TRABAJO DE FIN DE GRADO

Doble grado en Economía y Estadística

Título: Modelos de clasificación con datos desbalanceados: Estudio de la morosidad

Autor: Ayoub Bentohami Amiah

Director: Francisco Javier Sierra Martínez

Departamento: Econometría, Estadística y Economía aplicada

Convocatoria: Julio 2024

UNIVERSITAT DE BARCELONA

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat de Matemàtiques i Estadística

Resumen

Análisis del impacto de la morosidad en el pago de préstamos. Se examina los efectos a nivel macroeconómico y microeconómico respecto a entidades financieras, y se estudian metodologías para prevenir este fenómeno.

Por otro lado, se estudia como el *machine learning* contribuye a la detección de posibles morosos para reducir el riesgo de impago al que las entidades se exponen, teniendo en cuenta problemáticas como el desbalance de clases.

Palabras clave

Riesgo, morosidad, mercados financieros, bancos, créditos, sector financiero, préstamos, impago, machine learning, algoritmo, aprendizaje supervisado, bases de datos desbalanceadas.

Abstract

Analysis of the impact of late payment on loan repayment. It examines the effects at the macroeconomic and microeconomic level with respect to financial institutions, and studies methodologies to prevent this phenomenon.

On the other hand, we study how machine learning contributes to the detection of potential defaulters in order to reduce the risk of non-payment to which institutions are exposed, taking into account problems such as class imbalance.

Keywords

Risk, non-performing loans, financial markets, banks, credit, financial sector, loans, default, machine learning, algorithm, supervised learning, imbalanced databases.

Clasificación AMS

62H30 - Classification and discrimination; cluster analysis (statistical aspects); mixture models

91B05 - Risk models

Índice general

Índice de figuras

Introducción

El sector financiero es un pilar fundamental en el mundo, proporcionando las herramientas necesarias para el desarrollo económico. A menudo, las instituciones financieras se enfrentan a varios riesgos, y entre ellos, el riesgo de morosidad: es la posibilidad de que los prestatarios sean incapaces de afrontar y cumplir sus obligaciones de pago. Este riesgo afecta a las entidades financieras, aunque si no se mitiga a tiempo, puede desencadenar en problemáticas a gran escala.

Las entidades financieras abordan este problema con un amplio abanico de estrategias: desde análisis cualitativos del individuo donde se valoran aspectos personales como prestatario, análisis cuantitativos donde se mide su solvencia numéricamente, o planes de actuación para diversificar el riesgo o reducirlo mediante políticas de concesión más estrictas. Aunque, aun tomando este tipo de medidas, identificar a los clientes morosos no solo acertadamente, sino a tiempo, puede ser un desafío debido a la variabilidad y la complejidad de las variables que afectan al comportamiento financiero de los prestatarios. Este error puede provocarle al sector financiero pérdidas y coste de oportunidad con los beneficios que dejan de obtener por sus decisiones.

En este contexto, el auge de la inteligencia artificial y el machine learning crean varias vías para estudiar esta problemática, gracias a que permiten detectar con mayor precisión que la metodología tradicional a perfiles concretos de clientes mediante el manejo de cantidades de información masivas. El aprendizaje supervisado contribuye tanto a la detección de la morosidad como a la aportación de información para la toma de decisiones por parte de las entidades.

Sin embargo, la proporción de clientes de riesgo es muy baja, por lo que puede acarrear problemas a los algoritmos a la hora de clasificarlos, generando un sesgo hacia la clase mayoritaria. Esto puede ser un problema muy grave, porque puede conducir a las entidades

a tratar con clientes potencialmente morosos, suponiendo que no lo son. Es crucial entonces valorar el uso de técnicas de remuestreo para mejorar la capacidad predictiva, el modelo, y evitar el error tipo II.

Objetivos del trabajo

En este trabajo contextualizaré el sistema financiero en España, trabajando con datos de la morosidad a nivel sectorial y agregado, así como estudiaré la evolución en los últimos años. A continuación, analizaré y evaluaré mediante una serie de criterios una selección de algoritmos, aplicando a su vez de técnicas de muestreo para mitigar el error de clasificación causado por las bases de datos donde una clase está sobrerrepresentada.

Estructura del documento

El trabajo está dividido en capítulos, dedicando el primer capítulo a entender el sector financiero y el riesgo de morosidad, y posteriormente analizar la situación española tanto a nivel macroeconómico como microeconómico.

En el segundo capítulo, estudiaré *machine learning* y sus aplicaciones en el sector financiero. Explicaré la metodología usada en proyectos en este ámbito, describiendo cada paso y algoritmo que utilizaré más adelante.

Finalmente, en el tercer capítulo implementaré la teoría explicada en los anteriores apartados para realizar un análisis de morosidad sobre una base de datos, implementando los algoritmos previamente seleccionados para clasificar a una serie de clientes según su calificación crediticia.

Capítulo 1

El riesgo de crédito en la sociedad española

1.1. Mercados financieros y concesión de créditos

Los mercados financieros son una pieza esencial en el engranaje de la economía global. Actúan como un sistema circulatorio para la economía mundial, canalizando fondos desde aquellos que tienen excedentes hacia aquellos que tienen déficits. En este sentido, los mercados financieros son el corazón de la economía, "bombeando sangre" a través del sistema económico y manteniéndolo vivo y saludable, gracias a instrumentos que llamamos activos financieros. Se encargan de proporcionar un lugar donde los ahorradores pueden invertir su dinero y donde las empresas y los gobiernos pueden recaudar fondos para sus operaciones y proyectos de inversión. Al hacerlo, los mercados financieros facilitan el crecimiento económico al permitir la acumulación de capital, que es esencial para la expansión de la producción y el empleo.

1.1.1. Funciones de los mercados financieros

Los mercados financieros son una parte integral de la infraestructura económica global. Proporcionan un foro para la compra y venta de activos financieros como acciones, bonos, divisas, derivados y otros instrumentos financieros. Los mercados financieros son esenciales para el funcionamiento eficiente de la economía, ya que facilitan el flujo de fondos desde aquellos con excedentes de capital (ahorradores) hacia aquellos que necesitan capital (inversores). Estos mercados cumplen tres funciones principales (Mascareñas, 2012):

- **Determinación de precios**: Los mercados financieros permiten la interacción entre compradores y vendedores, lo que lleva a la determinación de los precios de los activos financieros en función de la oferta y la demanda. Este proceso de determinación de precios es fundamental para el funcionamiento eficiente de la economía, ya que los precios de los activos financieros reflejan la información disponible y las expectativas de los inversores sobre el futuro.
- **Facilitar transacciones**: Los mercados financieros proporcionan el mecanismo necesario para comprar o vender activos financieros. Esto incluye no solo la infraestructura física o virtual para realizar transacciones, sino también un conjunto de reglas y procedimientos que garantizan que las transacciones se realicen de manera justa y eficiente.
- **Proporcionar liquidez**: Al permitir la compra y venta rápida de activos, los mercados financieros proporcionan liquidez a los inversores. La liquidez es crucial para los inversores, ya que les permite ajustar rápidamente sus carteras en respuesta a cambios en sus necesidades o expectativas.

Además de estas funciones, los mercados financieros también desempeñan un papel en la transferencia de recursos, la gestión de riesgos y la promoción de la eficiencia del mercado. Aunque estos aspectos son menos relevantes para un estudio sobre el riesgo crediticio, siguen siendo componentes importantes del funcionamiento general de los mercados financieros.

1.2. Activos financieros

Un activo financiero (Mascareñas, 2012) es un contrato que da a su propietario un reclamo sobre los ingresos o activos de la entidad emisora. Los activos financieros son fundamentales para la economía, ya que representan la riqueza de los individuos y las empresas y facilitan el flujo de fondos en la economía. Hay una extensa variedad de activos, pero si nos regimos al enfoque hacia la concesión de crédito, podemos señalar varios tipos de activos financieros que son particularmente relevantes para nuestro estudio:

- **Préstamos**: Un préstamo es un activo financiero que representa la obligación de un prestatario de devolver una cantidad de dinero al prestamista. Los préstamos son una de las formas más comunes de activos financieros en la concesión de créditos.
- **Bonos**: Un bono es un activo financiero que representa la obligación de una entidad (como un gobierno o una empresa) de pagar a los tenedores del bono una serie de pagos de intereses y el valor nominal del bono en la fecha de vencimiento. Los bonos son una forma común de financiación para las empresas y los gobiernos.

Cada uno de estos activos financieros juega un papel parecido en la concesión de créditos: Los préstamos permiten a los bancos y otras instituciones financieras proporcionar financiación a los prestatarios; los bonos permiten a las empresas y los gobiernos recaudar fondos para sus operaciones y proyectos de inversión. La finalidad de estos es clara: los prestadores, son usuarios que prestan su capital excedente a prestatarios, usuarios que precisan de financiación para llevar a cabo objetivos personales o empresariales, desde comprarte una casa o un coche, hasta llevar a cabo una idea de negocio y emprender.

La **liquidez** y la **rentabilidad** son características importantes de los activos financieros. Por un lado, la rentabilidad es el rendimiento que dicho activo tiene. La liquidez se refiere a la facilidad con la que un activo puede ser convertido en efectivo, por lo tanto, es más líquido si puede ser vendido rápidamente sin afectar significativamente su precio. Por ejemplo, una acción en la bolsa nacional es un activo considerablemente líquido, ya que, salvo comisiones y tasas, el precio al que puedes venderlo prácticamente de manera instantánea no difiere del precio de mercado. Ahora bien, no podemos decir lo mismo de un piso; fijar el valor de la propiedad, encontrar un potencial comprador, negociar el precio, y demás obstáculos en el proceso no solo son un coste en términos de tiempo, sino

que el precio final de venta puede diferir bastante respecto a su valor real (supongamos un valor al que ha sido tasado).

En el contexto de la concesión de créditos, los prestamistas buscan activos que ofrezcan un buen equilibrio entre liquidez, rentabilidad y un tercer factor clave en nuestro análisis: el riesgo, que es esencialmente la posibilidad de que el prestatario no cumpla su obligación (por cualquier razón que por el momento no tendremos en cuenta) de devolver el capital prestado.

Estas tres características definen el precio del activo en sí, pero es importante tenerlo en consideración no solo desde el punto de vista monetario. Entremos en profundidad en entender como el riesgo es un rasgo tan importante de los activos financieros, basando nuestro enfoque hacia los ya mencionados bonos y préstamos. Los bonos (dejando de lado las empresas) tal y como hemos explicado previamente, pueden ser emisiones de deuda nacional; normalmente llevan implícito un riesgo muy bajo, ya que un Estado, por lo general, muy probablemente acabe cumpliendo su obligación. Pero, aun así, es interesante para los prestadores clasificar a los Estados según su solvencia con el objetivo de poder distinguir entre bonos mejores (o peores), y para ello utilizan un indicador llamado la prima de riesgo. Esta es básicamente la diferencia en la tasa de interés (o rentabilidad) que hay entre un bono sano, estable y fiable que se usa como referencia versus otro bono el cual queramos comparar.

Por otro lado, en cuanto a préstamos se trata, el enfoque no es tan sencillo. A diferencia de los bonos, el prestatario, en este caso, al ser un particular o empresa, lleva implícito mucho más riesgo de insolvencia, y se debe de llevar un análisis mucho más minucioso con tal de evaluar la concesión. Para entender este modus operandi, es adecuado conocer quiénes son los principales prestatarios de este tipo y analizar cómo funcionan.

1.3. Instituciones financieras de crédito

Las instituciones financieras son entidades que facilitan las transacciones financieras en la economía, como inversiones, préstamos y depósitos. Existen varios tipos de instituciones financieras, incluyendo bancos comerciales, cooperativas de crédito, compañías financieras, y sociedades de ahorro y préstamo. Sin embargo, para conveniencia del posterior análisis que llevaré a cabo, voy a centrarme en los bancos comerciales. Los bancos comerciales son, en esencia, instituciones que aceptan depósitos y conceden préstamos. En este sentido, los bancos comerciales desempeñan dos roles principales:

- **Intermediarios financieros**: Los bancos comerciales facilitan el flujo de dinero desde los ahorradores hasta los prestatarios a escala particular. En este aspecto, desempeñan un papel crucial en la canalización de los ahorros hacia las inversiones productivas, lo que es fundamental para el crecimiento económico de la sociedad en general.
- **Gestores de riesgos**: Los bancos comerciales evalúan el riesgo crediticio de los prestatarios potenciales y deciden si conceder o no un préstamo en función de esta evaluación. Este rol mejora la estabilidad del sistema financiero, ya que ayuda a prevenir la acumulación de riesgos excesivos.

1.3.1. Concesión crediticia e importancia del riesgo

La concesión de créditos es un proceso que implica varios pasos. Primero, el prestatario potencial solicita un préstamo al banco. Esta solicitud puede incluir información sobre el propósito del préstamo, la cantidad solicitada, y detalles sobre la situación financiera del prestatario.

A continuación, el banco evalúa la solicitud de préstamo, analizando varias componentes características del prestatario: **información económica** (relacionada con los ingresos, a su historial previo crediticio, su patrimonio...), **social** (relacionada con su estado civil, o cantidad de hijos...), y **otros aspectos** que la entidad pueda considerar relevantes. Esto implica verificar la información proporcionada por el prestatario, evaluar su solvencia, y determinar el nivel de riesgo asociado. Esta evaluación implica el uso de modelos de **calificación crediticia** y es clave para el funcionamiento de los bancos, ya que una

a mayor calidad de las decisiones que tome, mayores serán sus beneficios. Este paso es el objetivo a estudiar en mi trabajo, y posteriormente profundizaremos en, no solo un funcionamiento más detallado, sino que analizaremos minuciosamente su repercusión en las entidades financieras.

A partir de este punto, si el banco toma la decisión de conceder el préstamo, ya que concluye que es un **buen prestatario** (recordemos, un prestatario en este contexto será esencialmente mejor cuanto mayor sea la seguridad de que pueda responder a sus obligaciones), entonces puede aprobar la solicitud de préstamo. El banco y el prestatario entonces acuerdan los términos del préstamo, incluyendo la cantidad del préstamo, la tasa de interés, y el calendario de pagos. Una vez concedido, se monitorea el préstamo para asegurarse de que el prestatario está haciendo los pagos a tiempo. Si se diera el caso de que existieran retrasos a la hora de hacer los pagos, o incluso se dieran indicios de morosidad, el banco puede tomar medidas para recuperar el dinero prestado, como la ejecución hipotecaria de una propiedad o la venta de la deuda a una agencia de cobro. No obstante, estas acciones perjudican al prestador, puesto que está en riesgo de no recibir lo acordado, y, por ende, pierde dinero.

Es por esto por lo que hago hincapié en que la detección y gestión del riesgo es una parte esencial de este proceso. Los bancos deben equilibrar cuidadosamente el deseo de hacer préstamos (y así generar ingresos por intereses) con la necesidad de minimizar el riesgo de pérdidas por préstamos incobrables, puesto que una alta tasa de morosidad puede perjudicar seriamente la salud económica de un banco, y como hemos visto a lo largo de la historia, también puede ser un serio problema a nivel mundial.

1.4. Detección de potencial morosidad

La gestión de riesgos en el sector bancario es un elemento crucial para la estabilidad financiera y la continuidad operativa, en este caso, de los bancos. Gran parte del riesgo recae en la correcta determinación de la capacidad de pago de un potencial cliente, ya que un impago, por un lado, provoca pérdidas para el banco, ya que pierde los fondos que ha prestado; pero, ahora bien, considerar a un buen pagador como uno que no lo es también provoca una pérdida de oportunidades para la entidad de generar intereses.

Como hemos visto en la historia, un mal funcionamiento del sistema financiero puede tener consecuencias devastadoras en la sociedad, y como ejemplo relacionado con mi análisis, consideraremos la "reciente" crisis financiera del 2008, que tuvo como detonante principal la caída del gran banco estadounidense Lehman Brothers Holdings Inc. En resumidas cuentas, el gran error que se produjo en ese periodo de "vacas gordas" y bonanza económica fue precisamente la concesión de créditos sin tener en cuenta la capacidad de pago del prestatario: una persona podía llegar a tener acceso a una elevada cantidad de financiación sin necesariamente presentar la capacidad de retornar dicho capital en el futuro. Este hecho con el tiempo fue empeorando hasta llegar a tal magnitud que provocó un estallido de la burbuja y por consecuencia, la crisis financiera más grave hasta la fecha.

Por un lado, entendemos por morosidad «el hecho de que un deudor incumpla sus obligaciones de pago en el momento temporal prefijado»(«Qué es la morosidad y qué consecuencias tiene»). Por ende, el riesgo de crédito es «la probabilidad de pérdida debido al incumplimiento en los pagos de cualquier tipo de deuda de parte del deudor» («Riesgo del crédito: Qué es y por qué importa»). Normalmente, para cuantificar la morosidad se suele utilizar el siguiente ratio al que llamaremos tasa de morosidad:

$$\text{Tasa de morosidad} = \frac{\text{Saldo de la cartera de crédito en mora o dudoso cobro}}{\text{Saldo de la cartera de crédito total}} \tag{1.1}$$

En el siguiente apartado, profundizaré en por qué una alta tasa de morosidad puede ser nociva para la salud económica de un Estado, e incluso a nivel global. Para ello,

estudiaré tanto las causas como los efectos, además de que variables macroeconómicas están relacionadas con la tasa de morosidad (y de qué manera).

1.4.1. ¿Qué es la morosidad y cuál ha sido su evolución?

Es interesante para profundizar en el análisis del estado financiero español tener claro los conceptos tanto de morosidad, como del riesgo de crédito (así como los ratios derivados de estos), que nos ayudaran a entender la trayectoria de las finanzas españolas y como la crisis del COVID ha influido.

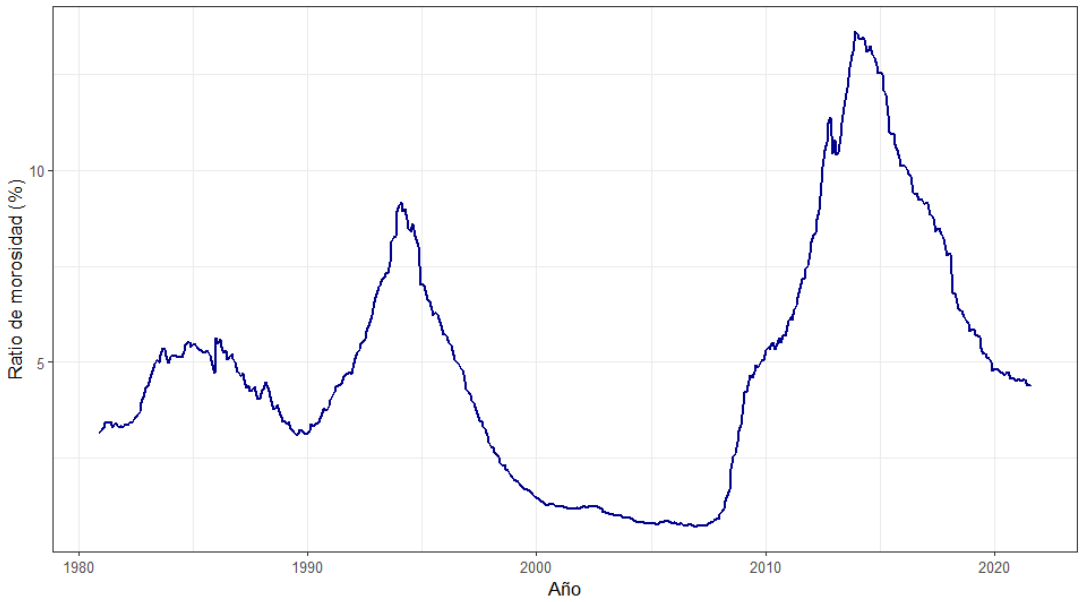


Figura 1.1: Evolución del ratio de mora entre los años 1980 y 2021

Fuente: Banco de España

Aun así, como podemos observar en este gráfico, vemos como la morosidad se ha disparado en dos momentos: alrededor del 1993 por la crisis del petróleo y otros factores mundiales, y la más reciente, una tendencia creciente desde 2008 hasta 2014, con la crisis financiera de 2008. Veamos como ha evolucionado durante los años el volumen de créditos concedidos.

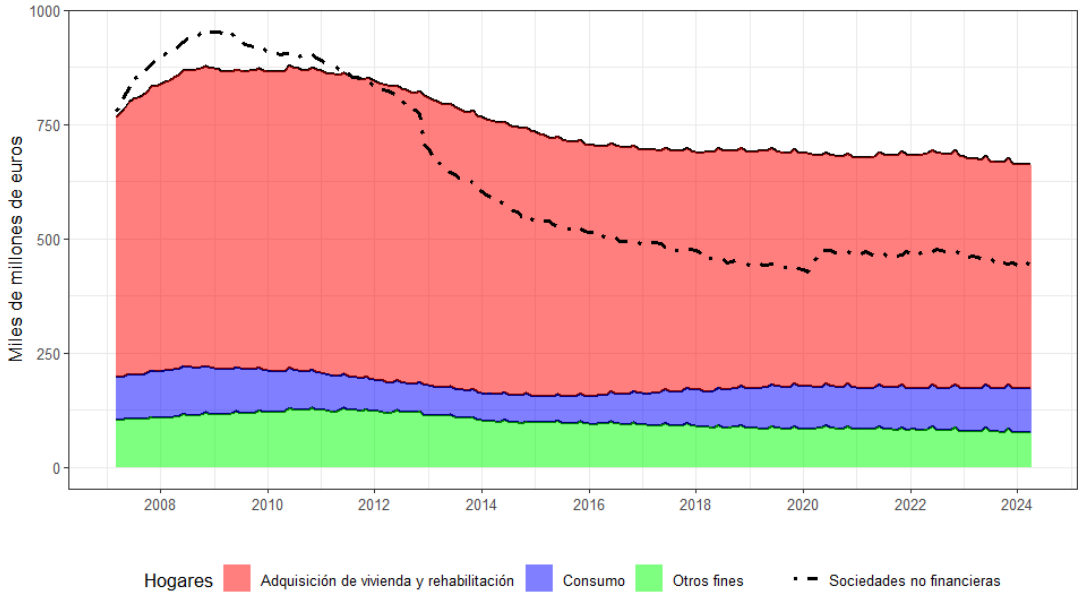


Figura 1.2: Volumen total de créditos a lo largo de los años en España

Fuente: Banco de España

Como podemos ver en el gráfico, el total de créditos destinados a uso particular (hogares) y empresarial, ha tenido un pico cerca del 2008 a nivel general, que coincide con el periodo previo a la crisis inmobiliaria de ese mismo año, puesto que la concesión de crédito era relativamente laxa. Posteriormente, existe una tendencia general a la baja en el volumen de préstamos a raíz de la crisis financiera global, que provocó una restricción del crédito y en general los bancos se volvieron más aversos al riesgo. Esta conducta se alinea también con las políticas de austeridad y las reformas financieras que se implementaron en España para estabilizar la economía, que las más significativas fueron:

- **Reestructuración bancaria**: Amplia reestructuración que incluyó la fusión de cajas de ahorros y la creación de la SAREB, la Sociedad de Gestión de Activos procedentes de la Reestructuración Bancaria, que en pocas palabras, era una entidad financiera dedicada a gestionar los activos tóxicos.
- **Reformas laborales y fiscales**: Se implementaron reformas para flexibilizar el mercado de trabajo, y demás ajustes fiscales para reducir el déficit público.
- **Medidas de austeridad**: Recortes en gasto público, también para reducir el déficit.
- **Ayudas al sector financiero**: Se concedieron ayudas a entidades financieras para evitar su quiebra y proteger los ahorros de las familias.

Después de este periodo, observamos una recuperación gradual y una mejora de la confianza en el sector financiero. Aun así, con la reciente crisis del COVID-19 y las materias primas a causa del conflicto entre Rusia y Ucrania, vemos como esta tendencia a la baja se frena y se estabiliza, ya que las familias y empresas vuelven a precisar de financiación para afrontar la pandemia y los efectos que ha tenido en la economía (con un descenso del PIB, dificultades en el mercado de trabajo, en la productividad de las empresas, ... (Torres & Fernández)). No obstante, la tasa de morosidad no ha aumentado gracias a medidas como los ERTE, moratorias y avales públicos por parte del Estado que han ayudado a mitigar los efectos. Pero como no todo lo que reluce es oro, que estas medidas hayan conseguido una estabilidad en la morosidad española no significa que a nivel agregado no se tengan en cuenta las diferencias entre sectores (ya que el COVID ha afectado más gravemente a algunos sectores que a otros), ni tampoco se valora la disparidad en función del destino del crédito.

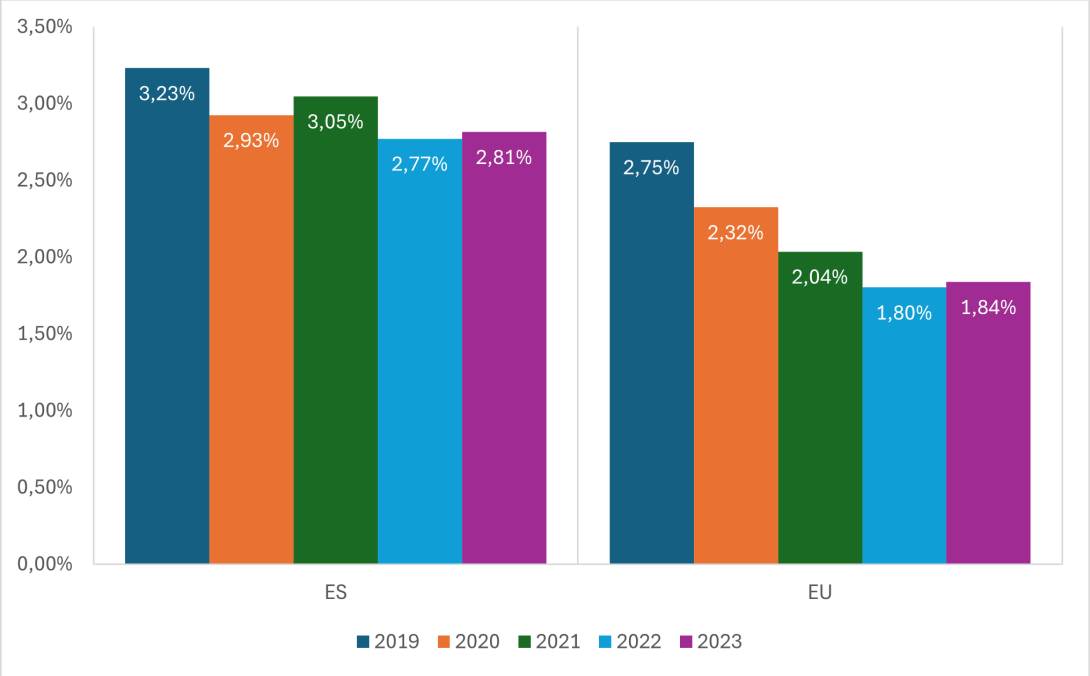


Figura 1.3: Tasa de morosidad de 2019 a 2023 en España y la Unión Europea

Fuente: European Banking Authority, EBA

Como vemos en el gráfico, la tasa de morosidad ha presentado una tendencia negativa desde antes del COVID. Además, en comparación a la media europea, España tiene de media casi 1 pp, presentando la menor diferencia en 2019 con 0.48 pp y la mayor en 2021 donde alcanza los 1.01 pp, estabilizándose alrededor de este valor los siguientes años.

No obstante, la tendencia negativa se estabiliza en 2023, llegando a crecer 0.4pp. Esto puede ser explicado por el lento crecimiento de la economía tras las recesiones, además que se trata de un país bastante expuesto a los cambios del tipo de interés, ya que gran parte de las hipotecas son a tipo variable (es decir, las subidas recientes de los tipos de interés se han traducido en una carga financiera mayor para los españoles).

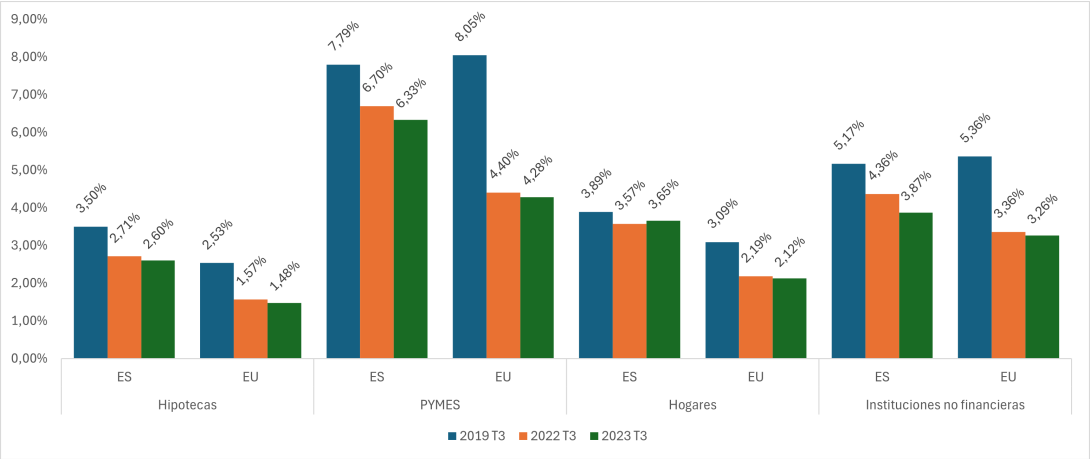


Figura 1.4: Tasa de morosidad de 2019 a 2023 en España y la Unión Europea por destino

Fuente: European Banking Authority, EBA

Si analizamos la tasa de morosidad según el destino del préstamo, vemos como presenta valores muy elevados en los préstamos otorgados a pequeñas y medianas empresas, doblando prácticamente la tasa de morosidad en hogares o destinados a hipotecas, y esto se puede explicar fácilmente, ya que las PYMES se enfrentan a riesgos comerciales (como los cambios en la economía, la demanda del mercado, la competencia...) y sufren más que los hogares durante las recesiones debido a la disminución de ingresos o la falta de liquidez. Además, tienen una dependencia del crédito superior a los hogares, puesto que dependen de los préstamos para expandirse o invertir, junto con mayores fluctuaciones de sus ingresos y gatos, lo que en conjunto puede afectar su solvencia. Volvemos a ver una vez más que a pesar de la pandemia, las tasas han disminuido tanto en España como en Europa.

