk Assessment.,» *International Journal of FinancialStudies,* nº 39, 2021.

Anexo A: Comparación predicciones sobre muestra sin balanceo y con balanceo.

- **Modelo Logit**

Muestra sin balanceo

Confusion matrix

Binary predictions

	0	1
0	567566	5865
1	11560	3620

Precision
0,970396408

Sensitivity
0,381655245

Specificity
0,980038886

Muestra con balanceo

Confusion matrix

Binary predictions

	0	1
0	558781	9650
1	8733	11447

Precision
0,968768847

Sensitivity
0,542588994

Specificity
0,984611833

- **Random Forest**

Muestra sin balanceo

Confusion matrix

Binary predictions

	0	1
0	573303	1937
1	128	13243

Precision
0,996491741

Sensitivity
0,872397892

Specificity
0,999776782

Muestra con balanceo

Confusion matrix

Binary predictions

	0	1
0	572461	1380
1	970	13800

Precision
0,99600755

Sensitivity
0,909090909

Specificity
0,998308428

- **Gradient Boosting**

Muestra sin balanceo

Confusion matrix

Binary predictions

	0	1
0	573229	2968
1	202	12212

Precision
0,99461444

Sensitivity
0,804479578

Specificity
0,999647734

Muestra con balanceo

Confusion matrix

Binary predictions

	0	1
0	572614	1785
1	817	13395

Precision
0,995579423

Sensitivity
0,882411067

Specificity
0,998575243

- **Red neuronal (Rprop+)**

Muestra sin balanceo

Confusion matrix

Binary predictions

	0	1
0	545341	7269
1	28090	7911

Precision
0,939928068

Sensitivity
0,521146245

Specificity
0,951014159

Muestra con balanceo

Confusion matrix

Binary predictions

	0	1
0	563278	5269
1	10153	9911

Precision
0,973799334

Sensitivity
0,652898551

Specificity
0,982294295

- **Red neuronal (CNN)**

Muestra sin balanceo

Confusion matrix

Binary predictions

	0	1
0	555341	5269
1	18090	9911

Precision
0,960315047

Sensitivity
0,652898551

Specificity
0,968453048

Muestra con balanceo

Confusion matrix

Binary predictions

	0	1
0	568278	3269
1	5153	11911

Precision
0,985691739

Sensitivity
0,784650856

Specificity
0,99101374

Anexo B: Tabla de Abreviaturas

Abreviatura	Significado
PD	Probabilidad de default
LDP	Low Default Portfolio
ML	Machine Learning
AIC	Akaike information criteria
XGBoost	Extreme Gradient Boosting
WoE	Weight of Evidence
AUC	Área bajo la curva ROC
DQ	Data Quality
DA	Data Availavility
FN	False negative
FP	False positive
TN	True negative
TP	True positive
TNR	True Negative Rate
TPR	True Positive Rate
FNR	False Negative Rate
FPR	False Positive Rate
IV	Information Value
KS	Kolmogorov-Smirnoff
OOS	Out of Sample
RF	Random Forest
IFRS9	Norma Internacional de Información Financiera
OOB	Out of Bag
IA	Inteligencia Artificial
ReLU	Función de Unidad Lineal Rectificada
LSTM	Long Short Term Memory
CNN	Convolutional Neural Networks
GRU	Gated Recurrent Unit
Rprop+	Resilient Propagation Plus
RNN	Recurrent Neural Networks

