Escuela Técnica Superior de Ingeniería (Comillas ICAI)

ESTUDIO DEL RIESGO DE CRÉDITO Y SU MARCO NORMATIVO PARA LA PREDICCIÓN DE LA PROBABILIDAD DE DEFAULT EMPLEANDO TÉCNICAS DE MACHINE LEARNING

Autor: Pedro González Schleissner
Director: María Coronado Vaca

MADRID | Junio 2023

Resumen

Este proyecto realiza un estudio del riesgo de crédito y de los factores que lo constituyen, tratando de sentar las bases teóricas del trabajo. A continuación, se desarrolla el panorama regulativo histórico y actual. Centrándonos en la normativa europea, se exponen los marcos normativos planteados por el comité de Basilea, concretamente Basilea I y Basilea II. Dentro de este último, se profundiza en los métodos basados en calificaciones internas (IRB). Se concluye el apartado teórico con la decisión de seleccionar la metodología IRB básica para la predicción de la probabilidad de default mediante el uso de modelos estadísticos y analíticos propios.

Los modelos seleccionados son un modelo de regresión logística, un modelo de red neuronal, un modelo de árbol de decisión y un modelo basado en XGBoost. Estos modelos son entrenados empleando los datos del banco especializado en préstamos entre particulares Lending Club. Primero, se lleva a cabo un tratamiento de los datos, donde reducimos el número de variables a emplear de 151 a 76. Posteriormente, se desarrollan y entrenan los cuatro modelos, cuyo desempeño demuestra una excelente capacidad predictiva.

Palabras clave

Riesgo de crédito – Basilea – IRB – Probabilidad de default – Machine learning – Regresión logística – Redes neuronales – Árboles de decisión – XGBoost

Abstract

This project conducts a study of credit risk and the factors that constitute it, trying to lay the theoretical foundations of the work. Next, the historical and current regulatory panorama is developed. Focusing on European regulations, the regulatory frameworks proposed by the Basel Committee, specifically Basel I and Basel II, are presented. Within the latter, the internal ratings-based approach (IRB) is discussed in depth. The theoretical section concludes with the decision to select the basic IRB methodology for the prediction of the probability of default through the use of statistical and analytical models.

The selected models are a logistic regression model, a neural network model, a decision tree model and a model based on XGBoost. These models are trained using data from the private lending bank Lending Club. First, a data treatment is carried out, where we reduce the number of variables to be used from 151 to 76. Subsequently, the four models are developed and trained, and their performance shows an excellent predictive capacity.

Key words

Credit risk – Basel – IRB – Probability of default – Machine learning – Logistic regression – Neural networks – Decision trees – XGBoost

Índice de contenidos

Índice de ilustraciones

Índice de ecuaciones

Índice de tablas

1 Introducción

Cuando los bancos, las entidades financieras o las entidades de crédito operan, están asumiendo una serie de riesgos. Estos pueden ser de numerosos tipos, pero los que más afectan a estas entidades son los riesgos financieros, los cuales se pueden clasificar a su vez en riesgo de mercado, de crédito, operativo o de liquidez. Para el desarrollo de este proyecto nos centraremos en el riesgo de crédito, que consiste en la incapacidad de una de las partes para pagar la deuda de un acuerdo con obligaciones contractuales (Banco Santander, 2022). Aunque el impago de cualquiera de las dos partes es posible y constituye un riesgo de crédito, la frecuencia con la que los bancos son impagados es notablemente superior. De hecho, dichos incumplimientos de las obligaciones financieras por parte de los deudores hacia los bancos forman parte de la actividad diaria de las entidades financieras, poniendo en riesgo la estabilidad financiera de las entidades. Es por ello por lo que se implantan medidas con el objetivo de estimar la cuantía de las posibles pérdidas y conseguir de esa manera estar preparados para afrontar los posibles descubiertos.

Una de esas medidas es el uso de modelos de riesgo de crédito, que tratan de estimar la probabilidad de impago de un deudor con el objetivo de poder cumplir las obligaciones de capital y de provisiones establecidas por las agencias regulatorias.

1.1 Objetivos

Los modelos de riesgo de crédito son una realidad en nuestro sistema financiero. Todas las entidades bancarias deben, de una forma u otra aplicar mecanismos de control del riesgo que están asumiendo al actuar como prestatario en un préstamo. La participación de todos los bancos se debe tanto al propio interés de las entidades de tener una actividad económica sana como a la imposición extensa normativa por parte de entidades regulatorias como la Autoridad Bancaria Europea.

El objetivo principal de este proyecto es la generación de diferentes modelos de riesgo de crédito con los cuales se tratará de predecir, de la forma más precisa posible, si un deudor cometerá un impago o no. Se realizarán cuatro modelos diferentes: una regresión

logística, una red neuronal, un árbol de decisión y un modelo basado en XGBoost. El objetivo es la creación de modelos con un alto poder predictivo, los cuales serán comparados para elegir, posteriormente, uno de ellos.

Para poder alcanzar dicho objetivo, deberemos primero entender una serie de conceptos e ideas que nos permitan alcanzar un nivel de entendimiento suficiente para poder desarrollar buenos modelos. Se pretende entender a rasgos generales cuál es el panorama actual en el mundo del riesgo de crédito y cuál es la normativa vigente. También se quiere comprender qué es un modelo de riesgo de crédito, analizando qué parámetros trata de predecir, qué variables usan y cómo se construyen. Finalmente, llevaremos a cabo un análisis teórico de los modelos que se desarrollaran más adelante en dicho proyecto, estudiando sus diferentes arquitecturas y las diferencias y similitudes entre ellos.

1.2 Metodología

La consecución de los objetivos se llevará a cabo en diferentes etapas. Primero se tratará de plantear un marco teórico. En él se realizará primero una introducción al riesgo de crédito, explicando qué es, qué subtipos de este riesgo existen y qué características lo constituyen. También trataremos de explicar de dónde viene la importancia que se le da a este riesgo y cuál es el marco regulatorio que rige el comportamiento de los bancos. Continuando con la línea de estudio planteada, en el siguiente punto del marco teórico se centrará el análisis en los modelos de riesgo de crédito que se emplean a día de hoy por las entidades bancarias, atendiendo a los parámetros que se pretenden predecir, como se predicen y qué información nos aportan. Finalmente, se llevará a cabo un análisis de las diferentes técnicas que se van a emplear para el desarrollo de los modelos. Los parámetros que se tienen que definir vienen determinados por la normativa, pero ésta no define qué técnica es la que debe emplearse para obtener la predicción. En este proyecto se emplearán y compararán cuatro técnicas diferentes, las cuales son regresión logística, redes neuronales, árboles de decisión y clasificador mediante XGBoost.

Una vez definido el marco teórico se procederá al estudio empírico, que consistirá principalmente en el desarrollo de los modelos empleando las tres técnicas mencionadas previamente. Para ello se seguirán los siguientes pasos. Primero se llevará a cabo un análisis y estudio exploratorio de los datos a emplear en los modelos. Con ello se pretende

entender las variables con las que se va a afrontar el problema, así como llevar a cabo las correcciones o manipulaciones de los datos que sean necesarias para el adecuado desarrollo de las posteriores etapas. Una vez se haya comprendido el conjunto de datos que se van a utilizar y éstos estén en una condición óptima, se realizará un procedimiento similar para cada una de las técnicas. Se desarrollará un primer modelo incluyendo todas las variables y sin realizar ningún tipo de ajuste sobre la arquitectura, con el objetivo de tener un modelo base con el que poder comparar los resultados posteriormente. Tras obtener los resultados del modelo base, se procederá a refinar los modelos basándonos en el estudio de la significancia de las variables y atendiendo a diferentes indicadores que nos muestren si son necesarias modificaciones en las arquitecturas o en los hiperparámetros de los modelos. Finalmente, se procederá a realizar una comparación de los resultados de los diferentes modelos, esperando que se consiga determinar cuál de las técnicas es la óptima para afrontar este tipo de problemas.

1.3 Motivación

El riesgo de crédito es uno de los principales problemas que se encuentra un banco en su actividad cotidiana. Prácticamente todas las actividades económicas que lleva a cabo un banco conllevan en mayor o menor medida este riesgo.

Además, la estabilidad económica de la que gozamos a día de hoy depende íntegramente del correcto funcionamiento de las entidades bancarias, dependiendo en gran medida de la gestión del riesgo que se realice. Un claro ejemplo de ello es la crisis del 2008, donde un exceso en el riesgo asumido a la hora de conceder hipotecas resultó en una crisis a nivel global.

Las nuevas tecnologías permiten que el riesgo se controle con una precisión que nunca antes había sido posible, cuando los bancos se veían obligados a dejar márgenes superiores a los necesarios debido a la falta de capacidad de estimación.

La importancia de este campo junto con las posibilidades que han abierto las nuevas tecnologías hace que, desde nuestro punto de vista, este sea un excelente tema de estudio.

1.4 Estructura

El trabajo está dividido en cinco capítulos. El primero, de función introductoria, tiene como objetivo sentar las bases del trabajo. A continuación, en el segundo capítulo, se profundizará en todo aquel conocimiento teórico que sea necesario para comprender el desarrollo posterior. Este capítulo tratará el concepto de riesgo de crédito, el marco normativo actual y la teoría que hay detrás de los modelos que se van a emplear.

Una vez asimilada la teoría, se comenzará con el desarrollo de los modelos, para lo cual será necesario llevar a cabo primero el tratamiento y el análisis de los datos. Tras completar la depuración de los mismos se entrenarán los cuatro modelos seleccionados y sus resultados serán comparados en el cuarto capítulo.

Finalmente, en el quinto capítulo se expondrán las conclusiones a las que se ha llegado, en las cuales se analizarán si se han alcanzado los objetivos planteados y las posibles vías de mejora.

2 Marco teórico

2.1 Riesgo de crédito

El riesgo de crédito, como ya se ha mencionado previamente, es la posibilidad de que el prestatario no cumpla sus obligaciones de pago (Banco Santander, 2023). Es decir, es la probabilidad de que la persona o entidad que recibe el dinero prestado no sea capaz de devolver el capital o los intereses estipulados en el contrato. Cuando no se realiza un pago o serie de pagos diremos que se ha producido una situación de impago o de default, términos que usaremos indistintamente a partir de este momento. Es de vital importancia para los bancos tener un profundo entendimiento y control del riesgo de crédito al que están expuestos y al que desean exponerse. Un claro ejemplo de lo que puede suceder si no se asume riesgo de una forma controlada es el que fue el causante de la crisis financiera de 2008. A grandes rasgos, a raíz de una gran burbuja inmobiliaria, se comenzó a otorgar hipotecas, es decir, préstamos, a personas cuya capacidad de pago no era lo suficientemente buena. Esto supuso un aumento en el riesgo de crédito que se estaba asumiendo y, como era probable, los deudores con mal historial crediticio comenzaron a hacer incumplir en sus obligaciones de pago. Esto causó finalmente el hundimiento de uno de los mayores bancos de inversión de aquel momento, Lehman Brothers, y el comienzo de la crisis del 2008 (Ramskogler, 2015).

Para ser capaces de establecer de forma efectiva el riesgo de crédito que se está asumiendo se deben estudiar otros riesgos que componen o que pueden aumentar el riesgo de crédito que está asumiendo una empresa o una entidad financiera. Según (Brown & Moles, 2014) dos componentes importantes del riesgo de crédito son el riesgo de concentración y el riesgo de liquidación. El primero hace referencia al riesgo que conlleva tener diferentes exposiciones con características similares, lo cual puede desencadenar en situaciones catastróficas si un sector al que nos encontramos expuestos en numerosas operaciones cae. Este fue el caso de diferentes crisis inmobiliarias que resultaron en el cierre de numerosos bancos en los países escandinavos en la década de los 90 o el de la crisis del 2008 (Grippa & Gornicka, 2016). Por otro lado, el riesgo de liquidación surge cuando una tercera parte se encarga de procesar una transacción para otras partes. Esto aumenta

el riesgo ya que las partes ya no dependen únicamente de su capacidad de cumplir con sus obligaciones, sino que también debe cumplir el intermediario.

Además, en (Brown & Moles, 2014) se expone que se debe tener en cuenta que el crecimiento de una empresa conlleva una serie de nuevos factores que aumentan el riesgo de crédito, especialmente en empresas que operan en varios países. Los nuevos riesgos a los que se enfrentan las empresas y que aumentan el riesgo de crédito son:

- **Riesgo-país**: nació en la década de los 50, cuando los grandes bancos comenzaron a operar a grandes escalas a nivel internacional. Éste consiste en el riesgo asociado a las posibles pérdidas debidas a factores de diversa naturaleza, como macroeconómicos, políticos o legales, en ambientes en los que dichos factores difieren son diferentes a aquellos del país de origen de la entidad bancaria, por ejemplo, la diferencia en las leyes o gobiernos (Iranzo, 2008). Podemos identificar cuatro diferentes componentes dentro del riesgo país:
    - **Riesgo político:** Término que se empezó a usar en la década de los 60, a raíz de eventos como la Guerra Fría y el comienzo de la descolonización, hace referencia al riesgo no económico que puede provenir de decisiones políticas, especialmente de los nacionalismos económicos. Este tipo de riesgo es prevalente principalmente en países en vías de desarrollo, ya que estos suelen caracterizarse por una mayor inestabilidad política (Sottilotta, 2012).
    - **Riesgo económico:** Riesgo que abarca un amplio espectro de conceptos, pudiendo agrupar la mayoría en riesgos comerciales y riesgos institucionales. El primero engloba aquellos riesgos que nacen de los posibles cambios en precios de las materias primas o de los productores o cambios en la demanda de los productos o servicios. Por el contrario, los riesgos institucionales surgen de factores ajenos al mercado, como políticas monetarias, fiscales o sociales (Platon et al., 2014).
    - **Riesgo de divisa:** Cuando una empresa opera en diferentes países es muy frecuente, salvo que opere en una unión monetaria, que deba realizar cambios de divisa, tanto por la necesidad de realizar pago en el extranjero como por cambiar la moneda en la que recibe pagos. Esto supone un riesgo, puesto que los tipos de cambio fluctúan, pudiendo provocar situaciones de pérdidas económicas (Brown & Moles, 2014).

    - **Riesgo de cumplimiento:** Es el riesgo que nace de posibles incumplimientos de leyes o regulaciones que pueden resultar en pérdidas económicas o incluso en sanciones civiles o penales (Committee of Sponsoring Organizations of the Treadway Commission (COSO), 2020). Aunque este riesgo puede ocurrir en cualquier geografía en la que se opere, incluso en la que la empresa o entidad financiera esté acostumbrada a operar, el riesgo aumenta cuando las operaciones tienen lugar en marcos legales con los que se está menos familiarizado.
- **Riesgo de industria:** Relacionado con el riesgo de concentración, engloba aquellos riesgos que emanan de una situación negativa en la economía de un país, la cual afecta principalmente a un grupo de industrias (Brown & Moles, 2014). Un claro ejemplo de ello es el efecto que ha tenido la guerra entre Rusia y Ucrania, la cual ha producido una escalada en los precios de ciertas materias primas, viéndose este incremento reflejado en los precios de gran cantidad de productos y por tanto en la notable crisis inflacionaria que se está viviendo en Europa.

Entendiendo qué es el riesgo de crédito y cuáles son sus principales componentes, es el momento de estudiar cómo se cuantifica este riesgo y, por lo tanto, como se trata. Como hemos mencionado previamente, el riesgo de crédito es uno de los factores más determinantes en las operaciones de las entidades bancarias, por lo que existen numerosas técnicas o procedimientos para evaluar la calidad crediticia de un aplicante. La forma de reducir el riesgo que está dispuesto a asumir la entidad es tan sencilla como aceptar o denegar el crédito a aquellos aplicantes que conlleven una probabilidad de impago superior a la que se quiere asumir. Según (Brown & Moles, 2014), podemos agrupar los procedimientos mediante los cuales se determina si a un participante se le otorga un préstamo en seis principales grupos, dentro de los cuales se encuentras las técnicas empleadas. Los tipos de procedimientos de evaluación de crédito son los siguientes:

- Métodos basados en el juicio de uno o varios asesores con experiencia en la materia, los cuales llegan a una decisión a raíz de su criterio personal.
- Métodos basados en comités de expertos. Estos se diferencian de la categoría previa en que sus decisiones se formalizan siguiendo procedimientos y sistemas de préstamos.
- Métodos analíticos. Emplean información cuantitativa para obtener una decisión.

- Métodos estadísticos. Llevan a cabo inferencia estadística para establecer relaciones que les permitan llegar a una decisión.
- Modelos de comportamiento, los cuales estudian los comportamientos previos de los aplicantes al préstamo, con el objetivo de establecer relaciones que les permitan llegar a una decisión.
- Modelos de mercado. Estos se basan en diferentes indicadores de los mercados financieros para estimar la solvencia de los aplicantes.

2.2 Modelos de riesgo de crédito. Marco normativo actual

Previamente se ha expuesto qué es el riesgo de crédito y que el objetivo de los bancos radica en controlar dicho riesgo. Sin embargo, a mayor riesgo mayor rentabilidad pueden llegar a obtener las entidades financieras, al cobrar intereses mayores en operaciones que impliquen mayor riesgo. Por este motivo en 1974 se crea el Comité de Basilea, a raíz de una serie de eventos que pusieron en jaque a los mercados, como la caída de los mercados o la crisis del petróleo, ambos en 1973. El objetivo del comité era crear un organismo de supervisión bancaria internacional mediante mecanismos que permitían el adecuado intercambio de información y mediante la creación de directrices y estándares (Moody's Analytics, 2011). Es así como en 1988 se emite el primer conjunto de estándares, conocido como Basilea I, en el cual se establecen unos requerimientos mínimos de capital que los bancos deben poseer para poder afrontar pérdidas relacionadas con los activos que poseían (Chatterjee, 2015).

Basilea I se publica en julio del 1988 y establece que los requerimientos mínimos de capital se calcularán siguiendo la *Ecuación 1.*

$$Requerimiento\ mínimo\ de\ capital = 8\% * \sum activos\ ponderados\ al\ riesgo$$

Ecuación 1: Requerimiento mínimo de capital

Básicamente, se clasificaban los diferentes activos de la entidad bancaria en diferentes categorías, según el riesgo que solían tener ese tipo de activos. Las ponderaciones posibles eran 0%, para los activos con menor riesgo, 20%, 50% y finalmente 100% (European Central Bank [ECB], 2004). Un posible ejemplo de activos con bajo riesgo

podría ser deuda soberana, mientras que deuda privada o las hipotecas reciben una ponderación mayor como resultado del mayor riesgo que conllevan. Se calcula entonces el valor de cada uno de los activos multiplicado por su ponderación y el 8% de dicho valor es el capital mínimo que Basilea I estipula que deben tener las entidades bancarias (Chatterjee, 2015).

Una de las contrapartidas de obligar a los bancos a reservar capital para cubrir posibles pérdidas es que estos disponen de menor liquidez para realizar inversiones. Por ello, se estudió el desarrollo de un estándar que fuera más preciso que Basilea I, el cual se realizaba mediante un enfoque poco granular, es decir, se realizaban muchas generalizaciones y aproximaciones. Es por ello por lo que en junio del 2004 se publica Basilea II.

Las directrices de Basilea II se distribuyen en tres pilares diferentes, según (ECB, 2004):

- **Pilar I:** El pilar I hace referencia a los requerimientos mínimos de capital que los bancos deben guardar para hacer frente al riesgo de crédito, que hace referencia al riesgo de que se incumplan las obligaciones de pago (Banco Santander, 2023); al riesgo de mercado, el cual trata el riesgo de que los activos pierdan valor como consecuencia de factores de mercado como cambios en los tipos de interés o en los tipos de cambio (African Development Bank, 2007); y al riego operativo, aquellos riesgos que derivan de errores en procesos internos, fallo humano o factores externos como procedimientos legales (Federal Deposit Insurance Corporation, 2006).
- **Pilar II:** El pilar II se encarga de la supervisión y revisión de que los requisitos de capital mínimo definidos en el Pilar I se estén llevando a cabo de manera adecuada y de acuerdo con las directrices definidas.
- **Pilar III:** El tercer y último pilar está relacionado con la transparencia. Se requiere que los bancos faciliten a los inversores la información necesaria para que estos puedan determinar el riesgo del banco.

A continuación, entraremos más en detalle en el Pilar I, puesto que es este el que más íntimamente relacionado está con el propio desarrollo de los modelos de riesgo de crédito.

En Basilea I se estipulaba que los bancos debían tener en reservas el 8% del valor de sus activos ponderados al riesgo. Sin embargo, en Basilea II, se argumenta que se deben tener en cuenta otra serie de riesgos, de crédito, de mercado y operacionales. Por ello se debe calcular el Capital Adequacy Ratio (CAR), valor que como mínimo tendrá que ser el 8% definido en Basilea I. Para ello, los activos ponderados al riesgo, también conocidos como RWA por sus siglas en inglés, se dividen ahora en RWA ponderados al riesgo de crédito, RWA ponderados al riesgo de mercado y RWA ponderados al riesgo operativo (Moody's Analytics, 2011). Si dividimos el capital del que dispone un banco entre la suma de estos tres tipos de activos ponderados al riesgo obtendremos el CAR.

$$CAR = \frac{Capital}{RWA_{Riesgo\ de\ capital} + RWA_{Riesgo\ de\ mercado} + RWA_{Riesgo\ operativo}}$$

Ecuación 2: Capital Adequacy Ratio

Ecuación de la cual podemos derivar la siguiente formula, que nos indica el capital del que deben disponer los bancos:

$$Capital = CAR \times (RWA_{Riesgo\ de\ capital} + RWA_{Riesgo\ de\ mercado} + RWA_{Riesgo\ operativo})$$

Ecuación 3: Capital

La siguiente diferencia de Basilea II y Basilea I reside en las posibilidades del cálculo de los activos ponderados al riesgo. En (ECB, 2004) se indican las metodologías que se pueden emplear para los tres tipos de riesgo a tener en cuenta en el cálculo de los RWA:

- Riesgo de crédito: existen el enfoque estándar y los métodos basados en calificaciones internas (IRB), los cuales son dos, el enfoque IRB básico y el enfoque IRB avanzado.
- Riesgo de mercado: existen el enfoque estándar y el método basado en modelos internos.
- Riesgo operativo: existen el enfoque estándar, el enfoque basado en indicadores básicos y el enfoque de mediciones avanzadas.

Siguiendo con el objetivo del proyecto, nos centraremos en los enfoques empleados para el cálculo de riesgo de crédito.

El método estándar se basa en la clasificación de los deudores basándose en sistemas de calificación de crédito proporcionados por agencias externas (Chatterjee, 2015). Sin embargo, Basilea especifica una lista de seis criterios para aceptar los diferentes sistemas de calificación de crédito, tratando de mitigar la posibilidad de calificaciones optimistas que no cuantifiquen el riesgo de manera adecuada tratando de reducir el capital que los bancos deben reservar. Los criterios son objetividad, independencia, acceso internacional y transparencia, divulgación de información, recursos y credibilidad (Van Roy, 2005). Como consecuencia de los estrictos requisitos, las únicas agencias aceptadas por el comité de Basilea son S&P, Moody's y Fitch (Van Roy, 2005). En la *Imagen 1* podemos observar una comparación entre las calificaciones de estas tres agencias de calificación de crédito. La calidad del crédito va disminuyendo a medida que bajamos en la tabla, siendo aquellas calificaciones en la zona de non-investment grade como no recomendables para la inversión debido a su alto riesgo.