Centrándonos más en las empresas, observemos la tasa de morosidad por tipo de actividad comercial de 2019 a 2023:

	Tercer trimestre de 2019		Segundo trimestre de 2022		Tercer trimestre de 2023		Variación T32019 - T22022		Variación T22022-T32023	
	ES	EU	ES	EU	ES	EU	ES	EU	ES	EU
A Agricultura, silvicultura y pesca	5,50%	7,14%	5,00%	3,97%	5,07%	4,00%	-0,50%	-3,17%	0,07%	0,02%
B Explotación de minas y canteras	3,20%	6,53%	2,57%	5,52%	0,66%	3,91%	-0,63%	-1,01%	-1,91%	-1,61%
C Industria manufacturera	4,83%	5,64%	3,42%	3,63%	3,15%	3,52%	-1,41%	-2,01%	-0,27%	-0,11%
D Suministro de energía eléctrica, gas, vapor y aire acondicionado	3,26%	2,67%	2,89%	1,39%	2,40%	1,56%	-0,37%	-1,28%	-0,49%	0,17%
E Suministro de agua y actividades de saneamiento, gestión de residuos y descontaminación	2,57%	3,35%	2,33%	1,69%	1,52%	1,24%	-0,25%	-1,66%	-0,80%	-0,45%
F Construcción	9,91%	14,11%	7,47%	6,75%	6,22%	5,99%	-2,44%	-7,36%	-1,24%	-0,76%
G Comercio al por mayor y al por menor	5,76%	6,12%	4,66%	3,87%	4,52%	3,90%	-1,10%	-2,25%	-0,14%	0,03%
H Transporte y almacenamiento	4,09%	6,61%	4,39%	4,20%	3,99%	3,38%	0,30%	-2,41%	-0,40%	-0,81%
I Hostelería	5,15%	8,74%	9,26%	8,67%	7,18%	7,37%	4,11%	-0,07%	-2,08%	-1,29%
J Información y telecomunicaciones	2,62%	3,23%	1,94%	1,97%	1,70%	2,06%	-0,68%	-1,26%	-0,24%	0,09%
K Actividades financieras y de seguros	2,65%	3,18%	1,40%	2,08%	2,22%	2,22%	-1,24%	-1,10%	0,81%	0,14%
L Actividades inmobiliarias	4,61%	3,25%	3,77%	1,96%	3,08%	2,17%	-0,85%	-1,29%	-0,69%	0,20%
M Actividades profesionales, científicas y técnicas	5,29%	4,60%	6,35%	3,57%	4,34%	3,15%	1,05%	-1,03%	-2,00%	-0,42%
N Actividades administrativas y servicios auxiliares	3,78%	3,17%	3,91%	3,26%	3,47%	2,95%	0,12%	0,09%	-0,44%	-0,31%
O Administración Pública y defensa; SS obligatoria	0,82%	1,12%	0,52%	0,67%	0,43%	0,94%	-0,29%	-0,45%	-0,10%	0,27%
P Educación	6,43%	4,24%	5,53%	3,56%	4,44%	3,48%	-0,90%	-0,68%	-1,09%	-0,08%
Q Actividades sanitarias y de servicios sociales	3,20%	3,18%	3,11%	2,22%	4,47%	5,59%	-0,09%	-0,96%	1,36%	3,38%
R Actividades artísticas, recreativas y de entretenimiento	7,12%	7,47%	14,75%	7,16%	8,91%	5,59%	7,64%	-0,31%	-5,85%	-1,57%
S Otros servicios	3,98%	4,96%	2,60%	3,00%	2,94%	2,77%	-1,38%	-1,95%	0,34%	-0,23%

Figura 1.5: Tasa de morosidad por tipo de actividad laboral

Fuente: European Banking Authority, EBA

Por lo general podemos observar una tendencia decreciente en todos las ramas especificadas, aunque tal y como hemos visto en el gráfico anterior, podemos encontrar casos en los que, a pesar de que la tasa haya disminuido de antes de la pandemia al 2022, repuntan ligeramente durante 2023. Ahora bien, hay actividades que han experimentado un aumento considerable. Por un lado, tenemos la hostelería española, con un aumento del 5.15 % al 9.26 %. Esto viene causado por las restricciones durante la pandemia, conduciendo así a una disminución de ingresos (y además, de entrada, el sector hostelero presenta una alta competencia y márgenes bastante ajustados). Por otro lado, las actividades artísticas presentan las tasas más altas tanto en 2022 como en 2023, siendo la variación más alta entre 2019 y 2022 de 7.64 pp, alcanzando el 14.75 %. En ambas actividades, aunque presenten

aumentos considerables, observamos que en 2023 disminuyen 2 pp y 6 pp respectivamente. En general, la pandemia ha afectado las actividades de ocio, la hostelería, el transporte y almacenamiento. Destaca el hecho de que en las actividades comentadas durante la pandemia, a pesar del aumento en España, a nivel europeo se ha dado una situación contraria, disminuyéndose así la tasa de morosidad tanto en el sector hostelero como en las actividades artísticas. Esto puede ser debido a que las restricciones relacionadas con el COVID no se aplicaron uniformemente en toda Europa, así como otras variantes como la estructura del sector.

1.4.1.1. Causas de la morosidad

La morosidad se puede dar por varios casos, ya sea por negligencia, intencionadamente, o circunstancialmente, es decir, el prestatario tiene intención de cumplir sus obligaciones, pero dadas ciertas circunstancias no puede hacerlo en el momento pactado. En pocas palabras, la capacidad de hacer frente a sus deudas determinará la solvencia de un particular o empresa. Pero, ¿cuáles son las causas de que no se pague a tiempo (o directamente, no se pague)? Podemos diferenciar según a quién las imputamos; por ejemplo, se puede dar una situación de morosidad si la actividad de la empresa acreedora provoca una reacción negativa en el prestatario, traducida en una negación al pago de facturas, como la calidad o el servicio. Por otro lado, hay causas que se pueden imputar a terceros, como a una agencia de transportes, por ejemplo. Ahora bien, las que me interesa para este análisis son aquellas relacionadas con el deudor, ya que el objetivo de esta tesis es identificar a deudores basándome en sus características. Aquí me gustaría tener en cuenta tanto a particulares como a profesionales, ya que cada grupo puede presentar unas causas diferentes.

Por un lado, las personas físicas supondremos que son prestatarios que financian compras de uso doméstico, como podría ser una casa, un coche, unas vacaciones o un teléfono móvil. Las causas que pueden perjudicar su solvencia (o su riesgo de morosidad) tienen relación directa en su mayoría con su fuente de ingresos:

- **Endeudarse por encima de sus posibilidades**, y al darse la situación donde gastas más que ingresas desemboca en no poder afrontar tus compromisos de pago.
- Entrar en situación de **desempleo** o situaciones parecidas en las que tus ingresos disminuyen (enfermedad o motivos de salud, motivos familiares, etc.).

Por otro lado, valoramos el punto de vista de profesionales, autónomos y empresas:

- Inversiones ambiciosas y **sobreendeudamiento**.
- **Gastos demasiado elevados** sin dejar margen a cumplir las obligaciones con los acreedores.
- Otros **motivos coyunturales** o relacionados con la salud, entre otros, que impidan llevar a cabo la actividad económica con normalidad.

De todas las causas, estas últimas son relativas al aspecto microeconómico de la sociedad, ya que son situaciones relativas a la toma de decisiones tanto de particulares como de las empresas. Estas decisiones pueden ser conducidas por varios factores, como por ejemplo:

- **Ingresos del deudor**: La caída de los ingresos (tanto de un particular como de una empresa) es una de las causas principales que provocan la dificultad o imposibilidad de hacer frente a las obligaciones de pago. Los cambios inesperados en la economía como la reducción de empleo o cambio de las condiciones laborales (a peor) perjudican la capacidad de pago de las personas y empresas. Tanto es así que los deudores con ingresos variables o inestables son particularmente vulnerables a caer en esta situación.
- **Gestión financiera**: Los problemas derivados de la planificación financiera así como el control de las deudas pueden llevar a una situación de morosidad o insolvencia, principalmente provocados por falta de conocimientos financieros («Estado Actual de la Cultura Financiera en España») o una gestión ineficaz de los recursos disponibles (falta de ahorro, excesivo crédito, acumulación de deudas...). No hemos recibido instrucción formal sobre cómo pagar un recibo, gestionar nuestras deudas o invertir nuestros ahorros. Además, no solemos preocuparnos por comprender el funcionamiento de los intereses que pagamos ni somos capaces de analizar la cantidad de deuda que podemos soportar. (Pérez)
- **Condiciones del préstamo**: Si las condiciones de un préstamo no son favorables, los deudores pueden encontrar difícil cumplir con los términos y condiciones establecidos.

Por otro lado, podemos identificar causas a nivel macroeconómico, ya que son provocadas por variables a nivel nacional o gubernamental, y tienen un impacto en todo el país. Entre

ellas, podemos hablar de («La morosidad del crédito bancario en España») (Rego) («La morosidad en España en 2022»):

- **Coyuntura económica**: La morosidad está afectada por la situación económica general de un país, ya que durante periodos de recesión o crisis, la capacidad de pago o solvencia de los deudores es perjudicada notoriamente. Un aumento de la tasa de paro reduce los ingresos de las familias, por lo que su consumo cae, así como se eleva la morosidad frente a las obligaciones.
- **Inflación**: La subida de los precios erosiona el poder adquisitivo de los deudores, por lo que si sus ingresos no se ajustan correctamente, puede provocar que, una vez los precios de los bienes y servicios aumentan, la cantidad de dinero disponible para el pago de deuda se vea reducido.
- **Política monetaria**: Los cambios en los tipos de interés pueden tener un impacto significativo en la morosidad, ya que un aumento de estos encarecen el costo de los préstamos, lo que puede provocar que las familias tengan dificultades a la hora de pagar.
- **Regulación financiera**: Las autoridades financieras pueden tomar decisiones que influyan en la morosidad, como por ejemplo el determinar condiciones de concesión laxas (como se dio en la crisis financiera del 2008), pueden dar acceso a financiación a particulares que tengan un elevado riesgo.

Veamos la siguiente matriz de correlaciones entre las variables **IPC** (representando la inflación), el **PIB** (la producción) y la **ratio de morosidad**. Se han utilizado datos entre 2002 y 2021 extraídos de la base de datos del Banco de España:

	Ratio de morosidad	**IPC**	**PIB**
Ratio de morosidad	1.0000000		
IPC	0.7265040	1.0000000	
PIB	-0.1147011	-0.2882708	1.0000000

Cuadro 1.1: Matriz de correlaciones entre las variables PIB, IPC y el ratio de morosidad

Fuente: Banco de España

Como podemos observar, los resultados verifican las premisas anteriores: el ratio de morosidad y el IPC están relacionados de manera positiva, implicando así una aumento de la morosidad en ciclos inflacionarios. Por otro lado, la relación entre el ratio de morosidad

y el PIB es algo más débil, pero podemos concluir lo mismo: están relacionados negativamente y una disminución de la producción de un país provoca un aumento en la tasa de morosidad. Aunque no pueda calcularlo debido a que no hay datos suficientes, un artículo de Botta et al., 2018 indica que la correlación podría ser incluso más fuerte cuando se usan valores retardados.

1.4.1.2. Efectos de la morosidad

Para valorar los efectos de la morosidad, podemos diferir entre los efectos macroeconómicos y posteriormente estudiar los efectos en el sector financiero y los hogares.

A nivel macroeconómico, la morosidad afecta negativamente de varias maneras. Por un lado, puede conducir a una reducción de la oferta de crédito, lo que provoca que las empresas encuentren dificultades para financiar sus proyectos y expansión, afectando así a la inversión y el consumo; en resumen, frena el crecimiento económico. Además, si los inversores perciben un riesgo de impago suficientemente alto, puede llevar una fuga de capitales y a una mayor volatilidad de los mercados financieros.

Por otro lado, aunque profundizaré más adelante, una morosidad elevada puede afectar la solidez de las instituciones financieras aumentando el riesgo de insolvencia y crisis bancarias, lo que puede afectar a la economía general (Solutions, 2009); el gobierno puede verse obligado a intervenir si la situación en el sistema financiero es crítica, lo que puede resultar en un aumento del gasto público y presión sobre las finanzas públicas, y en el hipotético caso de que se tuviera la necesidad de imprimir dinero, esto podría conducir a una inflación.

De hecho, las instituciones financieras cuando tienen una proporción considerable de préstamos incobrables pueden ser menos propensos a prestar, lo que hace menos efectivas las medidas de estímulo monetario (como una disminución de los tipos de interés); Las empresas, ante la restricción del crédito, pueden verse forzadas a reducir o incluso cesar su actividad, lo que aumentaría la tasa de desempleo.

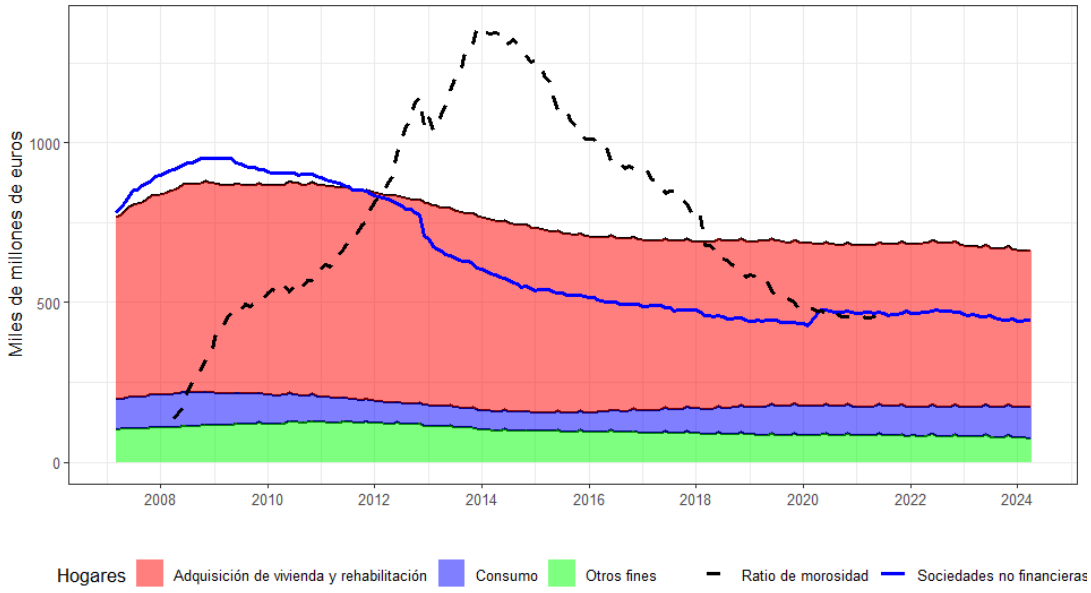


Figura 1.6: Volumen de créditos junto a la tasa de morosidad en los últimos años

Fuente: Banco de España

Como vemos en el gráfico, a medida que la tasa de morosidad aumenta, el volumen de créditos concedidos disminuye.

Respecto a las entidades financieras, el impacto que pueda llegar a alcanzar la morosidad de los clientes puede ser profundo y multifacético. Aparte de los efectos que he comentado anteriormente (como la reducción de la concesión de créditos), hay otras consecuencias a nivel microeconómico.

Para empezar, las entidades deben de apartar una cantidad significativa de dinero para cubrir las posibles pérdidas que puedan tener a causa de los impagos, hecho que reduce la rentabilidad (ya que esos fondos se podrían utilizar para generar rendimientos). Esta tarea puede ser especialmente desafiante en épocas de recesión económica.

Por otro lado, la liquidez de las entidades financieras se ve afectada negativamente, puesto que los ingresos que se habían previsto no llegarán (o se demoran), lo que provocará que tengan dificultades no solo para otorgar nuevos créditos, sino que puede llegar a tal punto de que se presenten dificultades para cumplir sus propias obligaciones financieras.

Según el «Observatorio de Morosidad 2T 2021» de la **Confederación Española de la Pequeña y Mediana Empresa** (CEPYME), el coste de oportunidad financiero de la morosidad empresarial alcanzó los 1.472 millones de euros durante el primer trimestre de 2021, lo que supone un aumento interanual del 3.5 %.

	1T 2020	1T 2021	VAR 20-21
Deuda comercial	350.547	351.610	0,3%
Coste financiero de la deuda comercial	2.510	2.537	1,1%
Deuda comercial con retraso de pago	239.420	241.577	0,9%
Intereses de demora exigibles por retraso de pago	1.422	1.472	3,5%

Figura 1.7: Coste financiero de la morosidad (millones de euros y % var.)

Fuente: CEPYME-Afi a partir del BdE, CESCE e Informa

Por otra parte, la percepción de una elevada tasa de morosidad puede afectar negativamente la reputación de una entidad financiera y afectar la confianza de los inversores y depositantes. Esto podría generar una fuga de capitales y obstaculizar la captación de nuevos fondos. A su vez, deteriora la calidad del crédito, hecho que puede afectar negativamente su calificación crediticia y aumentar el costo de financiamiento.

Por último, y no menos importante, la gestión de los créditos morosos puede implicar costos operativos adicionales.

1.4.2. Métodos tradicionales en la detección de la morosidad

El análisis de crédito implica una variedad de procedimientos por parte del banco, el cual les permite evaluar la capacidad del prestatario de cumplir sus obligaciones financieras. Este proceso en esencial, ya que no solo ayuda a gestionar riesgos (y evitar las posibles pérdidas a raíz de la morosidad), sino que también es crucial a la hora de tomar decisiones. Dividiremos el análisis crediticio en dos grandes grupos según las variables que estudian (AnalystPrep):

- **Análisis cualitativo**: Evalúa aspectos cualitativos del solicitante y de su entorno.
  - **Características del solicitante**: Entre ellas podemos encontrar algunas financieras como su historial de crédito y referencias personales de otros acreedores (para conocer su comportamiento en otras situaciones similares como prestatario) y otras características personales y contextuales como su estilo de

vida o educación, o la estabilidad laboral que presenta.

- **Factores externos**: Factores que presenta la economía a nivel general, como la tasa de desempleo o inflación, factores que tienen una fuerte correlación con la solvencia de los individuos (Solutions, 2009).

- **Análisis cuantitativo**: Por otro lado, este análisis utiliza información numérica del individuo para evaluar la solvencia. Utiliza valores como los ingresos, los ahorros, los activos, e inclusive *scorescards*[1] de crédito (una puntuación que refleja el historial crediticio y el comportamiento del solicitante).

Hoy en día, los bancos tienden a preferir la implementación de modelos mixtos que combinan ambos tipos de análisis, como las conocidas 5 C's (Segal):

- **Carácter (*Character*)**: Analizan, en pocas palabras, la reputación del prestatario mediante historiales de pago, informes laborales, entre otras referencias.
- **Capacidad (*Capacity*)**: Describe si el prestatario es capaz de, con sus ingresos, hacer frente a sus obligaciones de pago, y se evalúa mediante el análisis de sus ingresos, gastos y deudas.
- **Capital (*Capital*)**: Reúne los activos y ahorros que el prestatario puede poner a disposición. Por ejemplo, cuanto mayor sea la entrada en una hipoteca, menor riesgo correrá el prestamista.
- **Condiciones (*Conditions*)**: Incluye los aspectos relacionados con el préstamo, teniendo en cuenta tanto el propósito del crédito, las tasas de interés, estabilidad del mercado, y otros factores macroeconómicos.
- **Colateral (*Collateral*)**: Reúne los activos que el prestatario ofrece como garantía del préstamo; un aval.

[1] Un *scorecard* es un modelo basado en datos sobre características del usuario, para predecir las probabilidades de incumplimiento de la obligación (Bequé et al., 2017)

Podemos intuir que estos análisis implican un elevado volumen de datos, y trabajar con estos de manera manual y no sistematizada puede llevar a aumentar tanto el costo como el tiempo relacionados con la concesión de créditos. La implementación de algoritmos de *machine learning* en el sector financiero han permitido trabajar con bases de datos masivas, además de tener en cuenta muchas más variables para poder extraer conclusiones más precisas y con una capacidad predictora más sólida. En el siguiente capítulo, nos adentraremos en entender qué es el *machine learning* y cómo funcionan los principales algoritmos que usaremos en nuestro análisis.

Capítulo 2

Uso de aprendizaje automático para detectar posibles casos de morosidad

2.1. Machine Learning en el sector financiero

El aprendizaje automático (o *machine learning*) está transformando el mundo de las finanzas al introducir nuevas formas de analizar datos, predecir comportamientos y automatizar procesos. Esta tecnología permite a las instituciones financieras procesar grandes cantidades de datos de forma rápida y precisa, mejorando la toma de decisiones y la gestión de riesgos. Por ejemplo, en el análisis crediticio, permite a los bancos utilizar una amplia gama de variables e información histórica para evaluar la solvencia de un prestatario con una precisión sin precedentes; Los modelos predictivos pueden identificar a los solicitantes que tienen más probabilidades de incumplir, ayudando a las instituciones a gestionar mejor el riesgo y diseñar productos financieros adaptados a las necesidades específicas de cada cliente. Es necesario entonces entender cómo funciona el *machine learning* y qué abanico de posibilidades nos ofrece.

2.1.1. ¿Qué es el Machine Learning?

El aprendizaje automático o *machine learning* es una rama de la inteligencia artificial[1] y se encarga de estudiar algoritmos y técnicas con el fin de automatizar soluciones a problemas complejos que las metodologías tradicionales no podrían llevar a cabo (Rebala et al., 2019). Los algoritmos de aprendizaje automático permiten a los ordenadores aprender a realizar tareas sin estar programadas para hacerlo; en cambio, utilizan los datos proporcionados para identificar patrones de conocimiento, lo que ayuda a la toma de decisiones (Gutpa). Estos algoritmos de *machine learning* usan una metodología la cual mejora a medida que se añadan al proceso nuevos datos, dando pie a un aprendizaje más robusto cuyas conclusiones serán más precisas cuanto mayor sea el volumen de información con la que se trabaja.

Esto último nos puede dar a entender el porqué de su auge en los últimos años: Las tecnologías recientes han permitido reunir y manejar cantidades desmesuradas de datos, así como mejoras sustanciales en la potencia computacional. Gracias a lo mencionado, empresas que han ido acumulando una gran cantidad de datos a lo largo del tiempo (la cual nos podemos referir a ella como *Big Data*) pueden usarlos para tomar decisiones inteligentes en su negocio, así como hacer sugerencias más personalizadas a sus clientes, predecir tendencias, y encontrar patrones que puedan ayudar a su forma de proceder como empresa (Mani).

El siguiente ejemplo relacionado a *tech companies* nos será útil para complementar la explicación de la magnitud de las aportaciones que nos da el análisis de una gran cantidad de datos: grandes empresas tecnológicas como Amazon, Facebook, Spotify o Netflix, descubren gracias al aprendizaje automático qué es lo que realmente buscan sus usuarios, desde añadir nuevas funciones. Según un artículo («How Does Netflix Use Machine Learning»), Netflix utiliza el *machine learning* por lo menos para 5 diferentes objetivos: recomendaciones más precisas de contenido para cada usuario, generación automática de miniaturas para resumir de manera óptima una serie o película, mejorar la calidad del *streaming* mediante técnicas que reducen el *buffering*, mejorar la calidad del contenido (ya sea la interfaz de usuario o producciones originales), y finalmente, les ayuda a seleccionar lugares para grabar las series o películas de las cuales Netflix es productora.

[1] «Disciplina científica que se ocupa de crear programas informáticos que ejecutan operaciones comparables a las que realiza la mente humana, como el aprendizaje o el razonamiento lógico.» (Española)

Podemos resumir un proyecto de *machine learning* en las siguientes fases («The Machine Learning Life Cycle Explained»):

Definición del problema

En primer lugar, es esencial saber qué objetivos perseguimos o a qué problemática nos enfrentamos. Dentro de esta planificación, debemos de valorar la disponibilidad de los datos, los recursos de los que disponemos, y otros aspectos a considerar como alternativas menos costosas o sencillas, o identificar las limitaciones legales a la hora de recoger datos, aplicar resultados, entre otros.

Recolección de los datos

En esta fase, debemos de reunir los datos que creemos necesarios para el proyecto. La precisión de nuestros resultados serán mayores como más volumen, calidad, variedad tengan nuestros datos («The Four V's of Big Data»); es muy importante poder asegurar hasta cierta medida unos buenos datos, ya que es la materia prima de nuestro proyecto. Por el contrario, las conclusiones que extraigamos pueden ser erróneas y poco robustas. Un ejemplo muy trivial, si quisiéramos ejecutar un algoritmo para identificar correos *spam*, sería alimentarlo con correos que efectivamente lo sean y que sean representativos de los correos que suelen ser no deseados; identificar palabras que se repitan como 'compra', u 'oferta', entre otras («How To Design A Spam Filtering System with Machine Learning Algorithm»).

Preparación de los datos o *preprocessing*

Los datos que hemos recolectado pueden presentarse en formatos dispares, y/o con registros que pueden ser problemáticos para nuestro análisis, ya sean datos erróneos debido a problemas en su observación, o datos extremos u *outliers* que pueden influenciar más de lo debido en nuestro resultado. En este proceso llevamos a cabo una limpieza de datos erróneos o inconsistentes (valores faltantes, duplicados, etc.), codificación (algunos algoritmos no trabajan con variables categóricas, por lo que es necesario transformarlas en valores numéricos), e incluso se puede reducir la dimensionalidad siempre que el volumen de datos sea demasiado alto por lo que se escogen las variables más importantes. De todos modos, profundizaremos más adelante con esta fase, ya que puede es muy importante y

puede llegar a ocupar el 80 % de todo un proyecto [2] [3] [4].

Análisis preliminar de las variables

A continuación, es interesante visualizar nuestros datos mediante gráficos bivariantes y univariantes. Esto nos puede resultar muy útil, ya que ante grandes volúmenes de datos, podemos explorar nuestros datos y encontrar comportamientos interesantes para nuestro proyecto, así como encontrar relaciones entre nuestra variable respuesta y las explicativas. Es muy común utilizar diagramas de barras, de área, gráficos de caja (*boxplots*), histogramas, mapas de calor, en análisis de texto usar nubes de palabras, etc.

[2] Wikipedia
[3] «¿Qué es el preprocesamiento de datos? Definición, importancia y pasos»
[4] «¿Qué es el Preprocesamiento de Datos (Data Preprocessing)?»

Especificar nuestros datos de entrenamiento y de prueba

Antes de empezar a ejecutar nuestros modelos de aprendizaje automático, debemos de generar particiones en nuestros datos («Training data vs Testing data») («Conjuntos de entrenamiento y prueba: división de datos»), y dedicar una parte al entrenamiento del modelo (es decir, para que el algoritmo estudie y aprenda el comportamiento de los datos, identificando patrones y relaciones) y otra parte a la prueba (es decir, con lo aprendido en la anterior fase de entrenamiento, el modelo ahora deberá de ponerse a prueba para ver que tan bien funciona). Esto se hace principalmente para evitar el sobreajuste; el modelo puede aprender demasiado sobre los datos, lo cual hará una tarea difícil implementar el algoritmo en nuevos datos (aunque esta situación también se puede dar si los datos de entrenamiento no representan bien a la población). Esta técnica que divide los datos se llama validación cruzada (o *cross-validation*), y podemos considerar diferentes variantes (Wikipedia):

- **Validación *holdout***: Esta variante es la más simple y la que ya hemos mencionado previamente. Simplemente, se divide la muestra en dos subconjuntos, normalmente usando una proporción de entrenamiento - prueba de 80 %-20 % o similares, siempre ajustándonos a la cantidad y tipo de datos que tenemos. La principal ventaja de este método es su rapidez, aunque no es muy preciso debido a la variación que presenta entre conjuntos de datos.

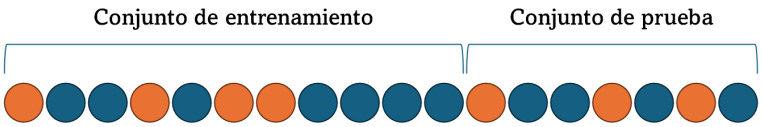


Figura 2.1: Ejemplo gráfico del método *holdout*.