Anexo C: Código

```
##### Este código está escrito en R y pertenece al TFM de Raquel
Martínez Quiroga - UC3M ##############
##### Título: Modelización del riesgo de crédito con técnicas de
machine y deep learning ################################

library(readr)
library(lubridate)
data <- read_delim("/Workspace/Val_POPS/Raquel/TFM/Muestra_total.csv",
                       delim = ";"), trim_ws = TRUE)
names(data) = c("Cod_persona", "var_obj", "refinanciaciones",
"Max_dias_incumplido_12m", "Max_dias_incumplido_3m",
"Credito_dispuesto", "Liquidez_cliente",  "saldo_medio_disponible_3m",
"Media_lineas_Activo", "Cuota_amortizar", "Veces_default_12m",
"Recibos_devueltos_12m",
"Max_recursos_totales","Max_saldos_descubiertos_3m",
"Max_saldos_descubiertos_12m", "Score", "fecha")
summary(data)


################## MISSINGS

colSums(is.na(data))
## maximo dias incumplido a 12 m se sustituye por 0
data$Max_dias_incumplido_12m[is.na(data$Max_dias_incumplido_12m)] <- 0
# Recibos_devueltos_12m SUSTITUIDO POR 0
data$Recibos_devueltos_12m[is.na(data$Recibos_devueltos_12m)] <- 0
## El SCORE se sustituye por la media de scores
data$Score[is.na(data$Score)] <- mean(data$Score, na.rm = TRUE)
#######
data$Liquidez_cliente[is.na(data$Liquidez_cliente)] <-
mean(data$Liquidez_cliente, na.rm = TRUE)
data$saldo_medio_disponible_3m[is.na(data$saldo_medio_disponible_3m)]
<- mean(data$saldo_medio_disponible_3m, na.rm = TRUE)
data$Media_lineas_Activo[is.na(data$Media_lineas_Activo)] <-
mean(data$Media_lineas_Activo, na.rm = TRUE)

######

data$Max_saldos_descubiertos_12m[is.na(data$Max_saldos_descubiertos_12
m)] <- mean(data$Max_saldos_descubiertos_12m, na.rm = TRUE)
data$Max_saldos_descubiertos_3m[is.na(data$Max_saldos_descubiertos_3m)
] <- mean(data$Max_saldos_descubiertos_3m, na.rm = TRUE)
data$Max_recursos_totales[is.na(data$Max_recursos_totales)] <-
mean(data$Max_recursos_totales, na.rm = TRUE)
##  CUOTA AMORTIZAR se sustituye por la media de scores
data$Cuota_amortizar[is.na(data$Cuota_amortizar)] <-
mean(data$Cuota_amortizar, na.rm = TRUE)

# eliminamos outliers de la variable score
#hist(data$Score)
count_score <- sum(data$Score < 100, na.rm =TRUE)
data <- subset(data, Score >= 100)


##################################
##########  ENTRENAMIENTO Y PRUEBA
##################################
```

```
total_defaults <- sum(data$var_obj == 1)
total_no_defaults <- sum(data$var_obj == 0)
default_rate_total <- total_defaults / nrow(data)
# Separa los casos de default y no default en dos conjuntos
defaults <- data[data$var_obj == 1, ]
no_defaults <- data[data$var_obj == 0, ]

# entrenamiento y prueba manteniendo el default rate
set.seed(223)
train_defaults <- defaults[sample(nrow(defaults), 0.7 *
total_defaults), ]
test_defaults <- defaults[setdiff(rownames(defaults),
rownames(train_defaults)), ]
train_no_defaults <- no_defaults[sample(nrow(no_defaults), 0.7 *
total_no_defaults), ]
test_no_defaults <- no_defaults[setdiff(rownames(no_defaults),
rownames(train_no_defaults)), ]
train <- rbind(train_defaults, train_no_defaults)
test <- rbind(test_defaults, test_no_defaults)
default_rate_train <- sum(train$var_obj == 1) / nrow(train)
default_rate_test <- sum(test$var_obj == 1) / nrow(test)


############  TRAINING SAMPLE BALANCEADO ##################


training_goods2 <- subset(train, var_obj == 0)
training_bads2 <- subset(train, var_obj == 1)

# Eliminamos el 50% de la muestra para submuestreo de los no default

n_subsample <- round(0.50 * nrow(training_goods2))
indices_subsample <- sample(nrow(training_goods2), n_subsample,
replace = FALSE)
training_goods_subsampled <- training_goods2[indices_subsample, ]
training_goods <- training_goods2[-indices_subsample, ]

# Sobremuestreo de defaults
# tamaño inicial del remuestreo (40% de los datos originales)

initial_sample_size <- round(0.40 * nrow(training_bads2))
resampled_bads <- training_bads2[0, ]

# remuestreo

while (nrow(resampled_bads) < 0.85 * nrow(training_bads2)) {# Se toma
un 10% de los datos originales y se generan un 10% más hasta alcanzar
el 85% del total
  sample_indices <- sample(1:nrow(training_bads2),
initial_sample_size, replace = TRUE)
  sampled_data <- training_bads2[sample_indices, ]
  resampled_bads <- rbind(resampled_bads, sampled_data)
}

# Unir datasets

training_bads <- rbind(training_bads2, resampled_bads)
train_b <- rbind(training_bads, training_goods)

#### EXPLORACION NUEVA MUESTRA
table(train_b$var_obj)
```

```
##### graficos y outliers
#######gráfico de correlaciones

install.packages("ggplot2")
library(ggplot2)
install.packages("corrplot")
library(corrplot)

# matriz de Correlación

data1 <- data[, !names(data) %in% c("Cod_persona", "fecha")]
cor_matrix <- cor(data1)
dev.new(width = 10, height = 10)
plot.new()
dev.off()
corrplot(cor_matrix, method = "number", type = "upper", diag = TRUE,
tl.cex = 0.7)


####################################
################## HISTOGRAMAS


muestra <- data
hist(muestra$refinanciaciones)
hist(muestra$Max_dias_incumplido_12m)
hist(muestra$Max_dias_incumplido_3m)
hist(muestra$Credito_dispuesto)
hist(muestra$Liquidez_cliente)
hist(muestra$saldo_medio_disponible_3m)
hist(muestra$Media_lineas_Activo)
hist(muestra$Cuota_amortizar)
hist(muestra$Veces_default_12m)
hist(muestra$Recibos_devueltos_12m)
hist(muestra$Max_recursos_totales)
hist(muestra$Max_saldos_descubiertos_3m)
hist(muestra$Max_saldos_descubiertos_12m)
hist(muestra$Score)
count_score <- sum(muestra$Score < 100, na.rm =TRUE)
summary(muestra$Veces_default_12m)
boxplot(muestra$Max_saldos_descubiertos_12m)
quantile(muestra$Max_saldos_descubiertos_12m, 1)
sum(data$Max_saldos_descubiertos_12m > 100000 )
#data2 <- subset(muestra, Max_saldos_descubiertos_12m <= 100000)

############################
install.packages("car")
library(car)

########## LOGIT ##########