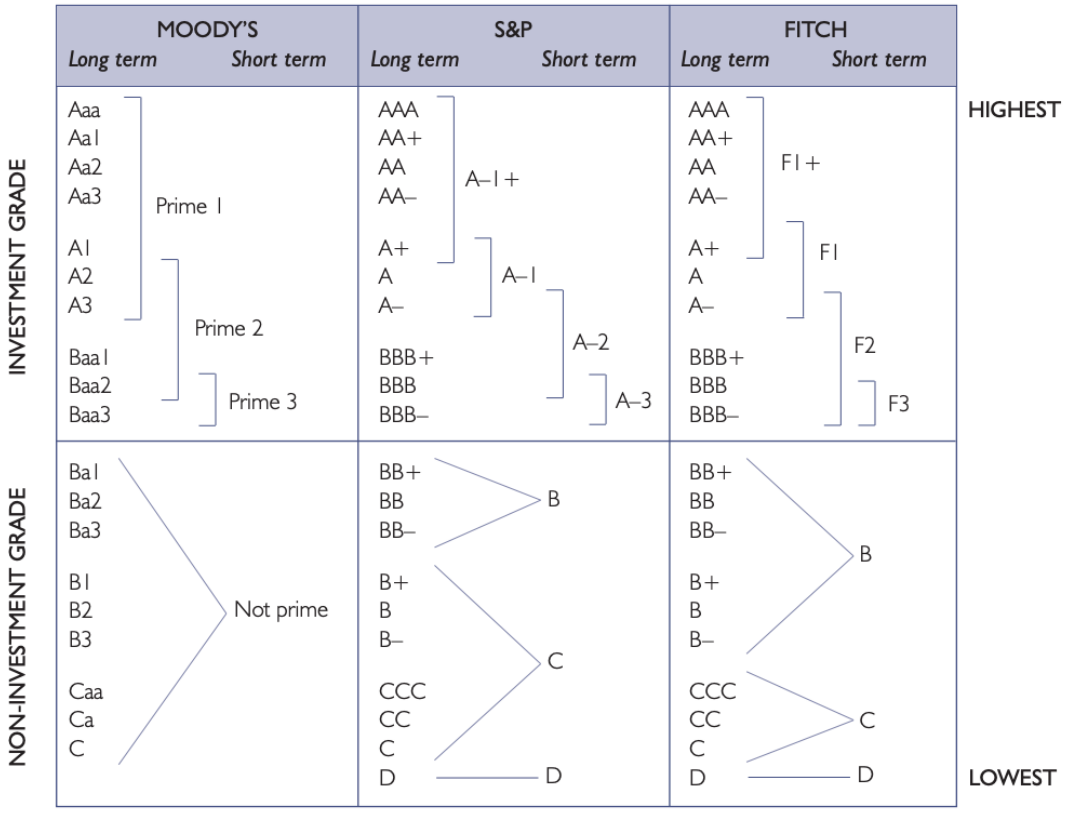

Ilustración 1: Comparación calificaciones de crédito

Fuente: (Santos, 2008)

Los métodos basados en calificaciones internas (IRB) permiten a los bancos prescindir de agencias de calificación externas, pudiendo desarrollar sus modelos propios para calcular el capital mínimo requerido (Chatterjee, 2015). En IRB se definen cuatro

parámetros, la probabilidad de impago o default (PD), que estima la probabilidad de que un deudor cometa impago; la exposición en caso de default (EAD), es decir, la cantidad del préstamo que queda por pagar, la pérdida en caso de default (LGD), que mide la parte de la exposición que es probable que se pierda en caso de impago y el vencimiento del préstamo (ECB, 2004).

La principal diferencia entre el enfoque IRB básico y el avanzado es que en el básico los bancos únicamente estiman la probabilidad de default, usando estimaciones proporcionadas por los reguladores para los parámetros LGD y EAD. Por el contrario, en el enfoque avanzado de los métodos basados en calificaciones internas, se permite que los bancos estimen los tres parámetros (Basel Committee on Banking Supervision, 2001).

El desarrollo del resto del trabajo se enfocará en el enfoque básico de IRB, es decir, se tratará de predecir la probabilidad de default de una serie de clientes. El principal motivo detrás de esta decisión es la dificultad a la hora de obtener datos que permitan estimar los parámetros de la pérdida en caso de default y de la exposición en caso de default. En la siguiente sección estudiaremos los diferentes modelos analíticos que se van a emplear en la estimación de la probabilidad de default.

2.3 Tipos de modelos

La metodología que se va a emplear para la predicción de la probabilidad de default va a consistir en el entrenamiento de modelos que sean capaces de predecir la probabilidad de impago media de una serie de clientes. Para ello, trataremos de predecir con los modelos si un cliente cometerá un impago o no, es decir, trataremos el impago como una variable binaria. Una vez el modelo se haya entrenado realizaremos las predicciones de qué deudores cometerán impago y dividiendo esa cifra entre el total de clientes de la muestra podremos obtener la probabilidad de default de la muestra. La idea es que estos modelos se puedan aplicar a carteras de préstamos similares y obtener sus PD medias. Por ejemplo, si la información que tenemos es de hipotecas a particulares en España, podríamos conocer su probabilidad de impago media.

Los modelos que se han elegido para la predicción del impago son la regresión logística, los árboles de decisión, las redes neuronales y XGBoost, una librería que optimiza los árboles de decisión consiguiendo generalmente mejores resultados.

2.3.1 Regresión logística

Actualmente, los modelos de regresión logística son una de las técnicas más extendidas a la hora de realizar modelos de riesgo de crédito. Su sencillez combinada con los buenos resultados a los que es capaz de llegar es tipo de modelo hace que sea una excelente opción a la hora de afrontar esta clase de problemas.

La regresión logística es un modelo de clasificación, el cual es capaz de distinguir entre varias clases, aunque el uso más frecuente que se le da es en problemas de dos clases, es decir, es un tipo de modelo diseñado principalmente para la clasificación de variables binarias o dicotómicas (Park, 2013). Además, no solo es capaz de determinar si la variable dependiente pertenecerá a la clase 1 o a la clase 0, sino que esta labor la realiza mediante el cálculo de la probabilidad de que la variable dependiente, a la que a partir de ahora nos referiremos como Y, sea 1 (Shalizi, 2013). Esto hace que sea un modelo idóneo para el riesgo de crédito, puesto que no solo es capaz de predecir si ocurrirá un impago o no, sino que también nos indica la probabilidad de que dicho impago ocurra para cada una de las observaciones en nuestros datos.

La regresión logística se basa en la regresión lineal, uno de los modelos más sencillos, pues emplea una combinación lineal de las variables independientes con el objetivo de predecir la variable dependiente (Foong et al., 2018). La expresión de una regresión lineal es la mostrada en la *Ecuación 4.*

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k + \varepsilon$$

Ecuación 4: Regresión lineal

Donde y representa la variable dependiente, x_i representa el conjunto de k variables independientes que son usadas para predecir y, donde $i = 1,2, \ldots, k$. β_j representa los coeficientes de cada una de las variables independientes, para los cuales $j = 0,1,2, \ldots, k$. El término β_0 no acompaña a ninguna variable independiente y es conocido como la

ordenada en el origen (Montgomery et al., 2021). ε hace referencia al error, es decir es la diferencia entre la predicción realizada por el modelo y el valor real de la variable dependiente. La expresión del valor predicho de y, representado como $\hat{y}$, se muestra en la *Ecuación 5* (Hilbe, 2016).

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k$$

Ecuación 5: Valor predicho por regresión lineal

Podemos concluir por lo tanto que $\varepsilon = y - \hat{y}$.

La regresión logística no es más que una transformación de la regresión lineal. Como hemos explicado anteriormente, la regresión logística predice la probabilidad de que la variable dependiente sea igual a 1, probabilidad a la cual no referiremos como p. Un modelo de regresión lineal está prediciendo el indicado en la *Ecuación 6*.

$$\ln\left(\frac{p}{1-p}\right) = \hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k$$

Ecuación 6: Transformación de regresión lineal a regresión logística

Por ello, para obtener la probabilidad de que la variable dependiente sea uno debemos realizar una transformación de la expresión (Park, 2013):

$$p = \frac{e^{\hat{y}}}{1 + e^{\hat{y}}} = \frac{e^{(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k)}}{1 + e^{(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k)}} = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k)}}$$

Ecuación 7: Regresión logística

La expresión de la *Ecuación 7* es la más comúnmente empleada para expresar un modelo de regresión logística. Su peculiaridad, la cual hace a este modelo idóneo para representar probabilidades, es que los valores de dicha expresión se encuentran acotados entre 0 y 1, tal y como podemos apreciar en la siguiente gráfica.

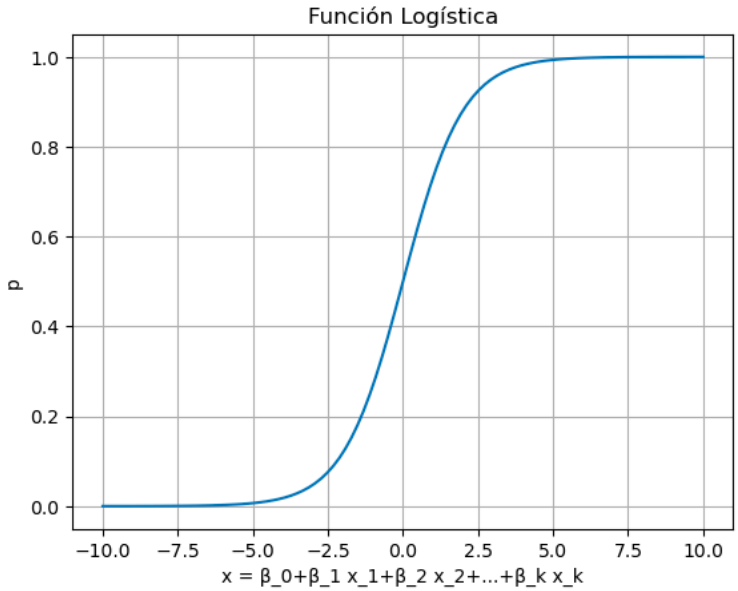

Ilustración 2: Gráfica de la función logística

Fuente: Elaboración propia

Para ser capaces de clasificar la variable entre los valores 0 y 1 a partir de su probabilidad, el procedimiento es tan simple como establecer un umbral de probabilidad a partir del cual se cambie de clase (Shalizi, 2013). Por ejemplo, podríamos decir que $y = 1$ si $p \geq 0{,}5$ e $y = 0$ si $p < 0{,}5$. Aun así, el umbral se puede definir en cualquier probabilidad dependiendo del grado de confianza que se quiera tener para afirmar que $y = 1$.

2.3.2 Redes neuronales

El siguiente método que emplearemos para el desarrollo de modelos para la predicción de impagos serán las redes neuronales. Este término se acuñó por el intento de la red de replicar el comportamiento del cerebro humano, estando su arquitectura compuesta por nodos o neuronas interconectadas entre sí (Mehlig, 2021). Actualmente representa el estado del arte en numerosas aplicaciones, sin embargo, en el ámbito del riesgo de crédito no son especialmente usadas, debido principalmente a la complejidad que pueden llegar a alcanzar.

Las neuronas o nodos son los elementos principales de las redes neuronales. Éstas reciben una serie de estímulos como entrada, agregan y transforman la información y, finalmente, transmiten el resultado obtenido. El funcionamiento de una neurona se puede entender analizando sus distinta partes y funciones. En (Munt, 2018) se detalla el funcionamiento

de estos nodos. Comenzamos con los **valores de entrada**, los cuales pueden ser los datos originales con los que se entrenará la red o pueden provenir de otras neuronas. Cada uno de ellos será multiplicado por un **peso**, encontrándose estos pesos en todas las conexiones que existen en la red y pudiendo tomar valores positivos, negativos o cero, en cuyo caso la conexión entre esas dos neuronas no se encontrará activa (Mehlig, 2021). Posteriormente, las multiplicaciones de los pesos por los valores de entrada podrán ser sumadas, multiplicadas o se podrá seleccionar el producto cuyo valor sea máximo. A pesar de que existes más criterios de selección de entrada, estos tres son los más comunes (Matich, 2001). El valor obtenido se pasa por una función conocida como **función de activación**. Previamente se ha mencionado que las multiplicaciones de los valores de entrada por los pesos pueden ser sumada, multiplicadas o que se realice una selección del término máximo. Sin embargo, el método más extendido es el de la suma, lo cual supone un problema: la red sería un conjunto de regresiones lineales y únicamente sería capaz de replicar comportamientos lineales. Es por ello por lo que se emplean las funciones de activación, funciones las cuales son no lineales, permitiendo que la red sea capaz de predecir comportamientos tanto lineales como no lineales (Sharma et al., 2017). Existen gran variedad de funciones de activación, teniendo cada una de ellas una serie de propiedades que las hacen adecuadas para diferentes tipos de aplicaciones. Entre las funciones de activación, las más comunes son las siguientes:

- Función escalón: función de activación adecuada para la clasificación de variables dicotómicas. La función devuelve un 1 si el valor de entrada es positivo o cero y un 0 en el caso en el que el valor de entrada sea negativo (Sharma et al., 2017).

$$f(x) = \begin{cases} 0 \; si \; x < 0 \\ 1 \; si \; x \geq 0 \end{cases}$$

Ecuación 8: Función escalón

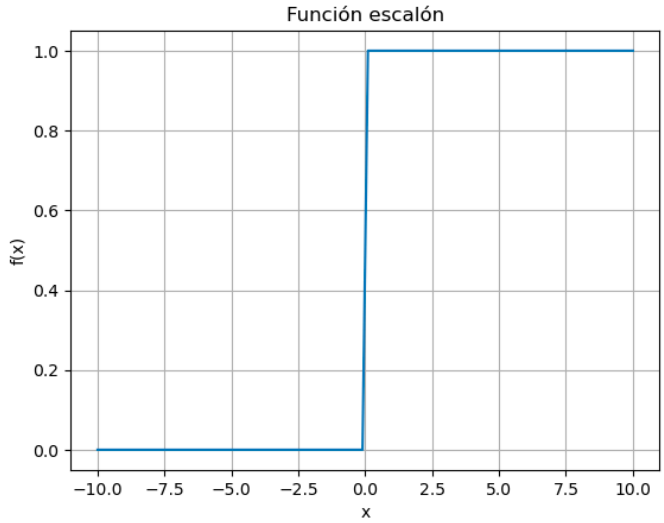

Ilustración 3: Gráfica de la función escalón

Fuente: Elaboración propia

- Función sigmoide: de la misma forma que la función escalón, esta función de activación es empleada en casos donde la variable dependiente es binaria. Sin embargo, presenta una serie de particularidades que la diferencia de la previa. A pesar de estar acotada entre los valores 1 y 0, esta función puede devolver cualquier valor comprendido entre dicho intervalo (Dubey et al., 2022). Gracias a ello, esta función resulta adecuada para casos en los que tratamos de predecir probabilidades. Es por esta razón por la cual será la función de activación elegida en la capa de salida para el desarrollo del modelo de riesgo de crédito que tiene como objetivo este trabajo.

$$f(x) = \frac{1}{1 + e^{-x}}$$

Ecuación 9: Función sigmoide

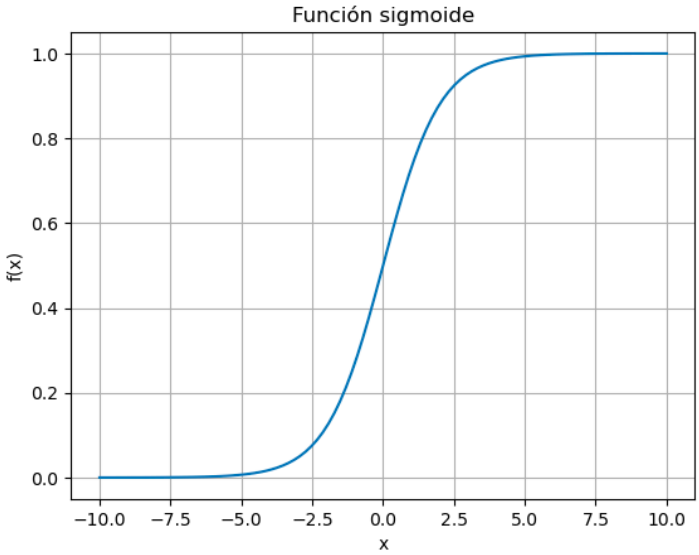

Ilustración 4: Gráfica de la función sigmoide

Fuente: Elaboración propia

- Función tangente hiperbólica: esta función de activación se presenta como solución a un problema presentado por la función sigmoide. El problema, conocido como problema de desvanecimiento de gradiente, sucede cuando, tras el entrenamiento prolongado de la red, las salidas de las funciones de activación que son cero dificultan el aprendizaje de la red (Pascanu et al., 2013). La función de tangente hiperbólica, al acotar sus salidas entre -1 y 1 reduce considerablemente la magnitud de este problema (Dubey et al., 2022).

$$f(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$

Ecuación 10: Función tangente hiperbólica

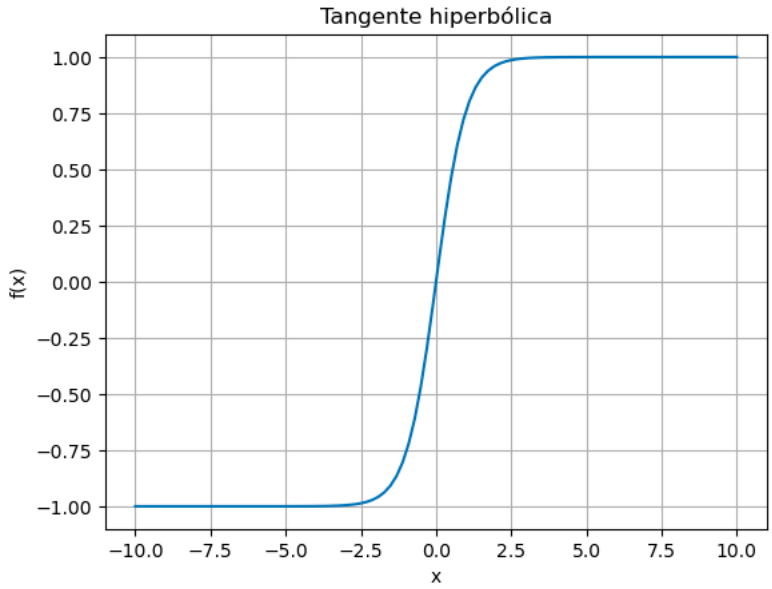

Ilustración 5: Gráfica de la función tangente hiperbólica

Fuente: Elaboración propia

- Función ReLU: esta función, de forma similar a la función escalón, toma el valor cero para aquellas entradas negativas y devuelve el propio valor de entrada a la función para las entradas iguales o mayores que cero (Sharma et al., 2017). Su utilidad reside en que, al devolver un cero para las entradas negativas, se está desactivando la neurona. Esto es una efectiva técnica con el overfitting, es decir, el sobreajuste de la red a los datos de entrenamiento el cual resulta en una pérdida en la capacidad de generalización del modelo (Srivastava et al., 2014).

$$f(x) = \max(x, 0)$$

Ecuación 11: Función ReLU

El valor obtenido como salida de la función de activación se compara con un **umbral**, mediante el cual se determinará si el valor obtenido se sigue transmite a la siguiente neurona o si se desactiva la neurona. Para que suceda lo primero se debe superar o igualar el umbral definido, en cuyo caso la neurona tiene como salida el valor de la función de activación (Munt, 2018).

Comprendido el comportamiento de una única neurona podemos tratar de comprender como se construyen redes más complejas. Para ello debemos únicamente agrupas varias neuronas en diferentes capas. Existen tres tipos diferentes de capas, la capa de entrada,

una o varias capas ocultas y la capa de salida. La capa de entrada es aquella cuyas neuronas reciben como valores de entrada los datos que estemos empleando para entrenar el modelo. La capa de salida, por el contrario, es aquella en la que las salidas de sus neuronas constituyen la salida de la red neuronal, es decir, el resultado del modelo. Por último, las neuronas de las capas ocultas son aquellas que reciben como entrada la salida de otras neuronas y que, además, su salida será la entrada de las neuronas de la siguiente capa (Munt, 2018).

Finalmente, el último concepto básico que nos queda por estudiar para poder comprender el funcionamiento de estas redes son los mecanismos que existen para su aprendizaje o entrenamiento. Cabe recalcar que las redes neuronales pueden emplearse tanto para tareas de aprendizaje supervisado, es decir, conociendo el valor real que tratamos de predecir, como de aprendizaje no supervisado. En el caso de estudio de este proyecto tratamos con datos supervisados, pues conocemos qué clientes han cometido default y cuáles no. Consecuentemente, nos centraremos en explicar los métodos de entrenamiento de las redes neuronales supervisadas. Debido a la complejidad de las redes con capas ocultas, el algoritmo de entrenamiento es complejo. Sin embargo, para el alcance de este trabajo, es suficiente con entender que el resultado obtenido de la red se compara con el valor real y la diferencia entre ambos se propaga de salida a entrada de la red, empleando un algoritmo conocido como Backpropagation, ajustando los pesos de toda la red de forma proporcional al error (Munt, 2018).

2.3.3 Árboles de decisión

El tercer tipo de modelo que se propone para el desarrollo de los modelos de predicción de la probabilidad de default son los árboles de decisión. Existen árboles de decisión que se pueden emplear tanto en tareas de clasificación como de regresión (Loh, 2011). Ciñéndonos al caso de estudio de este trabajo, la explicación se centrará en los árboles de clasificación únicamente. Estos se basan en la representación de los datos mediante un árbol compuesto por nodos u hojas. El primer nodo contiene todas las observaciones del conjunto de datos con el objetivo de segmentar dicho conjunto de datos en subconjuntos en función del valor que adopten en alguna de las variables. El objetivo es alcanzar nodos con conjuntos de observaciones lo suficientemente homogéneas como para poder extraer reglas (Hagenlocher, 2017). Un ejemplo de regla, para el caso de modelos de riesgo de

crédito, sería una serie de atributos que identifiquen grupos que cometerán impago o que no.

Las divisiones de los nodos se realizan empleando las como criterio de partición aquella variable que den como resultado nodos hijo los más homogéneos posibles en cuanto a su clase. El concepto de homogeneidad o pureza del nodo se mide de diferentes formas dependiendo del algoritmo que se emplee. Los tres algoritmos que vamos a tratar son C4.5 y CART. El algoritmo C4.5 emplea la entropía como la medida de pureza del nodo, mientras que CART usa el índice de Gini.