- **Validación cruzada de K iteraciones**: En este caso, los datos se dividen en las proporciones indicadas en K grupos. Para cada una de las K iteraciones, se escoge un grupo de prueba y el resto de entrenamiento ($K-1$ grupos). Finalmente, se hace una media aritmética de los K resultados que obtenemos. A diferencia del anterior método, es mucho más preciso, pero el proceso es computacionalmente más exigente.

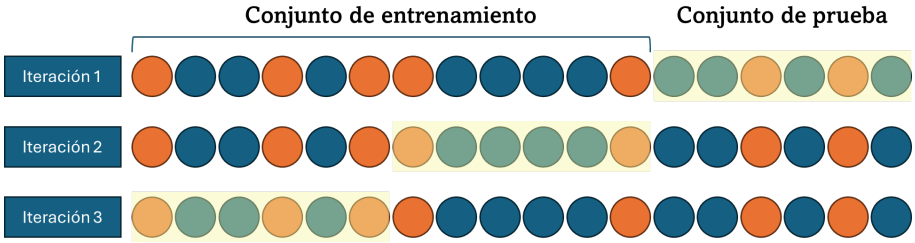


Figura 2.2: Ejemplo gráfico del método *kfold* usando $k = 3$.

- **Validación cruzada aleatoria**: En esta variante, aunque presenta parecido con la anterior, se diferencia en que el subconjunto de entrenamiento se escoge al azar en cada iteración. Esto, por un lado, nos permite no tener que hacer tantas iteraciones como pliegues (o *folds*) tengamos, pero nos comporta la desventaja de que puede haber individuos que se pueden quedar sin evaluar, o de la misma manera, que haya muestras que se "sobreevaluen".

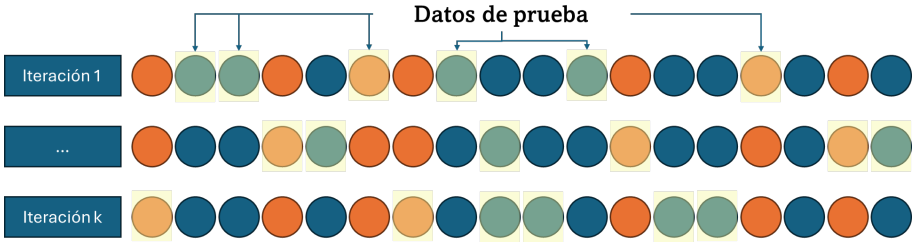


Figura 2.3: Ejemplo gráfico del método *random cross-validation*.

- **Validación cruzada dejando uno fuera**: En esta variante habrá tantas iteraciones como datos en la muestra. En cada iteración, se escoge un dato como dato de prueba, y se deja el resto como subconjunto de entrenamiento. Aunque presenta un error mucho más bajo que el resto, se puede ver claramente como supone un coste exagerado computacionalmente (haciéndolo prácticamente inviable con bases de datos grandes).

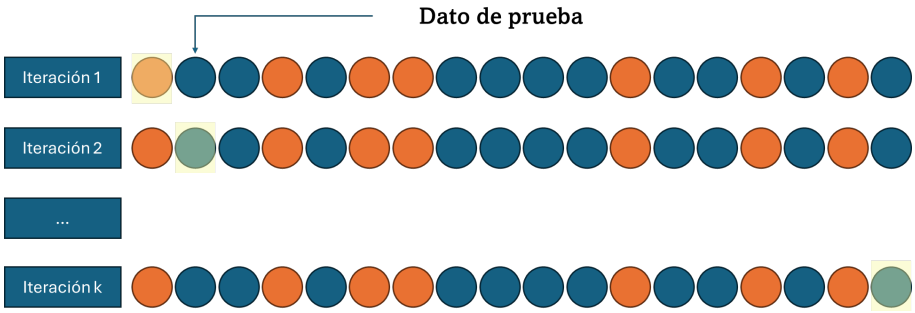


Figura 2.4: Ejemplo gráfico del método *Leave-one-out cross-validation (LOOCV)*.

Este será la estructura del código que utilizaré para plantear los pliegues o *folds*.

```
#Entrenar el modelo usando tecnicas de validacion cruzada con k pliegues e i
↪  repeticiones
ctrl <- trainControl(method = "repeatedcv", number = k ,repeats = i)
```

Como veremos más adelante, este es un punto clave cuando nos enfrentamos a un conjunto de datos desbalanceados, es decir, cuando un grupo está mucho más representado que el otro, ya que la forma en la que trabajemos condicionará nuestros resultados. Veremos en más detalle qué problemas generan los datos desbalanceados y cuáles son las maneras más comunes de trabajar con ellos.

Fase de entrenamiento

Una vez llegados a este punto, deberemos de entrenar nuestro programa mediante los algoritmos que seleccionemos. La elección del algoritmo no es una decisión sencilla, ya que dependerá del tipo y cantidad de datos que estemos usando, el objetivo que deseamos y el contexto (Metwalli). Como veremos en la parte práctica, hay algoritmos que frente a tipos de datos concretos son mejores con otros; la elección del algoritmo influye significativamente en la precisión de los resultados.
El punto clave del aprendizaje automático es, valga la redundancia, la capacidad de los algoritmos de ganar experiencia y aprender de manera automática y autónoma con la adición progresiva de datos nuevos.

Validación y despliegue

Llegamos a la última fase, donde valoraremos qué tan bien lo ha hecho nuestra máquina. Para ello, usaremos unas métricas de validación que nos ayudarán a medir de manera precisa la capacidad del programa para clasificar. Dependiendo qué objetivo tengamos o con qué datos estemos trabajando, usaremos unas métricas u otras, las cuales, profundizaremos en un apartado posterior, ya que es conveniente entender como funciona cada una de ellas. Si consideramos que está validado, es decir, cumple nuestras expectativas (o se acerca a ellas) y el objetivo, podemos valorar lanzarlo y usarlo en casos prácticos reales (si seguimos el ejemplo de los correos spam, una vez desarrollemos un buen programa de clasificación, podemos implementarlo en nuestro correo electrónico). En caso contrario, debemos de volver al paso anterior y seguir trabajando.

2.2. Preparación de los datos o *preprocessing*

Como he mencionado brevemente, para poder confiar en obtener unos resultados robustos y de calidad, debemos de trabajar con datos adecuadamente preparados y transformados para nuestro análisis. Como hemos visto, esta parte supone el mayor tiempo de un proyecto, llegando a alcanzar hasta el 80 % («Data preparation part in data mining projects»).

En primer lugar, después de echar un vistazo general a la base de datos, tenemos que especificar la población a estudiar, o dicho de otra manera, seleccionar las filas de la matriz que consideremos adecuadas. De la misma manera, también deberemos decidir qué variables (o columnas) seleccionar, utilizando criterios de exclusión o inclusión, filtros, o decidiendo respecto variables externas como el coste de obtención de la información (Gibert et al., 2016). Además, debemos de tener cuidado a la hora de leer los datos con nuestro software escogido (en este análisis utilizaré principalmente R y Excel), comprobar que no ha habido errores de interpretación de variables, por ejemplo que se codifiquen variables categóricas a numéricas, que se lean variables numéricas en formato de texto, idioma estándar con caracteres que no sean problemáticos (como tildes o puntos), entre otros.

Habrá situaciones en las que tendremos celdas vacías en nuestra base de datos **(conocidos como *missings* o valores faltantes)**, y para solucionarlo deberemos de saber la naturaleza de estos («Valores perdidos: MCAR, MAR y MNAR»):

- Valores faltantes aleatorios (MAR & MCAR): Este tipo de *missings* son debidos a razones aleatorias, es decir, no siguen ningún patrón identificable (o al menos, no del todo como veremos). Por un lado, tenemos los *Missings Completely at Random* o valores faltantes completamente aleatorios. Supongamos que un virus infecta nuestra base de datos y destruye datos al azar; que haya un valor faltante no depende de ninguna variable observada. Por otro lado, tenemos los Missings at Random o valores faltantes aleatorios pueden estar condicionados a una variable observada. Por ejemplo, se pueden dar valores faltantes relacionados con el grupo de edad en el que está el sujeto, o relacionados con la especie de una planta; aun así, al dividir la muestra en esos grupos que causan esta "problemática", los valores faltantes en

cada grupo pasan a ser MCAR. Por suerte, no son problemáticos este tipo de valores faltantes, ya que se pueden imputar fácilmente siguiendo la distribución de los datos presentes.

- Valores faltantes NO aleatorios (MNAR): Son valores faltantes que siguen un patrón, que normalmente provienen de una parte concreta de la población de estudio. En este caso, son mucho más difíciles de imputar, ya que es difícil inducir la distribución que siguen respecto a los datos que hay presentes. Por ejemplo, un doctor, ante el gran volumen de pacientes, decide tomar la presión de solo aquellos que tengan una presión por encima de cierto valor. En este caso, imputarlos será una tarea mucho más difícil y delicada, por lo que hay que tener en cuenta el contexto, el objetivo, y el procedimiento a seguir.

Para saber si nuestros valores faltantes son aleatorios o no, utilizamos el test MCAR de Little (Gibert, 2009), que contrasta las siguientes hipótesis:

$$\begin{cases} H_0: & \text{Valores faltantes completamente aleatorios (MCAR)} \\ H_1: & \text{Valores faltantes NO aleatorios} \end{cases}$$

Una vez sepamos la naturaleza de nuestros *missings*, deberemos de imputarlos, es decir, rellenar las celdas vacías con la finalidad de completar la información, sin distorsionar las características de nuestra base de datos. Para ello, tenemos varias maneras de abordar el problema. Para las variables cualitativas lo más recomendable es crear una nueva categoría asociada a la falta de valor, por ejemplo «Unknown». En el caso de las variables numéricas, tenemos una amplia variedad de maneras (Gopalan) (Gibert et al., 2016):

- **Métodos básicos:** Usar un 0, la mediana global, o la mediana local por grupos de la variable.
- **Vecinos más próximos (KNN):** Con el método de *K-nearest neighbours* se imputan los valores faltantes de cada observación en relación con otras observaciones parecidas (o cercanas), usando la distancia euclidiana (u otras distancias entre puntos).

- **Imputación multivariable por ecuaciones encadenadas (MICE):** Este método más avanzado reemplaza los *missings* con valores basados en la distribución de máxima verosimilitud basados en métodos Montecarlo con cadenas de Markov (MCMC) [5] (Royston, s.f.) (Prabhakaran) . En primera instancia, reemplaza todos los valores faltantes por un reemplazo temporal como puede ser la media. A continuación, reemplazar de nuevo a faltante los valores de una de las columnas que presentaban valores faltantes, y se construye un modelo de regresión con el resto. Se repite para cada columna que presente *missings*, pero teniendo en cuenta los valores que se han imputado previamente con las regresiones. Esta iteración se repetirá tantas veces como queramos, ya que mejorará de manera sucesiva.

Para comprobar qué método ha sido efectuado con mayor éxito, podemos visualizar las densidades de las variables y compararla con la original; la más parecida será la mejor, porque habremos conseguido imputar los valores faltantes sin alterar apenas la distribución de los datos.

A continuación, debemos de **revisar la base de datos para encontrar *outliers* o valores extremos**, que son observaciones raras y/o fuera de rango, ya que estos pueden condicionar o sesgar nuestras conclusiones. Para identificarlos, podemos usar análisis estadísticos (como Rosner (Fissuh)) tanto para detectarlos como para identificar su influencia en la distribución de los datos (distancia de Cook o apalancamiento (Dhakal, 2017)). Por otro lado, también podemos visualizarlos de manera gráfica, usando histogramas, gráficos de puntos, diagramas de caja (hay que tener en cuenta que para estos últimos tenemos que asegurarnos que los datos siguen una distribución normal, o al menos, normalizar los datos previamente). Para tratarlos, debemos de tener en cuenta su origen, y a raíz de ahí, decidir si eliminamos la observación (mala codificación, entre otros) o la mantenemos o transformamos a valor faltante (registro extremo pero plausible, valor de una población diferente a la de estudio, entre otros). Es muy importante tomar esta decisión meticulosamente, ya que una mala praxis nos puede llevar a conclusiones erróneas.

[5] «En estadística, los métodos de Montecarlo basados en cadenas de Markov (MCMC por sus siglas en inglés, Markov chain Montecarlo) comprenden una clase de algoritmos para el muestreo de una distribución de probabilidad. Construyendo una cadena de Markov que tiene la distribución deseada como su distribución de equilibrio, se puede obtener una muestra de la distribución deseada registrando estados de la cadena» Wikipedia, 2024c

Este fragmento de (Moore & McCabe, 2005) sobre la detección de agujeros en la capa de ozono nos puede ayudar a entender la importancia del tema:

> «*In 1985 British scientists reported a hole in the ozone layer of the earth's atmosphere over the South Pole. This is disturbing, since ozone protects us from cancer-causing ultraviolet radiation. The British report was at first disredarded, since it was based on ground instruments looking up. More comprehensive observations from satellite instruments looking down had shown nothing unusual. Then examination of the satellite data revealed that the South Pole ozone readings were so low that the computer software used to analyze the data had automatically suppressed these values as erroneous outliers. Readings dating back to 1979 were reanalized and showed a large and growing hole in the ozone layer that is unexplained and possibily dangerous Computers analyzing large volumnes of data are often programmed to suppress outliers as protection againts errors in the data. As the example of the hole in the ozone layer illustrated, suppressing an outliers without inv estigating it can kleep valuable information out of the sight.*» [6]

Finalmente, seleccionamos las variables que consideramos más útiles para nuestro análisis. Para ello, podemos analizar la correlación entre la variable respuesta y las variables explicativas, y descartar aquellas que no presenten relación. También podemos analizar la correlación entre las variables explicativas, ya que podemos reducir la dimensionalidad de nuestra base de datos en caso de que dos o más variables estén muy correlacionadas (y por ende aportan la misma información), o crear nuevas variables artificiales que nos ayuden a interpretar nuestro análisis, ya sean de carácter binario, categórico, o numéricas.

Una vez finalizado el proceso de preprocesamiento de los datos, podemos ejecutar los algoritmos de clasificación que hemos seleccionado.

[6] El artículo de este suceso se reportó en el New York Times (Gleick, 1986)

2.3. Algoritmos de clasificación

En el aprendizaje supervisado encontramos dos grandes grupos de algoritmos («Algoritmos de aprendizaje automático»). Por un lado, están los algoritmos no supervisados, los cuales intentan agrupar o describir la estructura de los datos, por ejemplo, para encontrar patrones de comportamiento, o identificar grupos con individuos semejantes en un ecosistema. Un ejemplo puede ser identificar patrones entre clientes de un establecimiento, con la intención de ofrecer un servicio más personalizado.

Por otro lado, están los algoritmos supervisados, que a diferencia de los anteriores, estos están etiquetados y solemos encontrar una variable respuesta. Conociendo así el resultado, este tipo de algoritmos pueden elaborar predicciones basadas en información histórica. Los algoritmos de clasificación son uno de los métodos de aprendizaje supervisado, es decir, las predicciones se realizan a partir de datos históricos, con el objetivo de asignar cada observación a una clase u otra. En nuestro análisis relacionado con la morosidad utilizaremos principalmente este tipo, ya que en nuestra base de datos (la cual comentaré más adelante) cada observación está etiquetada según si es o no una persona morosa.

A continuación, introduciré brevemente los algoritmos que utilizaré en este análisis, así como también explicaré la estructura de las funciones en R que utilizaré (en teoría, si para mi análisis preciso de hacer algún cambio al código general, lo especificaré en su respectivo apartado).

2.3.1. Máquinas de Vectores de Soporte (SVM)

Las máquinas de vectores de soporte o su nombre en inglés *Support Vector Machines* pertenecen al grupo de los ya mencionados algoritmos de aprendizaje supervisado, desarrollado por Vladimir Vapnik y su equipo. El modus operandi de este algoritmo es encontrar el hiperplano que separe dos clases diferentes de puntos, maximizando el margen o distancia que hay desde el hiperplano hasta cualquiera de las dos clases. Este margen es la anchura máxima de la región paralela al hiperplano.

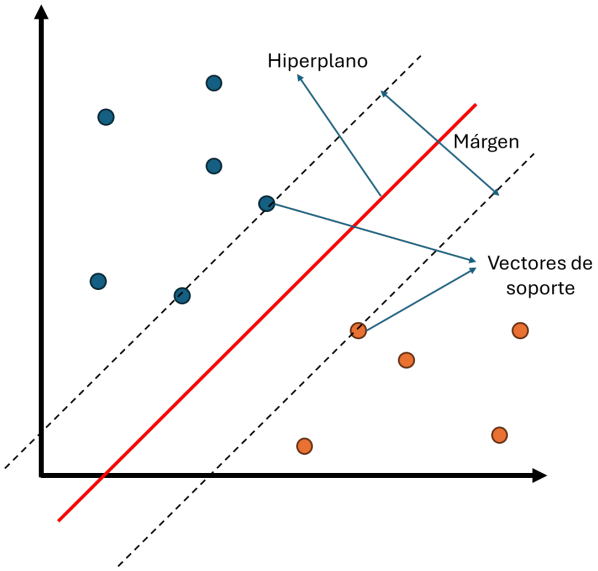


Figura 2.5: Ejemplo gráfico del funcionamiento de los *Sector Vector Machines*.

Normalmente, este algoritmo está enfocado a problemas de clasificación binaria, por lo que los problemas multiclase (más de dos clases) se reducen a una serie de problemas binarios. Además, los SVM pertenecen a una clase de algoritmos llamados *kernel*, es decir, utilizan una función para transformar las características de los datos, asignándolos a espacios dimensionales diferentes, con el objetivo de facilitar la clasificación.

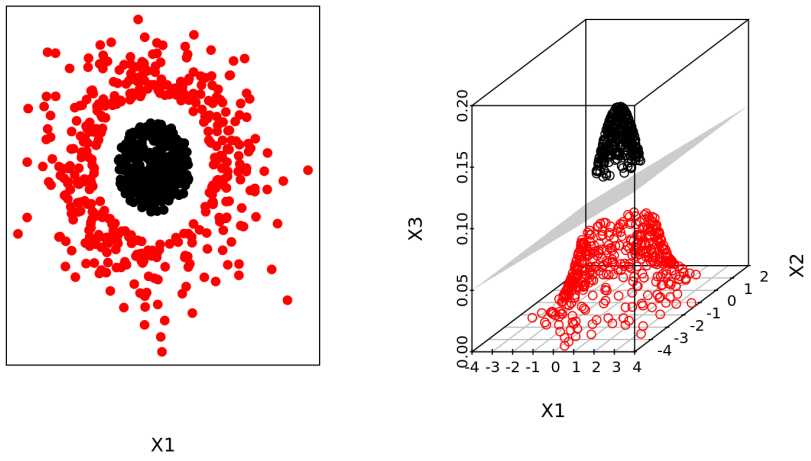


Figura 2.6: Ejemplo gráfico del funcionamiento de los *Kernel*.

Fuente: Joaquín Amat Rodrigo, RPubs

En la anterior imagen podemos observar como en primera instancia, los datos no se podían separar con ningún hiperplano, pero si aumentamos la dimensión encontramos una nueva manera de segregar ambas clases. Gracias a los *kernel* no nos restringimos solamente a los problemas lineales, sino que también podemos abarcar clasificaciones de naturaleza no lineal (usando los *kernels* gaussiano, polinómico, sigmoide...).

Para este algoritmo (y el resto de los que implementaré), utilizaremos el lenguaje de programación R y el paquete `caret`, y la función sigue la siguiente estructura.

```
1 mod <- train(respuesta~., data=datos, method = "svmkernel", trControl = ctrl)
```

2.3.2. Regresión Logística (Logit)

La regresión logística (IBM) es un algoritmo de clasificación probabilístico, es decir, en lugar de asignar cada observación a una clase, lo que nos devuelve es la probabilidad que tiene una observación de pertenecer a una clase u otra. Predice (o asigna) la clase de la observación en función de las variables explicativas. Este modelo utiliza datos distribuidos binomialmente, ya que buscamos saber el número de éxitos (observaciones pertenecientes a una clase) sobre el total de observaciones (muestra). Siguen la siguiente estructura:

$$logit(p_i) = ln(\frac{p_i}{1-p_i}) = \beta_0 + \beta_1 x_{1,i} + \cdots + \beta_k x_{k,i} \tag{2.1}$$

Donde p_i es la probabilidad de pertenecer a la clase positiva para la observación i, β_k son los parámetros (normalmente estimados por máxima verosimilitud) y x_k es el valor que toma la variable explicativa k.

La función de enlace es una transformación matemática que se utiliza para relacionar la variable dependiente (a qué clase pertenece cada observación), con los predictores lineales del modelo, que en este caso, son $\beta_0 + \beta_1 x_{1,i} + \cdots + \beta_k x_{k,i}$. Para nuestro análisis, nos centraremos en la función de enlace anteriormente mencionada *logit*, ya que pretendemos transformar unos valores en una escala continua (valor del predictor lineal) en un valor entre 0 y 1 que simbolice la probabilidad.

$$\begin{aligned} &\textbf{Función de enlace} && g(p_i) = \eta = X\beta \\ &\textbf{Función de enlace } \textbf{\textit{logit}} && ln(\frac{p_i}{1-p_i}) \\ &\textbf{Función de distribución} && p_i(\eta) = g_i^{-1}(\eta) = \frac{exp(\eta)}{1+exp(\eta)} \end{aligned} \tag{2.2}$$

Donde $p_i(\eta)$ es la probabilidad de la observación i de pertenecer a la clase positiva, $g(p_i)$ la función de enlace y η el predictor lineal ($\beta_0 + \beta_1 x_{1,i} + \cdots + \beta_k x_{k,i}$) . Ahora que he detallado los fundamentos matemáticos de este modelo (ya que los considero muy útiles a la hora de entender como funciona), procederemos a ver la estructura que tiene el código que implementaré:

```
mod <- glm(respuesta ~., family=binomial(link='logit'),  data = datos )
m0.1 <- step(mod,direction="both",k=log(nrow(datos)),trace=FALSE)
```

Como vemos, especificamos la fórmula (variable dependiente versus explicativas), la familia de la distribución (que en nuestro caso es la binomial), y finalmente podemos ver el argumento **step**, que se refiere a la técnica *stepwise*. Es una herramienta muy útil, ya que mediante el cálculo del AIC o Criterio de información de Akaike [7] construye nuevos modelos predictivos de manera iterativa, añadiendo y quitando variables al modelo hasta seleccionar el mejor posible.

2.3.3. Naïve Bayes

El clasificador bayesiano ingenuo (IBM), o su nombre en inglés *Naïve Bayes*, es un algoritmo de clasificación que busca modelar la distribución de los datos introducidos, y a diferencia de otros algoritmos, no aprende qué características son más importantes para diferenciar entre clases. Se fundamenta principalmente en el teorema de Bayes, el cual relaciona la probabilidad condicionada de un suceso A a un suceso B, con la probabilidad condicionada de un suceso B a un suceso A:

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{P(B)} \tag{2.3}$$

Donde $P(A_i)$ son las probabilidades a priori, $P(B|A_i)$ es la probabilidad de B suponiendo que ha sucedido A_i y $P(A_i|B)$ son las probabilidades a posteriori. El adjetivo “ingenuo"viene dado porque este algoritmo asume que los atributos son condicionalmente independientes a la clase. Estas suposiciones, aunque no siempre se cumplen en la vida real, por lo general este algoritmo funciona bastante bien. Para su ejecución, el código seguirá esta estructura:

```
    library(naivebayes)
    nb <- naive_bayes(respuesta ~ ., data = datos)
    nb_kernel <- naive_bayes(x = x, y = y, usekernel = TRUE, laplace = 1)
```

[7] El criterio de información de Akaike (AIC) es una medida utilizada para comparar la calidad de diferentes modelos estadísticos. Evalúa el equilibrio entre la bondad de ajuste del modelo y su complejidad, penalizando los modelos con demasiados parámetros para evitar el sobreajuste (Wikipedia, 2024a)

Como veis, hemos introducido también la posibilidad de usar un kernel, ya que dada la situación que el algoritmo supone que los datos numéricos se distribuyen normalmente, el kernel nos permite transformar la distribución de los datos consiguiendo que el *Naïve Bayes* realice predicciones más precisas.

2.3.4. K-Nearest Neighbours (kNN)

El algoritmo de K vecinos más próximos (o en inglés *K-Nearest Neighbours*) (IBM), siendo k un número impar (para evitar empates), selecciona la clase que predomine entre las k observaciones cerca de la observación a clasificar. Para ello, calcula la distancia entre las observaciones, que en nuestro caso, utilizaremos la distancia de *Gower* (Gower, 1971) (en lugar de la euclidiana, que se usa por defecto), ya que permite trabajar con datos numéricos y categóricos a la vez. Esta distancia se calcula como el promedio de las disimilitudes parciales entre individuos:

$$D(x,y) = \frac{1}{p}\sum_{j=1}^{p} s_j(x_j, y_j) \tag{2.4}$$

Donde $D(x,y)$ es la distancia de Gower, p es el número total de variables, $s_j(x_j, y_j)$ es la función de similitud calculada para cada variable, que presenta una forma diferente según si se tratan de datos numéricos o categóricos. En caso de los numéricos:

$$s_j(x_j, y_j) = \frac{|x_j - y_j|}{R_j} \tag{2.5}$$

Donde:

$|x_j - y_j|$ = Diferencia absoluta entre los valores de la variable j en los registros x e y
R_j = Rango (diferencia entre el valor máximo y mínimo) de la variable j

Por otro lado, si se trata de datos categóricos:

$$s_j(x_j, y_j) = \frac{2c}{a+b} \tag{2.6}$$

Donde:

a y **b** = Número de presencias de la categoría j en los registros x e y
c = Número de coincidencias en ambos registros

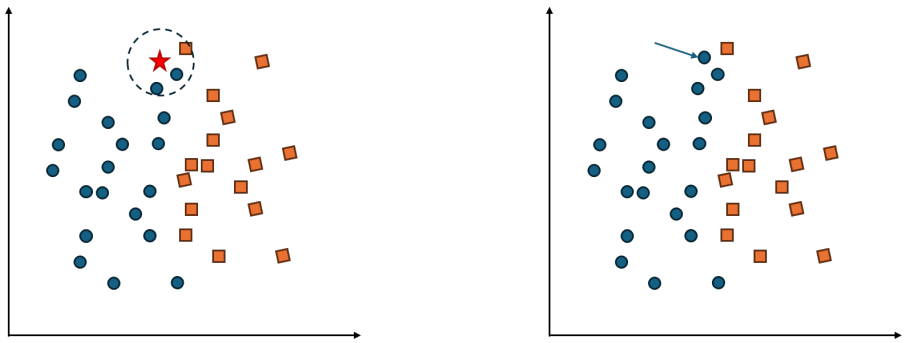


Figura 2.7: Ejemplo gráfico del funcionamiento del kNN usando $k = 3$.

Como podemos ver en esta representación, la estrella (que representa un dato nuevo en nuestro conjunto), identifica sus $k = 3$ vecinos más próximos, y su categoría pasará a ser la que predomine entre sus vecinos. El código que usaremos para ejecutar este algoritmo en R será:

```
library(caret)
train(respuesta ~., data = datos, method = "knn", trControl = ctrl, preProcess =
↪  c("nzv", "BoxCox"), tuneLength = 10)
```

2.3.5. Árboles de Decisión (CART) y Bosques Aleatorios (Random Forest)

Los árboles de decisión o *decision trees* son un algoritmo de clasificación muy sencillo de implementar en cualquier base de datos, además de que a efectos de interpretabilidad es muy cómodo de visualizar. Por una parte, pueden manejar una gran cantidad de datos sin implicar un uso excesivo de carga computacional. Aun así, tienden a "sobreajustarse" a los datos, son muy sensibles a los cambios y pueden presentar un sesgo elevado si se encuentran demasiado influenciados por alguna variable (Gupta)

Los elementos que forman los árboles de decisión son:

- **Nodo raíz**: Es el primer nodo de todos, desde el cual la muestra se divide de acuerdo a varias variables. En el ejemplo, el nodo raíz clasifica según si la observación es de sexo masculino o femenino.
- **Nodo interno**: Estos son los nodos que resultan de generar las ramas desde el nodo raíz, donde puede darse el caso en el que se pueda dividir varias veces o acabar en nodos hoja. En el ejemplo, observamos que si la observación es de sexo masculino,

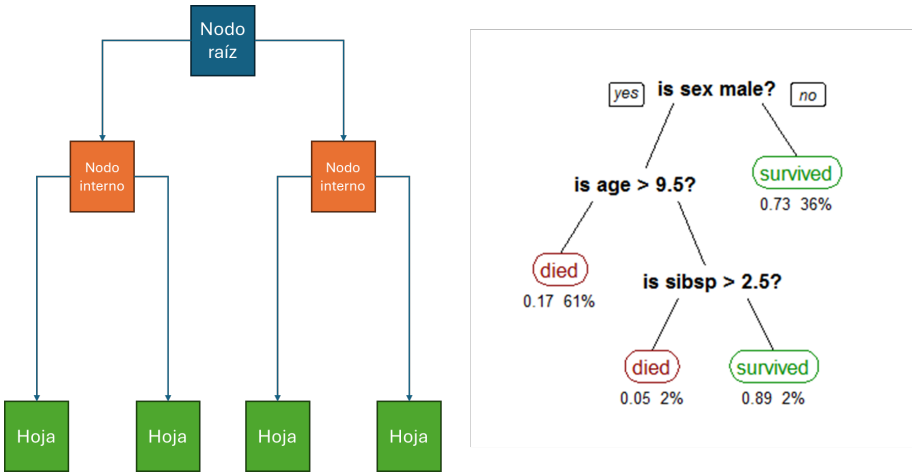


Figura 2.8: Representación de un árbol de decisión y un ejemplo

Fuente: Elaboración propia y (Gupta)

genera un nodo interno donde se divide según si la variable *age* es superior a 9.5, y si no lo es, vuelve a generarse un nodo interno, donde esta vez la condición es si la variable *sibsp* es mayor a 2.5 o no.