# se elimina Max_saldos_descubiertos_3por no ser significativa
modelo_logistico <- glm(var_obj ~
refinanciaciones+Max_dias_incumplido_12m+

Max_dias_incumplido_3m+Credito_dispuesto+Liquidez_cliente+

saldo_medio_disponible_3m+Media_lineas_Activo+Cuota_amortizar+
                          Veces_default_12m+Recibos_devueltos_12m+
                          #Max_recursos_totales+
```

```
                          Max_saldos_descubiertos_12m+Score,
                        train, family = binomial)

summary(modelo_logistico)
residuals <- modelo_logistico$residuals
summary(residuals)
quantile(residuals, 0.0001)
quantile(residuals, 0.1)
quantile(residuals, 0.25)
quantile(residuals, 0.5)
quantile(residuals, 0.75)
quantile(residuals, 0.9)
quantile(residuals, 0.95)
quantile(residuals, 0.99)
indice_max <- which.max(residuals)
observacion_max <- train[indice_max, ]
datos_sin_max <- train[-indice_max, ]

### eliminamos outliers -- reisudos > quantil 99%  (valores muy
extremos en los resiudos)

percentil_99 <- quantile(residuals, 0.99)
indices_a_eliminar <- residuals > percentil_99
datos_filtrados <- train[!indices_a_eliminar, ]
modelo_logistico_actualizado <- glm(var_obj ~ refinanciaciones +
Max_dias_incumplido_12m +
                                      Max_dias_incumplido_3m +
Credito_dispuesto + Liquidez_cliente +
                                      saldo_medio_disponible_3m +
Media_lineas_Activo + Cuota_amortizar +
                                      Veces_default_12m +
Recibos_devueltos_12m +
                                      #Max_recursos_totales +
                                      Max_saldos_descubiertos_12m +
Score,
                                    datos_filtrados, family =
binomial)
summary(modelo_logistico_actualizado)
summary(modelo_logistico_actualizado$residuals)

#residuals_99 <- modelo_logistico_actualizado$residuals
### eliminamos outliers -- reisudos < quantil 1%  (valores muy
extremos en los resiudos)
residuals2 <- modelo_logistico_actualizado$residuals
percentil_01 <- quantile(residuals2, 0.01)
indices_a_eliminar2 <- residuals2 < percentil_01
datos_filtrados2 <- datos_filtrados[!indices_a_eliminar2, ]
modelo_logistico_actualizado2 <- glm(var_obj ~ refinanciaciones +
Max_dias_incumplido_12m +
                                      Max_dias_incumplido_3m +
Credito_dispuesto + Liquidez_cliente +
                                      saldo_medio_disponible_3m +
Media_lineas_Activo + Cuota_amortizar +
                                      Veces_default_12m +
Recibos_devueltos_12m +
                                      #Max_recursos_totales +
                                      Max_saldos_descubiertos_12m +
Score,
                                    datos_filtrados2, family =
binomial)
summary(modelo_logistico_actualizado2)
```

```
summary(modelo_logistico_actualizado2$residuals)
modelo_logistico <- modelo_logistico_actualizado2

################## VIF
vif(modelo_logistico)


############################### PREDICCIONES


#  predicciones en el conjunto de datos de prueba
predicciones_test <- predict(modelo_logistico, test, type =
"response")
mean(predicciones_test)
summary(predicciones_test)
hist(predicciones_test)
# umbral de decisión
umbral <- 0.5
predicciones_binarias <- ifelse(predicciones_test > umbral, 1, 0)
# matriz de confusión
conf_matrix <- table(test$var_obj, predicciones_binarias)
# precisión
precision <- sum(diag(conf_matrix)) / sum(conf_matrix)
# curva ROC (AUC-ROC)
auc_roc <- auc(roc(test$var_obj, predicciones_test))
# sensibilidad y la especificidad
sensibilidad <- conf_matrix[2, 2] / sum(conf_matrix[2, ])
especificidad <- conf_matrix[1, 1] / sum(conf_matrix[1, ])


########## LOGIT BALANCEADO##########


# se elimina Max_saldos_descubiertos_3por no ser significativa
modelo_logistico_b <- glm(var_obj ~
refinanciaciones+Max_dias_incumplido_12m+

Max_dias_incumplido_3m+Credito_dispuesto+Liquidez_cliente+

saldo_medio_disponible_3m+Media_lineas_Activo+Cuota_amortizar+

Veces_default_12m+Recibos_devueltos_12m+Max_recursos_totales+
                          Max_saldos_descubiertos_12m+Score,
                        train_b, family = binomial)

summary(modelo_logistico_b)
residuals_b <- modelo_logistico_b$residuals
summary(residuals_b)
quantile(residuals_b, 0.0001)
quantile(residuals_b, 0.1)
quantile(residuals_b, 0.25)
quantile(residuals_b, 0.5)
quantile(residuals_b, 0.75)
quantile(residuals_b, 0.9)
quantile(residuals_b, 0.95)
quantile(residuals_b, 0.99)
indice_max <- which.max(residuals_b)
observacion_max <- data[indice_max, ]
datos_sin_max <- data[-indice_max, ]
```

```
### eliminamos outliers -- reisudos > quantil 99%  (valores muy
extremos en los resiudos)

percentil_b_99 <- quantile(residuals_b, 0.99)
indices_b_eliminar <- residuals_b > percentil_b_99
datos_filtrados_b <- train_b[!indices_b_eliminar, ]
modelo_logistico_b2 <- glm(var_obj ~ refinanciaciones +
Max_dias_incumplido_12m +
                                     Max_dias_incumplido_3m +
Credito_dispuesto + Liquidez_cliente +
                                     saldo_medio_disponible_3m +
Media_lineas_Activo + Cuota_amortizar +
                                     Veces_default_12m +
Recibos_devueltos_12m +
                                     #Max_recursos_totales +
                                     Max_saldos_descubiertos_12m +
Score,
                                   datos_filtrados_b, family =
binomial)
summary(modelo_logistico_b2)
summary(modelo_logistico_b2$residuals)
residuals_b2 <- modelo_logistico_b2$residuals
### eliminamos outliers -- reisudos > quantil 1%%  (valores muy
extremos en los resiudos)
percentil_b_01 <- quantile(residuals_b2, 0.01)
indices_b2_eliminar <- residuals_b2 < percentil_b_01
datos_filtrados_b2 <- datos_filtrados_b[!indices_b2_eliminar, ]
modelo_logistico_b3 <- glm(var_obj ~ refinanciaciones +