La entropía es una medida que cuantifica el grado de incertidumbre de una variable (Vajapeyam, 2014). Si en un nodo todas las observaciones pertenecen a la misma clase, su grado de incertidumbre o su entropía será cero. Esto viene definido en la *Ecuación 12*, desarrollada por Shannon en 1948 (Wang et al., 2016).

$$H(X) = -\sum_{i=1}^{n} p(x_i) \ln\ p(x_i)$$

Ecuación 12: Entropía de Shannon

La expresión de la entropía, $H(X)$, es el resultado del sumatorio, para todas las diferentes clases que se encuentran en un nodo, representadas por n, de la probabilidad de aparición de dicha clase x_i multiplicado por el logaritmo de esa misma probabilidad. Si estudiamos los casos extremos que se pueden dar, es decir, que todas las observaciones sean de una clase o que una clase no tenga observaciones, comprobaremos que para ambos casos la entropía es cero, pues uno de los dos términos del sumatorio será cero. Esto tiene sentido, pues no hay incertidumbre si no tenemos una clase o si todos los casos pertenecen a dicha clase. Es por ello por lo que se dividirá el nodo en base a aquella variable que más reduzca la entropía con respecto a la entropía del nodo del que procede.

Por otro lado, el índice de Gini, como hemos mencionado con anterioridad, es una medida que nos permite estudiar cómo de bien una variable separa los nodos según su pureza. Esto lo hace calculando el siguiente índice mediante la expresión indicada en la *Ecuación 13*.

$$Índice\ de\ Gini = 1 - \sum_{j} {p_j}^2$$

Ecuación 13: Índice de Gini

Donde p_j es la probabilidad que tiene la clase j dentro de un nodo. Para optimizar el árbol se tiene como objetivo reducir el índice de Gini tras cada división, siendo el límite de un árbol cuando todos sus nodos tienen un índice igual a cero (Sundhari, 2011).

Finalmente, es de vital importancia controlar la división de los nodos de un árbol, ya que, si se deja a los algoritmos actuar sin ninguna medida de control, dividirán el árbol hasta que todos los nodos contengan únicamente una clase por hoja. Esta situación sería ideal de no ser por la alta probabilidad de que el árbol se haya ajustado en exceso a los datos observados. El inconveniente es que su rendimiento será significativamente menor sobre datos no observados, los cuales, al fin y al cabo, son los que realmente deseamos predecir (Amro et al., 2021). Para solventar este problema se emplea una técnica conocida como prunning o poda, la cual se basa en establecer criterios que controlen la expansión del árbol. Dentro de los criterios de poda, los más comunes pueden ser establecer un número mínimo de observaciones por hoja, es decir, si una división resulta en un nodo con menos observaciones que las especificadas esa división no se realizará; o establecer una profundidad máxima del árbol (Esposito et al., 1997).

2.3.4 XGBoost

XGBoost es una librería que emplea gradient boosting para diversas aplicaciones de machine learning (XGBoost Developers, 2022), que en esta aplicación de estudio será empleado para la optimización de árboles de decisión con la finalidad de obtener predicciones acerca de la predicción de divisas.

La optimización de los árboles de decisión se lleva a cabo mediante el método de gradient boosting, el cual es un método de ensemble, es decir, el uso de numerosos árboles de decisión, los cuales por si solos no tienen un alto poder predictivo, sin embargo, al usarse de forma combinada los resultados mejorar sustancialmente (Ganaiea et al., 2022). El funcionamiento del método de gradient boosting se basa en la corrección de los errores

residuales del árbol previo. Primero se crea un árbol, cuyos resultados son comparados con los reales, obteniendo los errores residuales para cada una de las observaciones. A continuación, el siguiente árbol que se genere tendrá como objetivo corregir los errores del árbol previo. Una vez el nuevo árbol obtenga las nuevas predicciones habiendo tratado de mejorar los resultados del árbol que le precedía, las predicciones de todos los árboles generados hasta ese momento se combinarán para obtener la predicción final. Este proceso se seguirá repitiendo hasta que se alcance el número de árboles deseado (Fafalios et al., 2020).

3 Estudio empírico

En esta sección del proyecto nos centraremos en el desarrollo de los modelos, comenzando con el estudio de los datos y, posteriormente, llevando a cabo el desarrollo de los cuatro modelos planteados con anterioridad en este trabajo.

3.1 Datos

Esta sección desarrollará el estudio de los datos, comenzando con una introducción del origen de los mismos, para a continuación llevar a cabo un estudio exhaustivo de éstos. Se explicará el contenido de los datos, centrándonos en las variables que aparecen y su significado, para posteriormente llevar a cabo un análisis exploratorio. De forma paralela al estudio exploratorio de los datos se irán realizando las depuraciones necesarias de los mismos para poder utilizarlos en el entrenamiento de los modelos.

3.1.1 Introducción

El desarrollo de los modelos se llevará a cabo empleando los datos de Lending Club, una compañía estadounidense de préstamos entre particulares fundada en 2006. Los préstamos entre particulares se basan en el siguiente funcionamiento: cualquier persona aplica a un préstamo y son los inversores de la compañía los que financian a los deudores. Previamente, se analiza su riesgo y, en base a ello, se determina el interés que se aplicará a los créditos (Forbes, s.f.).

Actualmente la empresa ha cesado sus operaciones en el sector de los créditos entre particulares, también conocido por el término inglés peer-to-peer lending. Es por ello por lo que el acceso a los datos a través de su página web ya no es posible. Sin embargo, esta fuente de datos ha sido de uso público durante numerosos años, siendo empleada en gran cantidad de proyectos, blogs y competiciones de desarrollo de modelos de riesgo de crédito. Como consecuencia, los datos se encuentran disponibles en un gran número de plataformas. Para este proyecto los datos fueron obtenido de Kaggle.

Kaggle es una plataforma web cuya finalidad es promover competiciones en el ámbito del machine learning y de la ciencia de datos. Fundada en 2010, la plataforma contaba en

2021 con más de ocho millones de usuarios registrados (Uslu, s.f.), convirtiéndola en una referencia en el mundo de la ciencia de datos. Sus competiciones, a las cuales cualquier usuario se podía unir, fomentan un ambiente de innovación y comunidad que ha ayudado al sector a continuar con su acelerado desarrollo.

3.1.2 Análisis exploratorio y depuración de los datos

Los datos obtenidos de Kaggle están divididos en dos conjuntos de datos diferentes. El primero hace referencia a aquellas aplicaciones de préstamo que han sido aceptadas y de las cuales tenemos información relacionada con la evolución de los préstamos. El segundo conjunto de datos contiene aquellos aplicantes cuyas solicitudes han sido rechazadas y, como consecuencia de ello, no es posible conocer si han tenido lugar incumplimientos en el pago de las deudas o no. Es por este motivo que se empleará únicamente el primer conjunto de datos para el entrenamiento de los modelos.

Este conjunto de datos es un fichero *csv* con el nombre *accepted_2007_to_2018Q4.csv*, el cual incluye todos los préstamos que la compañía aceptó entre los años 2007 y 2018, ambos inclusive. En total cuenta con 2.260.701 créditos, de los cuales se dispone información de un total de 151 variables. El alto volumen de datos hace que el peso total del fichero que contiene los datos sea de +2,5 GB. Es probable que, debido al gran peso de dicho archivo, sea necesario en un futuro, cuando llegue el momento de entrenar los modelos, llevar a cabo un muestreo de los datos con el objetivo de reducir su tamaño. Esto será debido a que determinados modelos como las redes neuronales, los cuales contienen números muy elevados de parámetros, requieran de tiempos de computación demasiado elevados. Aun así, será necesario llevar a cabo primero la depuración de los datos, pues es muy probable que la falta de calidad de ciertas variables u observaciones nos lleve a retirar información y, como resultado, a reducir el tamaño de los datos.

3.1.2.1 **Variable dependiente**

Dentro de las 151 variables, podemos distinguir entre las variables independientes y la variable dependiente. La variable dependiente es la variable que queremos modelar para ser capaces de predecir su comportamiento. Esta variable es *loan_status* y hace referencia al estado del crédito. La variable es de tipo categórica, es decir, sus posibles valores hacen

referencia a deferentes clases que representan el estado en el que se encuentra el préstamo. En la siguiente tabla observamos las nueve diferentes clases y el número de observaciones de cada una de ellas.

Clase	Número de observaciones
Fully Paid	1076751
Current	878317
Charged Off	268559
Late (31-120 days)	21467
In Grace Period	8436
Late (16-30 days)	4349
Does not meet the credit policy. Status: Fully Paid	1988
Does not meet the credit policy. Status: Charged Off	761
Default	40

Tabla 1: Clases de la variable dependiente

Fuente: Elaboración propia

Los modelos que vamos a desarrollar son modelos de clasificación para una variable dicotómica, es decir, una variable que únicamente pueda tomar dos valores. Estos dos valores son el 1 y el 0, que hacen referencia a si se ha cometido un impago o si no, respectivamente. Necesitamos entonces agrupar las nueve posibles clases que se pueden dar dentro de las clases impago y no impago. Dentro de la clase 0, es decir, aquella que engloba los créditos sanos, incluiremos únicamente las observaciones de la clase *Fully Paid*, puesto que es la clase que representa a los créditos ya finalizados en los cuales los pagos se han realizado correctamente. Por otro lado, la clase *current* será descartada, ya que ésta recoge los préstamos que aún siguen abiertos y que no han tenido impagos. Es necesario retirar esta clase, ya que no representa ni créditos sanos, debido a que todavía pueden cometer default, ni créditos impagados. La clase *charged off* será la que represente aquellos expedientes que han cometido impago. El resto de las clases se descartarán, ya que no hacen referencia a impago, sino a diferentes tipos de morosidad, concepto que no se va a incluir en los modelos a desarrollar. Tras los reajustes en las etiquetas de las clases y la retirada de los expedientes abiertos que no han cometido impago o aquellos que hacen referencia a situaciones de morosidad, la distribución de las clases de la variable

loan_status es la siguiente: la clase 0, que se refiere a los expedientes cerrados sanos, tiene 1076751 observaciones, mientras que la clase 1, que representa los expedientes con algún tipo de impago, tiene 268559 observaciones. Por lo que la clase 0 representa el 80,04% de las observaciones y la clase 1 el 19,96%.

3.1.2.2 **Variables independientes**

Tras estudiar y llevar a cabo los ajustes necesarios sobre la variable dependiente, analizaremos la calidad de las variables independientes, es decir, aquellas que se emplearan para modelar variable dependiente. El objetivo será retirar aquellas que no contengan información útil para el modelo y poder llevar a cabo posteriormente un examen exhaustivo de las variables independientes que sí aporten información de calidad.

Comenzaremos la exploración estudiando el número de datos faltantes de las distintas variables independiente, ya que, tras una primera exploración visual de los datos, se ha observado que hay una serie de variables que se encuentran no informadas en un alto porcentaje de los datos. Se ha decidido retirar todas las variables que tengan más de un 10% de observaciones no informadas. Puede parecer que el umbral establecido es demasiado estricto y que se están eliminando variables que pueden aportar información. Aunque esto sea cierto, consideramos que ésta es la mejor solución a un problema con difícil arreglo. Si tratásemos de conservar variables con más datos faltantes, deberíamos llevar a cabo a posteriori alguna acción para tratar los NAs. Una opción sería rellenar los NAs con valores, pero desconocemos el motivo por el que no se tiene esa información y por lo tanto, estaríamos alterando los datos de manera arbitraria. La otra opción sería retirar las observaciones con datos faltantes, sin embargo, esto tendría dos efectos negativos. Por una parte, podríamos estar introduciendo un sesgo en los datos que haga que la predicción no se ajuste bien a los datos reales, mientras que, por otra parte, al haber tantos NAs, el tamaño de los datos se vería reducido considerablemente, pasando de más de un millón de observaciones a pocas decenas de miles.

Por lo tanto, se retirarán las variables con más de un 10% de NAs, las cuales son las siguientes:
member_id, desc, mths_since_last_delinq, mths_since_last_record, next_pymnt_d, mths_since_last_major_derog, annual_inc_joint, dti_joint, verification_status_joint,

open_acc_6m, open_act_il, open_il_12m, open_il_24m, mths_since_rcnt_il, total_bal_il, il_util, open_rv_12m, open_rv_24m, max_bal_bc, all_util, inq_fi, total_cu_tl, inq_last_12m, mths_since_recent_bc_dlq, mths_since_recent_inq, mths_since_recent_revol_delinq, revol_bal_joint, sec_app_fico_range_low, sec_app_fico_range_high, sec_app_earliest_cr_line, sec_app_inq_last_6mths, sec_app_mort_acc, sec_app_open_acc, sec_app_revol_util, sec_app_open_act_il, sec_app_num_rev_accts, sec_app_chargeoff_within_12_mths, sec_app_collections_12_mths_ex_med, sec_app_mths_since_last_major_derog, hardship_type, hardship_reason, hardship_status, deferral_term, hardship_amount, hardship_start_date, hardship_end_date, payment_plan_start_date, hardship_length, hardship_dpd, hardship_loan_status, orig_projected_additional_accrued_interest, hardship_payoff_balance_amount, hardship_last_payment_amount, debt_settlement_flag_date, settlement_status, settlement_date, settlement_amount, settlement_percentage y *settlement_term.*

A continuación, al haber retirado la mayoría de los NAs, podemos eliminar las observaciones con datos faltantes. Al hacer esto, pasamos de 1345341 observaciones a 1090611, el cual sigue siendo un número considerable de clientes para poder realizar un modelo robusto. Para confirmar que no se introduce un sesgo significativo al eliminar las observaciones, hemos analizado el cambio porcentual tanto en la media como en la desviación típica de las variables numéricas. En la mayoría de las variables la variación de los dos parámetros estadísticos es como máximo del orden de $\pm 3\%$. Por ello, damos por satisfactorio el proceso de tratamiento de datos faltantes.

A continuación, realizaremos un segundo estudio de las variables, eliminando aquellas que no aporten información relevante. Comenzamos realizando una inspección ocular de los datos y observamos que las variables *id, url y zip_code* no aportan información útil. La primera es únicamente un índice numérico que indica la posición dentro del conjunto de datos, la segunda es la dirección URL a la página de Lending Club que contiene los datos y la tercera es el código postal con dígitos censurados por motivos de privacidad

Observamos, tras realizar una descripción de los valores que toman las diferentes variables numéricas, que algunas toman un único valor para todos o la gran mayoría de

los casos. Se procederá a eliminar las variables homogéneas del conjunto de datos. Dichas variables son las siguientes:

out_prncp, out_prncp_inv, total_rec_late_fee, collections_12_mths_ex_med, policy_code, acc_now_delinq, chargeoff_within_12_mths, delinq_amnt, num_tl_120dpd_2m y *num_tl_30dpd.*

El siguiente paso que se va a tomar es el tratamiento de las variables categóricas. La mayoría de los modelos que se van a emplear necesitan que todos los datos que se introduzcan en sus arquitecturas sean numéricos. Es por ello por lo que debemos estudiar dichas variables y en los casos que sea apropiado, convertirlas para que puedan ser utilizadas. El primer paso será analizar las diferentes categorías que toma cada una de las variables, observando que las variables *pymnt_plan, hardship_flag, application_type, disbursement_method* y *debt_settlement_flag* presentan una única clase para todas o la gran mayoría de las observaciones, por lo tanto, serán eliminadas del conjunto de datos.

Por el contrario, se observa también la situación contraria, es decir, variables con un número de clases muy elevado. Se procederá a retirar estas clases del conjunto de datos, ya que no se podrá interpretar el verdadero significado de cada una de las categorías de la variable. Las variables en las que se da esta situación son *emp_title*, con 317413 clases posibles, y *title*, con 35168 clases.

Tras haber seleccionado las variables categóricas con las que deseamos trabajar, será necesario convertirlas a variables numéricas. Sin embargo, como este proceso no se deberá hacer para todos los modelos, se reservará su explicación para futuras secciones.

El último tipo de dato al que nos debemos enfrentar son las fechas. Actualmente, dentro del conjunto de datos existen cuatro variables de fecha: *issue_d, earliest_cr_line, last_pymnt_d* y *last_credit_pull_d*. Todas ellas se encuentran expresadas con el formato mmm-aaaa, es decir, los meses son indicados con tres letras y los años aparecen completos. Para su incorporación a los modelos, se ha decidido separar la fecha en dos nuevas columnas, una para el mes, que será expresado con su valor numérico del 1 al 12, y otra para el año. Tras realizar el cambio, se retiran las variables originales.

Finalmente, las variables que se emplearán para entrenar los modelos aparecen en la siguiente tabla, junto con el tipo de dato y una breve descripción. En total el conjunto de datos está compuesto por 76 variables, siendo una de ellas la variable dependiente, concretamente esta variable es *loan_status*.

Variable	**Tipo de dato**	**Descripción**
loan_amnt	float64	Cantidad del crédito a la que se aplica
funded_amnt	float64	Cantidad total comprometida en ese momento
funded_amnt_inv	float64	El importe total comprometido por los inversores para ese préstamo en ese momento
term	category	Número de pagos del crédito
int_rate	float64	Interés del crédito
installment	float64	Cantidad del pago mensual
grade	category	Calificación del crédito por parte de Lending Club
sub_grade	category	Sub-calificación del crédito por parte de Lending Club
emp_length	category	Tiempo que lleva el deudor empleado
home_ownership	category	Estatus de propiedad de la vivienda del deudor
annual_inc	float64	Ingresos anuales del deudor
verification_status	category	Indica si el deudor está verificado
loan_status	float64	Estado del crédito
purpose	category	Motivo del crédito
addr_state	category	Estado (localización) del deudor
dti	float64	Ratio calculado con los pagos mensuales y la deuda total
delinq_2yrs	float64	Número de retrasos de +30 días en los últimos 2 años
fico_range_low	float64	Límite inferior del rango FICO
fico_range_high	float64	Límite superior del rango FICO
inq_last_6mths	float64	Número de solicitudes de crédito en los últimos 6 meses
open_acc	float64	Número de líneas de crédito abiertas del deudor

pub_rec	float64	Número de registros públicos derogatorios
revol_bal	float64	Saldo total de crédito renovable
revol_util	float64	Tasa de utilización de la línea renovable
total_acc	float64	Número total de línea de crédito históricas del deudor
initial_list_status	category	Estado inicial del préstamo
total_pymnt	float64	Pagos recibidos hasta la fecha por la parte del importe total
total_pymnt_inv	float64	Pagos recibidos hasta la fecha por la parte del importe total financiado por los inversores
total_rec_prncp	float64	Principal recibido hasta la fecha
total_rec_int	float64	Intereses recibidos hasta la fecha
recoveries	float64	Recuperaciones
collection_recovery_fee	float64	Tasa de recuperaciones
last_pymnt_amnt	float64	Cantidad del último pago
last_fico_range_high	float64	Último límite inferior del rango FICO
last_fico_range_low	float64	Último límite superior del rango FICO
tot_coll_amt	float64	Importe total de los cobros
tot_cur_bal	float64	Saldo corriente total de todas las cuentas
total_rev_hi_lim	float64	Límite total de crédito alto renovable
acc_open_past_24mths	float64	Número de operaciones abiertas en los últimos 24 meses
avg_cur_bal	float64	Saldo corriente medio de todas las cuentas
bc_open_to_buy	float64	Total abierto a comprar con tarjetas bancarias renovables
bc_util	float64	Relación entre el saldo corriente total y el límite de crédito para todas las cuentas de tarjeta bancaria
mo_sin_old_il_acct	float64	Meses desde la apertura de la cuenta a plazos más antigua
mo_sin_old_rev_tl_op	float64	Meses desde la apertura de la cuenta rotatoria más antigua
mo_sin_rcnt_rev_tl_op	float64	Meses desde la apertura de la cuenta rotatoria más reciente
mo_sin_rcnt_tl	float64	Meses desde última apertura de cuenta

mort_acc	float64	Número de hipotecas
mths_since_recent_bc	float64	Meses desde la apertura de la cuenta bancaria más reciente
num_accts_ever_120_pd	float64	Número de cuentas en mora de 120 días o más
num_actv_bc_tl	float64	Número de cuentas de tarjeta bancaria activas
num_actv_rev_tl	float64	Número de operaciones renovables activas
num_bc_sats	float64	Número de cuentas de tarjeta bancaria sanas
num_bc_tl	float64	Número de cuentas de tarjeta bancaria
num_il_tl	float64	Número de cuentas a plazos
num_op_rev_tl	float64	Número de operaciones renovables abiertas
num_rev_accts	float64	Número de operaciones renovables
num_rev_tl_bal_gt_0	float64	Número de operaciones renovables con saldo >0
num_sats	float64	Número de cuentas sanas
num_tl_90g_dpd_24m	float64	Número de cuentas con 90 o más días de mora en los últimos 24 meses
num_tl_op_past_12m	float64	Número de cuentas abiertas en los últimos 12 meses
pct_tl_nvr_dlq	float64	Porcentaje de operaciones nunca morosas
percent_bc_gt_75	float64	Porcentaje de todas las cuentas de tarjeta bancaria > 75% del límite
pub_rec_bankruptcies	float64	Número de quiebras registradas públicamente
tax_liens	float64	Número de embargos fiscales
tot_hi_cred_lim	float64	Límite de crédito alto total
total_bal_ex_mort	float64	Saldo total del crédito, excluida la hipoteca
total_bc_limit	float64	Límite de crédito total de la tarjeta bancaria
total_il_high_credit_limit	float64	Límite de crédito total a plazos
issue_d_month	int64	Mes de fecha de emisión
issue_d_year	int64	Año de fecha de emisión
earliest_cr_line_month	int64	Mes de la primera línea de crédito
earliest_cr_line_year	int64	Año de la primera línea de crédito
last_pymnt_d_month	int64	Mes del último pago
last_pymnt_d_year	int64	Año del último pago
last_credit_pull_d_month	int64	Mes de la última solicitud de crédito
last_credit_pull_d_year	int64	Año de la última solicitud de crédito

Tabla 2: Descripción de las variables finales

Fuente: Elaboración propia

3.1.2.3 **Análisis exploratorio de las variables**

Finalizada la depuración de los datos originales, se procederá a realizar un estudio exploratorio de las variables que componen los datos. El objetivo de esta sección es comprender en mayor profundidad las variables que se usarán para entrenar los modelos, comprendiendo si existen relaciones entre ellas y si sus distribuciones presentan anomalías que haya que tener en cuenta a la hora de usar dichas variables.

Comenzaremos estudiando las distribuciones que presentan las variables numéricas ayudándonos para ello de histogramas.