- **Hojas**: Son nodos que no se dividen más, y ya representan una clase de la variable respuesta. En el ejemplo, podemos ver nodos hoja después del nodo raíz, y a continuación de los nodos internos.

Como vemos, en los nodos hoja encontramos dos valores numéricos: El de la izquierda es la probabilidad de pertenecer a la clase positiva (en este caso, sobrevivir) y el número de la derecha es el porcentaje de observaciones en ese nodo hoja concreto. La conclusión que podemos sacar observando este árbol, es que ser una mujer o un niño de como mucho 9 años y medio y menos de 3 hermanos maximiza la probabilidad de sobrevivir.

Por otro lado, están los bosques aleatorios o *random forests*, otro algoritmo supervisado en el cual se juntan varios árboles de decisión generados aleatoriamente y se promedia la precisión de todos, solucionando así un problema que presentan los árboles de decisión convencional: el sobreajuste u *overfitting*. Aunque se pierde interpretabilidad respecto a los árboles de decisión convencionales, son más estables frente a variaciones en los datos, entre otras mejoras.

El código a utilizar en R para ejecutar los dos algoritmos es:

```
#Arboles de decision
train(respuesta ~., data = datos, method = "rpart", trControl = ctrl, preProcess =
↪  c("nzv", "BoxCox"), tuneLength = 10)
#Random Forests
train(respuesta ~., data = datos, method = "rf", trControl = ctrl, preProcess =
↪  c("nzv", "BoxCox"), tuneLength = 10)
```

2.3.6. XGBoost

El algoritmo XGBoost, de las siglas en inglés *eXtreme Gradient Boosting*, y se basa principalmente en impulsar métodos débiles utilizando el descenso de gradiente [8], y es muy útil y popular por su capacidad de escalabilidad en una gran variedad de escenarios.

La hoja de ruta de este algoritmo es sencilla pero útil:

- Se construye un árbol o un modelo simple similar, para realizar unas predicciones preliminares.
- Calcula los errores de este modelo y construye otro árbol para corregirlos.
- Se repite esta iteración, donde cada nuevo modelo corrige los errores del anterior.

Como vemos, el algoritmo tiene un objetivo esencial: minimizar la función de pérdida (error). La estructura base del código que utilizaré será:

```
xgb_model <- xgb.train(params = params, data = datos, nrounds = iteraciones,
↪  watchlist = list(train = entrenamiento, test = prueba))
# Predecir
predictions <- predict(xgb_model, newdata = datos)
```

[8] «El descenso del gradiente o gradiente descendiente es un algoritmo de optimización iterativo de primer orden que permite encontrar mínimos locales en una función diferenciable» (Wikipedia, 2024b)

2.4. Métricas de validación

En este apartado hablaremos de las principales métricas que usaremos para validar el modelo, es decir, medir el rendimiento del modelo para poder evaluarlo y compararlo con el resto. Para ello explicaré las métricas más comunes y cómo se calculan.

Para calcular estas métricas, el punto de partida es la construcción de nuestra matriz de confusión (o *Confusion Matrix* en inglés). Esta tabla representa la clasificación de nuestro modelo y compara las predicciones con los valores reales.

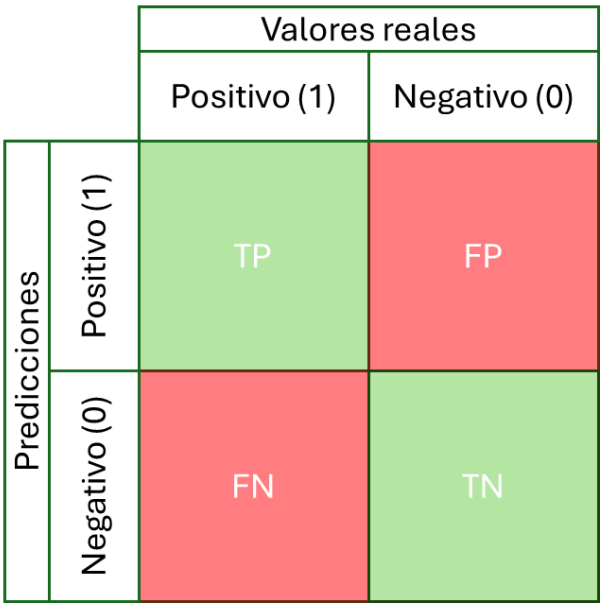


Figura 2.9: Estructura de la matriz de confusión

Las celdas verdes representan los éxitos del modelo, es decir, los valores verdaderamente positivos (**TP** o *True Positive* en inglés) y los valores verdaderamente negativos (**TN** o *True Negatives* en inglés). En cambio, las casillas rojas representan los errores que ha cometido el modelo: Por un lado, los falsos positivos (FP) se dan cuando el modelo considera positivo a una observación que realmente es negativa (por ejemplo, un test de COVID que detecta que tienes el virus cuando realmente no lo tienes); por otro lado, los falsos negativos se dan cuando la observación pertenece a la clase positiva, pero el modelo la clasifica como negativa (por ejemplo, un test de embarazo puede ser negativo cuando la persona está realmente embarazada). A raíz de esta tabla, calcularemos las siguientes métricas[9].

[9]Salvo excepción, el rango de los estadísticos es de 0 a 1, siendo 1 el mejor valor que puede tomar y 0 el peor.

Accuracy La exactitud o *accuracy* del modelo indica la cantidad de predicciones acertadas sobre el total de predicciones.

$$\text{Accuracy } = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.7}$$

Recall La sensibilidad, tasa de verdaderos positivos (TVP o *TPR*) o *Recall*, mide la proporción de positivos reales clasificados correctamente por el modelo, y describe la capacidad de nuestro modelo de clasificar correctamente la clase positiva.

$$\text{Recall } = \frac{TP}{TP + FN} \tag{2.8}$$

Precision La precisión es la proporción de predicciones positivas correctas respecto al total de clasificaciones positivas.

$$\text{Precision } = \frac{TP}{TP + FP} \tag{2.9}$$

Specificity La especificidad, tasa de verdaderos negativos (TVN o *TNR*) o *Specificity* mide que tan bien el modelo es capaz de clasificar los negativos.

$$\text{Specificity} = \frac{TN}{TN + FP} \tag{2.10}$$

F-Score El *F-Score* es la media armónica entre los previamente comentados *precision* y *recall*, y es una métrica que representa ambos valores a la vez.

$$\text{F-Score} = 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \tag{2.11}$$

ROC y AUC La curva ROC es un elemento gráfico muy útil, ya que muestra una representación de la relación entre sensibilidad y las falsas alarmas ($1 - especificidad$).

A raíz de los anteriores gráficos, podemos hacernos a la idea del comportamiento del modelo. La línea diagonal roja representa una aleatoriedad total, es decir, el modelo es tan preciso como lo sería lanzar una moneda al aire. Por otro lado, la esquina superior izquierda representa el modelo perfecto, en el que clasifica correctamente todas las observaciones.

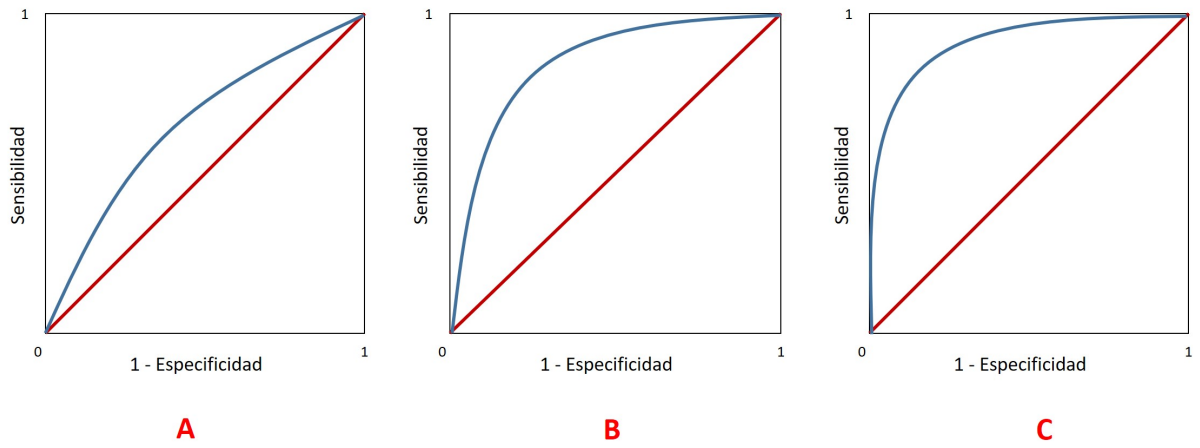


Figura 2.10: Diferentes escenarios de la curva ROC

Fuente: (Silva, 2021)

Por lo tanto, podemos ver como en el gráfico A, la curva ROC está muy cerca de la diagonal, por lo que el modelo sería mediocre o malo. A medida que avanzamos con los siguientes gráficos, los modelos son mejores, ya que la curva se aproxima más al punto de perfección. Para cuantificar esta valoración, lo que se hace es calcular el área debajo de la curva (o AUC). Se han especificado los siguientes valores para interpretar los valores de AUC:

AUC	**Evaluación**
[0.5]	Completamente aleatorio
(0.5 , 0.6]	Modelo malo
(0.6 , 0.75]	Modelo regular
(0.75 , 0.9]	Modelo bueno
(0.9 , 0.97]	Modelo muy bueno
(0.97 , 1]	Modelo excelente

Cuadro 2.1: Tabla de evaluación del valor AUC

Coeficiente de determinación $\mathbf{R}^2$ El R^2 es una medida estadística que indica la proporción de la varianza de la variable respuesta que es explicada por las variables predictoras. En este caso, no podemos calcularlo a raíz de la matriz de confusión, sino usando esta fórmula:

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \overline{y})^2} \tag{2.12}$$

Donde:

y_i = Observación i

$\overline{y}_i$ = Media de las observaciones

$\hat{y}_i$ = Predicción de la observación i

Cuanto mejor sea el modelo, el valor de R^2 será más cercano a 1, y de manera contraria, cuanto peor sea, más cercano al 0 será. En el mejor de los casos, las predicciones serán exactamente iguales a las observaciones, lo que anulará la fracción y el valor del coeficiente será 1.

Kappa de Cohen El estadístico Kappa se encarga de comparar los valores predichos con los observados, y mide la aleatoriedad de los aciertos. Es decir, mide si los aciertos que ha hecho nuestro modelo han sido por azar o por una correcta clasificación. La fórmula es la siguiente:

$$\kappa = \frac{2 \cdot (TP \cdot TN - FN \cdot FP)}{(TP + FP) \cdot (FP + TN) + (TP + FN) \cdot (FN + TN)} \tag{2.13}$$

Según el valor que tenga, concluiremos una cosa u otra:

κ	**Evaluación**
$< \mathbf{0}$	Desacuerdo
$\mathbf{(0.01\ ,\ 0.2]}$	Ligero acuerdo
$\mathbf{(0.2\ ,\ 0.4]}$	Acuerdo decente
$\mathbf{(0.4\ ,\ 0.6]}$	Acuerdo moderado
$\mathbf{(0.6\ ,\ 0.8]}$	Acuerdo sustancial
$\mathbf{(0.8\ ,\ 0.99]}$	Acuerdo prácticamente perfecto

Cuadro 2.2: Tabla de evaluación del valor κ

2.5. Problemática de los datos desbalanceados: Ejemplos y enfoques

Es importante saber como se distribuye nuestra variable respuesta, ya que los datos desbalanceados pueden ser desafíos significativos a la hora de clasificar. Esta situación se da cuando una de las dos clases está sobrerrepresentada respecto a la otra, lo cual nos puede llevar a (Dave, 2023):

- **Sesgo hacia la clase mayoritaria**: Los modelos tienden a favorecer a la clase con más muestras, lo que el rendimiento a la hora de detectar la clase minoritaria puede ser deficiente.
- **Dificultad en la generalización**: El modelo puede tener dificultades para generalizar datos nuevos, especialmente aquellos que pertenezcan a la clase subrepresentada.
- **Evaluación errónea del rendimiento**: Las métricas que hemos mencionado pueden dar resultados engañosos, ya que, por ejemplo, un modelo podría clasificar bien solamente la clase mayoritaria y aun así tener una *accuracy* bastante alta.

Por ejemplo, imaginemos que el caso que trabajaremos sobre la detección de posibles clientes morosos tiene un desbalance muy elevado: solamente el 1 % de los clientes llegan a ser morosos, frente al 99 % restante que pueden ser clasificados como buenos clientes. Un modelo que trabaje con estos datos podría aprender a clasificar todos los clientes como buenos, lo que resultaría en una alta tasa de falsos negativos (buen cliente cuando no lo es).

Por lo tanto, ante esta situación, debemos de reformular nuestra estrategia para poder mitigar esta problemática. Para ello, podemos usar diferentes métodos (Shirin, 2017):

- **Undersampling** : Para ello, lo que hacemos es seleccionar aleatoriamente una muestra de la clase predominante del tamaño de la clase minoritaria, para así equilibrar las dos clases. La principal desventaja de este método es que perdemos información potencialmente relevante en las observaciones que descartamos.

- **Oversampling** :Al contrario del anterior método, se duplican observaciones de manera aleatoria en la clase subrepresentada, hasta conseguir la misma cantidad en ambas clases. La principal desventaja es que aumentamos el riesgo de sobreajustar el modelo a los datos, lo cual conllevaría problemas de generalización con nuevos datos.
- **Synthetic minority oversampling technique (SMOTE)** : Es una técnica de *oversampling* que genera instancias artificiales seleccionando puntos que están en la línea que unen una observación rara (o minoritaria) con uno de sus vecinos más cercanos en el conjunto de características (Demir & Sahin, 2022).
- **Random Oversampling Examples (ROSE)** : Utiliza un *bootstraping* suavizado [10] creando nuevas muestras a partir del conjunto de características alrededor de la clase minoritaria (Demir & Sahin, 2022).

En nuestro análisis, llevaremos a cabo todos los métodos y los compararemos para saber cuál se comporta mejor.

[10]El bootstrap suavizado es una técnica de remuestreo que utiliza un estimador de densidad para generar muestras continuas a partir de una población, incorporando una perturbación controlada por un parámetro de suavizado (Abad & Casal, 2022)

Capítulo 3

Caso práctico: Datos crediticios alemanes

Llegamos a la parte práctica del análisis, en el que podremos comprobar como se implementa el *machine learning* en el sector financiero para clasificar prestatarios según su riesgo de crédito. Se espera que un modelo predictivo desarrollado a partir de estos datos proporcione a un director de banco orientación para tomar la decisión de aprobar o no (según su calificación crediticia) un préstamo a un posible solicitante en función de sus perfiles.

La base de datos *German Credit Data* contiene 20 variables y la clase según si el solicitante del préstamo se considera con buen riesgo crediticio (probablemente pagará la deuda) o no. Aquí dejaré el enlace a la fuente de la base de datos:

- https://archive.ics.uci.edu/dataset/144/statlog+german+credit+data [1]

Los modelos predictivos que llevaremos a cabo sobre esta base de datos pretenden proveer al director del banco información para tomar la decisión de si se debe aprobar o no el crédito al o la solicitante basado en su perfil.

[1] (Hofmann, 1994)

3.1. Definición de las variables utilizadas

Variable	**Tipo**	**Definición**	
account_status	Variable cualitativa politómica	Estado de la cuenta corriente	A11: En descubierto (menos de 0 u.m.), A12: Entre 0 y 200 u.m., A13: Al menos 200 u.m. o más, o nómina regular durante mínimo un año, A14: Sin cuenta corriente
months	Variable numérica continua	Duración del préstamo en meses	-
credit_history	Variable cualitativa politómica	Historial crediticio	A30 - No ha tenido créditos o los que ha tenido los ha devuelto debidamente, A31 - Todos los créditos en este banco reembolsados debidamente, A32 - Créditos existentes y están siendo devueltos debidamente hasta el momento, A33 - Retraso en el pago en el pasado A34 - Cuenta crítica y/o existen otros créditos (en otros bancos)
purpose	Variable cualitativa politómica	Propósito del crédito	A40 - coche (nuevo), A41 - coche (usado), A42 - mobiliario/equipo, A43 - radio/televisión, A44 - electrodomésticos, A45 - reparaciones, A46 - educación, A47 - vacaciones, A48 - reciclaje, A49 - negocios, A410 - otros
credit_amount	Variable numérica continua	Cuantía del crédito	-
savings	Variable cualitativa politómica	Ahorros	A61 : Menos de 100 u.m. A62 : Entre 100 y 500 u.m. A63 : Entre 500 y 1000 u.m. A64 : Más de 1000 u.m. A65 : Desconocido/ Sin ahorros
employment	Variable cualitativa politómica	Antigüedad en el actual trabajo	A71: Desempleado A72: Menos de un año A73: Entre un año y 4 A74: Entre 4 años y 7 A75: 7 o más años
installment_rate	Variable numérica continua	Proporción de los ingresos destinados al pago de deuda	
personal_status	Variable cualitativa politómica	Estado civil y género	A91: Hombre y divorciado, A92: Mujer y divorciada/casada, A93: Hombre soltero, A94: Hombre casado, A95: Mujer soltera
guarantors	Variable cualitativa politómica	Otros deudores o avalistas	A101: No, A102: Co-aplicante, A103: Avalista
residence	Variable numérica continua	Antigüedad en la actual residencia	-
property	Variable cualitativa politómica	Propiedades	A121: Bienes raíces, A122: Cuenta de ahorros de compañía o seguro de vida, A123: Coche u otros, A123: Sin propiedades o desconocido
age	Variable numérica continua	Edad en años	-
other_installments	Variable cualitativa politómica	Otras deudas	A141: Bancos, A142: Tiendas, A143: Ninguno
housing	Variable cualitativa politómica	Alojamiento	A151: Alquiler, A152: Propietario, A153: Gratis
credit_cards	Variable numérica continua	Número de créditos existentes en este banco	-
job	Variable cualitativa politómica	Trabajo	A171: Desempleado/no cualificado, no residente A172: No cualificado, residente A173: Cualificado / oficinista A174: Encargado / autónomo / altamente cualificado/ oficinista
dependents	Variable numérica continua	Número de personas dependientes del prestatario	-
phone	Variable cualitativa dicotómica	Teléfono	A191: No A192: Sí, registrado a su nombre
foreign_worker	Variable cualitativa dicotómica	Trabajador extranjero	A201: Sí, A202: No
credit_rating	Variable cualitativa dicotómica	Calificación crediticia	Buena o mala

Cuadro 3.1: Descripción de las variables

3.1.1. Preparación de los datos

En primer lugar, importamos los datos del archivo `german.data`, y para ello usaré la función `read_delim()` de `R`, que me permite importar archivos delimitados (en este caso, por espacios). Además, el nombre de las variables viene codificado desde $X1, \ldots, X21$, por lo que las renombraré para mayor comodidad, siguiendo la Tabla 3.1. Veamos si hay valores faltantes en la base de datos usando la función `colSums(is.na(x))`. En nuestro caso, los valores faltantes deben de verse directamente como `NA`, ya que al hacer una primera ojeada hay que observar de qué manera se codifican los valores faltantes y los *outliers*. Entonces:

Variables	**Número de missings**	**Variables**	**Número de missings**
account_status	0	property	0
months	0	age	0
credit_history	0	other_installments	0
purpose	0	housing	0
credit_amount	0	credit_cards	0
savings	0	job	0
employment	0	dependents	0
installment_rate	0	phone	0
personal_status	0	foreign_worker	0
guarantors	0	credit_rating	0
residence	0		

Cuadro 3.2: Cantidad de *missings* por variable

Como vemos, ninguna variable presenta ningún valor faltante, por lo que podemos empezar a ver como se comportan las variables, tanto individualmente como de manera bivariante con la respuesta.

3.2. Análisis descriptivo

Análisis univariante de las variables numéricas

Empezaremos con las variables numéricas, que son: *months, credit_amount, installment_rate, residence, age, credit_cards, dependents*. Por un lado, haremos una tabla resumen de los principales estadísticos como vienen siendo la media, las desviaciones estándar, los percentiles y los valores extremos.

Variable	N	Media	Std. Dev.	Min	Pctl. 25	Pctl. 75	Max
months	1000	21	12	4	12	24	72
credit_amount	1000	3271	2823	250	1366	3972	18424
installment_rate	1000	3	1.1	1	2	4	4
residence	1000	2.8	1.1	1	2	4	4
age	1000	36	11	19	27	42	75
credit_cards	1000	1.4	0.58	1	1	2	4
dependents	1000	1.2	0.36	1	1	1	2

Cuadro 3.3: Tabla resumen de las variables numéricas

La duración media de los préstamos es de 21 meses, aunque con una desviación estándar de 12, por lo que no nos sorprendería encontrar valores mayores a esta cantidad (lo veremos gráficamente). Por otro lado, lo mismo pasa con la cantidad prestada, que de media es de 3.271 u.m., alcanzando a máximos de casi 19.000 u.m. . El tipo de interés gira alrededor del 3 %, la edad ronda los 36 años, y suelen tener entre 1 y 2 tarjetas de crédito y 1 persona a su cargo. Veamos gráficamente estas variables para ver mejor como se distribuyen.

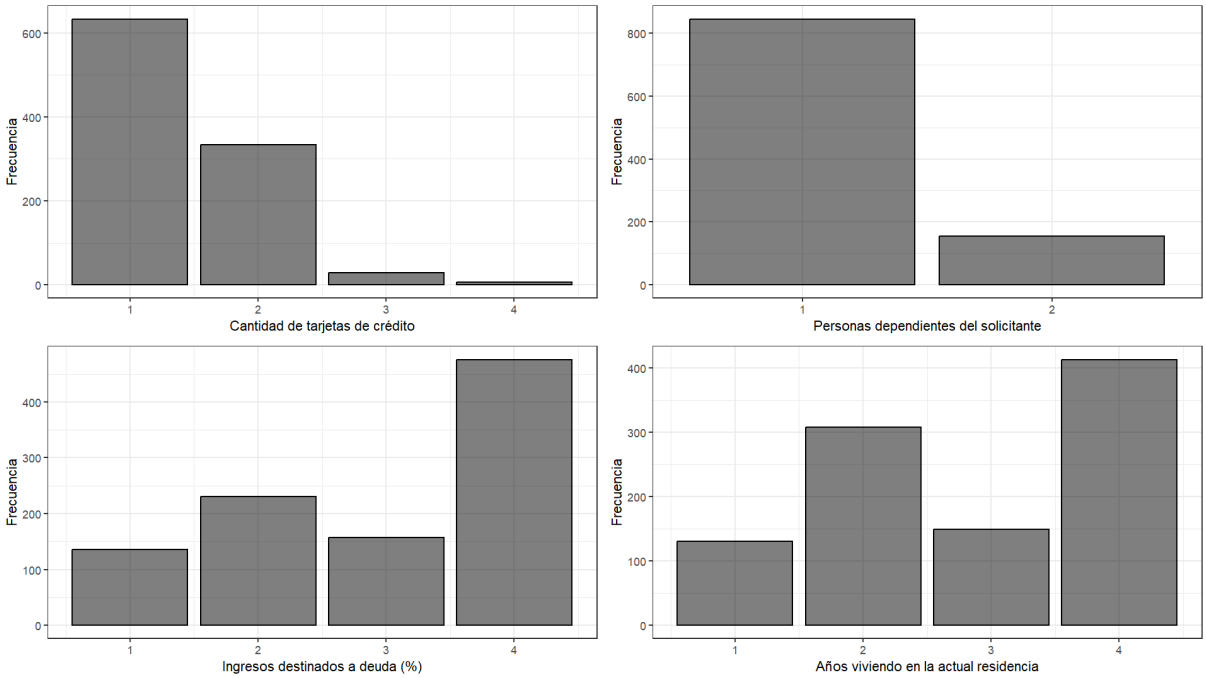


Figura 3.1: Gráfico de barras de las variables *installment_rate*, *credit_cards*, *dependents*, *residence*

En primer lugar, vemos que la cerca del total de los clientes tienen entre 1 o 2 tarjetas de crédito, habiendo muy pocos casos con 3 o 4. Por otro lado, la gran parte de las observaciones tienen solo una persona a cargo. Finalmente, más de la mitad de los préstamos están al 4 % de interés, mientras que la otra mitad esta equidistribuida entre 1 % y 3 %, y en cuanto a su estancia en su actual vivienda, el 40 % lleva 4 años y el 30 % 2 años, el restante están equidistribuidos entre 1 o 3 años.

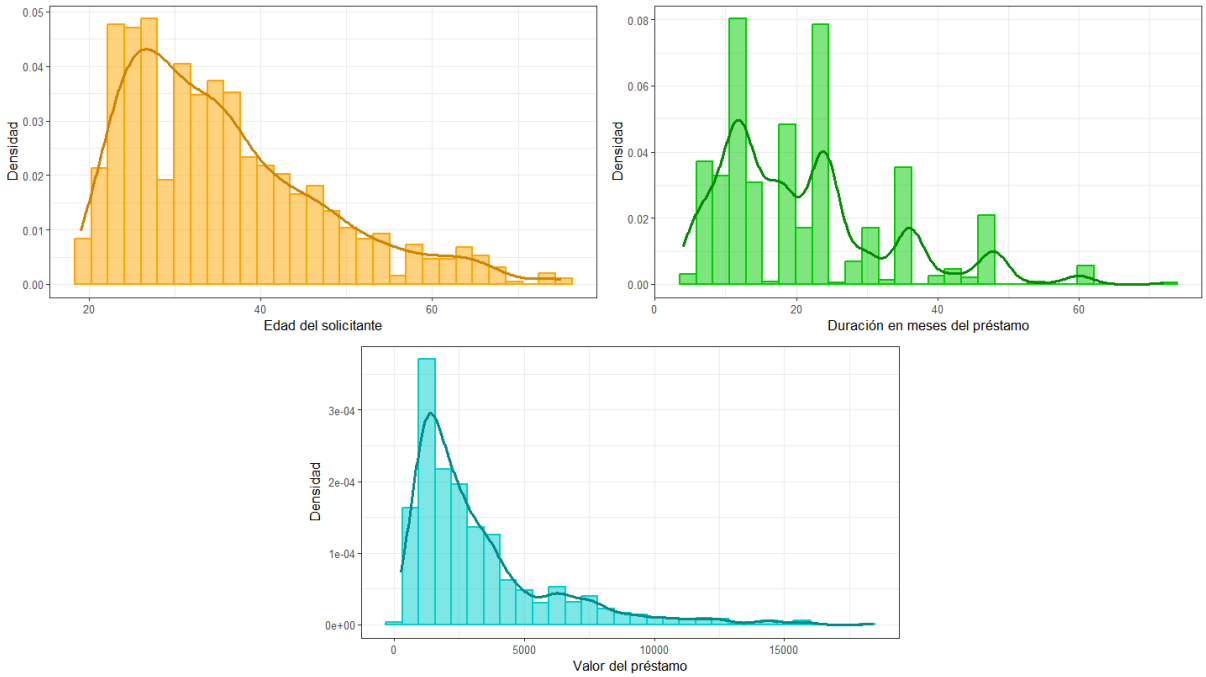


Figura 3.2: Histograma de las variables *age*, *months*, *credit_amount*

Por el lado de estas otras variables, vemos como la edad del solicitante, aunque presenta un rango de 20 a poco más de 70 años, vemos como la mayoría se concentran en la horquilla que comprende entre los 20 años y los 45. La duración del préstamo también tiene una asimetría hacia la izquierda, donde la mayoría de los préstamos de esta base de datos está por debajo de los 2 años (o 24 meses). Finalmente, el valor del préstamo viene concentrado por debajo de las 5.000 u.m., con pocos valores por encima.

Análisis univariante de las variables categóricas

A continuación analizaremos individualmente las variables categóricas, que son: *account_status, credit_history, savings, employement, personal_status, guarantors, property, other_installments, housing, job, phone, foreign_worker y credit_rating*. Por un lado, haremos una tabla resumen donde nos indicará como están distribuidas las observaciones en cada variable.