Max_dias_incumplido_12m +
                             Max_dias_incumplido_3m +
Credito_dispuesto + Liquidez_cliente +
                             saldo_medio_disponible_3m +
Media_lineas_Activo + Cuota_amortizar +
                             Veces_default_12m + Recibos_devueltos_12m
+
                             #Max_recursos_totales +
                             Max_saldos_descubiertos_12m + Score,
                           datos_filtrados_b2, family = binomial)
summary(modelo_logistico_b3)
summary(modelo_logistico_b3$residuals)


#########################################################################
######################## PREDICCIONES MODELO FINAL ######################


modelo_logistico_b <- modelo_logistico_b3

################## VIF

vif(modelo_logistico_b)

############################### PREDICCIONES

#  predicciones en el conjunto de datos de prueba
predicciones_test <- predict(modelo_logistico_b, test, type =
"response")
mean(predicciones_test)
summary(predicciones_test)
hist(predicciones_test)
library(pROC)
```

```
# umbral de decisión
umbral <- 0.5
predicciones_binarias <- ifelse(predicciones_test > umbral, 1, 0)

# matriz de confusión
conf_matrix <- table(test$var_obj, predicciones_binarias)

# precisión
precision <- sum(diag(conf_matrix)) / sum(conf_matrix)

# curva ROC (AUC-ROC)
auc_roc <- auc(roc(test$var_obj, predicciones_test))

# sensibilidad y la especificidad

sensibilidad <- conf_matrix[2, 2] / sum(conf_matrix[2, ])

especificidad <- conf_matrix[1, 1] / sum(conf_matrix[1, ])

precision_positiva <- conf_matrix[2, 2] / sum(conf_matrix[, 2])

F1_score <- 2 * (precision_positiva * sensibilidad) /
(precision_positiva + sensibilidad)


####################################
########### RANDOM FOREST #########


train$var_obj <- factor(train$var_obj)
levels(train$var_obj) <- make.names(levels(train$var_obj))
library(caret)

#as.Date(train$fecha)

fitControl = trainControl(## 10-fold CV
  method = "repeatedcv",
  number = 10,
  allowParallel = TRUE,
  classProbs = TRUE,
  repeats = 3)

# parametros
rfGrid <- expand.grid(
  mtry = c(5,7),
  min.node.size = c(5,15),
  splitrule = c("gini")
)

modellist <- list()
#

for (num.trees in (50, 100, 150, 200)) {
  set.seed(223)
  modelo_rf <- train(
    var_obj ~ ., data = train,
    trControl = fitControl,
    tuneGrid = rfGrid,
    method = "ranger",
    metric = "Accuracy",
    num.trees = num.trees,
```

```
    importance = 'impurity'  )
  key <- toString(num.trees)
  modellist[[key]] <- modelo_rf }

modellist

######################## PREDICCION

library(dplyr)
test_rf <- test
test_rf$var_obj <- factor(test_rf$var_obj)
predicciones_rf <- predict(modelo_rf, test_rf)
table(test_rf$var_obj)
table(predicciones_rf)

# matriz de confusión
predicciones_rf <- ifelse(predicciones_rf == "X0", 0, 1)
predicciones_rf <- factor(predicciones_rf)
table(predicciones_rf)
conf_matrix_rf <- confusionMatrix(predicciones_rf, test_rf$var_obj)
conf_matrix_rf

################### GRAFICO

library(ggplot2)

results_df <- modellist[["100"]][["results"]]
results_df$Accuracy <- as.numeric(as.character(results_df$Accuracy))
results_df$Kappa <- as.numeric(as.character(results_df$Kappa))
ggplot(data = results_df, aes(x = min.node.size, y = Accuracy, color =
as.factor(mtry), group = mtry)) +
  geom_line() +
  geom_point() +
  labs(title = "Accuracy vs. Minimal Node Size for Different mtry
Values",
       x = "Minimal Node Size",
       y = "Accuracy",
       color = "mtry") +
  theme_minimal()
ggplot(data = results_df, aes(x = min.node.size, y = Kappa, color =
as.factor(mtry), group = mtry)) +
  geom_line() +
  geom_point() +
  labs(title = "Kappa vs. Minimal Node Size for Different mtry
Values",
       x = "Minimal Node Size",
       y = "Kappa",
       color = "mtry") +
  theme_minimal()


##########################################
########### RANDOM FOREST BALANCEO #########


train_b$var_obj <- factor(train_b$var_obj)
levels(train_b$var_obj) <- make.names(levels(train_b$var_obj))
library(caret)

#as.Date(train_b$fecha)
fitControl = trainControl(## 10-fold CV
```

```
  method = "repeatedcv",
  number = 10,
  allowParallel = TRUE,
  classProbs = TRUE,
  repeats = 3)

rfGrid <- expand.grid(
  mtry = c(3,4,5,7),
  min.node.size = c(5,15,20,30),
  splitrule = c("gini"))
modellist_b <- list()

for (num.trees in 100) {
  set.seed(223)
  modelo_rf_b <- train(
    var_obj ~ ., data = train_b,
    trControl = fitControl,
    tuneGrid = rfGrid,
    method = "ranger",
    metric = "Accuracy",
    num.trees = num.trees
  )
  key <- toString(num.trees)
  modellist_b[[key]] <- modelo_rf_b
}

modellist_b

######################### PREDICCION

library(dplyr)
test_rf <- test
test_rf$var_obj <- factor(test_rf$var_obj)
predicciones_rf_b <- predict(modelo_rf_b, test_rf)
table(test_rf$var_obj)
table(predicciones_rf_b)

# matriz de confusión
predicciones_rf_b <- ifelse(predicciones_rf_b == "X0", 0, 1)
predicciones_rf_b <- factor(predicciones_rf_b)
table(predicciones_rf_b)
conf_matrix_rf_b <- confusionMatrix(predicciones_rf_b,
test_rf$var_obj)
conf_matrix_rf_b$overall["Accuracy"]
conf_matrix_rf_b$byClass["Sensitivity"]
conf_matrix_rf_b$byClass["Specificity"]
conf_matrix_rf_b$byClass["F1"]


###################### XG BOOST ########################


library(caret)
train$var_obj <- factor(train$var_obj)
levels(train$var_obj) <- make.names(levels(train$var_obj))

fitControl <- trainControl(
  method = "repeatedcv",
  number = 20,
  allowParallel = TRUE,
  classProbs =  TRUE,
```

```
  repeats = 3)

# parámetros
xgbGrid <- expand.grid(nrounds = 50,
                       max_depth = c(10, 20, 30),
                       colsample_bytree = c(0.5, 1),
                       eta = 0.1,
                       gamma= 1,
                       min_child_weight = 1,