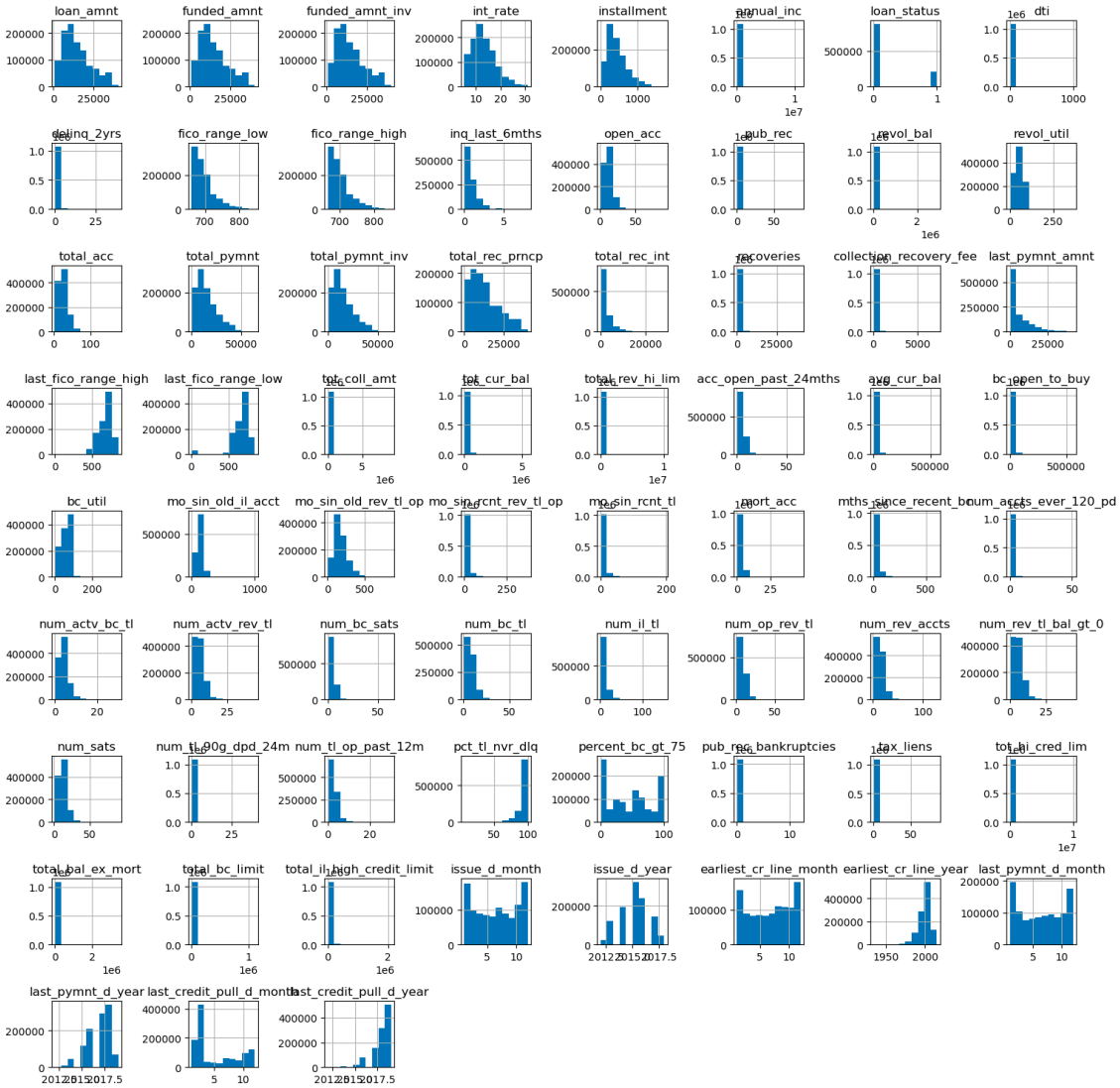

Ilustración 6: Conjunto de histogramas de las variables numéricas

Fuente: Elaboración propia

Lo primero que capta nuestra atención es que la distribución de la mayoría de las variables numéricas presenta un sesgo hacia la izquierda, lo cual es lógico, pues tratándose de variables relacionadas con créditos, es lógico que predominen las observaciones en las zonas bajas de capital, por ejemplo. Sin embargo, este hecho hace que debamos estudiar los posibles datos atípicos que pueden tener lugar en dichas variables, ya que estos son perjudiciales para las predicciones de los modelos. En los histogramas podemos identificar las variables que probablemente presenten datos atípicos o outliers de manera muy sencilla, si la distribución se encuentra sesgada hacia la derecha o la izquierda es debido a la existencia de outliers en el extremo opuesto.

Para llevar a cabo un análisis más detallado de cada una de las variables que consideramos que pueden presentar datos atípicos, procederemos a realizar gráficos de caja de cada una de ellas. Las variables que se estudiarán más a fondo son aquellas cuyo rango de valores es muy amplio, es decir, las siguientes: *anual_inc, dti, revol_bal, revol_util, total_rec_int, recoveries, collection_revovery_fee, tot_col_amt, tot_cur_bal, total_rev_hi_lim, avg_cur_bal, bc_open_to_buy, mo_sin_old_il_acct, tot_bi_cred_lim, total_bal_ex_mort, total_bc_limit* y *total_il_high_credit_limit*.

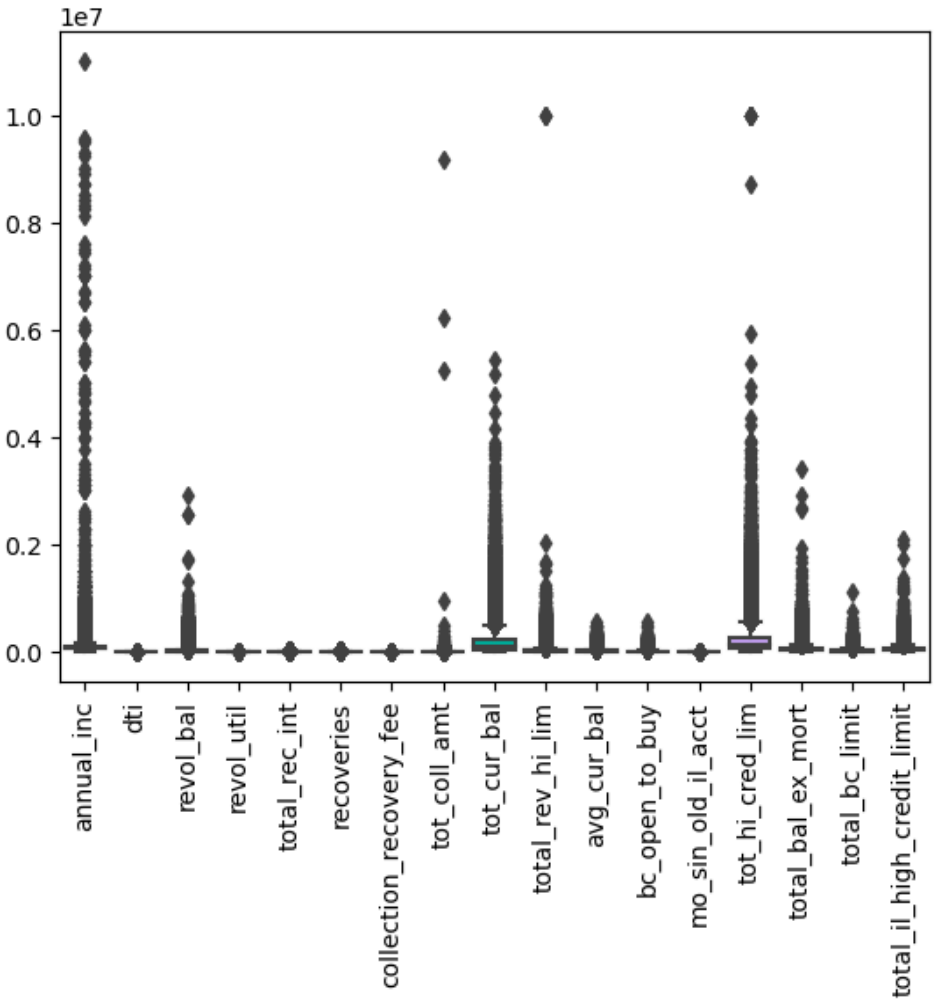

Ilustración 7: Gráficos de caja de las variables con valores atípicos

Fuente: Elaboración propia

Los gráficos de cajas muestran de manera clara la existencia de valores atípicos muy significativos en las siguientes variables: *anual_inc, revol_bal, tot_coll_amt, tot_cur_bal, total_rev_high_lim, tot_hi_cred_lim* y *total_bal_ex_mort*. Se retirará el 1% de los datos más extremos de cada variable, conservando el 99% restante de los datos. Las distribuciones de las variables sin los datos atípicos es la mostrada en la siguiente figura, donde podemos apreciar que las variables presentan distribuciones menos sesgadas. Aun así, el objetivo ha sido disminuir el sesgo, no eliminarlos, ya que este también nos aporta información sobre el comportamiento real y las características de los clientes.

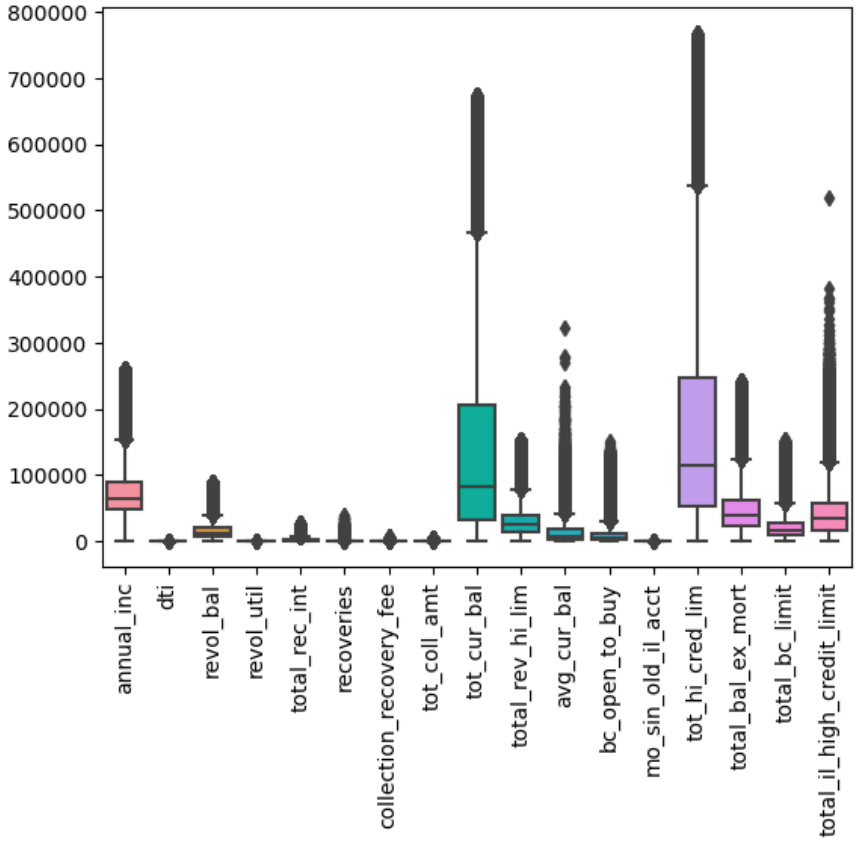

Ilustración 8: Gráficos de caja con las correcciones en las variables con valores atípicos

Fuente: Elaboración propia

Tras el estudio de las variables numéricas, analizaremos las variables categóricas. Para ello emplearemos gráficos de sector, que nos indiquen la distribución de las clases dentro de cada variable.

Comenzaremos con la variable *term*, la cual contiene dos clases indicando el número de pagos mensuales que se deben llevar a cabo. A pesar de que predominen los pagos de 36 meses, el desbalanceo de las clases es razonable y, por lo tanto, no se aplicará ningún cambio a esta variable.

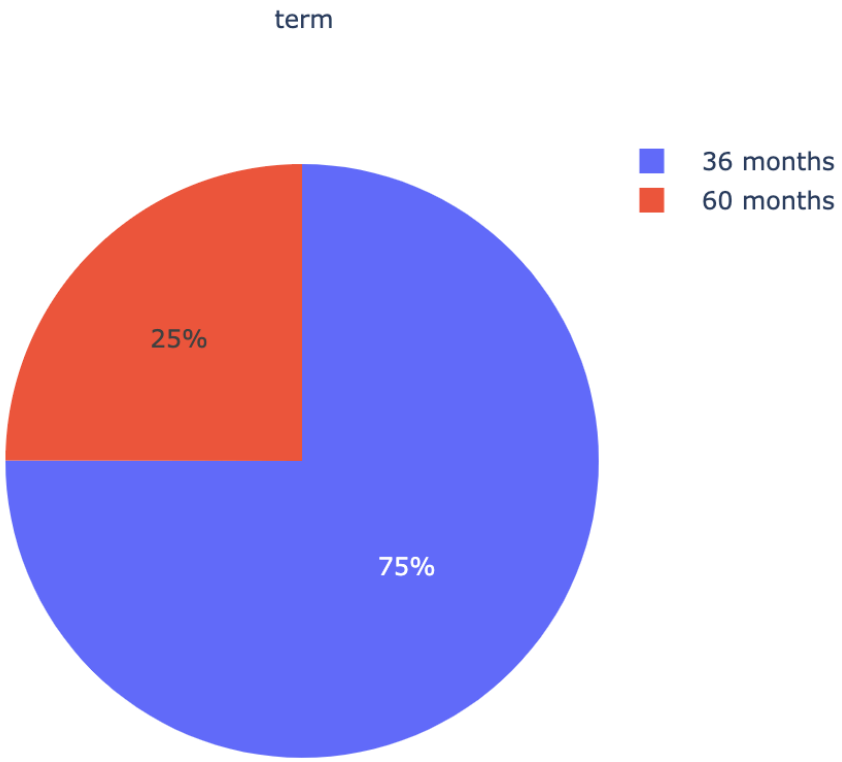

Ilustración 9: Distribución de las clases de la variable "term"

Fuente: Elaboración propia

La siguiente variable que se estudiará es *grade*, compuesta por las diferentes clasificaciones que Lending Club asigna a los créditos. La distribución se considera razonable y no se aplicarán cambios.

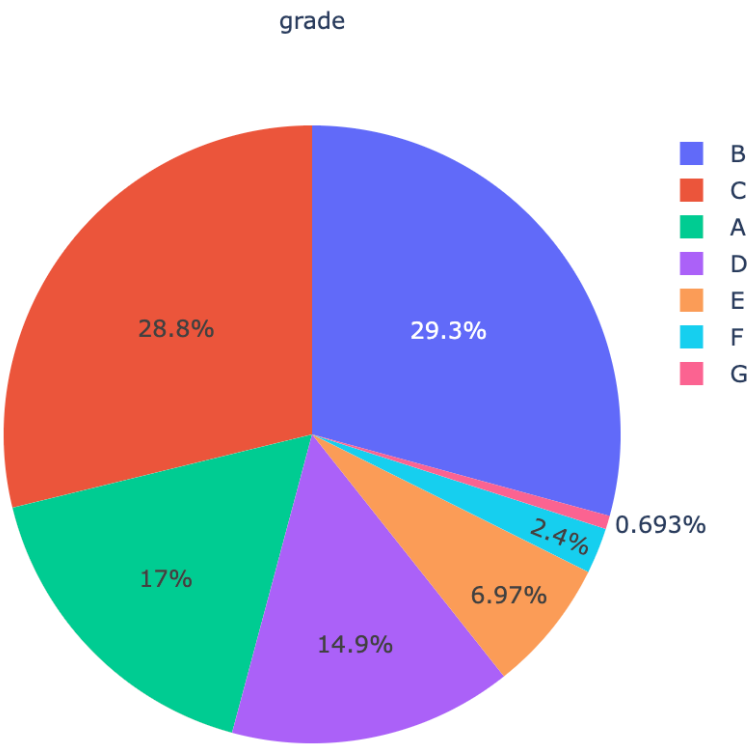

Ilustración 10:Distribución de las clases de la variable "grade"

Fuente: Elaboración propia

La variable *sub_grade* representa las subclasificaciones que Lending Club otorga a los créditos. Éstas provienen de dividir cada clasificación de la variable *grade*, estudiada previamente, en cinco clases nuevas. Observamos que la volumetría de determinadas clases es pequeña, por lo que se agruparán aquellas clases con menos de un 1% de los casos en una nueva clase conocida como *otros*. El objetivo es incluir únicamente clases representativas dentro de la variable, consiguiendo una mayor interpretabilidad por nuestra parte y por parte del modelo.

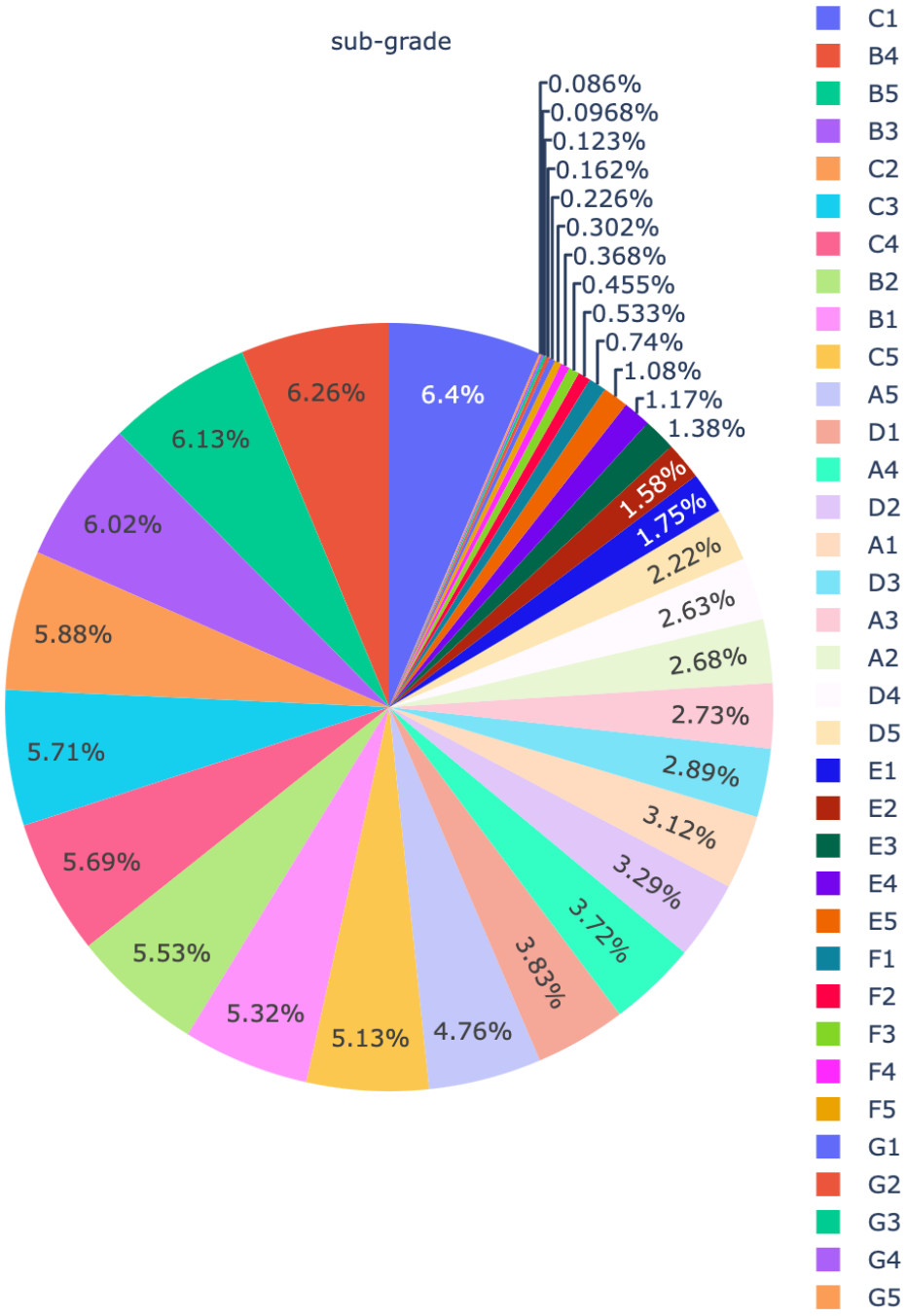

Ilustración 11: Distribución de las clases de la variable "sub_grade"

Fuente: Elaboración propia

La nueva distribución de *sub_grade* es la siguiente:

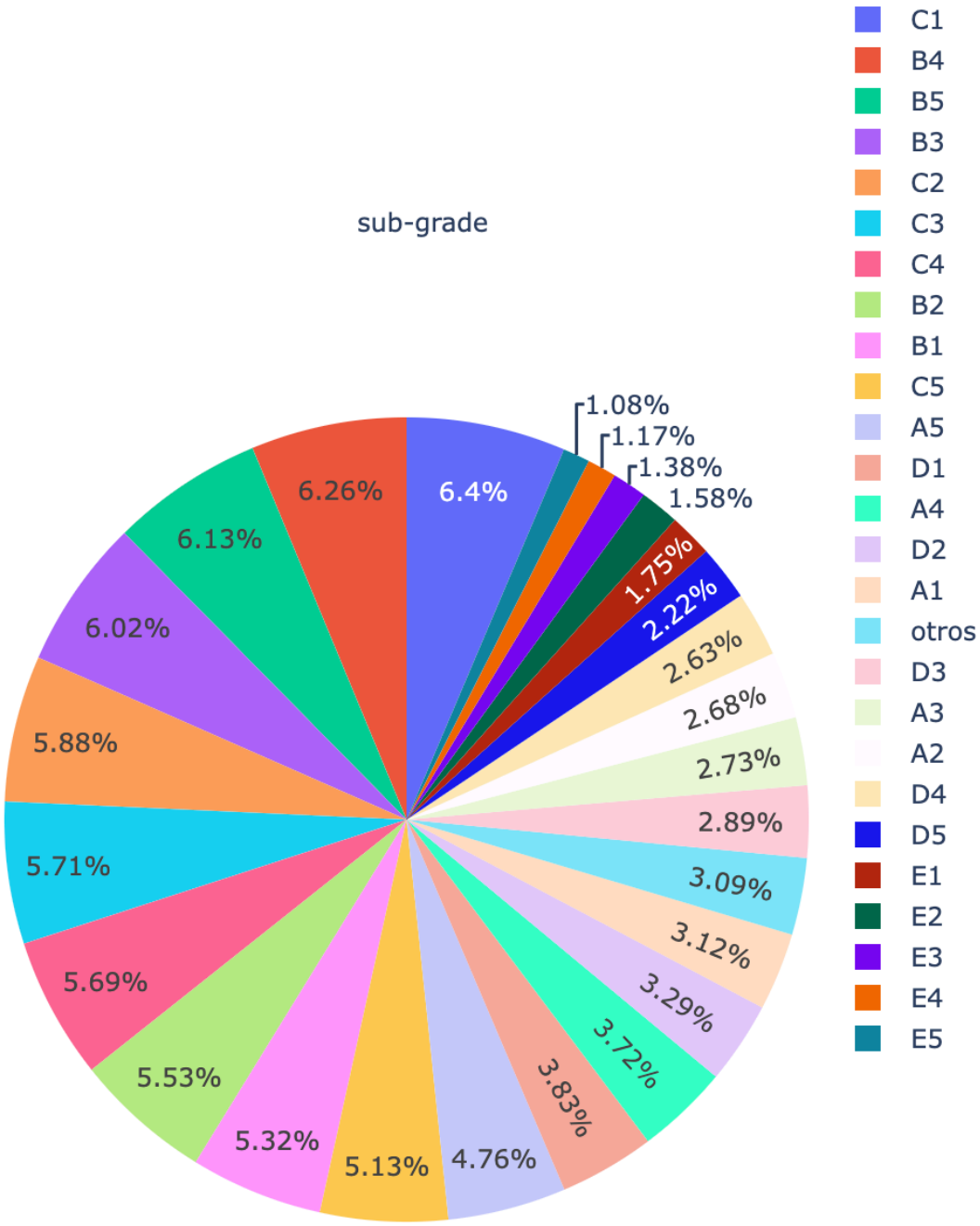

Ilustración 12: Distribución de las clases de la variable "sub_grade" tras la corrección

Fuente: Elaboración propia

La siguiente variable que analizaremos es *emp_length*, la cual indica la duración en años del empleo del aplicante. Sus clases se encuentran balanceadas y no necesitarán ninguna modificación.