Variable	N	Porcentaje	Variable	N	Porcentaje	Variable	N	Porcentaje
credit_rating	1000		**account_status**	1000		**purpose**	1000	
... Malo	300	30 %	... A11	274	27 %	... A40	234	23.4 %
... Bueno	700	70 %	... A12	269	27 %	... A41	103	10.3 %
credit_history	1000		... A13	63	6 %	... A42	181	18.1 %
... A30	40	4 %	... A14	394	39 %	... A43	280	28 %
... A31	49	5 %	**personal_status**	1000		... A44	12	1.2 %
... A32	530	53 %	... A91	50	5 %	... A45	22	2.2 %
... A33	88	9 %	... A92	310	31 %	... A46	50	5 %
... A34	293	29 %	... A93	548	55 %	... A47	0	0 %
savings	1000		... A94	92	9 %	... A48	9	0.9 %
... A61	603	60 %	**property**	1000		... A49	97	9.7 %
... A62	103	10 %	... A121	282	28 %	... A410	12	1.2 %
... A63	63	6 %	... A122	232	23 %	**job**	1000	
... A64	48	5 %	... A123	332	33 %	... A171	22	2 %
... A65	183	18 %	... A124	154	15 %	... A172	200	20 %
employment	1000		**other_installments**	1000		... A173	630	63 %
... A71	62	6 %	... A141	139	14 %	... A174	148	15 %
... A72	172	17 %	... A142	47	5 %	**phone**	1000	
... A73	339	34 %	... A143	814	81 %	... A191	596	60 %
... A74	174	17 %	**housing**	1000		... A192	404	40 %
... A75	253	25 %	... A151	179	18 %	**foreign_worker**	1000	
guarantors	1000		... A152	713	71 %	... A201	963	96 %
... A101	907	91 %	... A153	108	11 %	... A202	37	4 %
... A102	41	4 %						
... A103	52	5 %						

Cuadro 3.4: Tabla resumen de las variables categóricas

Comentaremos cada variable junto a unos gráficos de tarta que nos ayudará a visualizar las proporciones de cada variable, y así identificar comportamientos a priori. En primer lugar, podemos fijarnos en la variable respuesta: la variable *credit_rating* presenta un desbalance, ya que el 70 % de las observaciones pertenecen a la clase «Bueno».

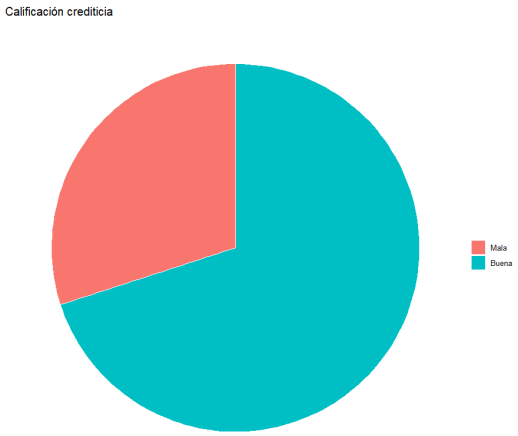


Figura 3.3: Gráfico de tarta de la variable *credit_rating*

A continuación observamos las variables referentes a la antigüedad laboral, al trabajo, y si se trata de un trabajador en el extranjero.

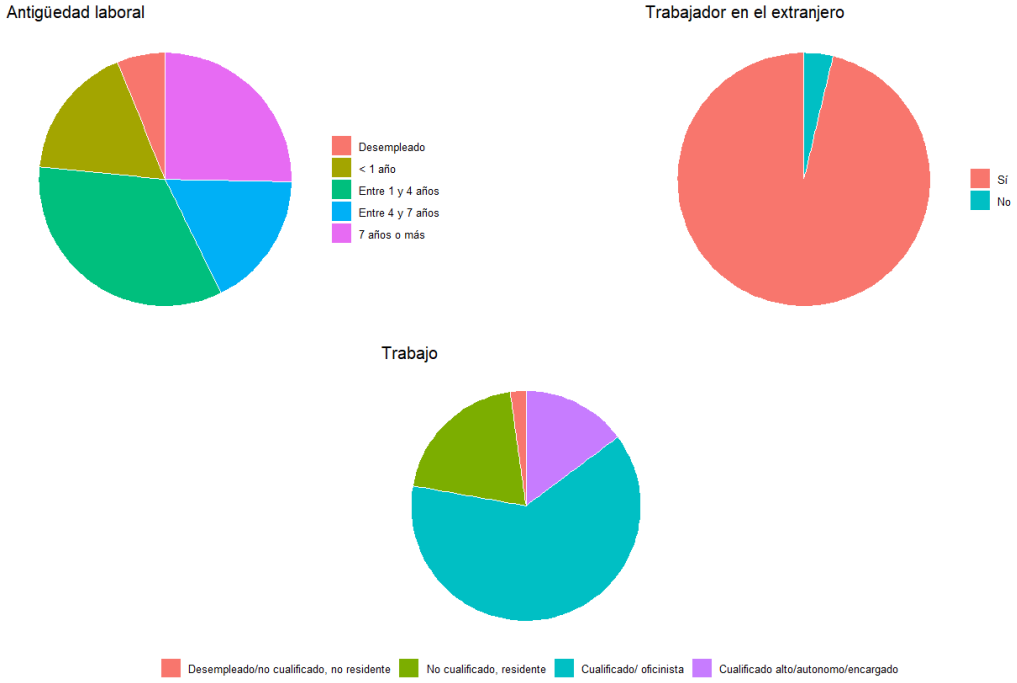


Figura 3.4: Gráfico de tarta de las variables *job, foreign_worker y employment*

Vemos como el 34 % de los solicitantes llevan entre 1 y 4 años en su trabajo actual, y poco más del 25 % llevan más de 7 años. Por otro lado, observamos como la gran mayoría de los clientes trabajan en el extranjero (o son extranjeros trabajando en la zona). Finalmente, el perfil más observado entre los sujetos es de un trabajo cualificado o como oficinista.

Las siguientes variables explican la situación financiera del cliente.

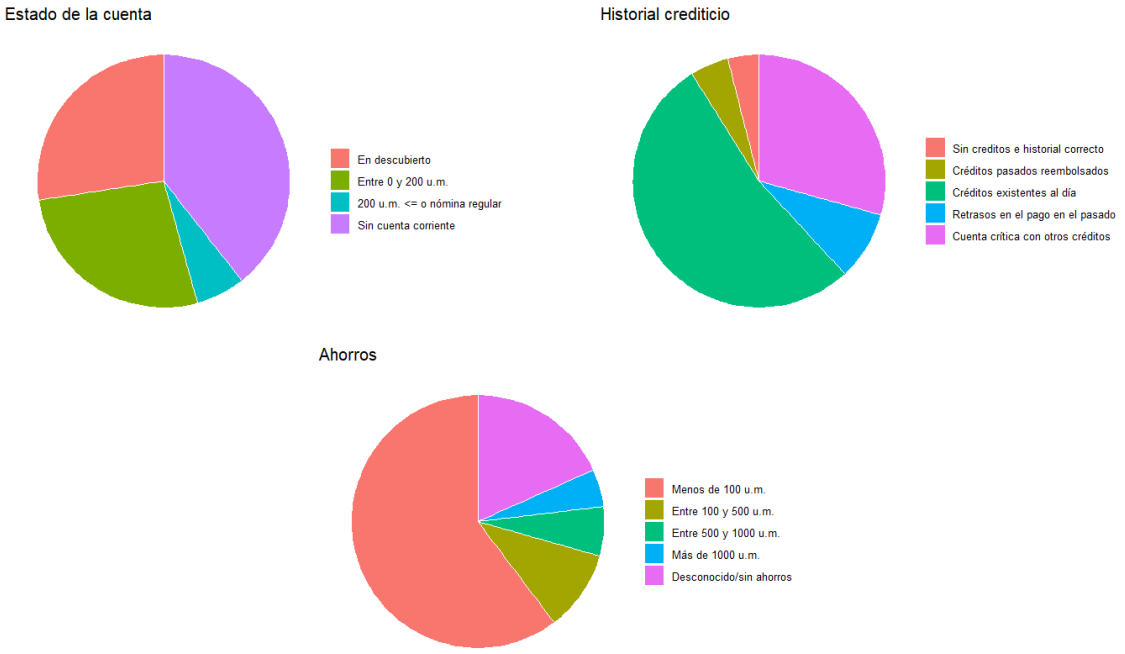


Figura 3.5: Gráfico de tarta de las variables *account_status, credit_history y savings*

Por un lado, vemos como el 39 % no tiene cuenta corriente, y del resto que sí, la mitad está en descubierto (es decir, con saldo negativo), y la otra mitad tiene entre 0 y 200 u.m. . Hay una pequeña proporción del 6 % que presenta más de 200 u.m. y una nómina domiciliada regular. Por otro lado, el 60 % de los individuos tienen pocos ahorros (por debajo de las 100 u.m.), y un 18 % se desconoce si quiera si tienen ahorros. Ahora bien, podemos observar como la mayoría tiene un historial crediticio bueno, aunque cerca del 38 % tienen varios créditos actuales o ha tenido retrasos en el pasado.

Estos gráficos representan las variables referentes a si son garantes, si tienen teléfono, o propiedades.

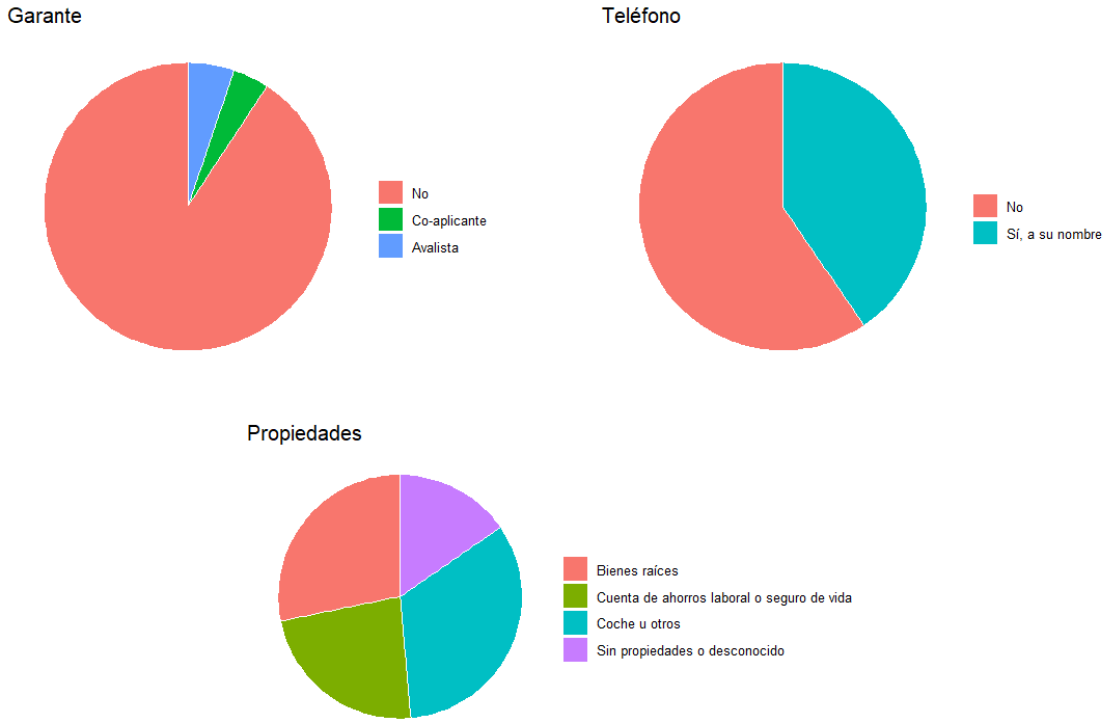


Figura 3.6: Gráfico de tarta de las variables *guarantors, phone y property*

Observamos que la gran mayoría de los individuos son los principales responsables de pagar el préstamo, mientras que hay un 5 % que actúa como avalista (se comprometen a asumir la responsabilidad si el prestatario no pueda pagar) y un 4 % que actúa como co-aplicante, es decir, solicita el crédito junto a otro individuo. Por otro lado, el 60 % no tiene teléfono (o al menos, no a su nombre), y el 40 % restante sí. Finalmente, podemos ver como muchos de ellos tienen propiedades, siendo la clase predominante el coche u otros vehículos, seguidamente de bienes raíces, y otros con cuentas de ahorros o seguros de vida (planes laborales). Hay un 15 % que no tiene ninguna propiedad.

Las últimas variables representan el sexo y estado civil, el propósito del crédito y la existencia de otras deudas.

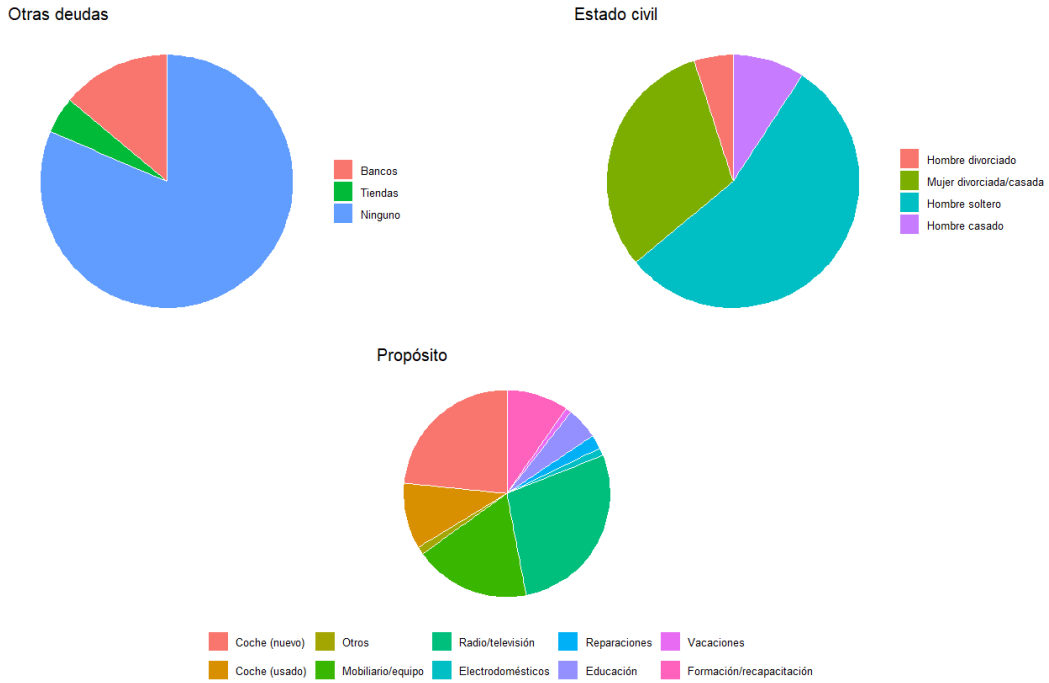


Figura 3.7: Gráfico de tarta de las variables *other_installments, status y purpose*

Finalmente, vemos como cerca del 81 % no presenta ninguna otra deuda o crédito en curso, y el 19 % restante tiene otros créditos bancarios o casualmente con tiendas. El estado civil y sexo de la mayoría de los individuos es de soltero y hombre, y el siguiente gran grupo es de mujer divorciada o casada. Resalta el hecho de que no haya mujeres solteras en esta base de datos, pero no podemos deducir a ciencia cierta el motivo. En cuanto al propósito del crédito, viene desglosado en varias razones: Alrededor del 28 % busca financiar la compra de una radio o televisión, cerca del 33 % financia la compra de un coche (en su mayoría, un coche nuevo con un 23.4 %). También hay una porción considerable de individuos que buscan financiar nuevo mobiliario o equipo, y el resto de individuos están repartidos entre si financian vacaciones, electrodomésticos, reparaciones, educación, ...

Análisis bivariante de las variables numéricas

Visualizaremos las principales estadísticas por cada clase:

Malo	N	Media	Std. Dev.	Min	Pctl. 25	Pctl. 75	Max
months	300	25	13	6	12	36	72
credit_amount	300	3938	3536	433	1352	5142	18424
installment_rate	300	3.1	1.1	1	2	4	4
residence	300	2.8	1.1	1	2	4	4
age	300	34	11	19	25	40	74
credit_cards	300	1.4	0.56	1	1	2	4
dependents	300	1.2	0.36	1	1	1	2

Bueno	N	Media	Std. Dev.	Min	Pctl. 25	Pctl. 75	Max
months	700	19	11	4	12	24	60
credit_amount	700	2985	2401	250	1376	3635	15857
installment_rate	700	2.9	1.1	1	2	4	4
residence	700	2.8	1.1	1	2	4	4
age	700	36	11	19	27	42	75
credit_cards	700	1.4	0.58	1	1	2	4
dependents	700	1.2	0.36	1	1	1	2

Además, seleccionaremos las variables que presenten una correlación significativa entre ellas y visualizaremos su comportamiento conjunto, además de realizar el mismo procedimiento, pero esta vez versus nuestra variable respuesta. Para ello, he hecho un mapa de calor que representa la correlación entre las variables, así como su significación (al $\alpha = 0{,}05$).

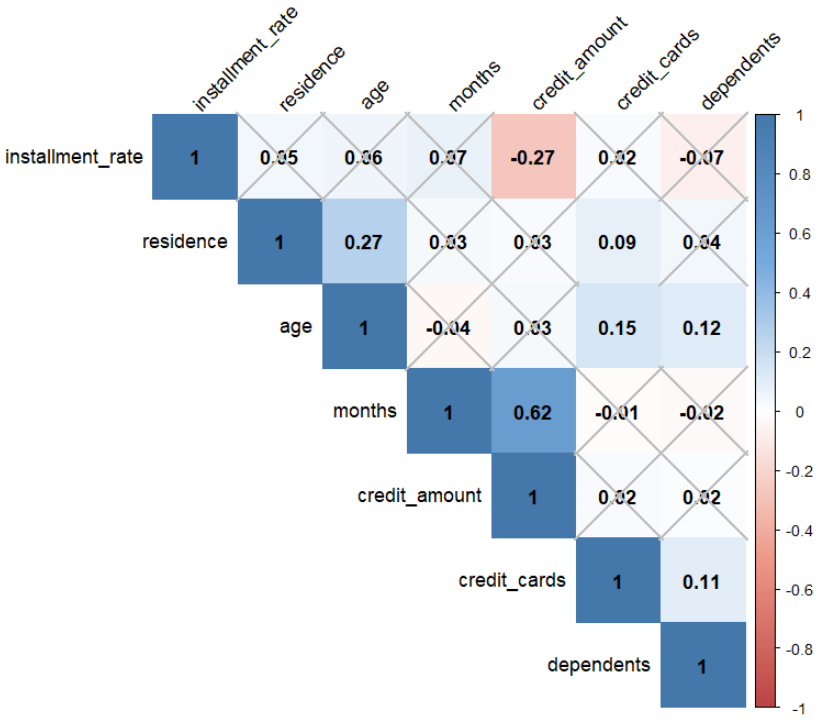


Figura 3.8: Mapa de calor de las correlaciones de Pearson de las variables numéricas

Vemos que hay varias relaciones que no están tachadas, eso significa que son significativas. Aun así, representaré solo las tres relaciones con una correlación más elevada, ya que las otras tres, aunque sean significativas, son muy bajas como para visualizar algo. Las relaciones más fuertes son: *months* y *credit_amount*, *residence* y *age*, *installment_rate* y *credit_amount*. Utilizaré el tipo de gráfico que encuentre conveniente para cada relación.

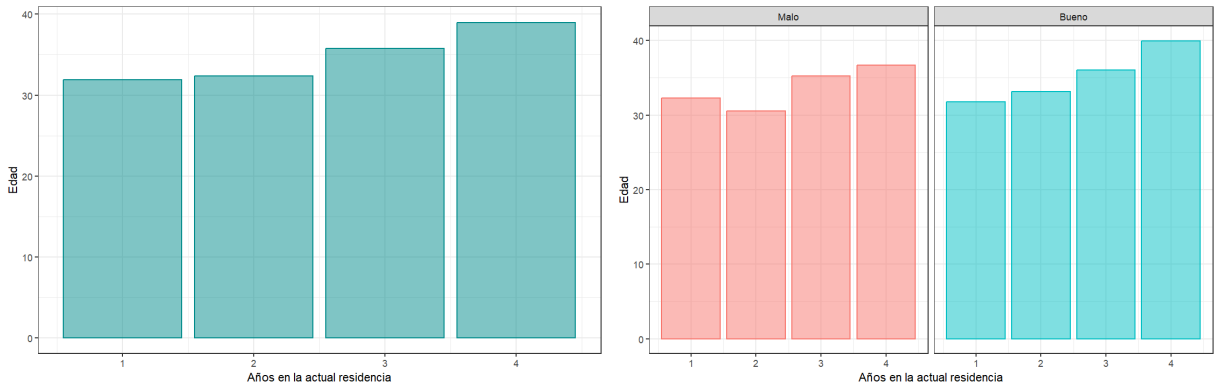


Figura 3.9: Edad y años en la actual residencia

Como podemos ver, hay una relación positiva entre la edad del cliente y los años que lleva viviendo en su actual residencia, lo que parece un razonamiento trivial por lo general. La relación se comporta de la misma manera tanto para clientes buenos como malos.

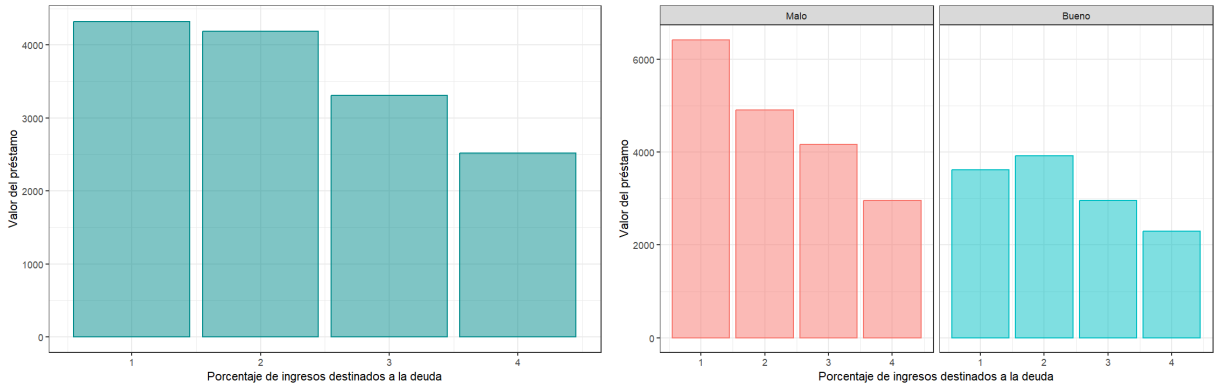


Figura 3.10: Valor del préstamo y proporción de los ingresos destinada a las deudas

Por otra parte, vemos como la cantidad de los ingresos que los clientes reservan para pagar las deudas disminuye cuanto mayor sea el préstamo; normalmente los préstamos de alto valor son concedidos a personas con altos ingresos (por lo tanto, el pago proporcional a la renta será menor), son longevos en el tiempo por lo que las cuotas mensuales son menores, entre otras posibles razones. Vemos una ligera diferencia entre los perfiles de los clientes, ya que mientras que los clientes con una buena calificación crediticia estan distribuidos prácticamente de igual manera, los clientes con mala calificación lo hacen de manera decreciente.

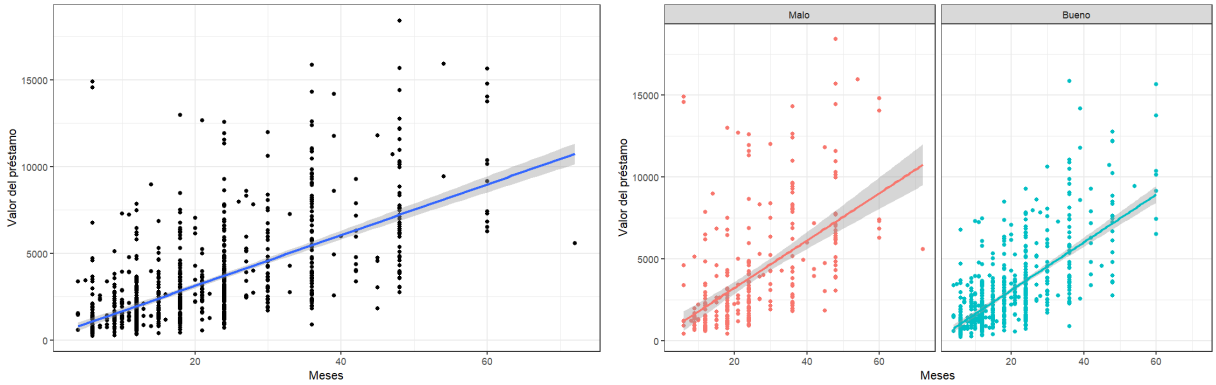


Figura 3.11: Valor del préstamo y meses que dura

Finalmente, vemos otra relación bastante lógica entre el valor del préstamo y los meses que dura; cuanto mayor sea el préstamo, es lógico pensar que el plazo para reembolsarlo será mayor. Vemos como la tendencia es parecida entre ambas clases de clientes.

A continuación, veremos como se comportan las variables numéricas frente a la variable respuesta, y comentaremos aquellas relaciones que nos puedan aportar algo de información.

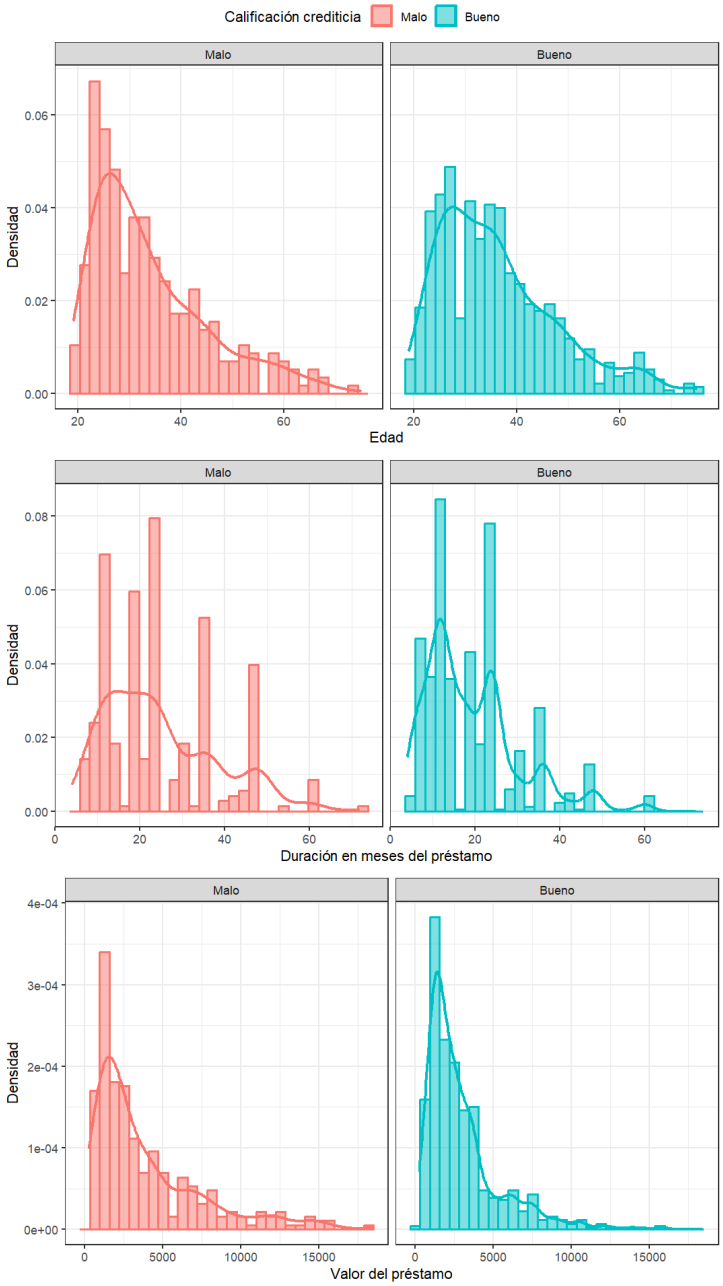


Figura 3.12: Gráficos bivariantes entre variables continuas y clase

Por un lado, la edad del cliente vemos como se distribuye de manera asimétrica hacia la izquierda en ambos casos, es decir, la mayor parte de los clientes es menor de 40 años. Ahora bien, los clientes con peor calificación son más jóvenes que los que tienen buena calificación. En cuanto a la duración del préstamo, no puedo encontrar una relación a simple vista entre las dos clases. Finalmente, el valor del préstamo parece seguir la misma dinámica en ambas clases, aunque podemos ver como la cola hacia la derecha parece más densa en los clientes con mala calificación (piden más prestado que los que tienen buena calificación).