                       subsample = 1)

set.seed(223)

xgb_model <-  train(
  var_obj ~ refinanciaciones + Max_dias_incumplido_12m +
Max_dias_incumplido_3m + Credito_dispuesto + Liquidez_cliente +
    saldo_medio_disponible_3m + Media_lineas_Activo + Cuota_amortizar
+ Veces_default_12m + Recibos_devueltos_12m +
    Max_recursos_totales + Max_saldos_descubiertos_3m +
Max_saldos_descubiertos_12m + Score,
  data = train,
  trControl = fitControl,
  tuneGrid = xgbGrid,
  method = "xgbTree",
  metric = "Accuracy"
)

if(!require(pROC)) install.packages("pROC")
if(!require(ggplot2)) install.packages("ggplot2")
library(pROC)
library(ggplot2)

# Predicciones
predicted_classes <- predict(xgb_model, newdata = test)
predicted_probs <- predict(xgb_model, newdata = test, type = "prob")

# Re-codificar las clases predichas

predicted_classes <- ifelse(predicted_classes == "X0", 0, 1)

predicted_classes <- factor(predicted_classes)

test$var_obj <- factor(test$var_obj)

c_matrix <- confusionMatrix(predicted_classes, test$var_obj)

f1_score <- c_matrix$byClass["F1"]


#################### XGBOOST BALANCEO #################


library(caret)
train_b$var_obj <- factor(train_b$var_obj)
levels(train_b$var_obj) <- make.names(levels(train_b$var_obj))

fitControl <- trainControl(
  method = "repeatedcv",
  number = 20,
  allowParallel = TRUE,
  classProbs =  TRUE,
```

```
  repeats = 3)

# parámetros
xgbGrid <- expand.grid(nrounds = 50,
                       max_depth = c(10, 20, 30),
                       colsample_bytree = c(0.5, 1),
                       eta = 0.1,
                       gamma= 1,
                       min_child_weight = 1,
                       subsample = 1)

set.seed(223)

xgb_model_b <-  train(
  var_obj ~ refinanciaciones + Max_dias_incumplido_12m +
Max_dias_incumplido_3m + Credito_dispuesto + Liquidez_cliente +
    saldo_medio_disponible_3m + Media_lineas_Activo + Cuota_amortizar
+ Veces_default_12m + Recibos_devueltos_12m +
    Max_recursos_totales + Max_saldos_descubiertos_3m +
Max_saldos_descubiertos_12m + Score,
  data = train_b,
  trControl = fitControl,
  tuneGrid = xgbGrid,
  method = "xgbTree",
  metric = "Accuracy"
)


predicted_classes_b <- predict(xgb_model_b, newdata = test)
predicted_probs_b <- predict(xgb_model_b, newdata = test, type =
"prob")
predicted_classes_b <- ifelse(predicted_classes_b == "X0", 0, 1)
predicted_classes_b <- factor(predicted_classes_b)
test$var_obj <- factor(test$var_obj)
actual_classes_b <- test$var_obj
table(Predicted = predicted_classes_b, Actual = actual_classes_b)
c_matrix_b<-confusionMatrix(predicted_classes_b, test$var_obj)
c_matrix_b$byClass["F1"]


###################################
####### REDES NEURONALES #########

if (!require(neuralnet)) {
  install.packages("neuralnet")
}
library(neuralnet)
train$var_obj <- as.factor(train$var_obj)
colSums(is.na(train))
indices_submuestra <- sample(1:nrow(train), 100)
datos_muestra <- train[indices_submuestra, ]

# red neuronal
formula <- as.formula(paste("var_obj ~ . - Cod_persona - fecha"))
red_neuronal <- neuralnet(formula,
                          data = train,
                          hidden = 5)
test_rn <- test
predicciones <- compute(red_neuronal, test_rn)
library(pROC)
real_values <- test_rn$var_obj
```

```
predicted_probs <- net.result[[1]]
predicted_class <- ifelse(predicted_probs > 0.5, 1, 0)

# Matriz de confusión
conf_matrix <- table(Predicted = predicted_class, Actual =
real_values)

# Curva ROC y AUC
roc_curve <- roc(real_values, predicted_probs)
auc_value <- auc(roc_curve)
print(conf_matrix)
print(auc_value)
library(keras)

#install_keras()
library(caret)

train <- train[, !(names(train) %in% c("Cod_persons", "fecha"))]
test <- test[, !(names(test) %in% c("Cod_persons", "fecha"))]
y_train <- train[["var_obj"]]
x_train <- train[, !names(train) %in% "var_obj"]
y_test <- test[["var_obj"]]
x_test <- test[, !names(test) %in% "var_obj"]
x_train <- array_reshape(x_train, c(nrow(x_train), ncol(x_train), 1))
x_test <- array_reshape(x_test, c(nrow(x_test), ncol(x_test), 1))


###################
##### LSTM ########


model_lstm <- keras_model_sequential() %>%
  layer_lstm(units = 100, input_shape = c(NULL, dim(x_train)[[3]]),
return_sequences = TRUE) %>%
  layer_dropout(rate = 0.2) %>%
  layer_lstm(units = 50, return_sequences = TRUE) %>%
  layer_dropout(rate = 0.2) %>%
  layer_lstm(units = 50) %>%
  layer_dense(units = 1, activation = 'sigmoid')

adam_optimizer <- optimizer_adam(lr = 0.001)
model_lstm %>% compile(
  loss = 'binary_crossentropy',
  optimizer = adam_optimizer,
  metrics = c('accuracy'))

history_lstm <- model %>% fit(
  x_train, y_train,
  epochs = 50,
  batch_size = 32,
  validation_split = 0.2)
score_lstm <- model %>% evaluate(x_test, y_test, verbose = 0)

# predicciones
predictions_lstm <- model_lstm %>% predict(x_test)
predictions_lstm <- ifelse(predictions_lstm > 0.5, 1, 0)
conf_matrix_lstm <- confusionMatrix(as.factor(predictions),
as.factor(y_test))

print(conf_matrix_lstm)
```

```
############################
######### CNN (GRU) ########

model_CNN <- keras_model_sequential() %>%
  layer_gru(units = 32, input_shape = c(ncol(x_train), 1),
return_sequences = TRUE) %>%
  layer_batch_normalization() %>%
  layer_dropout(rate = 0.2) %>%
  layer_gru(units = 32, return_sequences = TRUE) %>%
  layer_batch_normalization() %>%
  layer_dropout(rate = 0.2) %>%
  layer_gru(units = 32) %>%
  layer_batch_normalization() %>%
  layer_dropout(rate = 0.2) %>%
  layer_dense(units = 1, activation = 'sigmoid')

model_CNN %>% compile(
  loss = 'binary_crossentropy',
  optimizer = optimizer_adam(),
  metrics = c('accuracy'))
history_CNN <- model_CNN %>% fit(
  x_train, y_train,
  epochs = 10,
  batch_size = 32,
  validation_data = list(x_test, y_test))

# predicciones CNN (GRU)

predictions_CNN <- model_CNN %>% predict(x_test)