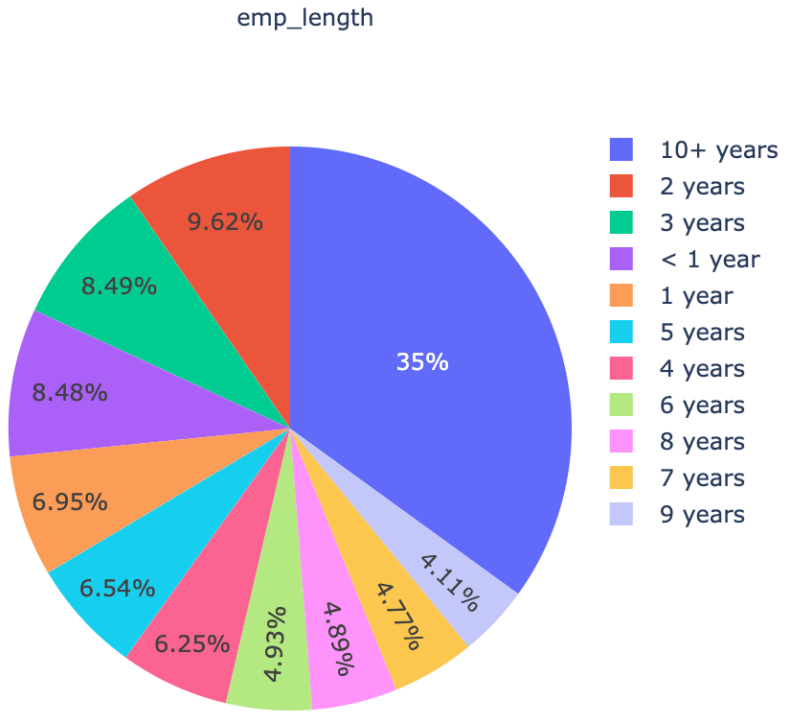

Ilustración 13: Distribución de las clases de la variable "emp_length"

Fuente: Elaboración propia

La variable independiente *home_ownership* representa la situación del deudor en relación con la propiedad de su inmueble. Observamos tres clases suficientemente pobladas, mientras que las clases *any, none* y *other* representan un porcentaje bajo de las observaciones. Estas tres clases se agruparán en una nueva clase bajo el nombre de *otros*.

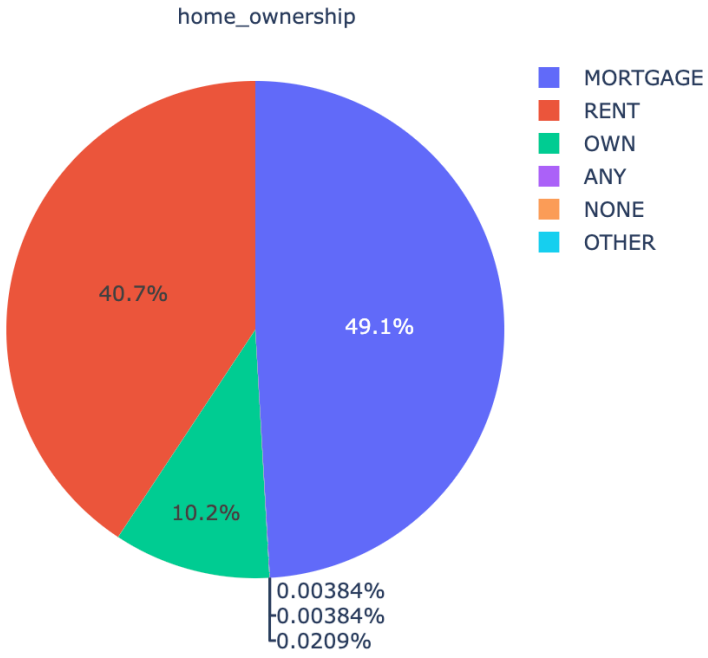

Ilustración 14: Distribución de las clases de la variable "home_ownership"

Fuente: Elaboración propia

La nueva distribución de las clases de *home_ownership* aparece a continuación:

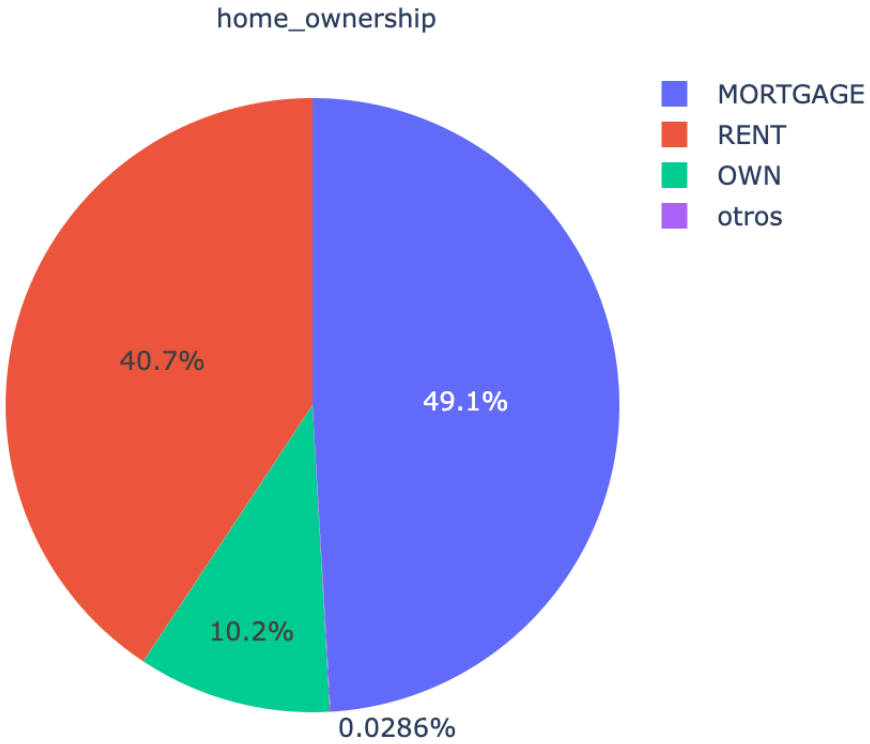

Ilustración 15: Distribución de las clases de la variable "home_ownership" tras la corrección

Fuente: Elaboración propia

El análisis de la variable *verification_status* muestra tres clases balanceadas. Sin embargo, dos de ellas representan el mismo concepto. Como consecuencia, las clases *Source Verified* y *Verified* se agruparán en la clase Verified.

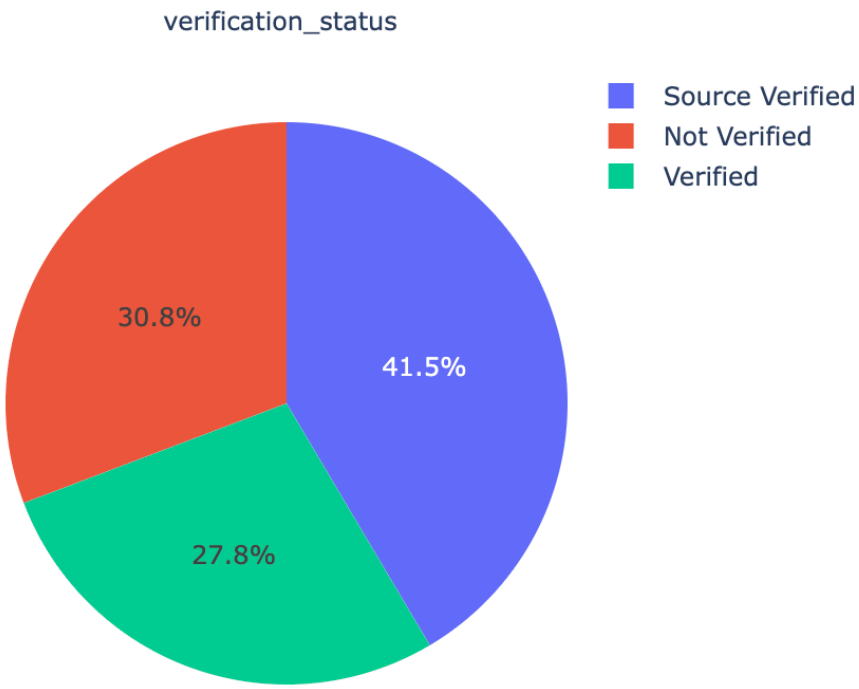

Ilustración 16: Distribución de las clases de la variable "verification_status"

Fuente: Elaboración propia

La nueva distribución se muestra a continuación.

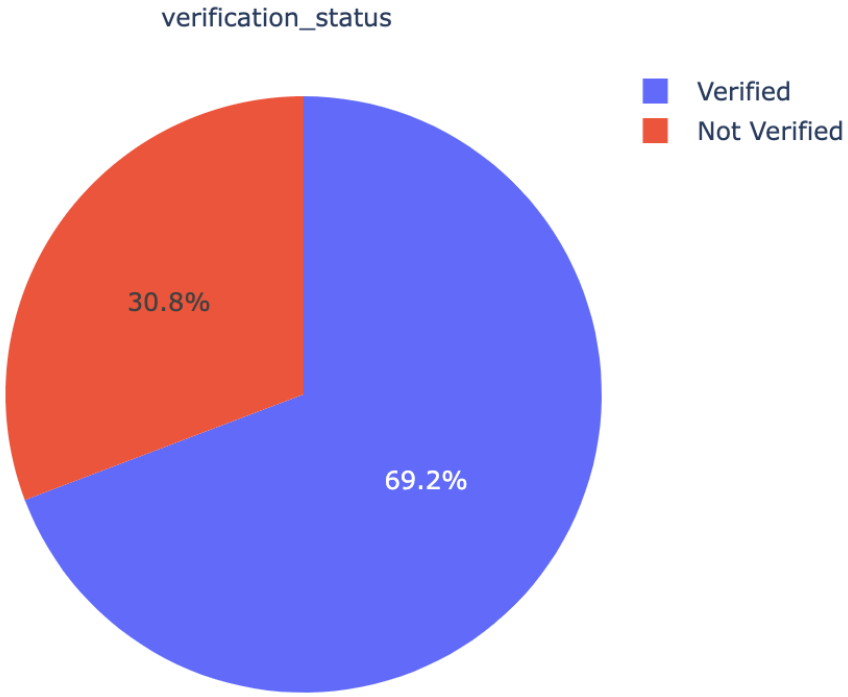

Ilustración 17: Distribución de las clases de la variable "verification_status" tras la corrección

Fuente: Elaboración propia

La finalidad del préstamo viene indicada por la variable *purpose*, en la cual podemos observar que una serie de clases representan un porcentaje pequeño de los casos. Por este motivo, las seis clases con menor volumen se agruparán bajo la nueva clase *otros*, siendo éstas *moving, vacation, house, wedding, renewable_energy* y *educational.*

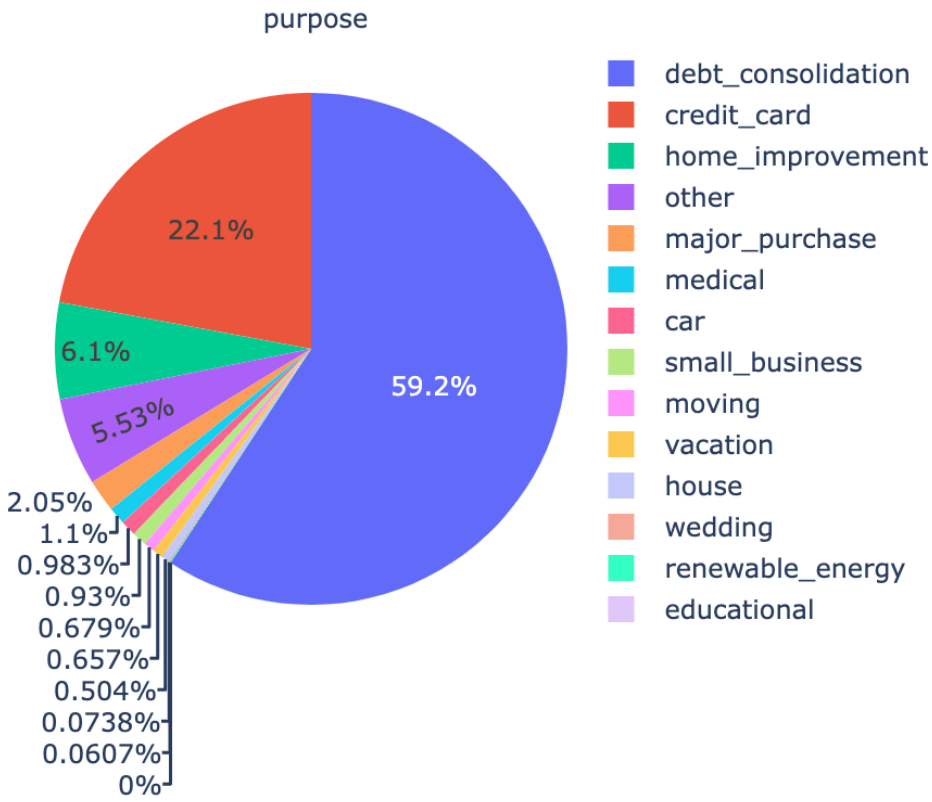

Ilustración 18: Distribución de las clases de la variable "purpose"

Fuente: Elaboración propia

Tras agrupar las clases con menos observaciones, la nueva distribución es la mostrada a continuación:

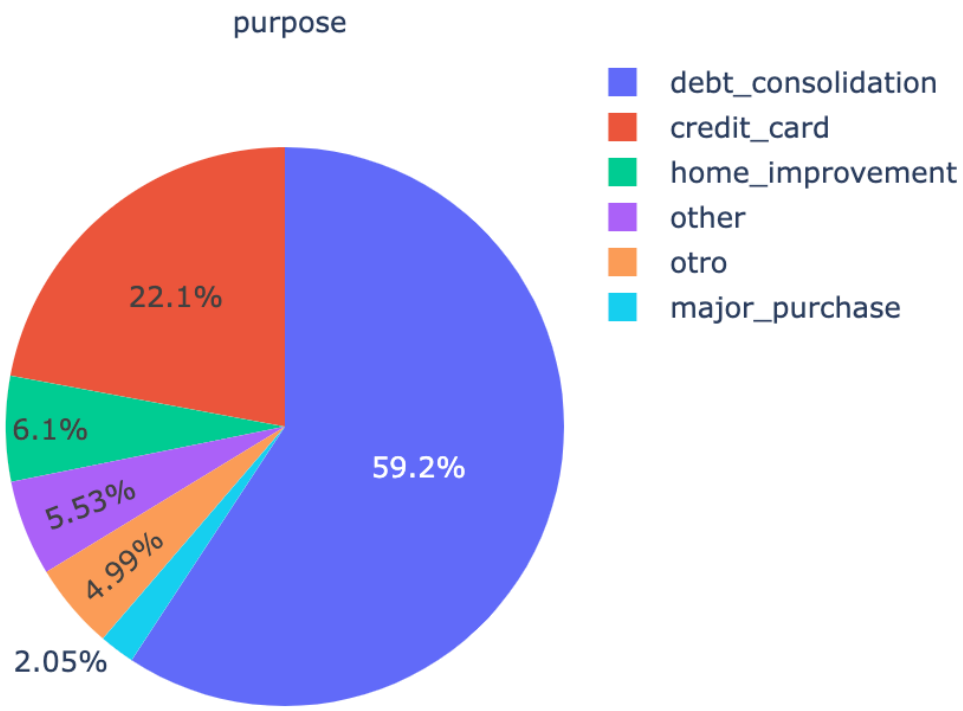

Ilustración 19: Distribución de las clases de la variable "purpose" tras la corrección

Fuente: Elaboración propia

El estado de Estados Unidos del que procede el deudor se representa mediante la variable *addr_state*. Esta variable incluye 51 clases, de las cuales algunas presentan una baja población. Todas las clases que representen menos del 1% de los datos serán agrupadas bajo la nueva clase *otro*.

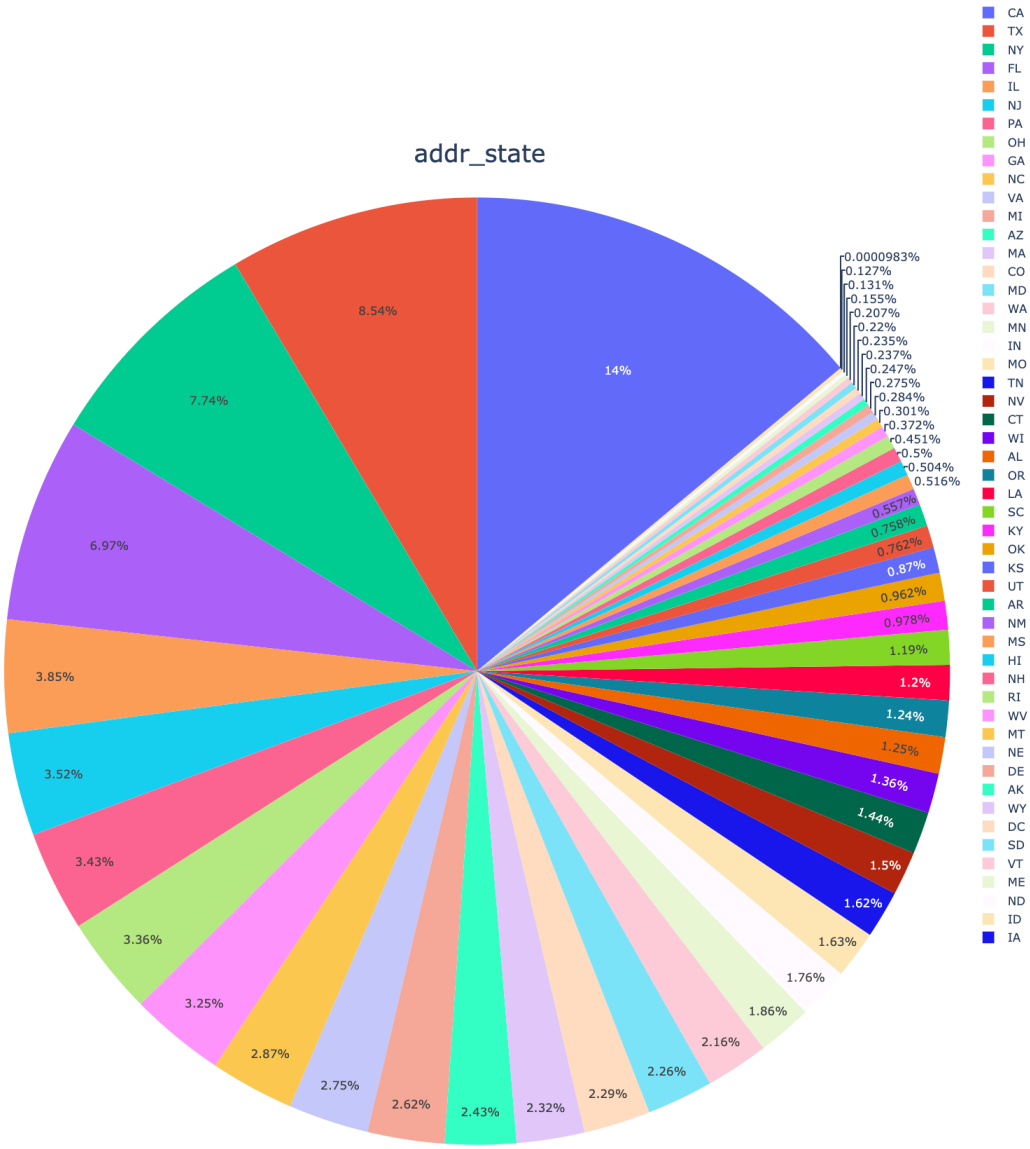

Ilustración 20: Distribución de las clases de la variable "addr_state"

Fuente: Elaboración propia

La distribución actualizada de los estados de origen de los deudores es la siguiente:

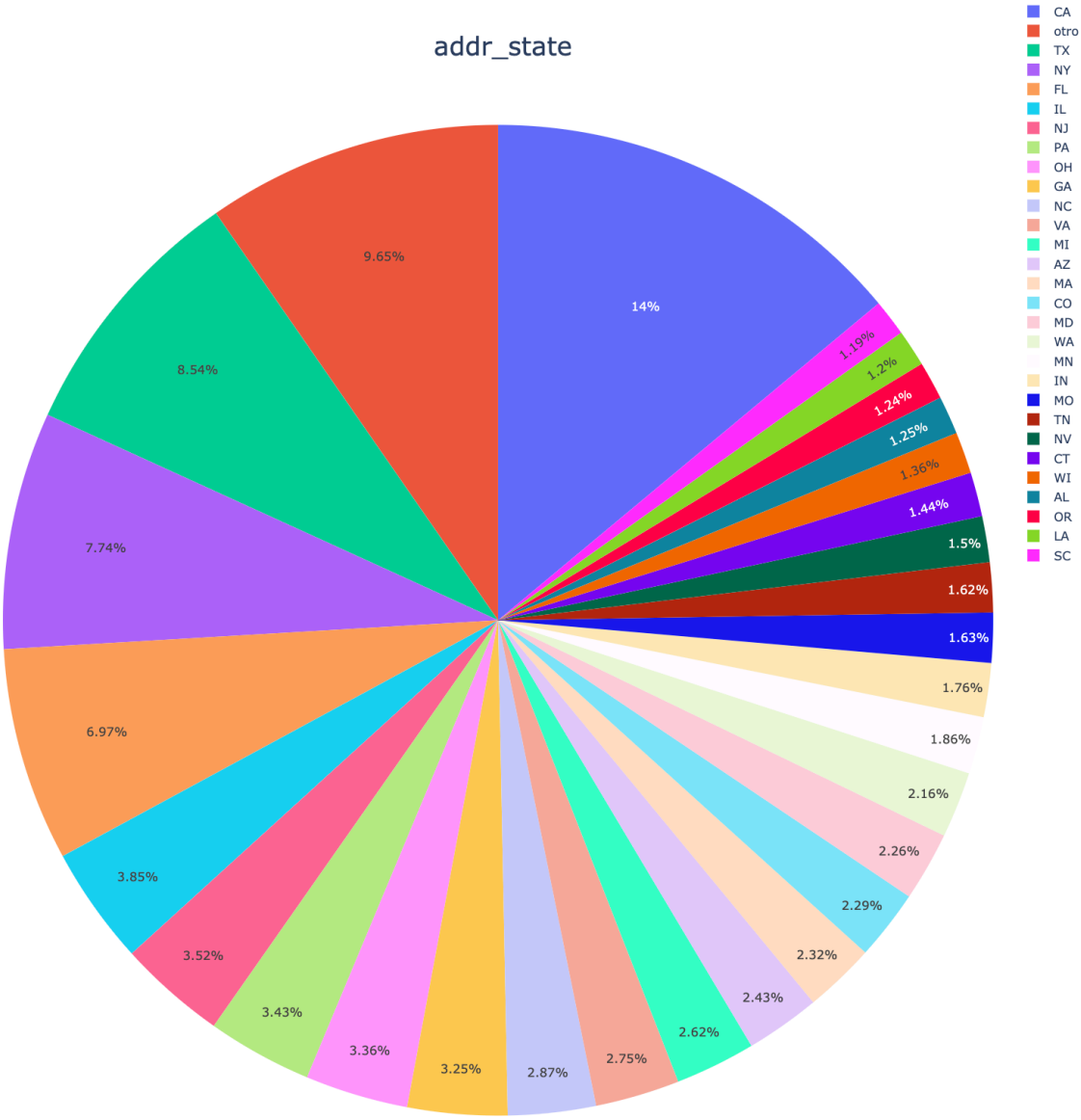

Ilustración 21: Distribución de las clases de la variable "addr_state" tras la corrección

Fuente: Elaboración propia

La variable *initial_list_status* representa los dos posibles estados de listado inicial, W y F. Desconocemos el significado de dichas clases, aun así, se conservará la clase y no se aplicarán cambios en ella.