Ahora haremos el mismo análisis para las variables categóricas, y veremos su comportamiento según la clase de cliente que se trate. En primer lugar, veremos las siguientes tablas resumen:

Variable	N	Porcentaje
credit_rating	700	
... Bueno	700	100 %
credit_history	700	
... A30	15	2 %
... A31	21	3 %
... A32	361	52 %
... A33	60	9 %
... A34	243	35 %
savings	700	
... A61	386	55 %
... A62	69	10 %
... A63	52	7 %
... A64	42	6 %
... A65	151	22 %
employment	700	
... A71	39	6 %
... A72	102	15 %
... A73	235	34 %
... A74	135	19 %
... A75	189	27 %
guarantors	700	
... A101	635	91 %
... A102	23	3 %
... A103	42	6 %

Variable	N	Porcentaje
account_status	700	
... A11	139	20 %
... A12	164	23 %
... A13	49	7 %
... A14	348	50 %
personal_status	700	
... A91	30	4 %
... A92	201	29 %
... A93	402	57 %
... A94	67	10 %
property	700	
... A121	222	32 %
... A122	161	23 %
... A123	230	33 %
... A124	87	12 %
other_installments	700	
... A141	82	12 %
... A142	28	4 %
... A143	590	84 %
housing	700	
... A151	109	16 %
... A152	527	75 %
... A153	64	9 %

Variable	N	Porcentaje
purpose	700	
... A40	145	21 %
... A41	86	12 %
... A410	7	1 %
... A42	123	18 %
... A43	218	31 %
... A44	8	1 %
... A45	14	2 %
... A46	28	4 %
... A48	8	1 %
... A49	63	9 %
job	700	
... A171	15	2 %
... A172	144	21 %
... A173	444	63 %
... A174	97	14 %
phone	700	
... A191	409	58 %
... A192	291	42 %
foreign_worker	700	
... A201	667	95 %
... A202	33	5 %

Cuadro 3.5: Tabla resumen de las variables categóricas - clase: bueno

Variable	N	Porcentaje
credit_rating	300	
... Malo	300	100 %
credit_history	300	
... A30	25	8 %
... A31	28	9 %
... A32	169	56 %
... A33	28	9 %
... A34	50	17 %
savings	300	
... A61	217	72 %
... A62	34	11 %
... A63	11	4 %
... A64	6	2 %
... A65	32	11 %
employment	300	
... A71	23	8 %
... A72	70	23 %
... A73	104	35 %
... A74	39	13 %
... A75	64	21 %
guarantors	300	
... A101	272	91 %
... A102	18	6 %
... A103	10	3 %

Variable	N	Porcentaje
account_status	300	
... A11	135	45 %
... A12	105	35 %
... A13	14	5 %
... A14	46	15 %
personal_status	300	
... A91	20	7 %
... A92	109	36 %
... A93	146	49 %
... A94	25	8 %
property	300	
... A121	60	20 %
... A122	71	24 %
... A123	102	34 %
... A124	67	22 %
other_installments	300	
... A141	57	19 %
... A142	19	6 %
... A143	224	75 %
housing	300	
... A151	70	23 %
... A152	186	62 %
... A153	44	15 %

Variable	N	Porcentaje
purpose	300	
... A40	89	30 %
... A41	17	6 %
... A410	5	2 %
... A42	58	19 %
... A43	62	21 %
... A44	4	1 %
... A45	8	3 %
... A46	22	7 %
... A48	1	0 %
... A49	34	11 %
job	300	
... A171	7	2 %
... A172	56	19 %
... A173	186	62 %
... A174	51	17 %
phone	300	
... A191	187	62 %
... A192	113	38 %
foreign_worker	300	
... A201	296	99 %
... A202	4	1 %

Cuadro 3.6: Tabla resumen de las variables categóricas - clase: malo

A continuación, una representación gráfica de las variables categóricas:

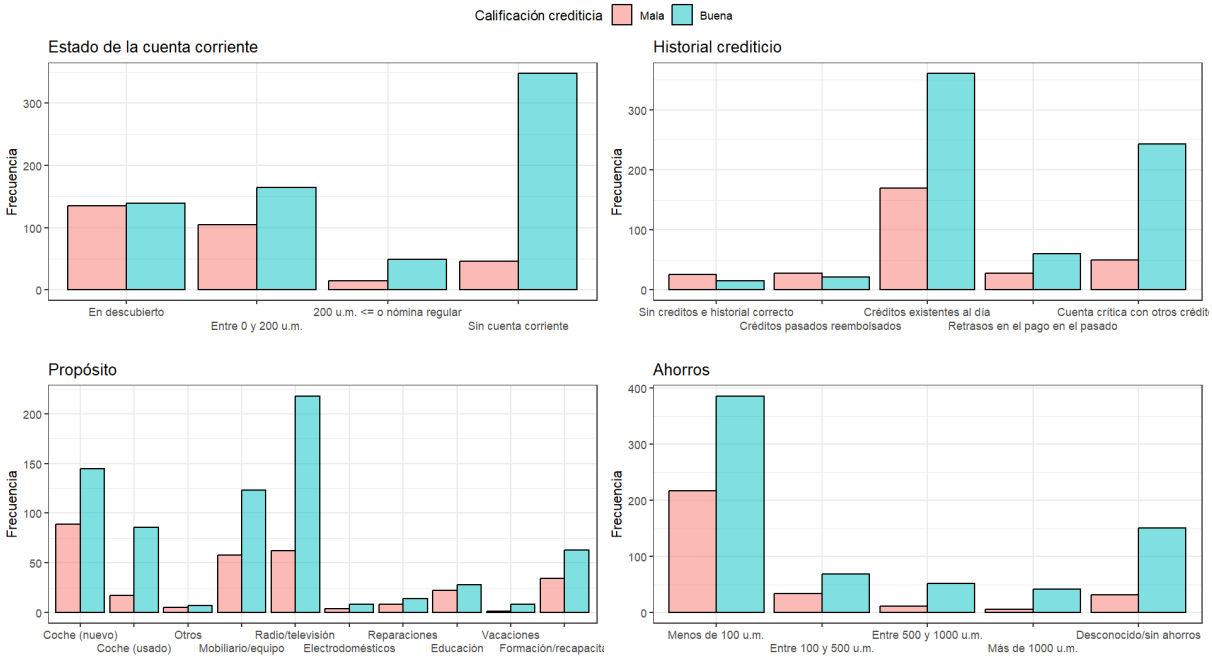


Figura 3.13: Gráficos de barras de las variables *account_status*, *credit_history*, *purpose* y *savings*

En cuanto al estado de la cuenta corriente, vemos una ligera diferencia en la distribución según se trate de la clase: Mientras que el 50 % de los clientes buenos no tienen cuenta corriente, el 85 % de los clientes con mala calificación presentan una cuenta con poco dinero o en números rojos. En cuanto a los ahorros, mientras que el 70 % de los malos clientes tienen menos de 100 u.m. ahorradas, los cerca de la mitad de los clientes con buena calificación tienen más que esa cantidad. En cuanto al propósito del préstamo, no puedo señalar ninguna diferencia significativa entre ambos grupos.

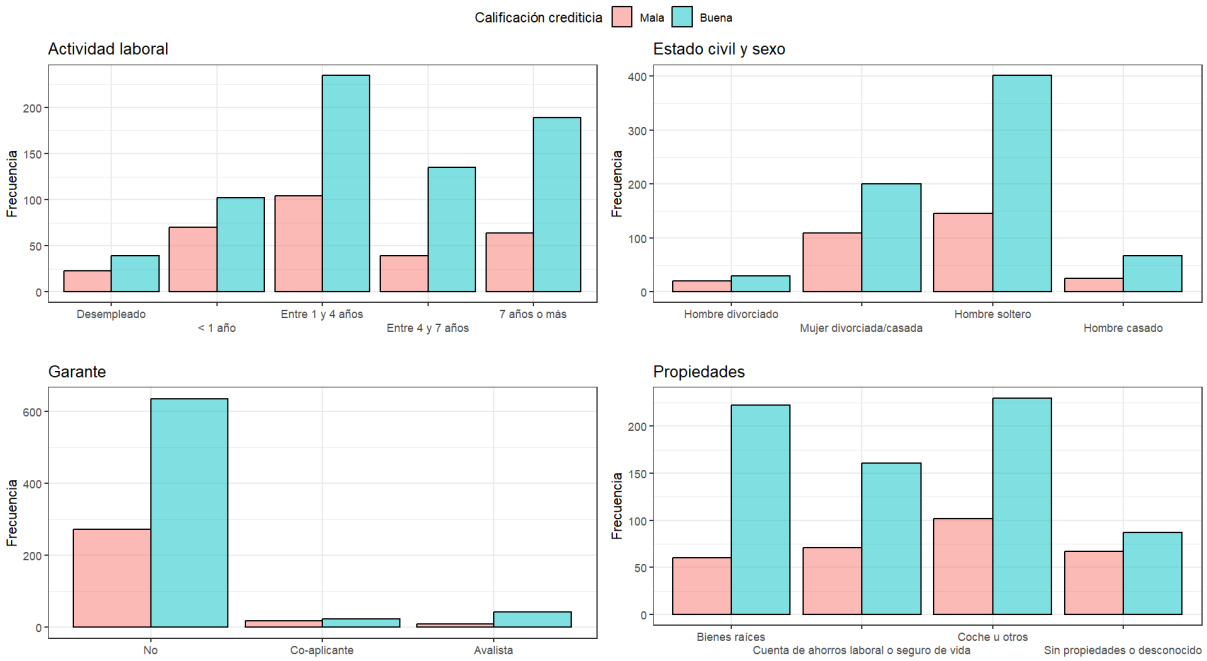


Figura 3.14: Gráficos de barras de las variables *employment*, *personal_status*, *guarantor* y *property*

Respecto a este conjunto de variables, tampoco puedo señalar ninguna diferencia significativa entre ambas clases.

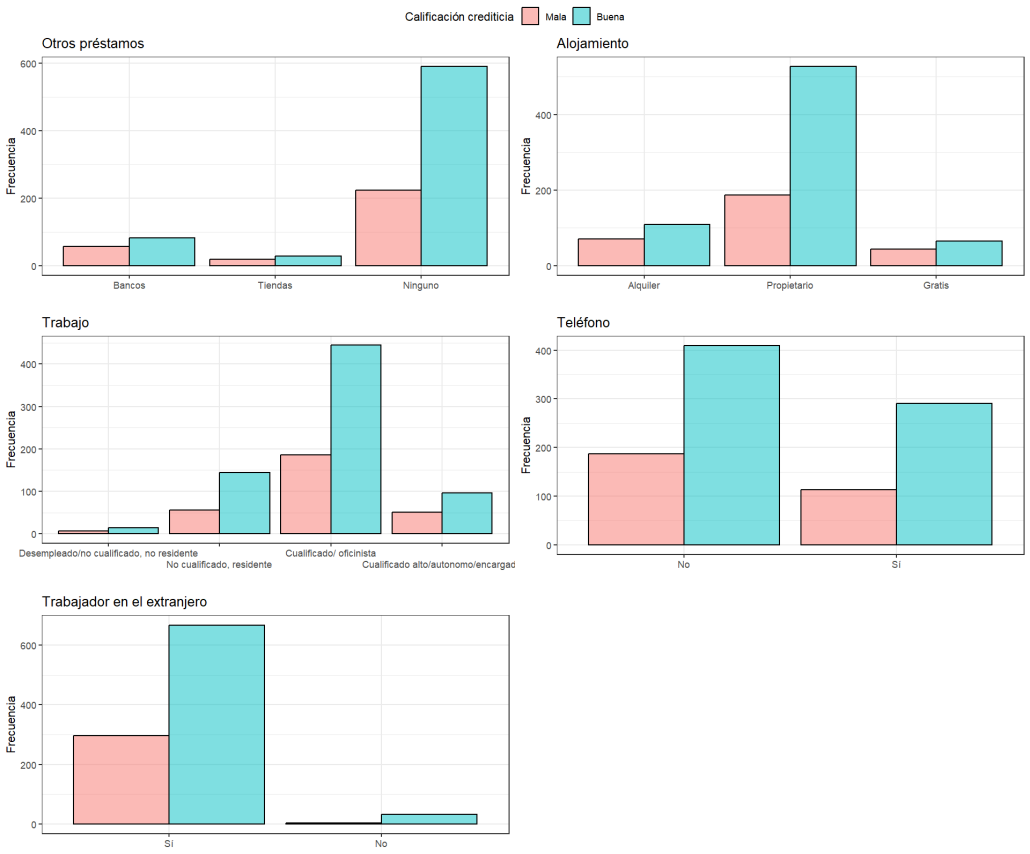


Figura 3.15: Gráficos de barras de las variables *other_installments*, *housing*, *job*, *phone* y *foreign_worker*

Finalmente, tampoco hay diferencias que pueda detectar a simple vista entre ambas clases respecto a estas variables. Por lo que hemos visto hasta ahora, podemos caracterizar el perfil promedio de cada clase:

- Un **cliente con una buena calificación crediticia** se caracteriza por ser un adulto de 36 años, y pedir préstamos de menos de 3000 u.m. y como hemos deducido con el análisis de la proporción de ingresos que dedican a saldar sus deudas, se puede de tratar de personas con altos ingresos. No suelen tener cuenta corriente, y tienen al menos 100 u.m. ahorradas.
- Un **cliente con una mala calificación crediticia** se caracteriza por ser algo más joven, y los valores de los préstamos que pide son algo más altos (y a consecuencia, con un periodo de madurez más alto). Además, dedican menor proporción de sus ingresos para saldar sus deudas, y no tienen apenas ahorros.

3.3. Preprocesamiento de los datos

Ahora, haremos las modificaciones necesarias para trabajar con la base de datos. Al no haber valores faltantes, no tenemos que imputarlos. Por otro lado, las variables categóricas que presenten muchas clases con baja frecuencia, trataré de fusionarlas con otros grupos parecidos. También llevaré a cabo un proceso de selección de variables (pero solo será de carácter "ilustrativo" para que veamos como se pueden excluir variables irrelevantes, ya que como no tengo tantas variables en la base de datos, puedo dejarlo en manos de cada modelo el hecho de excluir variables sin generar demasiado esfuerzo computacional) usando un **análisis de componentes principales** (PCA). Las variables numéricas están en diferentes escalas, pero normalizaré los datos en el momento de ejecutar cada algoritmo. Lo que sí haré será utilizar la codificación *One-hot* para transformar las variables categóricas en numéricas, creando una variable artificial para cada categoría de la columna.

Merging o fusión

Para la variable *purpose* que nos indica el propósito del crédito, he decidido juntar las categorías `A42,A44,A45` en una nueva `A411` que engloba tareas de mantenimiento del hogar. Por otro lado, las categorías `A46 y A48` las fusiono para crear la categoría `A412` que engloba la formación y educación. En el resto de variables no he considerado hacer ninguna modificación de este estilo. Así se ve nuestra nueva variable `purpose`:

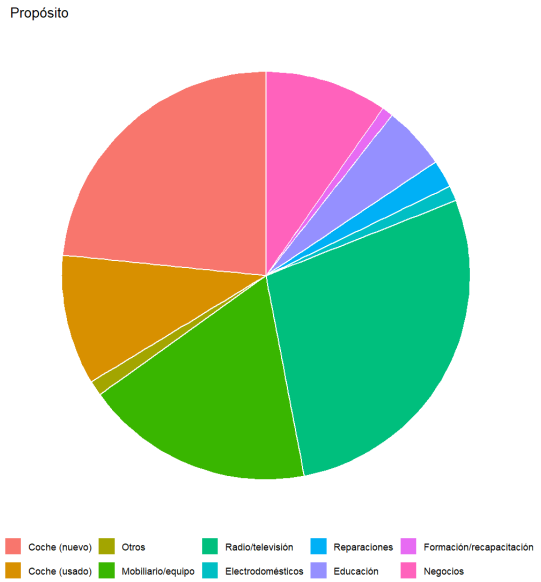


Figura 3.16: Gráfico de pastel de la variable *purpose* modificada

One-hot Encoding

Para la ejecución de nuestros algoritmos, es conveniente que toda nuestra base de datos tenga solamente variables numéricas, y esto lo podemos utilizar usando la metodología de "*One-hot Encoding*", que se trata de crear una variable artificial o *dummy* para cada una de las clases de una variable categórica. Para ello, usaré el paquete de `caret` que incluye una función para este procedimiento. El código a usar es el siguiente:

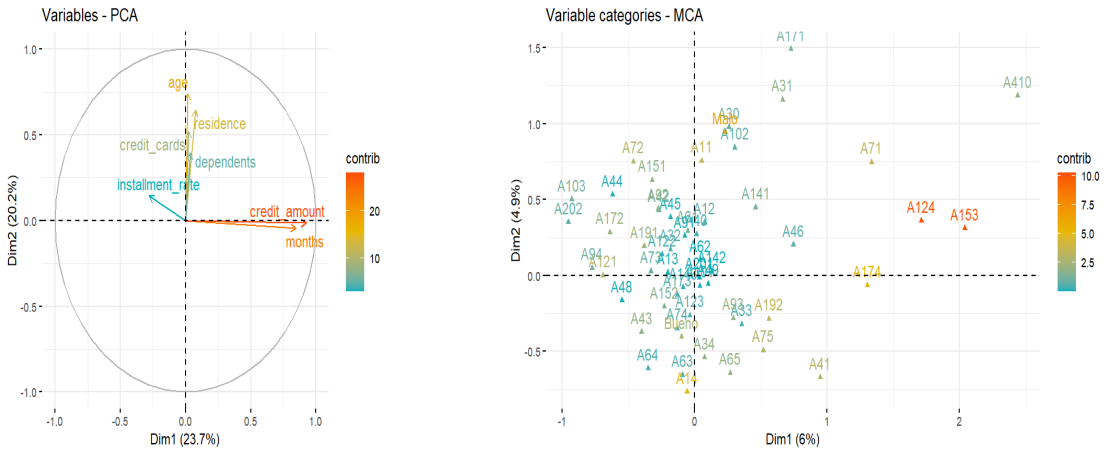
```
library(caret)
dmy <- dummyVars(" ~ .", data = data[,-21])
trsf <- data.frame(predict(dmy, newdata = data[,-21]))
trsf = data.frame(trsf,data$credit_rating)
saveRDS(trsf, file = "final_data.Rds")
```

Por lo tanto, una muestra del resultado sería el siguiente:

	account_statusA11	account_statusA12	account_statusA13	account_statusA14	months
1	1.00	0.00	0.00	0.00	6.00
2	0.00	1.00	0.00	0.00	48.00
3	0.00	0.00	0.00	1.00	12.00
4	1.00	0.00	0.00	0.00	42.00
5	1.00	0.00	0.00	0.00	24.00
6	0.00	0.00	0.00	1.00	36.00

PCA y MCA

Como he comentado anteriormente, aunque la base de datos utilizada no es tan extensa como para necesitar una reducción de dimensionalidad, simularé el procedimiento, ya que es muy común para ahorrar coste computacional. Para ello, llevaré a cabo un **Análisis de Componentes Principales (PCA)** y un **Análisis de Correspondencias Múltiples (MCA)**. Para empezar, el Análisis de Componentes Principales o PCA por sus siglas en inglés (*Principal Component Analysis*), es una técnica de reducción de dimensionalidad. El PCA transforma un conjunto de observaciones de variables (posiblemente correlacionadas) en un conjunto de valores de variables linealmente no correlacionadas llamadas componentes principales. El primer componente principal (PC1) tiene la mayor varianza posible, y cada componente sucesivo (PC2, PC3, ...) tiene la mayor varianza posible bajo la restricción de que sea ortogonal a los componentes anteriores (IBM, 2023). Se seleccionan tantos componentes hasta que al menos expliquen el 80 % de la varianza total de nuestro conjunto de datos. Por otro lado, el Análisis de Correspondencias Múltiples o MCA por sus siglas en inglés (*Multiple Correspondence Analysis*), cumple la misma función pero usando variables categóricas (IBM, 2024). Cabe destacar que si usamos el método PCA sobre variables categóricas codificadas usando *One-hot encoding*, llegaríamos a resultados similares que si llevamos a cabo un MCA. Una vez hecho, visualizaremos como se comportan las variables en las dos primeras dimensiones (que son las dos primeras que más varianza explican).



(a) PCA (b) MCA

Figura 3.17: Representación gráfica del PCA y MCA

Los criterios son parecidos en ambas análisis: Por un lado, cuanto más rojo, más contri-

buye la variable a explicar la varianza del conjunto, y cuanto más azul, más irrelevante es para ese par de dimensiones. Ahora bien, cuanto más cerca del origen estén, menos varianza explicaran en esas dimensiones, y si se da un solapamiento, querrá decir que ambas variables explican prácticamente lo mismo. En el PCA, podemos ver como las variables *credit_amount* y *months* van en la misma dirección y están cerca de solaparse; aun así, aunque un préstamo mayor normalmente se devuelve en más tiempo, no considero adecuado sustituir una variable por la otra. Lo mismo sucede con las variables *age* y *residence*, que aunque ser mayor puede implicar a veces llevar viviendo más tiempo en tu actual vivienda, no tiene necesariamente por qué. Por el lado de las categóricas, aunque hay un solapamiento en el origen que puede significar poca relevancia de esas categorías, podríamos seleccionarlas y decidir entre fusionarlas o eliminarlas. cv 10 3 repeticiones

3.4. Algoritmos

Finalmente, en este apartado ejecutaremos los algoritmos que he seleccionado previamente, interpretandolos y validándolos de manera individual, e intentando cuando sea posible explicar qué variables ayudan al modelo a tomar decisiones. Por lo general, he usado un procedimiento de *tunning* [2] de 10 repeticiones (excepto con el algoritmo `xgBoost` y el kernel polinómico del algoritmo SVM puesto que suponia un esfuerzo computacional demasiado elevado para mi equipo, por lo que lo he reducido a 2 repeticiones en estos casos), y una metodología de validación cruzada de 10 pliegues y 3 repeticiones. Todo el código utilizado está en el **anexo**, junto con únicamente las salidas de los modelos seleccionados como mejores. Para todos los algoritmos, he utilizado el paquete `caret`, puesto que tiene una interfaz intuitiva y personalizable que conecta con paquetes más especificos para cada algoritmo. He ejecutado cada algoritmo con las diferentes técnicas que hemos observado contra los datos desbalanceados, y utilizaré las métricas mencionadas en los apartados anteriores junto un gráfico de la curva ROC para seleccionar el más óptimo. La clase positiva para todos los algoritmos es una calificación crediticia «Mala».

[2] El algoritmo se ejecuta tantas veces como se especifique modificando los parámetros de entrada del modelo y escoge la mejor combinación

3.4.1. Máquinas de Vectores de Soporte (SVM)

Empezaremos con el algoritmo de *Support Vector Machines*, donde he utilizado tres kernels diferentes: el lineal (por defecto), el radial y el polinómico.

Kernel Lineal

	SVMLineal	Oversampling	Undersampling	SMOTE	ROSE
Kappa	0.242	0.414	0.34	0.421	0.504
Accuracy	0.73	0.74	0.695	0.75	0.765
Balanced Accuracy	0.602	0.719	0.687	0.717	0.785
F1	0.386	0.606	0.567	0.603	0.68
Recall	0.283	0.667	0.667	0.633	0.833
Precision	0.607	0.556	0.494	0.576	0.575
Neg Pred Value	0.75	0.844	0.832	0.836	0.912
Pos Pred Value	0.607	0.556	0.494	0.576	0.575
Specificity	0.921	0.771	0.707	0.8	0.736
Sensitivity	0.283	0.667	0.667	0.633	0.833

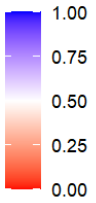


Figura 3.18: Tabla de validación para el kernel lineal

Como podemos ver, el algoritmo sin usar ningun tipo de tecnica contra el desbalance, tiene un `accuracy` moderadamente bueno, pero el resto de métricas rinden bastante mal; esto es principalmente provocado por el desbalance, y lo podemos comprobar viendo como presenta una capacidad para clasificar los malos clientes (sensibilidad) del `0.283`. Vemos que al usar remuestreo la situación mejora, siendo la técnica de ROSE la que presenta los mejores valores para este kernel: ahora vemos no solo una `accuracy` superior de `0.765`, sino que el `Recall` o sensibilidad se dispara hasta `0.833`. Cabe destacar también la sensibilidad de los modelos basados en sobremuestreo simple y submuestreo, pero seleccionaré el que usa ROSE por que en promedio, es mejor.

Kernel Radial

Estos han sido los resultados para el kernel radial:

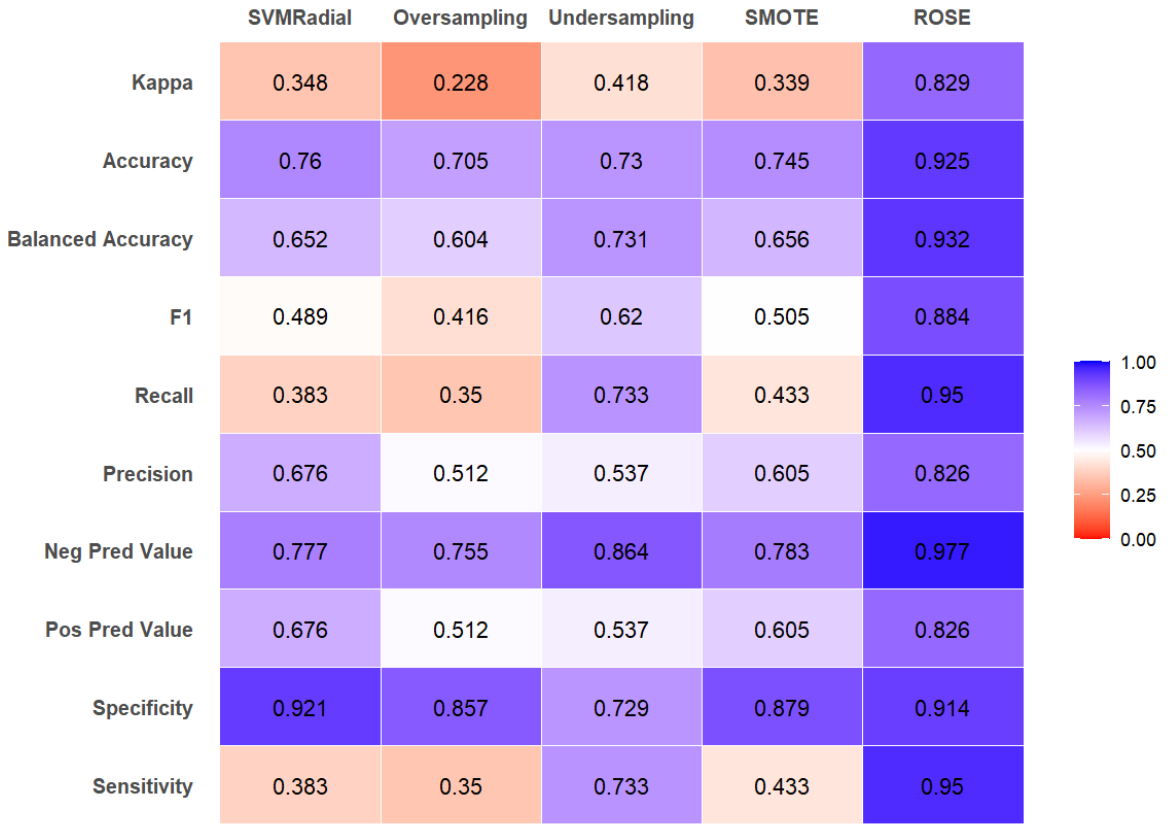


Figura 3.19: Tabla de validación para el kernel radial

Observamos que el modelo base se comporta mejor que el anterior kernel, presentando una `accuracy` del `0.76` y una sensibilidad del `0.383`, aunque siguen siendo valores muy pobres. Lo que salta a la vista son los valores excepcionales del método de remuestreo ROSE, presentando valores por encima del `0.9` en la `accuracy`, y sorprendentemente también en la especificidad y sensiblidad. Es un modelo muy bueno, siempre y cuando no se trate de un sobreajuste a los datos. En este caso, los valores de sensibilidad y especificidad solo destacan usando ROSE y submuestreo, aunque una vez más escogeré ROSE como modelo de kernel radial.

Kernel Polinómico

Estos han sido los resultados para el kernel polinómico:

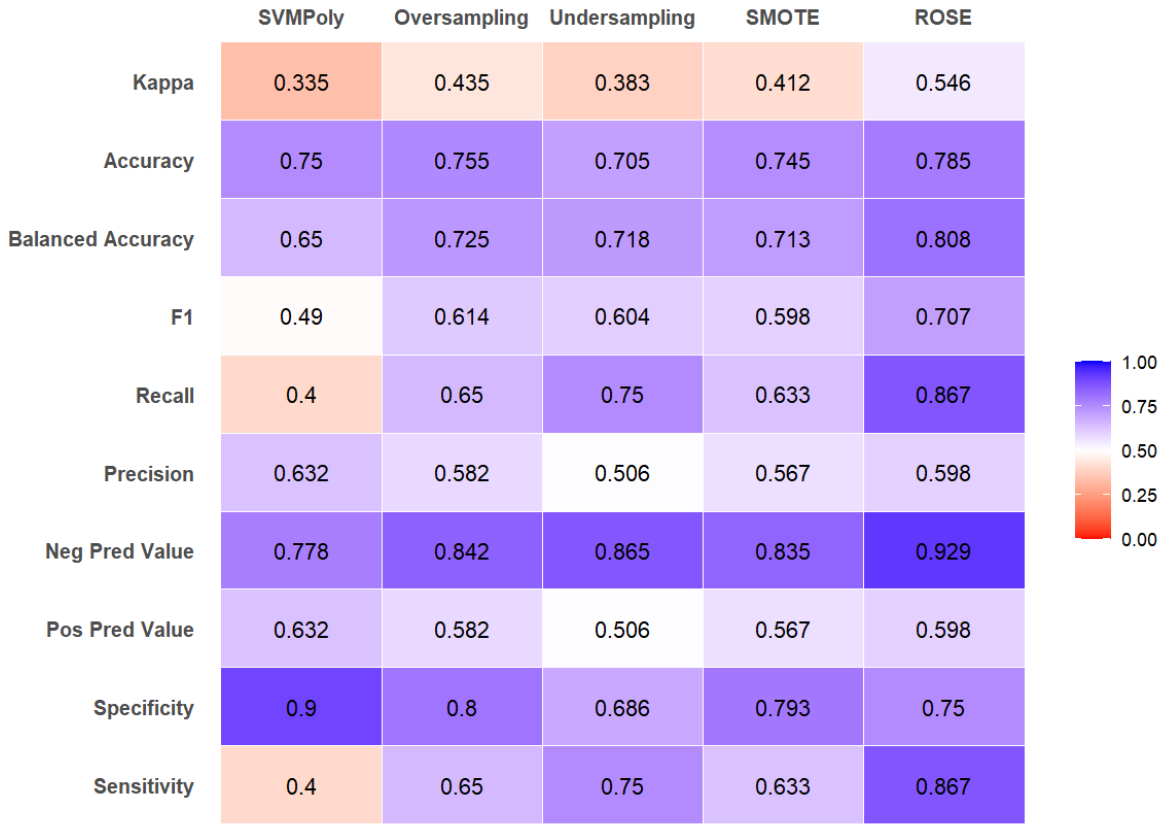


Figura 3.20: Tabla de validación para el kernel polinómico

Este kernel aunque presenta valores superiores al lineal, sigue siendo mediocre en su forma desbalanceada. No supera de manera notoria en ningún aspecto al kernel lineal, aunque podemos ver como todas las tecnicas contra los datos desbalanceados presentan valores mejores en promedio. Una vez más, destaca la técnica ROSE, pero esta vez sin tanta diferencia.