predictions_CNN <- ifelse(predictions_CNN > 0.5, 1, 0)
conf_matrix_CNN <- confusionMatrix(as.factor(predictions_CNN),
as.factor(y_test))
print(conf_matrix_CNN)


#####################################
######################### paquete h2o

library(h2o)
h2o.init()

data_h2o <- as.h2o(train)
x <- setdiff(names(data_h2o), "var_obj")
y <- "var_obj"
data_h2o[, "var_obj"] <- as.factor(data_h2o[, "var_obj"])
model_h2o <- h2o.deeplearning(x, y, training_frame = data_h2o, hidden
= c(100, 100, 100), epochs = 10)

# predicciones
test_h2o <- as.h2o(test)
predictions_h2o <- h2o.predict(model_h2o, test_h2o)
predicted_classes_h2o <- as.vector(ifelse(predictions_h2o$p1 > 0.5, 1,
0))

actual_classes <- as.vector(test_h2o$var_obj)
conf_matrix_h2o <- confusionMatrix(as.factor(predicted_classes_h2o),
as.factor(actual_classes))
print(conf_matrix_h2o)
```

```
########################################
####### REDES NEURONALES BALANCEO #########

library(pROC)

install.packages("neuralnet")
library(neuralnet)
train_b$var_obj <- as.factor(train_b$var_obj)
colSums(is.na(train_b))
indices_submuestra_b <- sample(1:nrow(train_b), 1000)
datos_muestra_b <- train_b[indices_submuestra_b, ]

# normalización de datos

columnas_a_mantener <- !names(datos_muestra_b) %in% c("var_obj",
"Cod_persona", "fecha")
datos_muestra_b_seleccionados <- datos_muestra_b[,
columnas_a_mantener]

datos_muestra_b_scaled <- scale(datos_muestra_b_seleccionados)
datos_muestra_b_scaled <- as.data.frame(datos_muestra_b_scaled)
datos_muestra_b_scaled <- cbind(datos_muestra_b_scaled, var_obj =
datos_muestra_b$var_obj)

# red neuronal
formula <- as.formula(paste("var_obj ~ ."))

# Entrenar la red
red_neuronal <- neuralnet(formula,
                          data = datos_muestra_b_scaled,
                          hidden = c(3, 5),
                          stepmax = 1e8,
                          algorithm = "rprop+",
                          rep = 3,
                          threshold = 0.01)


########################################################################
#######

test$var_obj <- as.factor(test$var_obj)
datos_test_seleccionados <- test[, columnas_a_mantener]
datos_test_scaled <- scale(datos_test_seleccionados, center = TRUE,
scale = TRUE)
datos_test_scaled <- as.data.frame(datos_test_scaled)
resultados_test <- compute(red_neuronal, datos_test_scaled)

predicciones <- resultados_test$net.result
clases_predichas <- ifelse(predicciones[, 2] > 0.5, 1, 0)
verdaderas_etiquetas <- test$var_obj
conf_matrix <- table(clases_predichas, verdaderas_etiquetas)
library(pROC)
library(caret)
clases_predichas <- as.factor(clases_predichas)
confusionMatrix(clases_predichas, test$var_obj)

f1_score <- 2 * (precision * sensitivity) / (precision + sensitivity)
f1_score
library(keras)

#install_keras()
```

```
train_b <- train_b[, !(names(train_b) %in% c("Cod_persons", "fecha"))]
test <- test[, !(names(test) %in% c("Cod_persons", "fecha"))]
y_train_b <- train_b[["var_obj"]]
x_train_b <- train_b[, !names(train_b) %in% "var_obj"]
y_test <- test[["var_obj"]]
x_test <- test[, !names(test) %in% "var_obj"]
x_train_b <- array_reshape(x_train_b, c(nrow(x_train_b),
ncol(x_train_b), 1))
x_test <- array_reshape(x_test, c(nrow(x_test), ncol(x_test), 1))

# modelo LSTM
model_b <- keras_model_sequential() %>%
  layer_lstm(units = 50, input_shape = c(ncol(x_train_b), 1)) %>%
  layer_dropout(rate = 0.2) %>%
  layer_dense(units = 1, activation = 'sigmoid')

# Configurar el optimizador Adam con ajustes de tasa de aprendizaje
adam_optimizer <- optimizer_adam(lr = 0.001)
model_b %>% compile(
  loss = 'binary_crossentropy',
  optimizer = adam_optimizer,
  metrics = c('accuracy')
)

# Entrenar el modelo
history_b <- model %>% fit(
  x_train_b, y_train_b,
  epochs = 50,
  batch_size = 32,
  validation_split = 0.2
)

# Evaluar el modelo
score_b <- model_b %>% evaluate(x_test, y_test, verbose = 0)

# predicciones
predictions_b <- model_b %>% predict(x_test)
predictions_b <- ifelse(predictions > 0.5, 1, 0)
library(caret)
conf_matrix_b <- confusionMatrix(as.factor(predictions_b),
as.factor(y_test))
print(conf_matrix_b)


######### WOE AND IV ##########

library(dplyr)
data <- datos_filtrados
table(data$var_obj)
total_defaults <- sum(data$var_obj == 1)
total_no_defaults <- sum(data$var_obj == 0)
default_rate_total <- total_defaults / nrow(data)
defaults <- data[data$var_obj == 1, ]
no_defaults <- data[data$var_obj == 0, ]

library(dplyr)
library(tidyr)

nombres <- c("refinanciaciones", "Max_dias_incumplido_12m",
"Max_dias_incumplido_3m", "Credito_dispuesto", "Liquidez_cliente",
```

```
            "saldo_medio_disponible_3m", "Media_lineas_Activo",
"Cuota_amortizar", "Veces_default_12m", "Recibos_devueltos_12m",
"Max_recursos_totales",
            "Max_saldos_descubiertos_3m",
"Max_saldos_descubiertos_12m", "Score")

# WoE y preparación para IV para cada variable predictora

woe_list <- lapply(nombres, function(var_name) {
  woe_data <- data %>%
    group_by(!!sym(var_name)) %>%
    summarise(Buenos = sum(var_obj == 0),
              Malos = sum(var_obj == 1),
              Total = n()) %>%
    mutate(DistBuenos = Buenos / sum(Buenos),
           DistMalos = Malos / sum(Malos)) %>%
    filter(DistBuenos > 0 & DistMalos > 0) %>%
    mutate(WoE = log(DistBuenos / DistMalos)) %>%
    select(!!sym(var_name), WoE, DistBuenos, DistMalos) %>%
    tidyr::complete(!!sym(var_name), fill = list(WoE = NA, DistBuenos
= 0, DistMalos = 0))  # Completa con valores por defecto
  woe_data$Variable <- var_name  # Añade el nombre de la variable
  return(woe_data)
})

calculate_iv <- function(data) {
  iv <- sum((data$DistBuenos - data$DistMalos) * data$WoE, na.rm =
TRUE)
  return(iv)}
iv_list <- lapply(woe_list, calculate_iv)
variables <- nombres
iv_values <- unlist(iv_list)