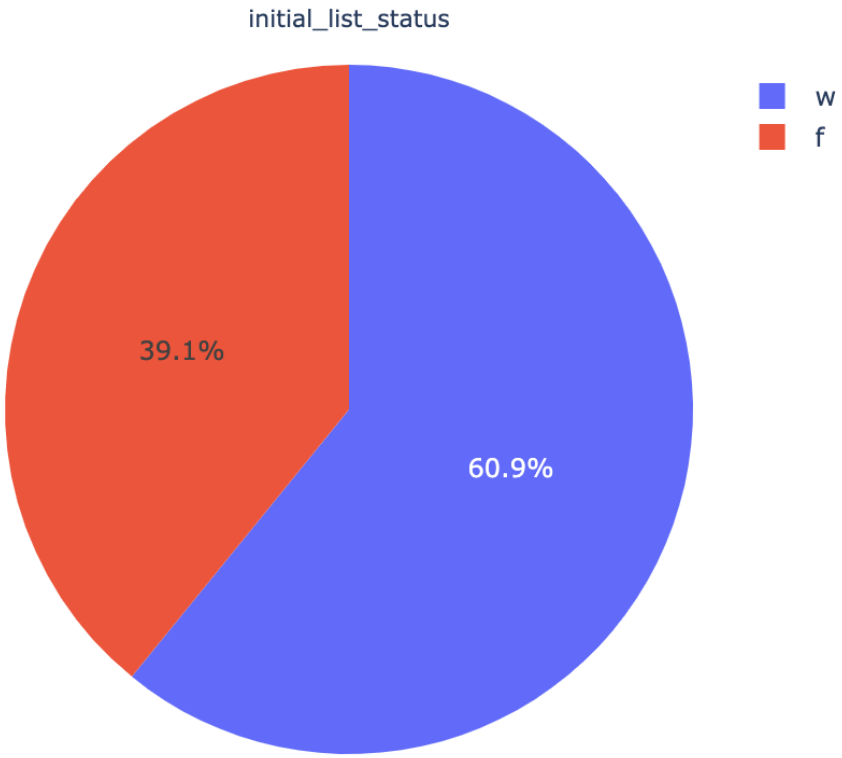

Ilustración 22: Distribución de las clases de la variable "initial_list_status"

Fuente: Elaboración propia

Se ha concluido el análisis exploratorio, pudiendo proceder con el desarrollo de los modelos de predicción de aquellos clientes que cometerán impago.

3.2 Modelos

En esta sección del proyecto se desarrollarán los cuatro modelos planteados para la predicción del impago de clientes. Previo desarrollo de los modelos se desarrollarán dos puntos. El primero será un breve apartado acerca de las métricas que serán empleadas para la evaluación de los modelos, con el objetivo de ser capaces de entender los valores que se obtengan. Después, se realizará la adaptación de las variables categóricas, convirtiéndolas en numéricas, para su uso en los modelos que así lo requieran. Una vez estos dos puntos se hayan completado, se procederá primero con la regresión logística, seguido de la red neuronal, el árbol de decisión y, finalmente, el algoritmo XGBoost.

3.2.1 Métricas de evaluación

Los resultados obtenidos por los modelos necesitan ser interpretados para determinar la eficiencia de los mismos. Esta interpretación se realiza mediante el uso de métricas de evaluación, las cuales cuantifican el error que han cometido los modelos. Existe un gran número de métricas de evaluación, cuyo uso adecuado depende del tipo de problema. Para el caso de estudio que se plantea en este trabajo, el cual es un problema de clasificación, requeriremos del uso de métricas de clasificación. Dentro de los problemas de clasificación, es necesario determinar si la variable a modelar es dicotómica o, por el contrario, si es multiclase. Como se ha mencionado en repetidas ocasiones a lo largo del proyecto, nuestra variable de interés toma únicamente dos valores, el 1 y el 0. Entonces, teniendo en cuenta que se trata de un problema de clasificación binaria, emplearemos las métricas derivadas de la matriz de confusión para la evaluación de los modelos.

Una matriz de confusión es una forma de representar las cuatro posibles situaciones de clasificación (Visa et al., 2011):

- Positivo verdadero (TP): representa aquellas observaciones cuya clase real y la clase predicha coinciden y, además, dicha clase es la clase positiva, es decir, 1.
- Falso positivo (FP): representa las observaciones cuya clase real es negativa (0) pero el modelo ha predicho que la observación era positiva (1).
- Negativo verdadero (TN): representa aquellas observaciones cuya clase real y la clase predicha coinciden y, además, dicha clase es la clase negativa, es decir, 0.
- Falso negativo (FN): representa las observaciones cuya clase real es positiva (1) pero el modelo ha predicho que la observación era negativa (0).

Clase		Clase real	
		Positivo	Negativo
Clase predicha	Positivo	TP	FP
	Negativo	FN	TN

Tabla 3: Posibles clasificaciones de una clase binaria

Fuente: Elaboración propia

De estas cuatro posibles clasificaciones obtenemos las siguientes métricas de evaluación para problemas de clasificación binaria (Vujović, 2021):

- Exactitud: representa el porcentaje de acierto general y se calcula como el total de aciertos dividido por el total de casos.

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

Ecuación 14: Exactitud

- Exhaustividad: es el total de casos reales positivos predichos correctamente (TP) entre todos los casos positivos reales (TP y FN).

- Precisión: es el total de casos positivos predichos correctamente (TP) entre todos los casos positivos predichos (TP y FP).

- F-Score: es una métrica que combina precisión y exhaustividad. Alcanza su valor máximo cuando ambas son 1.

$$F - Score = \frac{2 \times Exhaustividad \times Precision}{xhaustividad + Precision}$$

Ecuación 15: F-Score

- Especificidad: es el total de casos reales negativos predichos correctamente (TN) entre todos los casos negativos reales (TN y FP).

- Ratio de falsos positivos: es el total de falsos positivos (FP) dividido entre todos los casos negativos reales (TN y FP).

- Área ROC: combina la métrica de exhaustividad con el ratio de falsos positivos. Se representa mediante una curva, donde el área encerrada debajo de ella es el valor de la métrica. La curva se representa en un sistema de referencia cuyo eje x es el ratio de falsos positivos, mientras que el eje y es la exhaustividad. El objetivo es que la curva alcance los mayores valores posibles en el eje y para los menores valores posibles del eje x. En el caso ideal la exhaustividad sería 1 y el ratio de falsos positivos 0, resultando en un área debajo de la curva o en un área ROC de 1.

El uso de estas métricas dependerá del caso de estudio, concretamente del error que se quiera penalizar más. En nuestro caso primaremos la predicción correcta de los casos positivos sobre la de los casos negativos, por lo tanto, el error que debemos evitar son los falsos negativos, es decir, queremos evitar predecir que un cliente no va a cometer un impago cuando en realidad sí. Por ello, la métrica en la que nos debemos fijar principalmente será la exhaustividad, puesto que tiene premia las observaciones positivas clasificadas correctamente y penaliza las observaciones positivas clasificadas de forma errónea. Debido a ello, las métricas F-score y Área ROC también serán óptimas al incluir la exhaustividad en su cálculo. El resto de las métricas se seguirán computando para evaluar los modelos con el objetivo de obtener información adicional, pero siempre primando aquellas métricas relacionadas con la exhaustividad.

3.2.2 Transformación de las variables categóricas

Modelos como las regresiones logísticas o las redes neuronales requieren que todos los datos que emplean en su entrenamiento sean numéricos. En el conjunto de datos seleccionado hay variables categóricas, por ello, es necesario convertir estas variables de categóricas a numéricas.

Para ello, emplearemos variables dummy. Su funcionamiento se basa en dividir la variable categórica en tantas variables numéricas como clases haya e indicar con un 1 en la columna de la clase correspondiente si la observación pertenecía a dicha clase. Por ejemplo, la variable *term* estaba compuesta de dos clases: *36 months* y *60 months*. Por lo tanto, se crean dos columnas nuevas, *term_36 months* y *term_60 months*, las cuales indicarán con un 1 si cada observación pertenecía a su clase y con un 0 en caso contrario.

Después de incluir las variables dummies de todas las variables categóricas, el conjunto de datos ha pasado de 76 variables a 156.

3.2.3 Modelo de regresión logística

El primer modelo desarrollado es un modelo de regresión logística o logit. Partiendo de los datos previamente depurados y cuyas variables son numéricas en su totalidad, el primer paso es separar la variable dependiente o variable a predecir del resto de variables.

El motivo es que al modelo se le introducirá el conjunto de variables predictivas, al que denominaremos X, y las predicciones obtenidas se compararán con la vector *y* que contiene la variable dependiente, calculando el error para así llevar a cabo el entrenamiento.

Tras separar las variables independientes de la dependiente es necesario normalizar los datos del conjunto X, puesto que las variables tienen diferentes escalas, pudiendo generar dos problemas: malas predicciones y que el modelo no converja, es decir, que el modelo no finalice el entrenamiento.

Una vez listos los datos, se precede a la creación de los conjuntos de entrenamiento y de test. Para ello se realiza una división en la cual el 70% de los datos irán al conjunto de entrenamiento y el 30% al de test.

Finalmente, entrenamos el modelo con el conjunto de entrenamiento y, una vez entrenado validamos su rendimiento con el conjunto de test, el cual está compuesto por datos que el modelo desconoce, simulando así una situación de aplicación real.

Los resultados obtenidos mediante las métricas de evaluación expuestas previamente aparecen en la siguiente tabla. También se mostrará la matriz de confusión y la curva ROC.

Métricas				
Exhaustividad	F-Score	Área ROC	Precisión	Exactitud
99,47%	99,72%	99,73%	99,96%	99,89%

Tabla 4: Resultados del modelo de regresión lineal

Fuente: Elaboración propia

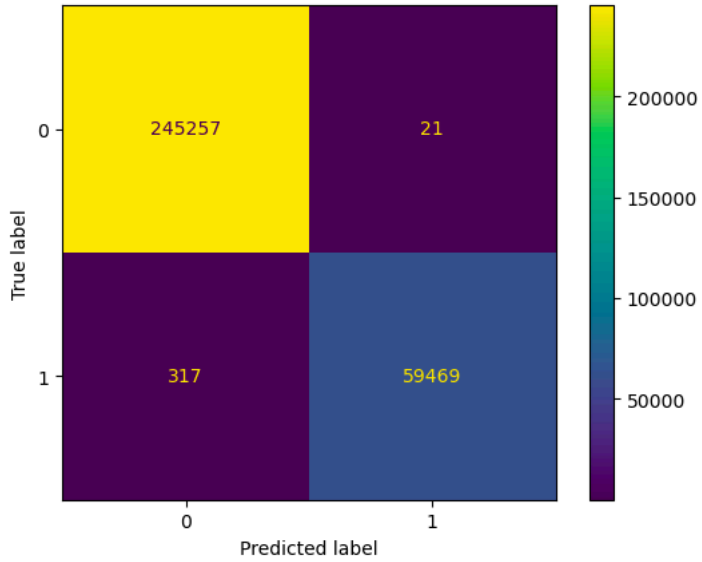

Ilustración 23: Matriz de confusión del modelo de regresión logística

Fuente: Elaboración propia

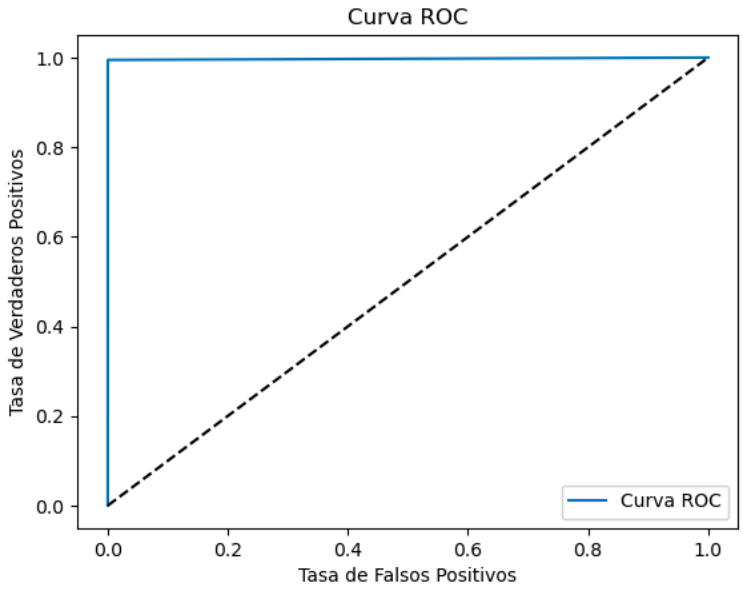

Ilustración 24: Curva ROC del modelo de regresión logística

Fuente: Elaboración propia

Los resultados del modelo son excelentes, obteniendo predicciones casi perfectas. El error que se buscaba evitar, el falso negativo, el significativamente pequeño, fallando únicamente en la predicción de 317 clientes de los 59.786 que cometieron impago.

En cuanto a la predicción de la probabilidad de default media los resultados también son excelentes. La PD media real es de 19,60%, mientras que la PD predicha es de 19.49%, cometiendo únicamente un error de 0,11%.

3.2.4 Modelo de red neuronal

El modelo de red neuronal requería de una preparación de los datos idéntica a la de la regresión logística. Se separó el conjunto de datos en la matriz de variables independientes X y en el vector que contiene a la variable dependiente *y*. Posteriormente se normalizaron los datos y se realizó la separación entre el conjunto de entrenamiento y test con una proporción de 70% y 30%, respectivamente.

La arquitectura de la red se definió teniendo en cuenta la gran cantidad de datos que iban a ser empleados en el entrenamiento. Se decidió crear una red sencilla para evitar problemas de overfitting y altos tiempos de entrenamiento. Se eligieron tres capas, una de entrada, una oculta y la capa de salida. La primera capa recibía 155 entradas, correspondiéndose con las 155 variables de la matriz X, y contaba con 32 neuronas cuya función de activación era ReLU, debido a la capacidad de reducir el overfitting. La capa oculta estaba compuesta por 16 neuronas, también con ReLU como función de activación. Finalmente, la capa de salida se componía de una única neurona, puesto que la variable a predecir es binaria, y su función de activación era una sigmoide, con el objetivo de que los resultados se encontraran acotados entre 0 y 1.

El entrenamiento de la red se realizó con el conjunto de entrenamiento y su posterior evaluación con el conjunto de test. Las predicciones obtenidas de la red eran valores acotados entre 0 y 1, por lo que para obtener las clases de estableció un umbral de 0.5. Si la predicción se encontraba por debajo del umbral, la clase asignada será la clase negativa o 0, mientras que, si se encontraba por encima o en el umbral, su clase sería la positiva.

Los resultados, empleando las métricas expuestas previamente, son los que aparecen la siguiente tabla. Además, se muestra la matriz de confusión y la curva ROC.

Métricas

Exhaustividad	F-Score	Área ROC	Precisión	Exactitud
99,30%	99,63%	99,64%	99,97%	99,86%

Tabla 5: Resultados del modelo de red neuronal

Fuente: Elaboración propia

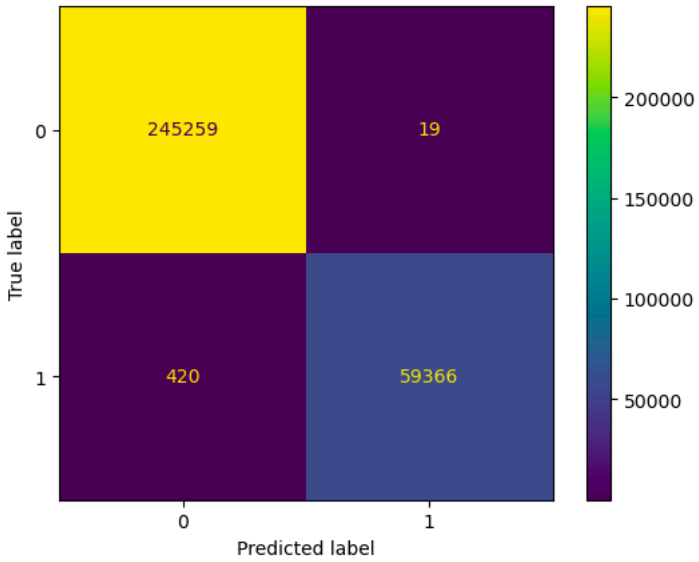

Ilustración 25: Matriz de confusión del modelo de red neuronal

Fuente: Elaboración propia

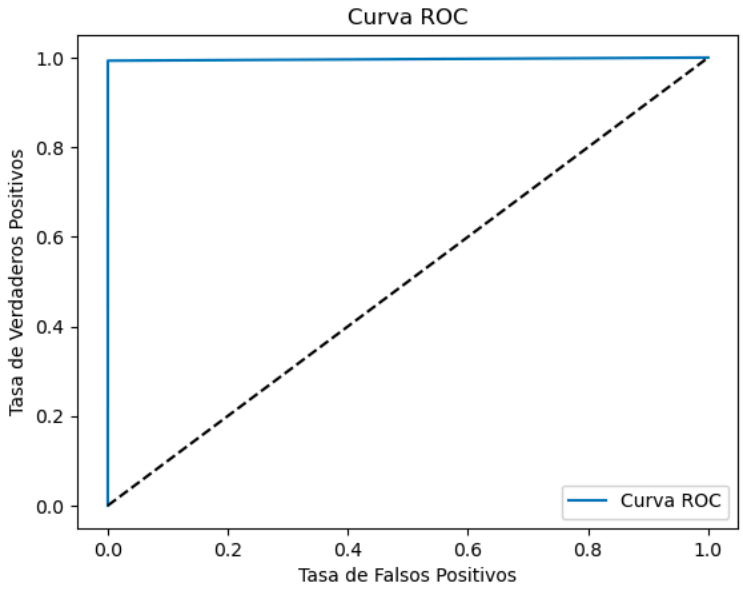

Ilustración 26: Curva ROC del modelo de red neuronal

Fuente: Elaboración propia

Los resultados del modelo son excelentes, siendo capaz de realizar predicciones sin apenas cometer falsos negativos. Además, la PD media estimada del conjunto de test es de 19,46% y la PD real de 19,60%, cometiendo un error de 0,14%.

3.2.5 Modelo de árbol de decisión

Teóricamente, los árboles de decisión son compatibles con variables categóricas, sin embargo, los modelos definidos por las librerías no aceptan esta clase de variables. Como consecuencia, para el modelo de árboles de decisión, se ha utilizado de nuevo el conjunto de datos numérico.

A pesar de ello, los árboles de decisión no requieren de la normalización de las variables, debido a que su entrenamiento no depende de la distancia sino de la homogeneidad de sus nodos.

Tras realizar la división de los datos en los conjuntos de entrenamiento y test, con un ratio de 70% para entrenamiento y 30% para test, se define y entrena el modelo, empleando Gini como medida de homogeneidad.

Los resultados de las métricas de evaluación aparecen a continuación, junto a la matriz de confusión y a la curva ROC.

Métricas				
Exhaustividad	F-Score	Área ROC	Precisión	Exactitud
99,18%	99,14%	99,48%	99,10%	99,66%

Tabla 6: Resultados del modelo de árbol de decisión

Fuente: Elaboración propia

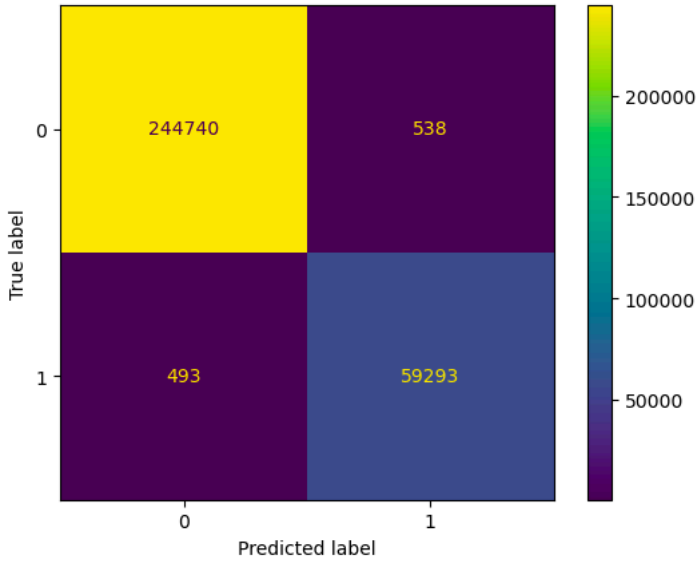

Ilustración 27: Matriz de confusión del modelo de árbol de decisión

Fuente: Elaboración propia

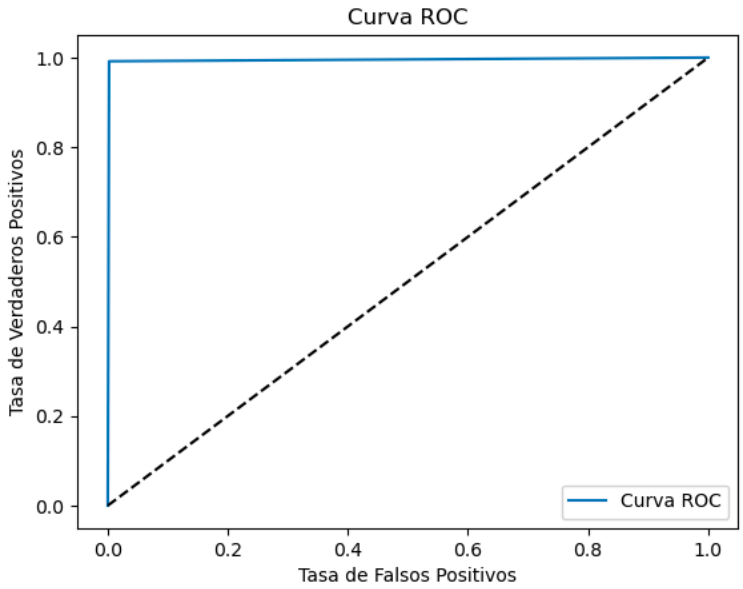

Ilustración 28: Curva ROC del modelo de árbol de decisión

Fuente: Elaboración propia

De nuevo, los resultados son excelentes. Los casos de falsos negativos son escasos y la diferencia entre la probabilidad de default media real, cuyo valor es de 19,60%, y la probabilidad de default media estimada, de valor 19,43%, es de 0,17%

3.2.6 Modelo basado en XGBoost

Finalmente, se empleará la librería XGBoost con el objetivo de mejorar los resultados obtenidos por el modelo de árbol de decisión previo. Aunque se esté actualizando la librería para que el modelo sea capaz de trabajar con variables categóricas, de momento esta función no se encuentra disponible. Como resultado se usará el conjunto de datos numérico, del cual obtenemos un conjunto de entrenamiento con un 70% de los datos y un conjunto de test con el 30%. Cabe mencionar que, al igual que en el modelo de árbol de decisión, no ha sido necesario normalizar las variables.