Para finalizar, visualizaremos la curva ROC del kernel radial, puesto que es el que mejor resultados ha tenido. Las curvas ROC de los otros dos kernel utilizados se podrán consultar en el **anexo**.

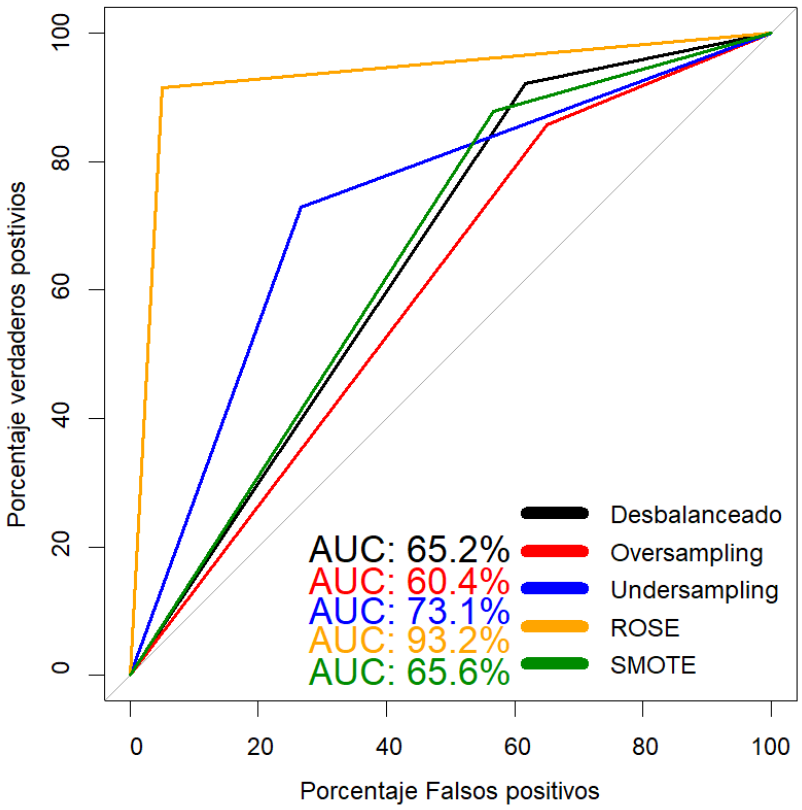


Figura 3.21: Curva ROC del kernel radial

Como podemos observar, la curva ROC del método ROSE es la mejor con diferencia, ya que presenta un AUC del `93.2%`. El resto de kernel han presentado valores del AUC más estables y parecidos entre cada técnica contra el desbalance, pero el radial con el método ROSE es el que finalmente selecciono debido a su buen comportamiento.

3.4.2. Regresión Logística (Logit)

Como he explicado anteriormente, en la regresión logística he implementado la metodología stepwise, lo cual excluirá o añadirá las variables al modelo con tal de minimizar el error (minimizar el AIC). El modelo resultante tiene la siguiente forma:

```
    Coefficients:
                          Estimate Std. Error z value Pr(>|z|)
(Intercept)              -0.01400    0.07789  -0.180 0.857350
account_status.A11       -0.80334    0.09812  -8.187 2.67e-16 ***
account_status.A12       -0.52573    0.09281  -5.665 1.47e-08 ***
account_status.A13       -0.13814    0.07771  -1.778 0.075472 .
months                   -0.25873    0.09065  -2.854 0.004315 **
credit_history.A30       -0.19535    0.09658  -2.023 0.043101 *
credit_history.A31       -0.24798    0.08953  -2.770 0.005609 **
credit_history.A32       -0.23444    0.10531  -2.226 0.026005 *
purpose.A41               0.47894    0.08868   5.401 6.64e-08 ***
purpose.A410              0.17370    0.08517   2.039 0.041418 *
purpose.A411              0.40211    0.09024   4.456 8.35e-06 ***
purpose.A412             -0.14585    0.08562  -1.703 0.088500 .
purpose.A43               0.35265    0.09197   3.834 0.000126 ***
credit_amount            -0.43916    0.10387  -4.228 2.36e-05 ***
savings.A64               0.15242    0.08840   1.724 0.084670 .
employment.A72           -0.15590    0.08272  -1.885 0.059470 .
employment.A74            0.50049    0.08743   5.724 1.04e-08 ***
installment_rate         -0.29071    0.08333  -3.488 0.000486 ***
personal_status.A91      -0.14497    0.08023  -1.807 0.070757 .
personal_status.A92      -0.21401    0.08408  -2.545 0.010916 *
guarantors.A101          -0.36756    0.11497  -3.197 0.001389 **
guarantors.A102          -0.42116    0.11383  -3.700 0.000216 ***
property.A121             0.19104    0.09090   2.102 0.035575 *
property.A123            -0.14020    0.09114  -1.538 0.123979
age                       0.14783    0.08413   1.757 0.078882 .
other_installments.A141 -0.33178    0.08143  -4.074 4.61e-05 ***
housing.A151             -0.25917    0.08402  -3.084 0.002039 **
credit_cards             -0.15908    0.09749  -1.632 0.102728
phone.A191               -0.13159    0.08300  -1.586 0.112844
foreign_worker.A201      -0.19100    0.09719  -1.965 0.049402 *
---
```

Para interpretar los coeficientes tenemos que entender que la clase positiva es ser una persona con una calificación crediticia mala, por lo que un coeficiente positivo, aumenta la probabilidad de serlo. En cambio, un coeficiente negativo disminuye la probabilidad, por lo que esos coeficientes aportarán a ser un buen cliente. Por lo tanto, cada coeficiente:

- **(Intercept)**: -0.014. Este es el término constante del modelo. No tiene una interpretación directa en términos de las variables predictoras.
- **account_status.A11*****: -0.803. No tener ninguna cuenta corriente disminuye la probabilidad de tener una mala calificación crediticia.
- **account_status.A12*****: -0.526. Tener una cuenta corriente con saldo menor a 200 u.m. disminuye la probabilidad de tener una mala calificación crediticia.
- **months****: -0.259. Cada mes adicional de duración del crédito disminuye la probabilidad de tener una mala calificación crediticia., ya que se puede considerar menos arriesgado debido a que el prestatario tiene más tiempo para pagar (credito diluido).
- **credit_history.A30***: -0.195. Un buen historial crediticio sin haber tomado creditos disminuye la probabilidad de tener una mala calificación crediticia.
- **credit_history.A31****: -0.248. Un buen historial crediticio habiendo pagado a tiempo todos los creditos y reflejando así un buen comportamiento financiero, disminuye la probabilidad de tener una mala calificación crediticia.
- **credit_history.A32***: -0.234. Un buen historial crediticio pagando actualtmente a tiempo todos los creditos y reflejando así un buen comportamiento financiero, disminuye la probabilidad de tener una mala calificación crediticia.
- **purpose.A41*****: 0.479. Tener el propósito A41 (compra de automóvil nuevo) aumenta la probabilidad de tener una mala calificación crediticia.
- **purpose.A410***: 0.174. Tener el propósito A410 (compra de electrodomésticos) aumenta la probabilidad de tener una mala calificación crediticia.
- **purpose.A411*****: 0.402. Tener el propósito A411 (mantenimiento del hogar) aumenta la probabilidad de tener una mala calificación crediticia.
- **purpose.A43*****: 0.353. Tener el propósito A43 (compra de automóvil usado) aumenta la probabilidad de tener una mala calificación crediticia.

- **credit_amount***: -0.439. Cada unidad adicional en la cantidad de crédito disminuye la probabilidad de tener una mala calificación crediticia, ya que los préstamos más grandes suelen ser otorgados a individuos con mejor perfil.
- **employment.A74***: 0.500. Estar empleado entre 4 y 7 años en el actual trabajo aumenta la probabilidad de tener una mala calificación crediticia.
- **installment_rate***: -0.291. Cuanto mayor sea el porcentaje de la renta disponible para pagar deudas, disminuira la probabilidad de tener una mala calificación crediticia.
- **personal_status.A92***: -0.214. Ser un hombre casado disminuye la probabilidad de tener una mala calificación crediticia, ya que puede reducir el riesgo percibido dada a su posible responsabilidad financiera.
- **guarantors.A101****: -0.368. No tener garante disminuye la probabilidad de tener una mala calificación crediticia.
- **guarantors.A102***: -0.421. Tener garante disminuye la probabilidad de tener una mala calificación crediticia.
- **property.A121***: 0.191. Tener una propiedad inmobiliaria aumenta la probabilidad de tener una mala calificación crediticia, si se interpreta como mayor carga financiera.
- **other_installments.A141***: -0.332. No tener otras deudas disminuye la probabilidad de tener una mala calificación crediticia, ya que a grandes rasgos, reduce la carga financiera general.
- **housing.A151****: -0.259. Tener vivienda propia disminuye la probabilidad de tener una mala calificación crediticia. Esto se puede deber a que una vivienda propia refleja estabilidad financiera.
- **foreign_worker.A201***: -0.191. Ser un trabajador extranjero disminuye la probabilidad de tener una mala calificación crediticia.

Observemos entonces los resultados que nos ha dado la regresión logística:

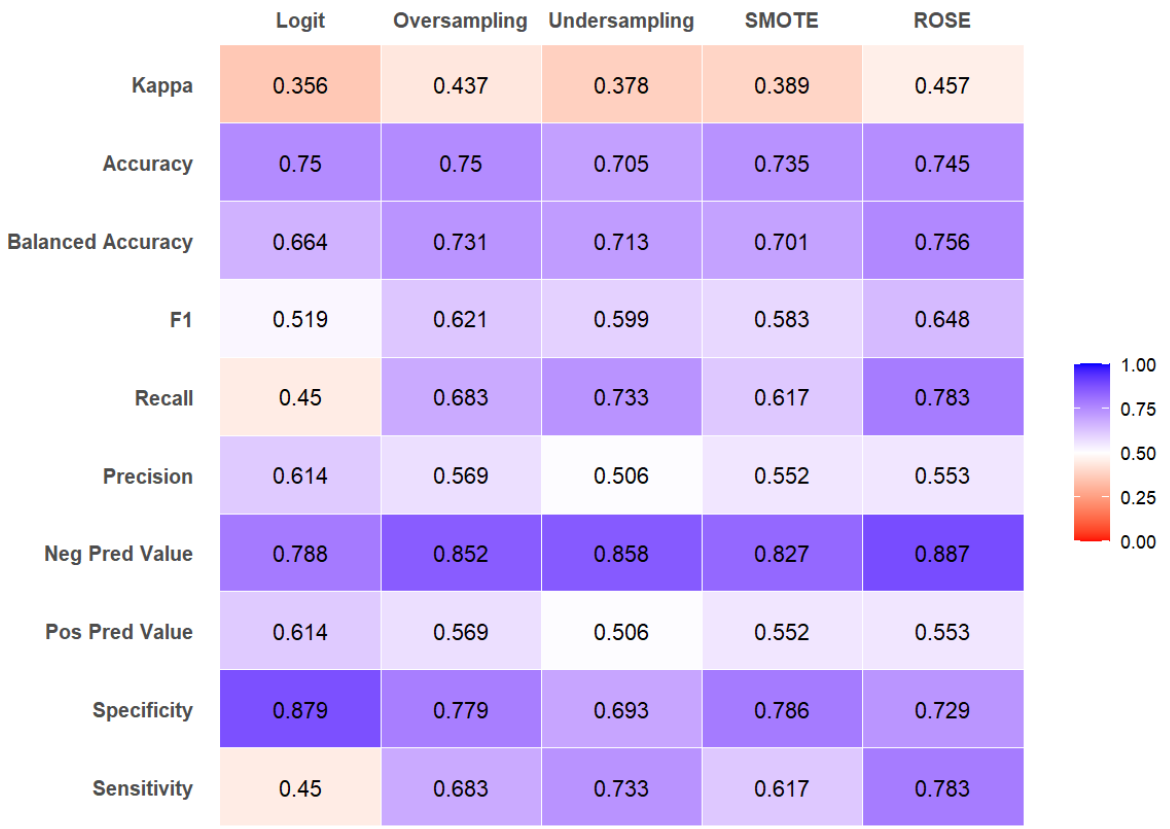


Figura 3.22: Tabla de validación para la regresión logística

En el caso de la regresión logística, podemos observar como presenta una `accuracy` de `0.75`, aunque una sensibilidad bastante mediocre por debajo del `0.5` . Ahora bien, aunque usando técnicas contra el desbalance vemos como mejoran los resultados, en promedio es un cambio bastante ligero respecto al algoritmo anterior. Aun así, vemos como se equilibran los valores de especificidad y sensibilidad, comportando resultados moderadamente buenos a la hora de clasificar correctamente tanto a buenos como malos clientes. De la misma manera que el anterior caso, seleccionaremos el algoritmo bajo la técnica ROSE, ya que presenta un comportamiento mejor en la mayoria de métricas.

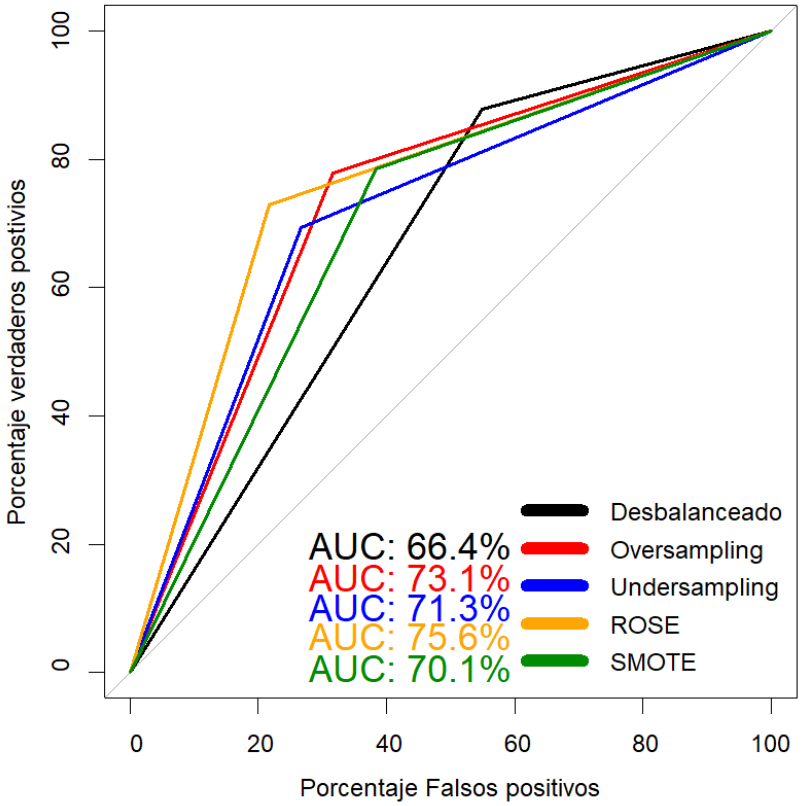


Figura 3.23: Curva ROC para la regresión logística

Si observamos las curvas ROC, podemos asegurarnos de que la técnica de ROSE es la que presenta un AUC mayor, aunque el resto de técnicas como el sobremuestreo simple también presenta valores acertados.

3.4.3. Naïve Bayes

Figura 3.24: Tabla de validación para Naïve Bayes

Podemos observar un comportamiento bastante curioso con el modelo desbalanceado: Ha decidido clasificar a todos los clientes como buenos, asegurando así un `accuracy` de `0.7`. No obstante, es el comportamiento que en apartados anteriores he explicado que se puede dar ante datos desbalanceados, dejando de lado la clase que realmente nos interesa identificar. Con el submuestreo aplicado, destacan las técnicas simple y ROSE, ya que no solo aumenta el `accuracy` rondando el `0.8`, sino que eoncontramos valores mejores tanto en la especificidad como la sensiblidad, presentando el método ROSE los mejores resultados con `0.829` y `0.683` respectivamente.

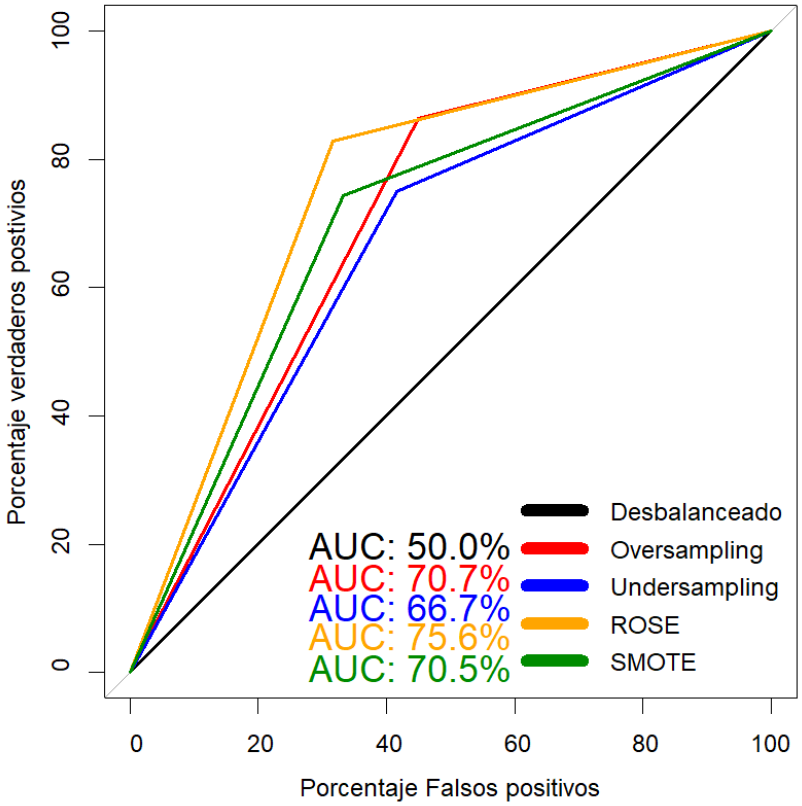


Figura 3.25: Curva ROC para para Naïve Bayes

Una vez mas, la metodología ROSE presenta la mejor curva ROC y valor AUC, por lo que seleccionaremos este modelo. Tal y como hemos visto en la tabla anterior, el modelo desbalanceado es el peor modelo posible estando por encima de la diagonal.

3.4.4. k-Nearest Neighbours (kNN)

El modelo final de este algoritmo tras el tunning de parámetros ha sido $k = 7$, es decir, se ha determinado la clase de la predicción en base a sus 7 observaciones más cercanas

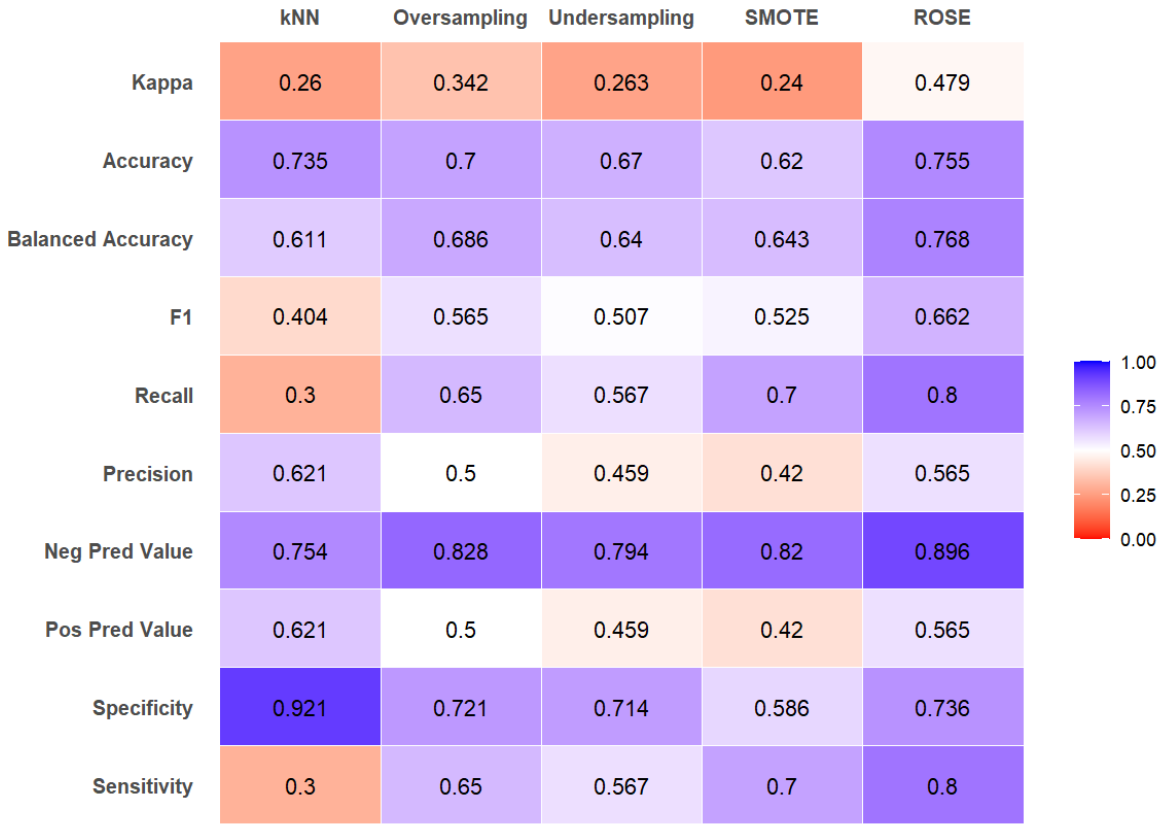


Figura 3.26: Tabla de validación para kNN

Vemos como el modelo desbalanceado presenta un `accuracy` de `0.735`, pero principalmente por lo bien que clasifica la clase mayoritaria que son los buenos clientes, ya que solo acierta con una proporción de `0.3` al detectar mala calificación crediticia. Utilizando las técnicas de muestreo obtenemos resultados mejores y más equilibrados entre ambas clases, siendo el sobremuestreo simple con una especificidad de `0.721` y una sensibilidad de `0.65`, y ROSE con especificidad de `0.736` y sensibilidad de `0.8` los mejores modelos.

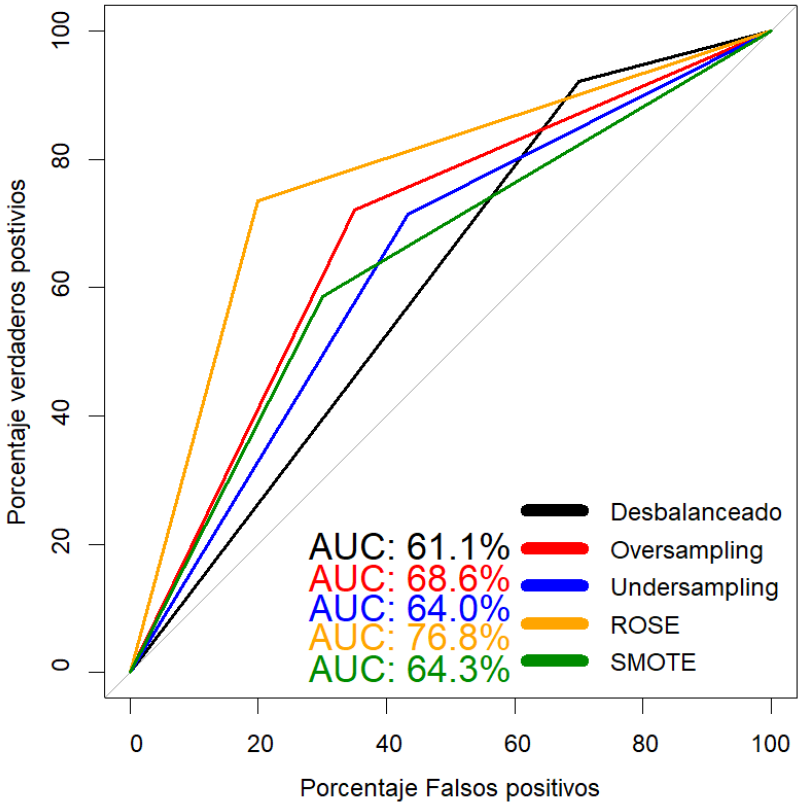


Figura 3.27: Curva ROC para kNN

Como vemos, el método ROSE vuelve a presentar una vez más los mejores valores de la curva ROC y del AUC, por lo que será el modelo seleccionado en este algoritmo.

3.4.5. Árboles de decisión (CART) y Bosques aleatorios (Random Forests)

Empezaremos ejecutando el algoritmo CART, ya que es más simple. En el **anexo** podemos encontrar el gráfico resultante.

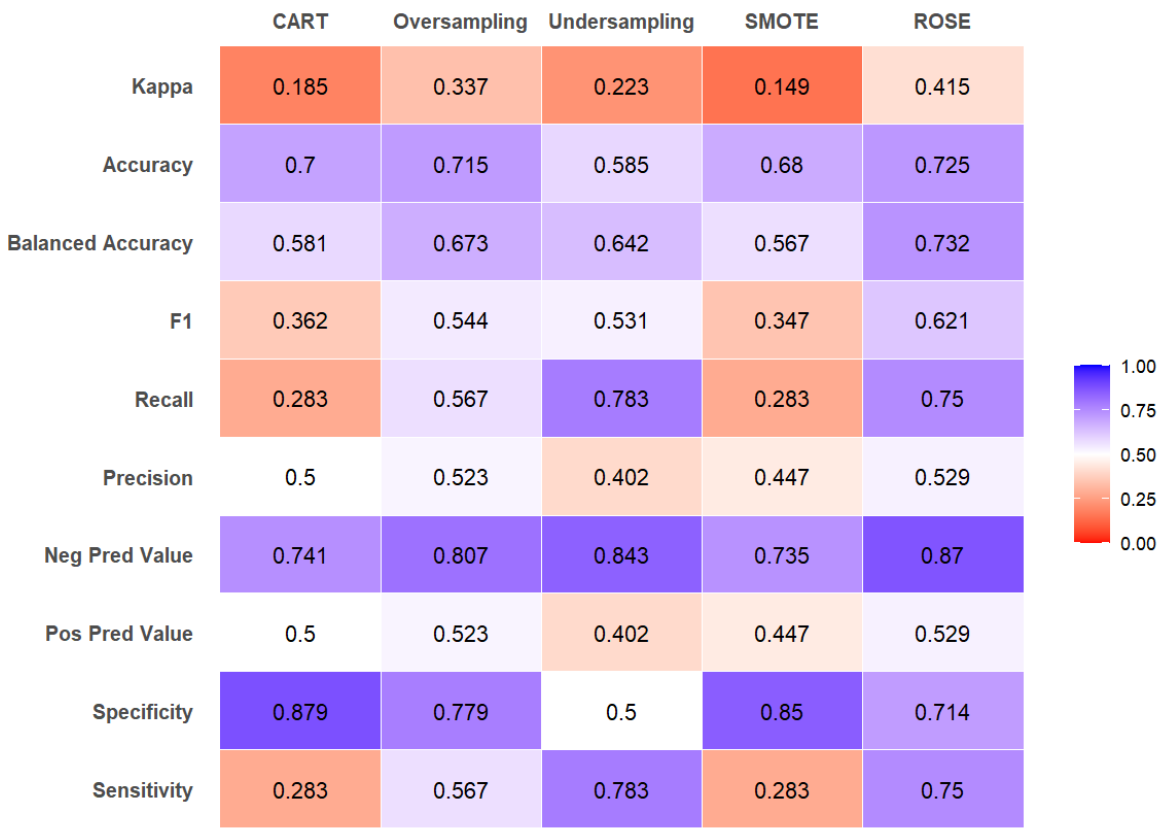


Figura 3.28: Tabla de validación para el árbol de decisión

El modelo desbalanceado presenta unos valores mediocres (aunque mejor que el algoritmo de naive bayes en su versión desbalanceada), con una `accuracy` del `0.7` y una sensibilidad del `0.283`, presentando dificultades para clasificar la clase minoritaria. Podemos observar como utilizando las técnicas de submuestreo y SMOTE presenta valores incluso peores que el modelo desbalanceado con `accuracy` y sensibilidades por debajo. Por otro lado, el sobremuestreo y la técnica ROSE estan bastante a la par con resultados decentes, siendo ROSE con una `accuracy` del `0.725` y una sensibilidad y especificidad de `0.75` y `0.714` respectivamente el mejor modelo.