iv_dataframe <- data.frame(Variable = variables, IV = iv_values)
print(iv_dataframe)


########## GRÁFICAS VALIDACIÓN MÉTRICAS #######

resultados <- data.frame(
  Modelo = c(
    "Logit", "Logit", "Logit", "Logit",
    "Random Forest", "Random Forest", "Random Forest", "Random
Forest",
    "XGBoost", "XGBoost", "XGBoost", "XGBoost",
    "Neural Net (Rprop+)", "Neural Net (Rprop+)", "Neural Net
(Rprop+)", "Neural Net (Rprop+)",
    "Neural Net (CNN)", "Neural Net (CNN)", "Neural Net (CNN)",
"Neural Net (CNN)" ),
  Metrica = rep(c("Precisión", "Sensibilidad", "Especificidad", "F1"),
times = 5),
  Valor = c(
    42.47, 30.55, 98.46, 35.54,
    99.04, 87.24, 99.98, 92.76,
    98.37, 80.44, 99.96, 88.51,
    55.30, 82.77, 98.22, 66.31,
    88.90, 85.40, 99.72, 87.11 ))
ggplot(resultados, aes(x = Metrica, y = Valor, fill = Modelo)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(
    title = "Comparación de Métricas de Rendimiento entre Modelos",
    x = "Métricas",
```

```
    y = "Valor (%)"  ) +
  theme_minimal() +
  scale_fill_brewer(palette = "Violet") + # Usar una paleta de colores
atractiva
  theme(
    plot.title = element_text(hjust = 0.5, size = 14, face = "bold"),
    axis.title = element_text(size = 12, face = "bold"),
    legend.title = element_text(size = 10, face = "bold"),
    legend.text = element_text(size = 9) )



############# VALIDACIÓN OOS #############


# librerías necesarias
library(caret)
library(xgboost)
library(pROC)
library(dplyr)

# muestra OOT
library(readr)
oot <- read_csv("TFM/OOT.csv", trim_ws = TRUE)
View(oot)
names(oot) = c("Cod_persona", "fecha", "var_obj", "refinanciaciones",
"Max_dias_incumplido_12m", "Max_dias_incumplido_3m",
"Credito_dispuesto", "Score", "Liquidez_cliente",
"saldo_medio_disponible_3m", "Media_lineas_Activo", "Cuota_amortizar",
"Veces_default_12m", "Recibos_devueltos_12m",
"Max_recursos_totales","Max_saldos_descubiertos_3m",
"Max_saldos_descubiertos_12m")

## maximo dias incumplido a 12 m se sustituye por 0

oot$Max_dias_incumplido_12m[is.na(oot$Max_dias_incumplido_12m)] <- 0
oot$Recibos_devueltos_12m[is.na(oot$Recibos_devueltos_12m)] <- 0
oot$Score[is.na(oot$Score)] <- mean(oot$Score, na.rm = TRUE)
oot$Liquidez_cliente[is.na(oot$Liquidez_cliente)] <-
mean(oot$Liquidez_cliente, na.rm = TRUE)
oot$saldo_medio_disponible_3m[is.na(oot$saldo_medio_disponible_3m)] <-
mean(oot$saldo_medio_disponible_3m, na.rm = TRUE)
oot$Media_lineas_Activo[is.na(oot$Media_lineas_Activo)] <-
mean(oot$Media_lineas_Activo, na.rm = TRUE)
oot$Max_saldos_descubiertos_12m[is.na(oot$Max_saldos_descubiertos_12m)
] <- mean(oot$Max_saldos_descubiertos_12m, na.rm = TRUE)
oot$Max_saldos_descubiertos_3m[is.na(oot$Max_saldos_descubiertos_3m)]
<- mean(oot$Max_saldos_descubiertos_3m, na.rm = TRUE)
oot$Max_recursos_totales[is.na(oot$Max_recursos_totales)] <-
mean(oot$Max_recursos_totales, na.rm = TRUE)
oot$Cuota_amortizar[is.na(oot$Cuota_amortizar)] <-
mean(oot$Cuota_amortizar, na.rm = TRUE)
oot <- subset(oot, Score >= 100)
oot <-na.omit(oot)

# Predicciones
predicted_classes_oot <- predict(xgb_model_b, newdata = oot)
predicted_probs_oot <- predict(xgb_model_b, newdata = oot, type =
"prob")

# Re-codificar las clases predichas
```

```
predicted_classes_oot <- ifelse(predicted_classes_oot == "X0", 0, 1)
predicted_classes_oot <- factor(predicted_classes_oot)
oot$var_obj <- factor(oot$var_obj)
c_matrix_oot <- confusionMatrix(predicted_classes_oot, oot$var_obj)
c_matrix_oot
f1_score <- c_matrix_oot$byClass["F1"]


########### KS Y GINI


predicted_probs <- predict(xgb_model, newdata = oot, type = "prob")
actual <- oot$var_obj

df <- data.frame(
  actual = actual,
  predicted_prob = predicted_probs_oot$X1  # Usar la probabilidad de
default (X1)
)

# Ordenar por probabilidad de default (X1)
df <- df %>%
  arrange(predicted_prob) %>%
  mutate(FDC_no_default = cumsum(actual == 0) / sum(actual == 0),
         FDC_default = cumsum(actual == 1) / sum(actual == 1))

# valor KS
KS_value <- max(abs(df$FDC_no_default - df$FDC_default))
KS_value


###### GINI

gini_xgboost <- 2 * AUC_default -1
gini_xgboost2 <- 2 * AUC_no_default - 1
gini_xgboost

####### AUC - ROC

tabla_matriz <- c_matrix$table

# Calcular TP, TN, FP, FN para No Default (X0)
TN <- tabla_matriz[1, 1]  # X0 predicho como X0
FP <- tabla_matriz[2, 1]  # X0 predicho como X1
FN <- tabla_matriz[1, 2]  # X1 predicho como X0
TP <- tabla_matriz[2, 2]  # X1 predicho como X1

TPR_no_default <- TN / (TN + FP)
FPR_no_default <- FP / (FP + TN)

TPR_default <- TP / (TP + FN)
FPR_default <- FN / (FN + TP)

points_no_default <- data.frame(
  FPR = c(0, FPR_no_default, 1),
  TPR = c(0, TPR_no_default, 1))

points_default <- data.frame(
  FPR = c(0, FPR_default, 1),
  TPR = c(0, TPR_default, 1))
```

```
AUC_no_default <- sum(diff(points_no_default$FPR) *
                        (points_no_default$TPR[-1] +
points_no_default$TPR[-nrow(points_no_default)]) / 2)

AUC_default <- sum(diff(points_default$FPR) *
                     (points_default$TPR[-1] + points_default$TPR[-
nrow(points_default)]) / 2)

roc_data <- rbind(
  cbind(points_no_default, Class = "No Default"),
  cbind(points_default, Class = "Default"))

ggplot(roc_data, aes(x = FPR, y = TPR, color = Class)) +
  geom_smooth(method = "loess", se = FALSE, size = 1.5) + # Smoothed
lines only
  geom_abline(slope = 1, intercept = 0, linetype = "dashed", color =
"grey") +
  labs(title = "Curva ROC XGBoost",
       x = "Tasa de Falsos Positivos (FPR)",
       y = "Tasa de Verdaderos Positivos (TPR)") +
  scale_color_manual(values = c("No Default" = "blue", "Default" =
"cyan")) +
  theme_minimal()

# Predicciones
predicted_classes_oot <- predict(modelo_rf, newdata = oot)
predicted_probs_oot <- predict(modelo_rf, newdata = oot, type =
"prob")

predicted_classes_oot <- ifelse(predicted_classes_oot == "X0", 0, 1)