Después del entrenamiento del modelo con el conjunto de entrenamiento y de su posterior validación con el conjunto de test, los resultados obtenidos son los siguientes.

Métricas				
Exhaustividad	F-Score	Área ROC	Precisión	Exactitud
99,82%	99,91%	99,91%	100%	99,96%

Tabla 7: Resultados del modelo basado en XGBoost

Fuente: Elaboración propia

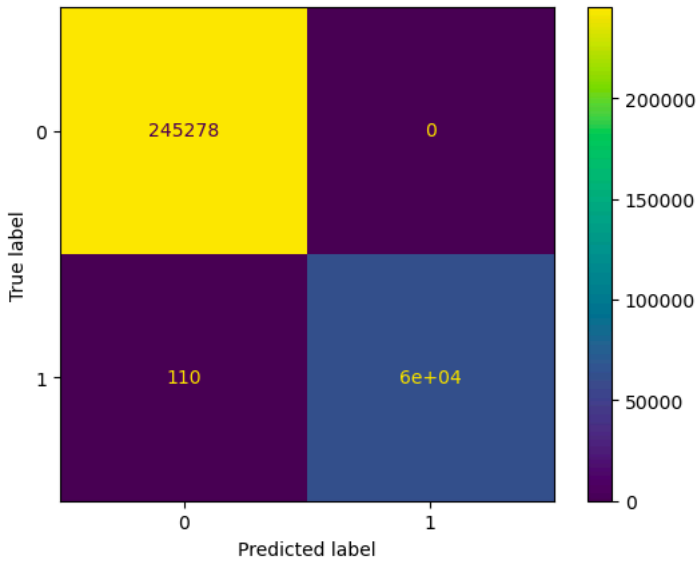

Ilustración 29: Matriz de confusión del modelo basado en XGBoost

Fuente: Elaboración propia

El valor exacto del número de verdaderos positivos, es decir, de aquellos cuya clase real y clase predicha es 1, es de 59.676.

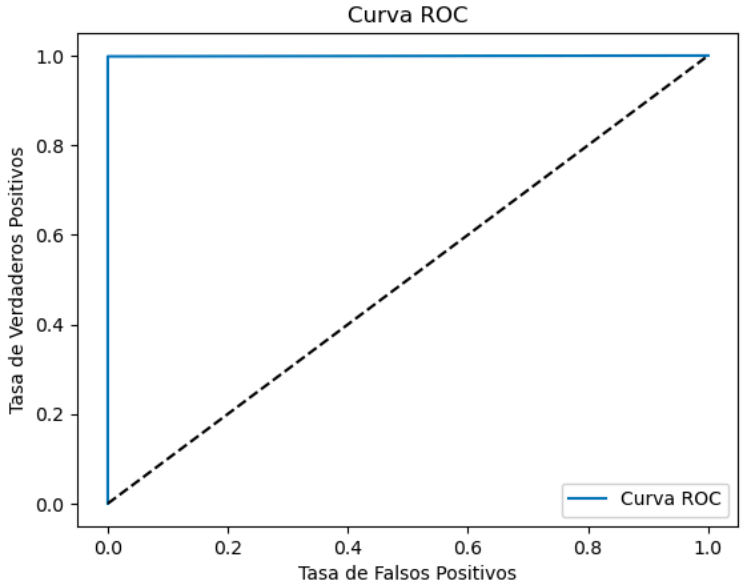

Ilustración 30: Curva ROC del modelo basado en XGBoost

Fuente: Elaboración propia

El modelo ha conseguido obtener unas predicciones casi perfectas, con apenas 110 casos positivos mal clasificados. Además, la PD predicha, de valor 19,56%, es casi igual a la PD real, de 19,60%.

4 Resultados

En este capítulo realizaremos una comparación de los resultados obtenidos para cada modelo a través de sus métricas y seleccionaremos un modelo definitivo.

	Métricas					
Modelo	Exhaustividad	F-Score	Área ROC	Precisión	Exactitud	Desviación en la PD
Regresión logística	99,47%	99,72%	99,73%	99,96%	99,89%	0,11%
Red neuronal	99,30%	99,63%	99,64%	99,97%	99,86%	0,14%
Árbol de decisión	99,18%	99,14%	99,48%	99,10%	99,66%	0,17%
XGBoost	99,82%	99,91%	99,91%	100%	99,96%	0,04%

Tabla 8: Comparación de los resultados de los modelos finales

Fuente: Elaboración propia

Como se puede observar, los resultados de los modelos son excepcionales, pudiendo ser cualquiera de ellos utilizado con excelentes resultados para la predicción de clientes que cometerán impago. A pesar de ello, la regresión logística es el tipo de modelo que menor capacidad computacional ha requerido, por lo que, si el tiempo o la capacidad de computación son un factor limitante para la entidad, recomendaría elegir este modelo. Si por el contrario se dispone de sistemas con alta capacidad de computación, el modelo a elegir sería el clasificador basado en XGBoost, debido a sus resultados prácticamente perfectos. También se podría intentar desarrollar una arquitectura más compleja para la red neuronal, sin embargo, la posible mejora de los resultados no compensaría el incremento en el coste de computación.

5 Conclusión

Los resultados del proyecto han sido satisfactorios. Se han conseguido alcanzar los diferentes objetivos planteados, desde aquellos teóricos hasta los más prácticos.

En el marco teórico hemos sido capaces de comprender el concepto de riesgo de crédito, estudiando que factores influyen en su constitución y de qué manera afecta a las entidades bancarias. Posteriormente, se realizó un análisis de la normativa que rige nuestro sistema bancario, adquiriendo un entendimiento más profundo de este ámbito tan crítico en las operaciones diarias de los bancos mundiales. Se consiguió entender las normativas que se aplican y así poder definir como se realizarían los modelos en los siguientes capítulos del proyecto.

Después del estudio teórico se seleccionó un conjunto de datos reales de la entidad crediticia Lending Club para el entrenamiento de los modelos. Al ser datos reales nos vimos obligados a realizar un tratamiento de los datos más exhaustivo del que se esperaba al empezar el trabajo. Sin embargo, gracias a ello, nos hemos podido acercar lo máximo posible a una situación de modelaje real. Finalmente, se procedió al desarrollo y evaluación de los modelos, cuyos resultados superaron la mejor de nuestras expectativas.

A pesar de haber obtenido resultados satisfactorios en los modelos, se considera que todavía se pueden realizar mejoras. Especialmente en el desarrollo de la arquitectura de los modelos mediante el uso de técnicas que permitan la optimización de los diferentes hiperparámetros.

6 Bibliografía

African Development Bank. (2007). *Market Risk Review.*

Amro, A., Al-Akhras, M., Hindi, K. E., Habib, M., & Shawar, B. A. (2021). Instance reduction for avoiding overfitting in decision trees. *Journal of Intelligent Systems*, 30(1), 438-459.

Banco Santander. (29 de Junio de 2022). *Riesgos financieros: qué son, tipos y consejos para enfrentarse a ellos*. Obtenido de Banco Santander: https://www.becas-santander.com/es/blog/riesgos-financieros.html

Banco Santander. (14 de Marzo de 2023). *Riesgo de crédito o crediticio: qué son y cuáles hay*. Obtenido de Banco Santander: https://www.santanderconsumer.es/blog/post/riesgo-de-credito-o-crediticio-que-son-y-cuales-hay

Basel Committee. (2023). *The Basel Framework.*

Basel Committee on Banking Supervision. (2001). The Internal Ratings-Based Approach.

Brown, K., & Moles, P. (2014). Credit risk management. Edinburgh Business School. Heriot-Watt University.

Chatterjee, S. (2015). *Modelling credit risk*. Handbooks, Centre for Central Banking Studies. Bank of England, number 34. December

Committee of Sponsoring Organizations of the Treadway Commission (COSO). (2020). Compliance risk management: Applying the COSO ERM framework.

Dubey, S. R., Singh, S. K., & Chaudhuri, B. B. (2022). Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing*, *503*(C), 92–108.

Esposito, F., Malerba, D., Semeraro, G., & Kay, J. (1997). A comparative analysis of methods for pruning decision trees. *IEEE transactions on pattern analysis and machine intelligence, 19*(5), 476-491.

European Central Bank. (2004). Financial stability review. European Central Bank.

Fafalios, S., Charonyktakis, P., & Tsamardinos, I. (2020). Gradient Boosting Trees. Gnosis Data Analysis PC, 1-3.

Federal Deposit Insurance Corporation. (2006). *Operational Risk Management: An Evolving Discipline.*

Foong, N. S., Ming, C. Y., Eng, C. P., & Shien, N. K. (2018) An Insight of Linear Regression Analysis. *Scientific Research Journal* 15(2):1

Forbes. (s.f.). *Lending Club*. Obtenido de Forbes: https://www.forbes.com/companies/lending-club/

Ganaie, M. A., Hu, M., Malik, A. K., Tanveer, M., & Suganthan, P. N. (2022). Ensemble deep learning: A review. *Engineering Applications of Artificial Intelligence, 115*, 105151.

Grippa, P., & Gornicka, L. (2016). Measuring concentration risk-A partial portfolio approach. IMF Working Papers 2016/158.International Monetary Fund.

Hagenlocher, P. (2017). Decision Tree Learning. Faculty of Informatics, TUM (Technische Universitat München).

Hilbe, J. M. (2016). *Practical guide to logistic regression*. CRC Press.

Iranzo Gutiérrez, S. (2008). Introducción al riesgo-país. Documentos ocasionales 0802, Banco de España.

Loh, W. Y. (2011). Classification and regression trees. WIRES (*Wiley interdisciplinary reviews): data mining and knowledge discovery*, *1*(1), 14-23.

Matich, D. J. (2001). Redes Neuronales: Conceptos básicos y aplicaciones. Universidad Tecnológica Nacional, Facultad Regional Rosario, Departamento de Ingeniería Química, Argentina.

Mehlig, B. (2021). *Machine learning with neural networks: an introduction for scientists and engineers*. Cambridge University Press.

Montgomery, D. C., Peck, E. A., & Vining, G. G. (2021). *Introduction to linear regression analysis*, 6th edition. John Wiley & Sons.

Moody's Analytics. (2011). *Regulation Guide: An Introduction.*

Morera Munt, A., & Alcalá Nalvaiz, J. T. (2018). Introducción a los modelos de redes neuronales artificiales El Perceptrón simple y multicapa (Doctoral dissertation, Tesis de pregrado, Universidad Saragoza]. Saguan Repositorio Institucional de Documentos https://cutt. ly/lEVVf7v).

Park, H. A. (2013). An introduction to logistic regression: from basic concepts to interpretation with particular attention to nursing domain. *Journal of Korean Academy of Nursing, 43*(2), 154-164.

Pascanu, R., Mikolov, T., & Bengio, Y. (2013, May). On the difficulty of training recurrent neural networks. Proceedings of the 30th International Conference on Machine Learning, PMLR 28(3):1310-1318.

Platon, V., Frone, S., & Constantinescu, A. (2014). Financial and economic risks to public projects. *Procedia Economics and Finance, 8*, 204-210.

Ramskogler, P. (2015). Tracing the origins of the financial crisis. *OECD Journal Financial Market Trends, 2014*(2), 47-61.

Santos, K. (2008). Corporate credit ratings: a quick guide. Treasurer's Companion, 45-49.

Shalizi, C. (2013). *Advanced data analysis from an elementary point of view*. Cambridge University Press, (pp. 231-236).

Sharma, S., Sharma, S., & Athaiya, A. (2017). Activation functions in neural networks. *Towards Data Sci, 6(*12), 310-316.

Sottilotta, C. E. (2012). Political risk: Concepts, definitions, challenges. Conference: LUISS School of Government Annual Conference "Investing in the Age of Political Risks".

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research, 15*(1), 1929-1958.

Sundhari, S. S. (2011, June). A knowledge discovery using decision tree by Gini coefficient. In 2011 International Conference on Business, Engineering and Industrial Applications (pp. 232-235). IEEE.

Uslu, Ç. (s.f.). *What is Kaggle?* Obtenido de Datacamp: https://www.datacamp.com/blog/what-is-kaggle

Vajapeyam, S. (2014). Understanding Shannon's entropy metric for information. arXiv preprint arXiv:1405.2061.

Van Roy, P. (2005). Credit ratings and the standardised approach to credit risk in Basel II. ECB (European Central Bank) Working Paper nº 517.

Visa, S., Ramsay, B., Ralescu, A. L., & Van Der Knaap, E. (2011). Confusion matrix-based feature selection. *Maics, 710*(1), 120-127.

Vujović, Ž. (2021). Classification model evaluation metrics. *International Journal of Advanced Computer Science and Applications, 12*(6), 599-606.

Wang, Y., Song, C., & Xia, S. T. (2016). Unifying the Split Criteria of Decision Trees Using Tsallis Entropy. arXiv preprint arXiv:1511.08136.

XGBoost Developers. (2022). *XGBoost Documentation*. Obtenido de XGBoost: https://xgboost.readthedocs.io/en/stable/

7 Anexos

7.1 Anexo 1: Código para las visualizaciones empleadas en el marco teórico

```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-10, 10, 100)
y = np.exp(x) / (1 + np.exp(x))

plt.plot(x, y)
plt.title('Función Logística')
plt.xlabel('x = β_0+β_1 x_1+β_2 x_2+...+β_k x_k')
plt.ylabel('p')
plt.grid(True)
plt.show()

x = np.linspace(-10, 10, 100)
y = np.heaviside(x, 0)

plt.plot(x, y)
plt.title('Función escalón')
plt.xlabel('x')
plt.ylabel('f(x)')
plt.grid(True)
plt.show()

x = np.linspace(-10, 10, 100)
y = np.exp(x) / (1 + np.exp(x))

plt.plot(x, y)
plt.title('Función sigmoide')
plt.xlabel('x')
plt.ylabel('f(x)')
plt.grid(True)
plt.show()

x = np.linspace(-10, 10, 100)
y = np.tanh(x)

plt.plot(x, y)
plt.title('Tangente hiperbólica')
plt.xlabel('x')
plt.ylabel('f(x)')
plt.grid(True)
plt.show()
```

7.2 Anexo 2: Código del desarrollo empírico

```
# Librerías
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots

from docx import Document

# Modelos

import pickle
from tensorflow.keras.models import save_model

#Regresión logística
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import precision_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.metrics import roc_curve, roc_auc_score

# Red neuronal
from tensorflow import keras
from tensorflow.keras.models import Sequential
```

```
from tensorflow.keras.layers import Dense

from sklearn.preprocessing import LabelEncoder

# Árbol de decisión
from sklearn.tree import DecisionTreeClassifier

# XGBoost
from xgboost import XGBClassifier

pd.set_option('display.max_columns', 200)
pd.set_option('display.max_rows', 200)

accepted = pd.read_csv("datos/kaggle/accepted_2007_to_2018Q4.csv")

# Retiramos el índice de los datos
accepted = accepted.reset_index(drop=True)

# Retiramos las últimas dos filas, puesto que no contienen información
accepted = accepted[:-2]
accepted

print("Información general del DataFrame:")
print(accepted.info())

print("\nPrimeras filas del DataFrame:")
print(accepted.head())

print("\nEstadísticas descriptivas:")
print(accepted.describe())

accepted["loan_status"].value_counts()

print(accepted["loan_status"].isna().sum())

# ### Tratamiento de la variable dependiente loan_status
# Retiramos las instacias de las clases de loan_status que no consideramos
como default
accepted_2 = accepted[accepted["loan_status"]!="Current"]
```

```
accepted_2 = accepted_2[accepted["loan_status"]!="Does not meet the credit
policy. Status:Fully Paid"]
accepted_2 = accepted_2[accepted["loan_status"]!="Late (31-120 days)"]
accepted_2 = accepted_2[accepted["loan_status"]!="In Grace Period"]
accepted_2 = accepted_2[accepted["loan_status"]!="Late (16-30 days)"]
accepted_2 = accepted_2[accepted["loan_status"]!="Does not meet the credit
policy. Status:Charged Off"]
accepted_2 = accepted_2[accepted["loan_status"]!="Default"]

accepted_2["loan_status"].value_counts()

accepted_2_1 = accepted_2.replace({"Fully Paid": 0,
                                   "Charged Off": 1})

print(accepted_2_1["loan_status"].value_counts())
print(accepted_2_1["loan_status"].value_counts(normalize=True))

# ### Tratamiento de las variables independientes

# #### Tratamiento missings

accepted_2_1 = accepted_2_1.reset_index(drop=True)

missings = accepted_2_1.isna().sum()

missings = missings.to_list()
columns = accepted_2_1.columns

diccionario_NAs = {k: v for k, v in zip(columns, missings)}
diccionario_NAs

# Comenzamos retirando las varibles con más de un 90% de NAs
variables_NA = []
PORCENTAJE_NAS = 0.1

for variable, num_NAs in diccionario_NAs.items():
    if num_NAs > len(accepted_2_1)*PORCENTAJE_NAS:
        variables_NA.append(variable)

variables_NA, len(variables_NA)
```

```
# Retiramos las variables con más de 10% missing
accepted_no_var_NAs = accepted_2_1.drop(variables_NA,axis=1)

accepted_no_var_NAs

accepted_no_NA = accepted_no_var_NAs.dropna()
accepted_no_NA

desc_1 = accepted_no_var_NAs.describe()

desc_2 = accepted_no_NA.describe()

desc_1_small = desc_1[1:3]
desc_1_small

desc_2_small = desc_2[1:3]
desc_2_small

# Estudiamos el porcentaje de cambio en la media y desviación típica de los
elementos al retirar los NAs
cambio_porcentaje = (desc_2_small - desc_1_small) / desc_1_small * 100

print(cambio_porcentaje)

# #### Estudio para retirar variables adicionales

accepted_no_NA.head()

# Columnas eliminadas por inspección ocular
accepted_no_NA_1 = accepted_no_NA.drop(["id","url","zip_code"],axis=1)
accepted_no_NA_1

accepted_no_NA.describe()

vars = ["out_prncp", "out_prncp_inv", "pub_rec",
"total_rec_late_fee","tot_coll_amt", "recoveries", "collection_recovery_fee",
"collections_12_mths_ex_med",
```

```
        "policy_code", "acc_now_delinq", "acc_open_past_24mths",
"chargeoff_within_12_mths", "delinq_amnt", "num_tl_90g_dpd_24m",
"num_accts_ever_120_pd",
          "num_tl_120dpd_2m", "num_tl_30dpd", "pub_rec_bankruptcies",
"tax_liens"]

for var in vars:

print("Variable:",var,"\n",accepted_no_NA[var].value_counts(normalize=True),"
\n")

# Columnas eliminadas por homogeneidad de valores

columnas_homogeneas = ["out_prncp", "out_prncp_inv", "total_rec_late_fee",
"collections_12_mths_ex_med", "policy_code", "acc_now_delinq",
                        "chargeoff_within_12_mths", "delinq_amnt",
"num_tl_120dpd_2m", "num_tl_30dpd"]

accepted_no_NA_2 = accepted_no_NA_1.drop(columnas_homogeneas,axis=1)
accepted_no_NA_2

# #### Variables categóricas

accepted_no_NA_2.head()

CATEGORICAL_COLS = ["term","grade","emp_title",     "emp_length",
"home_ownership","verification_status","issue_d","pymnt_plan", "purpose",
       "title", "addr_state",

"earliest_cr_line","initial_list_status","last_pymnt_d","last_credit_pull_d",
"application_type","hardship_flag","disbursement_method","debt_settlement_fla
g"]

for col in CATEGORICAL_COLS:
    print("Variable:
",col,"\n",accepted_no_NA_2[col].value_counts(normalize=True),"\n")

# Retiramos las variables categóricas que hemos considerado no útiles
```

```
accepted_cat_limpio = accepted_no_NA_2.drop(["pymnt_plan", "hardship_flag",
"debt_settlement_flag","emp_title","title","application_type","disbursement_m
ethod"],axis=1)
accepted_cat_limpio

# #### Tratamiento de fechas

accepted_fechas = accepted_cat_limpio.copy()

# Separamos las columnas de fecha en dos, una para el mes  y otra para el año
accepted_fechas['issue_d_month'] = accepted_fechas['issue_d'].str.split('-
').str[0]
accepted_fechas['issue_d_year'] = accepted_fechas['issue_d'].str.split('-
').str[1]
# Eliminamos la columna original de los datos
accepted_fechas = accepted_fechas.drop("issue_d",axis=1)

accepted_fechas['earliest_cr_line_month'] =
accepted_fechas['earliest_cr_line'].str.split('-').str[0]
accepted_fechas['earliest_cr_line_year'] =
accepted_fechas['earliest_cr_line'].str.split('-').str[1]
accepted_fechas = accepted_fechas.drop("earliest_cr_line",axis=1)

accepted_fechas['last_pymnt_d_month'] =
accepted_fechas['last_pymnt_d'].str.split('-').str[0]
accepted_fechas['last_pymnt_d_year'] =
accepted_fechas['last_pymnt_d'].str.split('-').str[1]
accepted_fechas = accepted_fechas.drop("last_pymnt_d",axis=1)

accepted_fechas['last_credit_pull_d_month'] =
accepted_fechas['last_credit_pull_d'].str.split('-').str[0]
accepted_fechas['last_credit_pull_d_year'] =
accepted_fechas['last_credit_pull_d'].str.split('-').str[1]
accepted_fechas = accepted_fechas.drop("last_credit_pull_d",axis=1)

accepted_fechas["earliest_cr_line_month"].value_counts()

# Procedemos a cambiar el mes de estar expresado en Mmm a su correspondiente
valor numérico
```

```
accepted_fechas["issue_d_month"] =
accepted_fechas["issue_d_month"].replace({"Jan": 1,
"Feb": 2,
"Mar": 3,
"Apr": 4,
"May": 5,
"Jun": 6,
"Jul": 7,
"Aug": 8,
"Sep": 9,
"Oct": 10,
"Nov": 11,
"Dec": 12})

accepted_fechas["earliest_cr_line_month"] =
accepted_fechas["earliest_cr_line_month"].replace({"Jan": 1,
"Feb": 2,
"Mar": 3,
"Apr": 4,
"May": 5,
"Jun": 6,
"Jul": 7,
"Aug": 8,
"Sep": 9,
"Oct": 10,
"Nov": 11,
"Dec": 12})

accepted_fechas["last_pymnt_d_month"] =
accepted_fechas["last_pymnt_d_month"].replace({"Jan": 1,
"Feb": 2,
"Mar": 3,
"Apr": 4,
"May": 5,
"Jun": 6,
"Jul": 7,
"Aug": 8,
"Sep": 9,
"Oct": 10,
"Nov": 11,
```

```
"Dec": 12})

accepted_fechas["last_credit_pull_d_month"] =
accepted_fechas["last_credit_pull_d_month"].replace({"Jan": 1,
"Feb": 2,
"Mar": 3,
"Apr": 4,
"May": 5,
"Jun": 6,
"Jul": 7,
"Aug": 8,
"Sep": 9,
"Oct": 10,
"Nov": 11,
"Dec": 12})

print(accepted_fechas["issue_d_month"].value_counts())
print(accepted_fechas["earliest_cr_line_month"].value_counts())
print(accepted_fechas["last_pymnt_d_month"].value_counts())
print(accepted_fechas["last_credit_pull_d_month"].value_counts())

# Convertimos los años a tipo int
accepted_fechas['issue_d_year'] = accepted_fechas['issue_d_year'].astype(int)
accepted_fechas['earliest_cr_line_year'] =
accepted_fechas['earliest_cr_line_year'].astype(int)
accepted_fechas['last_pymnt_d_year'] =
accepted_fechas['last_pymnt_d_year'].astype(int)
accepted_fechas['last_credit_pull_d_year'] =
accepted_fechas['last_credit_pull_d_year'].astype(int)

# #### Conversión final de tipos

columnas_finales = accepted_fechas.columns
tipos_columnas = accepted_fechas.dtypes

df_variables = pd.DataFrame({'Variable': columnas_finales, 'Tipo de dato':
tipos_columnas})
df_variables = df_variables.reset_index(drop=True)
df_variables
```

```
# Convertimos las variables categóricas de tipo object a tipo categorical
df_cat = df_variables[df_variables["Tipo de dato"]=="object"]
vars_cat = df_cat["Variable"]
vars_cat

accepted_final = accepted_fechas.copy()

for var in vars_cat:
    accepted_final[var] = accepted_final[var].astype('category')

# Guardamos el conjunto de datos final para tener fácil acceso a él
accepted_final.to_csv('accepted_final.csv',index=False)

# Celda para importar los datos en caso de que sea necesario
#prueba = pd.read_csv("accepted_final.csv")
#prueba

# #### Repaso de las variables finales

columnas_finales = accepted_final.columns
tipos_columnas = accepted_final.dtypes

df_variables_2 = pd.DataFrame({'Variable': columnas_finales, 'Tipo de dato':
tipos_columnas})
df_variables_2 = df_variables_2.reset_index(drop=True)
print(len(df_variables_2))
df_variables_2

# Exportar tabla con las variables y su tipo de datos a word
doc = Document()

table = doc.add_table(rows=df_variables_2.shape[0]+1,
cols=df_variables_2.shape[1])

for i, column_name in enumerate(df_variables_2.columns):
    table.cell(0, i).text = column_name

for i in range(df_variables_2.shape[0]):
    for j in range(df_variables_2.shape[1]):
        table.cell(i+1, j).text = str(df_variables_2.values[i, j])
```

```
doc.save('tabla.docx')

# ### Análisis exploratorio de las variables

# #### Variables numéricas

accepted_final.describe()

# Histogramas para las variables numéricas
accepted_final.hist(figsize=(15, 15))
plt.tight_layout()
plt.show()

# Variables con posibles datos atípicos
vars_outliers = ["annual_inc", "dti", "revol_bal", "revol_util",
"total_rec_int", "recoveries", "collection_recovery_fee",
                 "tot_coll_amt", "tot_cur_bal", "total_rev_hi_lim",
"avg_cur_bal", "bc_open_to_buy", "mo_sin_old_il_acct",
                 "tot_hi_cred_lim", "total_bal_ex_mort", "total_bc_limit",
"total_il_high_credit_limit"]

# Boxplots de las variables con posibles datos atípicos
df_boxplots = accepted_final[vars_outliers]

ax = sns.boxplot(data=df_boxplots)
ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
plt.show()

vars_outliers_2 = ["annual_inc", "revol_bal", "tot_coll_amt", "tot_cur_bal",
"total_rev_hi_lim", "tot_hi_cred_lim", "total_bal_ex_mort"]

accepted_final_2 = accepted_final.copy()

for var in vars_outliers_2:
    deciles = np.percentile(accepted_final_2[var], [0,99])
    limite_inferior = deciles[0]
    limite_superior = deciles[1]
    accepted_final_2 = accepted_final_2[(accepted_final_2[var] >=
limite_inferior) & (accepted_final_2[var] <= limite_superior)]
```

```
df_boxplots = accepted_final_2[vars_outliers]

ax = sns.boxplot(data=df_boxplots)
ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
plt.show()

# #### Variables categóricas

vars_cat

# Conteo del número de instancias de cada clase para cada variable
conteo_variable1 = pd.Series(accepted_final_2['term']).value_counts()
conteo_variable2 = pd.Series(accepted_final_2['grade']).value_counts()
conteo_variable3 = pd.Series(accepted_final_2['sub_grade']).value_counts()
conteo_variable4 = pd.Series(accepted_final_2['emp_length']).value_counts()
conteo_variable5 =
pd.Series(accepted_final_2['home_ownership']).value_counts()
conteo_variable6 =
pd.Series(accepted_final_2['verification_status']).value_counts()
conteo_variable7 = pd.Series(accepted_final_2['purpose']).value_counts()
conteo_variable8 = pd.Series(accepted_final_2['addr_state']).value_counts()
conteo_variable9 =
pd.Series(accepted_final_2['initial_list_status']).value_counts()

fig = go.Figure(
    data=[
        go.Pie(
            labels=conteo_variable1.index,
            values=conteo_variable1,
            )
        ]
    )

fig.update_layout(
    title_text="term",
    title_font_size=12,
    title_x=0.45,
    width=500,
```

```
    height=500
)
fig.show()

fig = go.Figure(
    data=[
        go.Pie(
            labels=conteo_variable2.index,
            values=conteo_variable2,
            )
        ]
    )

fig.update_layout(
    title_text="grade",
    title_font_size=12,
    title_x=0.5,
    width=500,
    height=500
)
fig.show()

fig = go.Figure(
    data=[
        go.Pie(
            labels=conteo_variable3.index,
            values=conteo_variable3,
            )
        ]
    )

fig.update_layout(
    title_text="sub-grade",
    title_font_size=12,
    title_x=0.5,
    title_y=0.73,
    width=550,
    height=1200
)
fig.show()
```

```
# Agrupación de las clases con menos de un 1% de los casos en la nueva clase
"otros"
umbral = 0.01
conteo_clases = accepted_final_2['sub_grade'].value_counts(normalize=True)

clases_pocas_observaciones = conteo_clases[conteo_clases <
umbral].index.tolist()

accepted_final_3 = accepted_final_2.copy()
accepted_final_3['sub_grade'] = accepted_final_3['sub_grade'].apply(lambda x:
'otros' if x in clases_pocas_observaciones else x)

# Nueva distribución de sub_grade
conteo_variable3_2 = pd.Series(accepted_final_3['sub_grade']).value_counts()

fig = go.Figure(
    data=[
        go.Pie(
            labels=conteo_variable3_2.index,
            values=conteo_variable3_2,
            )
        ]
    )

fig.update_layout(
    title_text="sub-grade",
    title_font_size=12,
    title_x=0.5,
    title_y=0.73,
    width=550,
    height=1200
)
fig.show()

fig = go.Figure(
    data=[
        go.Pie(
            labels=conteo_variable4.index,
            values=conteo_variable4,
```

```
            )
        ]
    )

fig.update_layout(
    title_text="emp_length",
    title_font_size=12,
    title_x=0.47,
    width=500,
    height=500
)
fig.show()

fig = go.Figure(
    data=[
        go.Pie(
            labels=conteo_variable5.index,
            values=conteo_variable5,
            )
        ]
    )

fig.update_layout(
    title_text="home_ownership",
    title_font_size=12,
    title_x=0.46,
    title_y=0.85,
    width=500,
    height=500
)
fig.show()

accepted_final_3["home_ownership"] =
accepted_final_3["home_ownership"].replace({"ANY": "otros",

"NONE": "otros",

"OTHER": "otros"})

# Nueva distribución de home_ownership
```

```
conteo_variable5_2 =
pd.Series(accepted_final_3['home_ownership']).value_counts()

fig = go.Figure(
    data=[
        go.Pie(
            labels=conteo_variable5_2.index,
            values=conteo_variable5_2,
            )
        ]
    )

fig.update_layout(
    title_text="home_ownership",
    title_font_size=12,
    title_x=0.46,
    title_y=0.85,
    width=500,
    height=500
)
fig.show()

fig = go.Figure(
    data=[
        go.Pie(
            labels=conteo_variable6.index,
            values=conteo_variable6,
            )
        ]
    )

fig.update_layout(
    title_text="verification_status",
    title_font_size=12,
    title_x=0.42,
    title_y=0.85,
    width=500,
    height=500
)
fig.show()
```

```
accepted_final_3["verification_status"] =
accepted_final_3["verification_status"].replace({"Source Verified":
"Verified"})

# Nueva distribución de verification_status
conteo_variable6_2 =
pd.Series(accepted_final_3['verification_status']).value_counts()

fig = go.Figure(
    data=[
        go.Pie(
            labels=conteo_variable6_2.index,
            values=conteo_variable6_2,
            )
        ]
    )

fig.update_layout(
    title_text="verification_status",
    title_font_size=12,
    title_x=0.42,
    title_y=0.85,
    width=500,
    height=500
)
fig.show()

fig = go.Figure(
    data=[
        go.Pie(
            labels=conteo_variable7.index,
            values=conteo_variable7,
            )
        ]
    )

fig.update_layout(
    title_text="purpose",
    title_font_size=12,
```

```
    title_x=0.40,
    title_y=0.83,
    width=500,
    height=500
)
fig.show()

# Agrupación de las clases con menos de un 1% de los casos en la nueva clase
"otro"
umbral = 0.015
conteo_clases = accepted_final_3['purpose'].value_counts(normalize=True)

clases_pocas_observaciones = conteo_clases[conteo_clases <
umbral].index.tolist()

accepted_final_3['purpose'] = accepted_final_3['purpose'].apply(lambda x:
'otro' if x in clases_pocas_observaciones else x)

# Nueva distribución de purpose
conteo_variable7_2 = pd.Series(accepted_final_3['purpose']).value_counts()

fig = go.Figure(
    data=[
        go.Pie(
            labels=conteo_variable7_2.index,
            values=conteo_variable7_2,
            )
        ]
    )

fig.update_layout(
    title_text="purpose",
    title_font_size=12,
    title_x=0.40,
    title_y=0.83,
    width=500,
    height=500
)
fig.show()
```

```
fig = go.Figure(
    data=[
        go.Pie(
            labels=conteo_variable8.index,
            values=conteo_variable8,
            )
        ]
    )

fig.update_layout(
    title_text="addr_state",
    title_font_size=25,
    title_x=0.495,
    title_y=0.8,
    width=1150,
    height=1800
)
fig.show()

# Agrupación de las clases con menos de un 1% de los casos en la nueva clase
"otro"
umbral = 0.01
conteo_clases = accepted_final_3['addr_state'].value_counts(normalize=True)

clases_pocas_observaciones = conteo_clases[conteo_clases <
umbral].index.tolist()

accepted_final_3['addr_state'] = accepted_final_3['addr_state'].apply(lambda
x: 'otro' if x in clases_pocas_observaciones else x)

# Nueva distribución de purpose
conteo_variable8_2 = pd.Series(accepted_final_3['addr_state']).value_counts()

fig = go.Figure(
    data=[
        go.Pie(
            labels=conteo_variable8_2.index,
            values=conteo_variable8_2,
            )
        ]
```

```
    )

fig.update_layout(
    title_text="addr_state",
    title_font_size=25,
    title_x=0.495,
    title_y=0.9,
    width=1150,
    height=1400
)
fig.show()

fig = go.Figure(
    data=[
        go.Pie(
            labels=conteo_variable9.index,
            values=conteo_variable9,
            )
        ]
    )

fig.update_layout(
    title_text="initial_list_status",
    title_font_size=11,
    title_x=0.5,
    title_y=0.85,
    width=500,
    height=500
)
fig.show()

# #### Guardamos el conjunto de datos definitivo

accepted_final_3.to_csv('accepted_definitivo.csv',index=False)

#Código para importar los datos en caso de que sea necesario
#accepted_definitivo = pd.read_csv("accepted_definitivo.csv")
#accepted_definitivo

# ### Transformación de las variables categóricas
```

```
df_dummies_1 = pd.get_dummies(accepted_final_3['term'], prefix='term')
accepted_num = pd.concat([accepted_final_3, df_dummies_1], axis=1)
accepted_num.drop('term', axis=1, inplace=True)

df_dummies_2 = pd.get_dummies(accepted_num['grade'], prefix='grade')
accepted_num = pd.concat([accepted_num, df_dummies_2], axis=1)
accepted_num.drop('grade', axis=1, inplace=True)

df_dummies_3 = pd.get_dummies(accepted_num['sub_grade'], prefix='sub_grade')
accepted_num = pd.concat([accepted_num, df_dummies_3], axis=1)
accepted_num.drop('sub_grade', axis=1, inplace=True)

df_dummies_4 = pd.get_dummies(accepted_num['emp_length'],
prefix='emp_length')
accepted_num = pd.concat([accepted_num, df_dummies_4], axis=1)
accepted_num.drop('emp_length', axis=1, inplace=True)

df_dummies_5 = pd.get_dummies(accepted_num['home_ownership'],
prefix='home_ownership')
accepted_num = pd.concat([accepted_num, df_dummies_5], axis=1)
accepted_num.drop('home_ownership', axis=1, inplace=True)

df_dummies_6 = pd.get_dummies(accepted_num['verification_status'],
prefix='verification_status')
accepted_num = pd.concat([accepted_num, df_dummies_6], axis=1)
accepted_num.drop('verification_status', axis=1, inplace=True)

df_dummies_7 = pd.get_dummies(accepted_num['purpose'], prefix='purpose')
accepted_num = pd.concat([accepted_num, df_dummies_7], axis=1)
accepted_num.drop('purpose', axis=1, inplace=True)

df_dummies_8 = pd.get_dummies(accepted_num['addr_state'],
prefix='addr_state')
accepted_num = pd.concat([accepted_num, df_dummies_8], axis=1)
accepted_num.drop('addr_state', axis=1, inplace=True)

df_dummies_9 = pd.get_dummies(accepted_num['initial_list_status'],
prefix='initial_list_status')
accepted_num = pd.concat([accepted_num, df_dummies_9], axis=1)
```

```
accepted_num.drop('initial_list_status', axis=1, inplace=True)

accepted_num

# ### Modelos

# #### Preparación de los datos para la regresión logística y la red neuronal

# Retiramos la variable dependiente del conjunto de datos
y = accepted_num["loan_status"]
X = accepted_num.drop("loan_status",axis = 1)

# Normalizamos los datos
scaler = StandardScaler()
X_norm = pd.DataFrame(scaler.fit_transform(X), columns=X.columns)

# Generamos los conjunto de entrenamiento y de test
X_train, X_test, y_train, y_test = train_test_split(X_norm, y, test_size=0.3,
random_state=50)

# #### Regresión logística

# Creamos y entrenamos el modelo de regresión logística
model = LogisticRegression(max_iter=10000)
model.fit(X_train, y_train)

# Guardamos el modelo
pickle.dump(model, open("modelo_logit", 'wb'))

# Leemos el modelo
# model = pickle.load(open("modelo_logit", 'rb'))

# Predicción del conjunto de test
y_pred = model.predict(X_test)

# Grafica de la matriz de confusión
matriz_de_confusion = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=matriz_de_confusion)
disp.plot()
```

```
# Evaluación del modelo
precision = precision_score(y_test, y_pred)
exhaustividad = recall_score(y_test, y_pred)
F_score = f1_score(y_test, y_pred)
exactitud = accuracy_score(y_test, y_pred)
ROC = roc_auc_score(y_test, y_pred)

print("La exhaustividad del modelo es: {:.2f}%".format(exhaustividad*100))
print("El F-Score del modelo es: {:.2f}%".format(F_score*100))
print("La precisión del modelo es: {:.2f}%".format(precision*100))
print("La exactitud del modelo es: {:.2f}%".format(exactitud*100))
print("El área ROC es: {:.2f}%".format(ROC*100))

# Gráfica de la curva ROC
fpr, tpr, thresholds = roc_curve(y_test, y_pred)

# Graficar la curva ROC
plt.plot(fpr, tpr, label='Curva ROC')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('Tasa de Falsos Positivos')
plt.ylabel('Tasa de Verdaderos Positivos')
plt.title('Curva ROC')
plt.legend(loc='lower right')
plt.show()

# #### Modelo de red neuronal

encoder = LabelEncoder()
encoder.fit(y)
encoded_Y = encoder.transform(y)

X_train, X_test, y_train, y_test = train_test_split(X_norm, encoded_Y,
test_size=0.3, random_state=50)

# Definimos la arquitectura de la red
model_NN = Sequential()

model_NN.add(Dense(32, activation='relu', input_shape=(155,)))
model_NN.add(Dense(16, activation='relu'))
model_NN.add(Dense(1, activation='sigmoid'))
```

```
# Entrenamos la red
model_NN.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
model_NN.fit(X_train, y_train, epochs=4, batch_size=32)

# Guardamos el modelo
save_model(model_NN, "model_NN.h5")

# Leemos el modelo
# model_NN = load_model('model_NN.h5')

# Predicciones de la red sobre el conjunto de test
y_pred_NN = model_NN.predict(X_test)
y_pred_NN = np.where(y_pred_NN >= 0.5, 1, 0)

# Grafica de la matriz de confusión
matriz_de_confusion_NN = confusion_matrix(y_test, y_pred_NN)
disp = ConfusionMatrixDisplay(confusion_matrix=matriz_de_confusion_NN)
disp.plot()

# Evaluación del modelo
precision_NN = precision_score(y_test, y_pred_NN)
exhaustividad_NN = recall_score(y_test, y_pred_NN)
F_score_NN = f1_score(y_test, y_pred_NN)
exactitud_NN = accuracy_score(y_test, y_pred_NN)
ROC_NN = roc_auc_score(y_test, y_pred_NN)

print("La exhaustividad del modelo es: {:.2f}%".format(exhaustividad_NN*100))
print("El F-Score del modelo es: {:.2f}%".format(F_score_NN*100))
print("La precisión del modelo es: {:.2f}%".format(precision_NN*100))
print("La exactitud del modelo es: {:.2f}%".format(exactitud_NN*100))
print("El área ROC es: {:.2f}%".format(ROC_NN*100))

# Gráfica de la curva ROC
fpr, tpr, thresholds = roc_curve(y_test, y_pred_NN)

# Graficar la curva ROC
plt.plot(fpr, tpr, label='Curva ROC')
plt.plot([0, 1], [0, 1], 'k--')
```

```
plt.xlabel('Tasa de Falsos Positivos')
plt.ylabel('Tasa de Verdaderos Positivos')
plt.title('Curva ROC')
plt.legend(loc='lower right')
plt.show()

# #### Árbol de decisión

# Creamos los conjuntos de entrenamiento y de prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=50)

# Creamos y entrenamos el modelo de árbol de decisión
model_DT = DecisionTreeClassifier()
model_DT.fit(X_train, y_train)

# Guardamos el modelo
pickle.dump(model_DT, open("modelo_DT", 'wb'))

# Leemos el modelo
# model_DT = pickle.load(open("modelo_DT", 'rb'))

# Realizamos las predicciones del modelo
y_pred_DT = model_DT.predict(X_test)

# Grafica de la matriz de confusión
matriz_de_confusion_DT = confusion_matrix(y_test, y_pred_DT)
disp = ConfusionMatrixDisplay(confusion_matrix=matriz_de_confusion_DT)
disp.plot()

# Evaluación del modelo
precision_DT = precision_score(y_test, y_pred_DT)
exhaustividad_DT = recall_score(y_test, y_pred_DT)
F_score_DT = f1_score(y_test, y_pred_DT)
exactitud_DT = accuracy_score(y_test, y_pred_DT)
ROC_DT = roc_auc_score(y_test, y_pred_DT)

print("La exhaustividad del modelo es: {:.2f}%".format(exhaustividad_DT*100))
print("El F-Score del modelo es: {:.2f}%".format(F_score_DT*100))
print("La precisión del modelo es: {:.2f}%".format(precision_DT*100))
```

```
print("La exactitud del modelo es: {:.2f}%".format(exactitud_DT*100))
print("El área ROC es: {:.2f}%".format(ROC_DT*100))

# Gráfica de la curva ROC
fpr, tpr, thresholds = roc_curve(y_test, y_pred_DT)

# Graficar la curva ROC
plt.plot(fpr, tpr, label='Curva ROC')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('Tasa de Falsos Positivos')
plt.ylabel('Tasa de Verdaderos Positivos')
plt.title('Curva ROC')
plt.legend(loc='lower right')
plt.show()

# #### XGBoost

# Cambiamos el nombre de la variable emp_length_< 1 year por
emp_length_less_1 year

X_2 = X.rename(columns={"emp_length_< 1 year": "emp_length_less_1 year"})

# Generamos el conjunto de entrenamiento y test
X_train, X_test, y_train, y_test = train_test_split(X_2, y, test_size=0.3,
random_state=50)

# Creamos y entrenamos el modelo de clasificación de XGBoost
model_XG = XGBClassifier()
model_XG.fit(X_train, y_train)

# Guardar modelo
model_XG.save_model("model_XG.json")

# Importar modelo
# model_XG = xgb.Booster()
# model_XG.load_model("model_XG.json")

# Realizamos las predicciones del modelo
y_pred_XG = model_XG.predict(X_test)
```

```
# Grafica de la matriz de confusión
matriz_de_confusion_XG = confusion_matrix(y_test, y_pred_XG)
disp = ConfusionMatrixDisplay(confusion_matrix=matriz_de_confusion_XG)
disp.plot()

# Valores exactos de la matriz de confusión
matriz_de_confusion_XG

# Evaluación del modelo
precision_XG = precision_score(y_test, y_pred_XG)
exhaustividad_XG = recall_score(y_test, y_pred_XG)
F_score_XG = f1_score(y_test, y_pred_XG)
exactitud_XG = accuracy_score(y_test, y_pred_XG)
ROC_XG = roc_auc_score(y_test, y_pred_XG)

print("La exhaustividad del modelo es: {:.2f}%".format(exhaustividad_XG*100))
print("El F-Score del modelo es: {:.2f}%".format(F_score_XG*100))
print("La precisión del modelo es: {:.2f}%".format(precision_XG*100))
print("La exactitud del modelo es: {:.2f}%".format(exactitud_XG*100))
print("El área ROC es: {:.2f}%".format(ROC_XG*100))

# Gráfica de la curva ROC
fpr, tpr, thresholds = roc_curve(y_test, y_pred_XG)

# Graficar la curva ROC
plt.plot(fpr, tpr, label='Curva ROC')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('Tasa de Falsos Positivos')
plt.ylabel('Tasa de Verdaderos Positivos')
plt.title('Curva ROC')
plt.legend(loc='lower right')
plt.show()
```