predicted_classes_oot <- factor(predicted_classes_oot)
oot$var_obj <- factor(oot$var_obj)
c_matrix_oot <- confusionMatrix(predicted_classes_oot, oot$var_obj)
f1_score <- c_matrix_oot$byClass["F1"]
predicted_probs <- predict(modelo_rf, newdata = test, type = "prob")
actual <- oot$var_obj

df <- data.frame(
  actual = actual,
  predicted_prob = predicted_probs_oot$X1  # Usar la probabilidad de
default (X1)
)

df <- df %>%
  arrange(predicted_prob) %>%
  mutate(FDC_no_default = cumsum(actual == 0) / sum(actual == 0),
         FDC_default = cumsum(actual == 1) / sum(actual == 1))

# valor KS
KS_value <- max(abs(df$FDC_no_default - df$FDC_default))
KS_value


###### GINI

gini_rf <- 2 * AUC_default -1
gini_rf <- 2 * AUC_no_default - 1
gini_rf
tabla_matriz <- c_matrix_oot$table

# TP, TN, FP, FN para No Default (X0)
```

```
TN <- tabla_matriz[1, 1]  # X0 predicho como X0
FP <- tabla_matriz[2, 1]  # X0 predicho como X1
FN <- tabla_matriz[1, 2]  # X1 predicho como X0
TP <- tabla_matriz[2, 2]  # X1 predicho como X1

# TPR y FPR para No Default (X0)
TPR_no_default <- TN / (TN + FP)
FPR_no_default <- FP / (FP + TN)

# TPR y FPR para Default (X1)
TPR_default <- TP / (TP + FN)
FPR_default <- FN / (FN + TP)

points_no_default <- data.frame(
  FPR = c(0, FPR_no_default, 1),
  TPR = c(0, TPR_no_default, 1))

points_default <- data.frame(
  FPR = c(0, FPR_default, 1),
  TPR = c(0, TPR_default, 1))

AUC_no_default <- sum(diff(points_no_default$FPR) *
                        (points_no_default$TPR[-1] +
points_no_default$TPR[-nrow(points_no_default)]) / 2)

AUC_default <- sum(diff(points_default$FPR) *
                     (points_default$TPR[-1] + points_default$TPR[-
nrow(points_default)]) / 2)

roc_data <- rbind(
  cbind(points_no_default, Class = "No Default"),
  cbind(points_default, Class = "Default"))

ggplot(roc_data, aes(x = FPR, y = TPR, color = Class)) +
  geom_smooth(method = "loess", se = FALSE, size = 1.5) + # Smoothed
lines only
  geom_abline(slope = 1, intercept = 0, linetype = "dashed", color =
"grey") +
  labs(title = "Curva ROC Random Forest",
       x = "Tasa de Falsos Positivos (FPR)",
       y = "Tasa de Verdaderos Positivos (TPR)") +
  scale_color_manual(values = c("No Default" = "darkgreen", "Default"
= "lightgreen")) +
  theme_minimal()

################### PERFIL DE RIESGO CARTERA DE CLIENTES
##################


# importancia de las variables
variable_importance <- importance(modelo_rf)


variable_importance_df <- data.frame(
  Variable = rownames(variable_importance),
  Importance = variable_importance[, "MeanDecreaseGini"])

# Ordenar por importancia descendente
variable_importance_df <- variable_importance_df[order(-
variable_importance_df$Importance), ]
```

```
print(variable_importance_df)

importancia_vars <- data.frame(
  Variable = c(
    "refinanciaciones", "Media_lineas_Activo",
"Max_dias_incumplido_12m",
    "Cuota_amortizar", "Score", "Liquidez_cliente",
"Max_recursos_totales",
    "Recibos_devueltos_12m", "saldo_medio_disponible_3m",
"Credito_dispuesto",
    "Max_dias_incumplido_3m", "Max_saldos_descubiertos_3m",
    "Max_saldos_descubiertos_12m", "Veces_default_12m"  ),
  Porcentaje = c(
    99.83, 98.85, 73.18, 72.06, 71.55, 71.07, 69.65,
    58.05, 54.94, 50.91, 45.56, 32.02, 25.83, 12.38 ))

ggplot(importancia_vars, aes(x = reorder(Variable, -Porcentaje), y =
Porcentaje)) +
  geom_point(color = "violet", size = 4) +
  geom_segment(aes(x = Variable, xend = Variable, y = 0, yend =
Porcentaje),
               color = "violet", linetype = "dashed") +
  coord_flip() +  # Voltear el gráfico para una mejor visualización
  labs(
    title = "Importancia de Variables en el Modelo de Random Forest",
    x = "Variables",
    y = "Importancia (%)"
  ) +
  theme_minimal() +
  theme(
    axis.text.y = element_text(size = 10, face = "bold"),
    plot.title = element_text(hjust = 0.5, size = 14, face = "bold"),
    axis.title = element_text(size = 12, face = "bold"))


####### SHAPELY #######


library(data.table)
#install.packages("ranger")
#install.packages("iml")
library(caret)
library(ranger)
library(iml)
library(dplyr)

train$var_obj <- factor(train$var_obj)
levels(train$var_obj) <- make.names(levels(train$var_obj))
train <- train%>% select (-Cod_persona, - fecha)

# trainControl
fitControl = trainControl(
  method = "repeatedcv",
  number = 10,
  allowParallel = TRUE,
  classProbs = TRUE,
  repeats = 3
)

# parámetros
```

```
rfGrid <- expand.grid(
  mtry = 7,
  min.node.size = 5,
  splitrule = "gini"
)

set.seed(223)
modelo_rf <- train(
  var_obj ~ ., data = train,
  trControl = fitControl,
  tuneGrid = rfGrid,
  method = "ranger",
  metric = "Accuracy",
  num.trees = 100,
  importance = 'impurity'
)

# impureza de Gini

importancia_vars <- varImp(modelo_rf)
print(importancia_vars)

train_data <- train %>% select(-var_obj)

predictor <- Predictor$new(modelo_rf, data = train_data, y =
train$var_obj)

set.seed(123)
sample_indices <- sample(1:nrow(train_data), 3000)
sample_data <- train_data[sample_indices, ]

# Shapley

shapley_values <- NULL
for (i in 1:nrow(sample_data)) {
  shapley_i <- Shapley$new(predictor, x.interest = sample_data[i, ])
  shapley_values <- rbind(shapley_values, shapley_i$results)
}

shap_dt <- as.data.table(shapley_values)
shap_dt[, phi := as.numeric(phi)]
shap_dt[, feature_value := as.numeric(sub(".*=", "", feature.value))]

# gráfico beeswarm
ggplot(shap_dt, aes(x = reorder(feature, -abs(phi), median), y = phi,
color = feature_value)) +
  geom_point(alpha = 0.5, size = 1.5, position = position_jitter(width
= 0.2, height = 0)) +
  scale_color_gradient(low = "blue", high = "red") +
  theme_minimal() +
  labs(x = "Feature", y = "SHAP value (impact on model output)", title
= "SHAP Summary Plot") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```