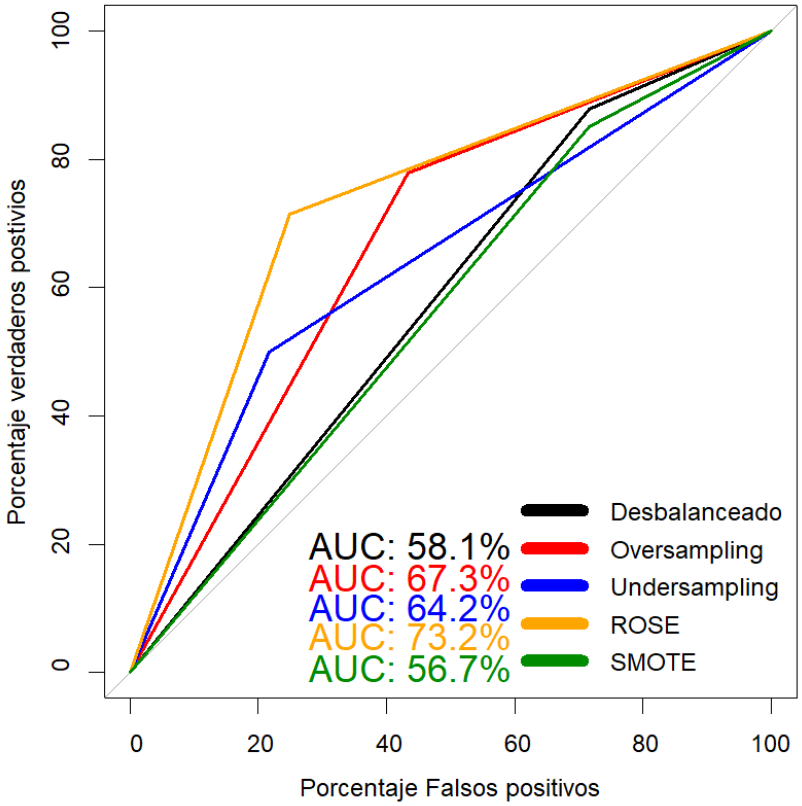


Figura 3.29: Curva ROC para el árbol de decisión

Podemos ver reflejado en la curva ROC las conclusiones anteriores, siendo el mejor una vez más el modelo que utiliza ROSE como técnica de balanceo.

De cara al algoritmo de *Random Forest*, podemos ver un comportamiento similar, pero con resultados mucho mejores.

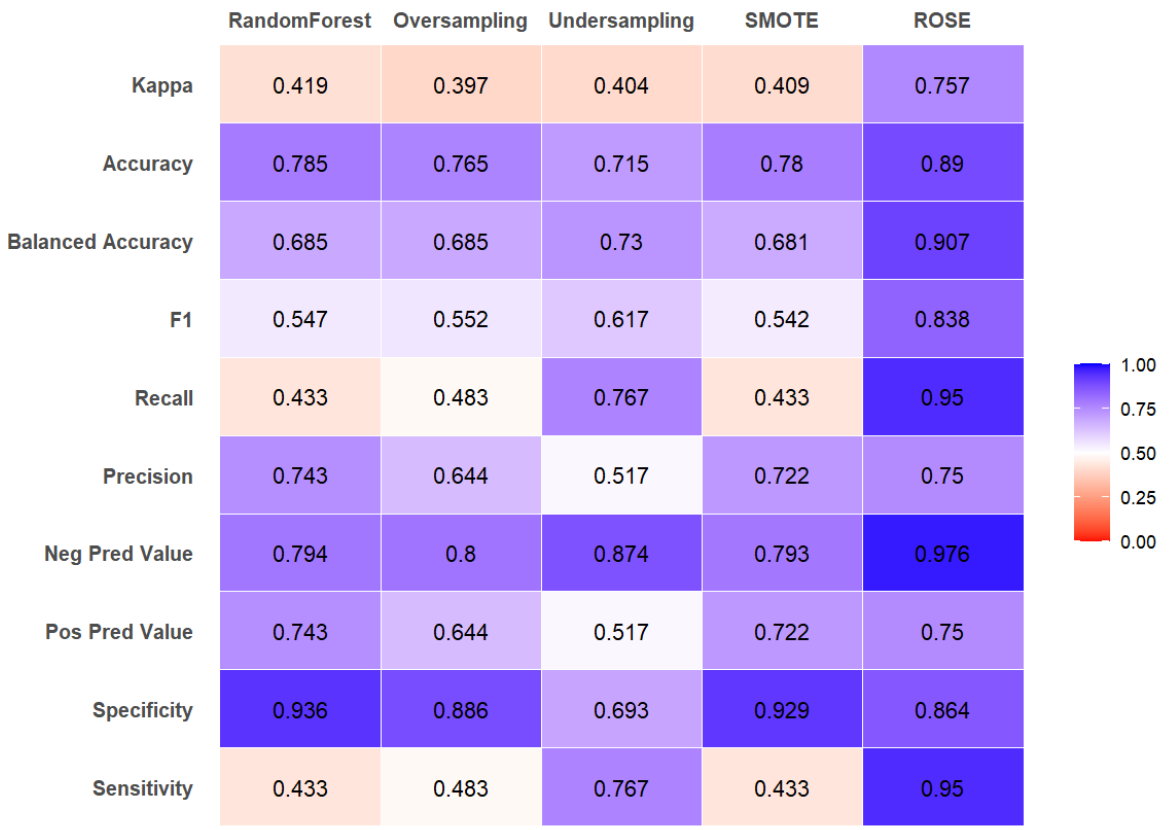


Figura 3.30: Tabla de validación para *Random Forest*

El modelo desbalanceado presenta unos resultados mejores que el anterior caso, con una `accuracy` de `0.785` y una sensibilidad del `0.433`, presentando dificultades para clasificar la clase minoritaria. El resto de técnicas ante desbalances presenta un comportamiento similar a excepción de la técnica ROSE, permitiendo al algoritmo tener resultados excepcionales. En este último, podemos ver una `accuracy` cerca del `0.9`, y una sensibilidad del `0.95`, siendo probablemente la mejor capacidad de clasificación de clase minoritaria juntamente con el kernel radial del SVM.

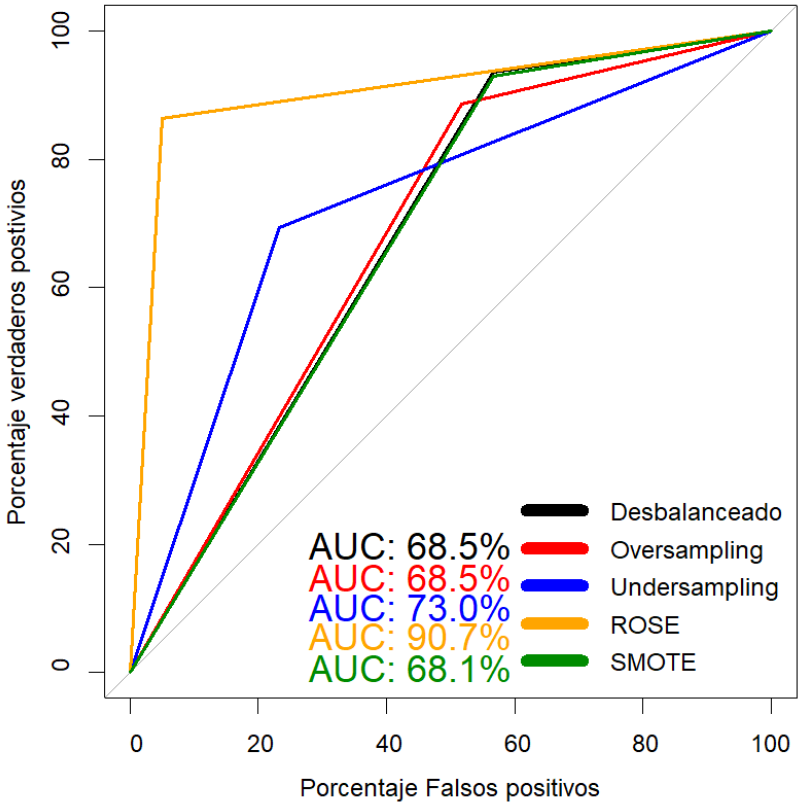


Figura 3.31: Curva ROC para *Random Forest*

La curva ROC corrobora nuestro análisis, ya que la curva del método ROSE está cerca de la esquina superior izquierda y presentando un AUC del `90.7%`. Seleccionaremos este modelo para el algoritmo de *Random Forest*.

3.4.6. XGBoost

Finalmente llegamos al último algoritmo que ejecutaré en mi trabajo, el XGBoost.

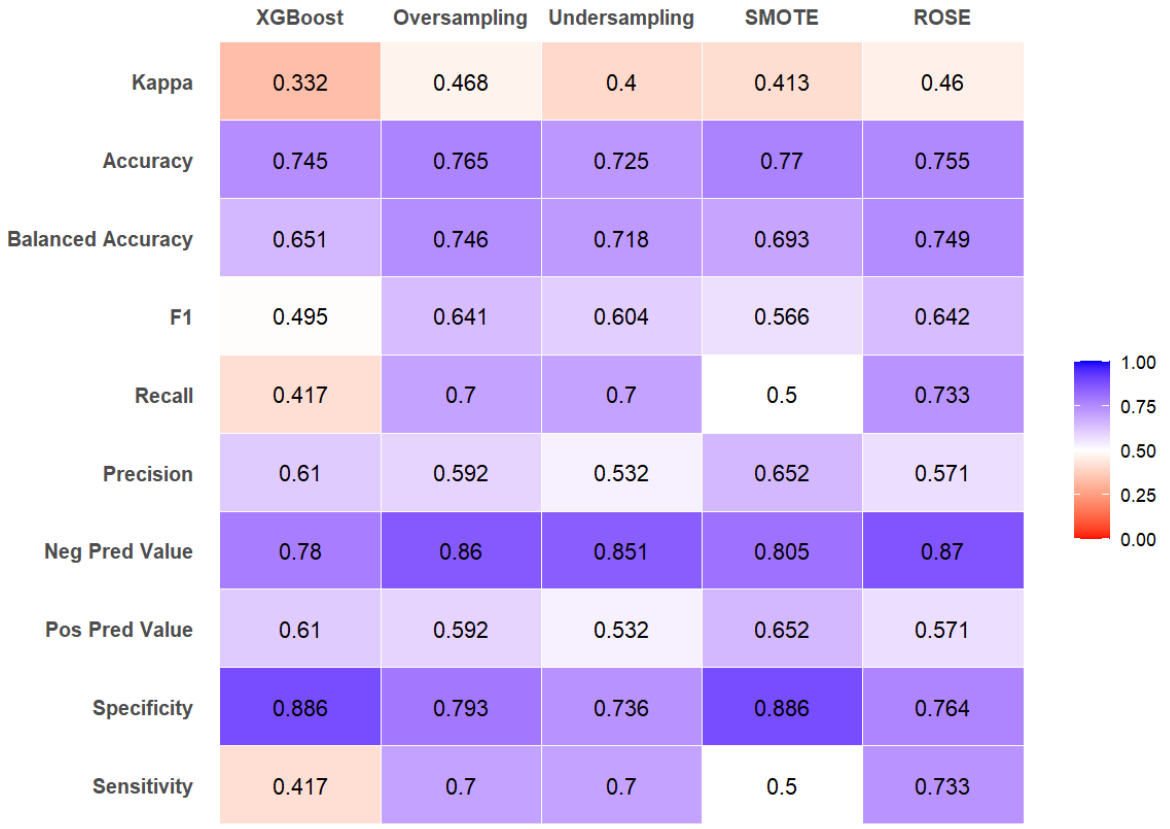


Figura 3.32: Tabla de validación para *XGBoost*

El modelo desbalanceado presenta un `accuracy` de `0.745` y una sensibilidad del `0.417`, es decir, presenta dificultad para clasificar la clase minoritaria. Usando las técnicas de muestreo el asunto mejora: aunque el submuestreo SMOTE presenta unos valores bastante malos en la sensibilidad (de `0.5`), el resto de métodos utilizados se comportan de manera similar, superando el `0.7` tanto en especificidad como en sensibilidad, indicando así que son moderadamente buenos clasificando ambas clases.

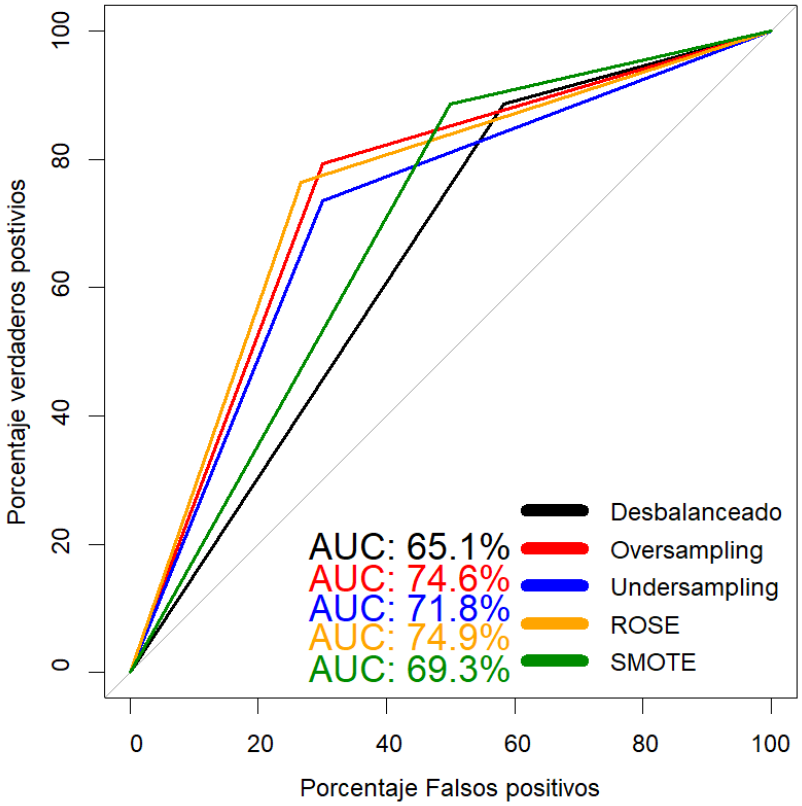


Figura 3.33: Curva ROC para *XGBoost*

La curva ROC del submuestreo y sobremuestreo simples y el método ROSE son bastante similares, tal y como he concluido con la tabla de métricas. De hecho, el AUC del sobremuestreo y del método ROSE són prácticamente iguales, diferenciandose solo por pocas décimas. Si nos tenemos que decantar por alguno, vuelvo a escoger el método ROSE una vez más.

3.5. Modelo final

Una vez ejecutados todos los algoritmos, haré una comparación final para escoger el modelo más adecuado para este conjunto de datos. Para ello, he construido una tabla comparativa similar a las anteriores pero con todos los datos:

	CART	kNN	Logit	NaiveBayes	RandomForest	SVM	XGBoost
Kappa	0.415	0.479	0.457	0.5	0.757	0.829	0.46
Accuracy	0.725	0.755	0.745	0.785	0.89	0.925	0.755
Balanced Accuracy	0.732	0.768	0.756	0.756	0.907	0.932	0.749
F1	0.621	0.662	0.648	0.656	0.838	0.884	0.642
Recall	0.75	0.8	0.783	0.683	0.95	0.95	0.733
Precision	0.529	0.565	0.553	0.631	0.75	0.826	0.571
Neg Pred Value	0.87	0.896	0.887	0.859	0.976	0.977	0.87
Pos Pred Value	0.529	0.565	0.553	0.631	0.75	0.826	0.571
Specificity	0.714	0.736	0.729	0.829	0.864	0.914	0.764
Sensitivity	0.75	0.8	0.783	0.683	0.95	0.95	0.733

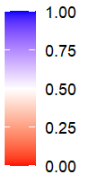


Figura 3.34: Tabla de validación para todos los modelos

Como hemos ido viendo, he seleccionado en todos los modelos la técnica ROSE, puesto que es la que mejores resultados ha dado. Como podemos ver, la mayoria de los algoritmos presentan resultados bastante buenos y parecidos, pero destacan dos que estan rozando la perfección: el bosque aleatorio o *Random Forest* y el *Support Vector Machines* usando el kernel radial. Ambos presentando `accuracy`'s cerca de `0.9` o incluso superandola en el caso del SVM, y sensibilidades y especificidades cerca también del `0.9` .

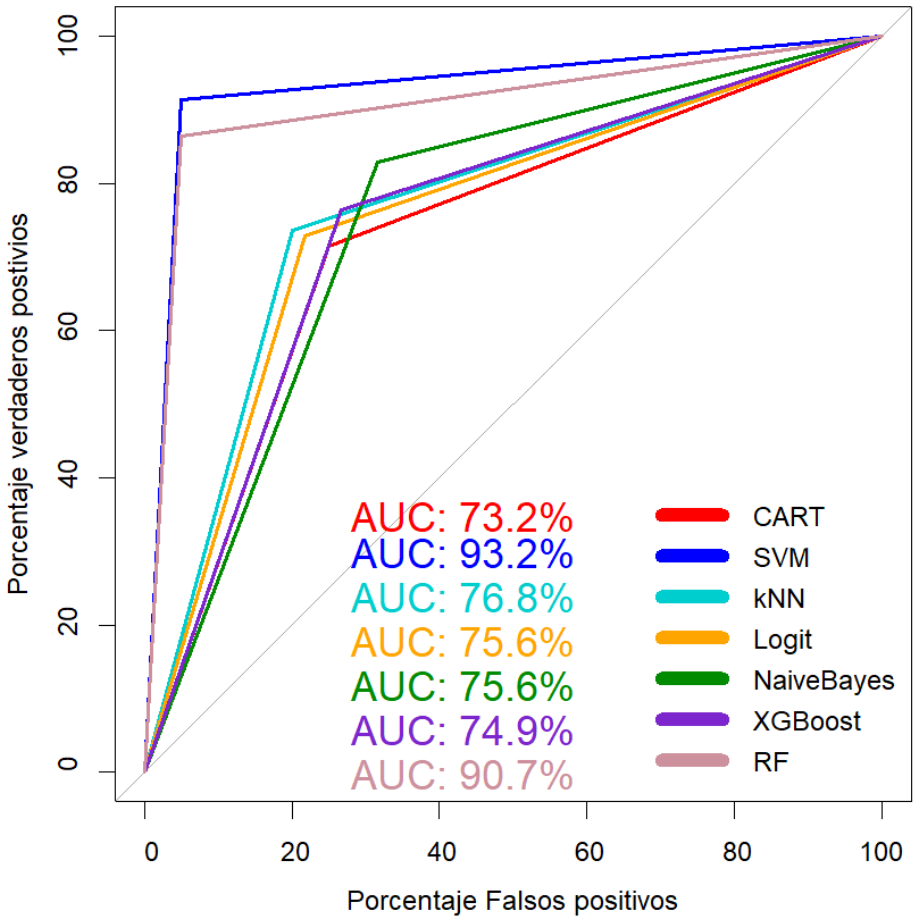


Figura 3.35: Curva ROC para todos los modelos

Podemos observar en las curvas ROC de los modelos el comportamiento similar entre ellos que he comentado, en su mayoria presentando valores AUC alrededor del 75 %. La excepción la observamos con los modelos SVM y *Random Forest*, presentando valores del AUC por encima del 90 % , que són los claros modelos ganadores ante esta base de datos. Si tuviera que escoger uno, sería el SVM de kernel radial, pero simplemente porque tiene resultados ligeramente mejores.

Conclusiones

En conclusión, con este trabajo se ha demostrado el potencial que tiene el machine learning en el sector financiero, ayudando a mitigar los daños y costos que provocan los impagos tanto en las entidades como en la economia de un Estado gracias a una detección a tiempo y precisa.

Por un lado, hemos podido entender qué es la morosidad, sus causas y sus efectos. La subida de los tipos de interés, la inflación, o la coyuntura económica son aspectos macroeconómicos que pueden ensalzar la tasa de morosidad, así como, de un punto de vista más microeconómico o particular, el sobreendeudamiento, la reducción de los ingresos en un hogar, o simplemente la mala gestión financiera de nuestros recursos. Con respecto a los efectos, hemos podido relacionar un alza en la tasa de morosidad con un declive en el crecimiento económico, puesto que ante un riesgo de impago elevado, las instituciones financieras frenan la concesión crediticia cosa que se traduce en una reducción tanto de la inversión como del consumo. Además, una situación extrema en la que el sistema financiero se tambalee debido a esta problemática, puede acarrear problemas aún peores en los que el Estado se vea obligado a intervenir, provocando así una presión fiscal más elevada (para financiar este gasto público) o un aumento de la inflación (si actua en forma de política monetaria).

En relación a los efectos, hemos estudiado también la situación española en los últimos años: El país ha presentado una tasa decreciente a pesar de crisis como la del COVID o la reciente crisis de las materias primas debido a el conflicto entre Rusia y Ucrania, gracias a las políticas como subvenciones, ERTEs o prórrogas. Si desglosamos por destino del crédito, hemos podido observar como las PYMES presentan la tasa de morosidad más alta, y no es de extrañar, puesto que asumen más riesgos (como los comerciales o de mercado) que los hogares. A nivel sectorial, las actividades que más han sido perjudicadas

a raíz del COVID han sido la hosteleria, las actividades artísticas, y en menor medida los transportes, siendo los dos primeros aquellos con una tasa de morosidad que doblaba cualquier otra. No obstante, la tendencia decreciente se ha pausado este último año, hecho que se puede explicar por la lenta recuperación tras las recesiones que ha tenido España, o la exposición que tiene la población a las fluctuaciones del tipo de interés.

Por otro lado, una vez que hemos visto el problema que puede suponer la morosidad a nivel macroeconómico y microeconómico, hemos introducido y aplicado algoritmos de aprendizaje supervisado con el objetivo de valorar o predecir el comportamiento de un cliente ante sus obligaciones financieras. También hemos podido comprobar que, puesto que normalmente los clientes morosos representan una proporcion muy baja de la población total, las clases desequilibradas en la aplicación de modelos de machine learning pueden suponer un problema y una mala interpretación de los datos: En todos los algoritmos, sin aplicar ningún método de remuestreo, el rendimiento era pobre y incorrecto, siendo sesgado hacia la clase mayoritaria. Aunque todas las técnicas de remuestreo mejoraban notoriamente el modelo base, cabe destacar el rendimiento de ROSE, llegando a presentar clasificaciones prácticamente perfectas.

Finalmente, a líneas futuras, sería interesante ir un paso más allá e implementar modelos de inteligencia artificial, ya que su excepcional manejo de datos y su capacidad de interpretar la información son tales, que podrían ser un cambio revolucionario en el sector financiero y en general en la economía de todo el mundo.

La integración de técnicas de análisis de datos y machine learning en los sistemas financieros representa una revolución en la manera en que entendemos y gestionamos las finanzas. Por un lado, mejora la precisión y eficiencia de nuestras decisiones, y por otro, está extendiendo muchas vías para innovar el mundo financiero. Además, considero que la capacidad que tienen estos sistemas de aprender y adaptarse a la información disponible es una ventaja crucial actualmente; todo avanza suficientemente rápido como para premiar las decisiones no solo precisas, sino que rápidas. No obstante, en mi opinión este tipo de tecnologías también presentan grandes desafíos; asegurar una información de buena calidad, construir modelos complejos, eficientes y robustos, o en general un proceso riguroso que exige de atención constante.

Personalmente, considero que en un futuro cercano el machine learning y la inteligencia artificial se usarán para hacer los actuales procesos y métodos más eficientes; pero la verdadera revolución será cuando se consiga innovar, es decir, cuando se llegue a crear nuevos procedimientos. Henry Ford solía decir que «Si hubiera preguntado a la gente qué querían, hubieran respondido: caballos más rápidos», y entonces, inventó el automóvil; para descubrir el verdadero potencial de esta "máquina de pensar", debemos de cambiar nuestra perspectiva en las tecnologías, y dar un paso más allá para explorar nuevas alternativas, metodologías, y estrategias.

Bibliografía

¿Qué es el Preprocesamiento de Datos (Data Preprocessing)? https://apuredigital.net/que-es-el-preprocesamiento-de-datos-data-preprocessing/ (acceso: 19/04/2024).

¿Qué es el preprocesamiento de datos? Definición, importancia y pasos. https://www.astera.com/es/type/blog/data-preprocessing/ (acceso: 31/05/2024).

Abad, R. C., & Casal, R. F. (2022). *Técnicas de Remuestreo.* https://rubenfcasal.github.io/book_remuestreo/modunif-boot-suav.html (acceso: 30/05/2024).

Algoritmos de aprendizaje automático. https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/what-are-machine-learning-algorithms (acceso: 14/04/2024).

AnalystPrep. *The Credit Decision.* https://analystprep.com/study-notes/frm/part-2/credit-risk-measurement-and-management/the-credit-decision/ (acceso: 19/04/2024).

Bequé, A., Coussement, K., Gayler, R., & Lessmann, S. (2017). *Approaches for credit scorecard calibration: An empirical analysis.* Knowledge-Based Systems.

Botta, A., Russo, A., Gallegati, M., & Caverzasi, E. (2018). INEQUALITY AND FINANCE IN A RENT ECONOMY. *Greenwich papers in political economy*, 28.

Conjuntos de entrenamiento y prueba: división de datos. https://developers.google.com/machine-learning/crash-course/training-and-test-sets/splitting-data?hl=es-419 (acceso: 31/05/2024).

Data preparation part in data mining projects. https://www.kdnuggets.com/polls/2003/data_preparation.htm (acceso: 06/06/2024).

Dave, P. (2023). *From Bias to Balance: Solving Imbalanced Data Issues.* https://priyankaddit.medium.com/how-to-deal-with-imbalanced-dataset-86de86c49 (acceso: 15/06/2024).

Demir, S., & Sahin, E. (2022). Evaluation of Oversampling Methods (OVER, SMOTE, and ROSE) in Classifying Soil Liquefaction Dataset based on SVM, RF, and Naïve Bayes. *European Journal of Science and Technology*, 142-147. https://doi.org/10.31590/ejosat.1077867

Dhakal, C. P. (2017). Dealing with outliers and influential points while fitting regression. *Journal of Institute of Science and Technology*, *22*(1), 61-65.

Española, R. A. *Diccionario de la lengua española, 23.a ed.* https://dle.rae.es/inteligencia (acceso: 24/05/2024).

Estado Actual de la Cultura Financiera en España. https://origensf.es/cultura-financiera-espana/ (acceso: 30/05/2024).

Fissuh, A. *Outliers detection in R.* https://rpubs.com/Alema/1000582 (acceso: 14/04/2024).

The Four V's of Big Data. https://opensistemas.com/en/the-four-vs-of-big-data/ (acceso: 19/04/2024).

Gibert, K. (2009). *Estadística: Contexto histórico e introducción a la descriptiva.* Serveis Gràfics Copisteria Imatge S. L.

Gibert, K., Sànchez-Marrè, M., & Izquierdo, J. (2016). *A Survey on Pre-processing Techniques in the Context of Environmental Data Mining. Artificial Intelligence in Communications.* https://doi.org/10.3233/AIC-160710

Gleick, J. (1986). Hole in ozone over South Pole worries scientists. *The New York Times.*

Gopalan, B. *MICE algorithm to Impute missing values in a dataset.* https://www.numpyninja.com/post/mice-algorithm-to-impute-missing-values-in-a-dataset (acceso: 30/05/2024).

Gower, J. C. (1971). A General Coefficient of Similarity and Some of Its Properties. *Biometrics*, *27*(4), 857-871. Consultado el 14 de junio de 2024, desde http://www.jstor.org/stable/2528823

Gupta, P. *Decision Trees in Machine Learning.* https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052 (acceso: 30/05/2024).

Gutpa, M. *What is Machine Learning?* https://www.geeksforgeeks.org/ml-machine-learning/ (acceso: 30/04/2024).

Hofmann, H. (1994). *Statlog (German Credit Data).* https://doi.org/https://doi.org/10.24432/C5NC77

How Does Netflix Use Machine Learning. https://www.geeksforgeeks.org/how-does-netflix-use-machine-learning/ (acceso: 30/04/2024).

How To Design A Spam Filtering System with Machine Learning Algorithm. https://towardsdatascience.com/email-spam-detection-1-2-b0e06a5c0472 (acceso: 19/04/2024).

IBM. (2023). *What is principal component analysis (PCA)?* https://www.ibm.com/topics/principal-component-analysis (acceso: 14/04/2024).

IBM. (2024). *Multiple Correspondence Analysis.* https://www.ibm.com/docs/en/spss-statistics/29.0.0?topic=application-multiple-correspondence-analysis (acceso: 14/04/2024).

IBM. *¿Qué es el algoritmo de k vecinos más cercanos?* https://www.ibm.com/es-es/topics/knn (acceso: 30/05/2024).

IBM. *¿Qué es la regresión logística?* https://www.ibm.com/es-es/topics/logistic-regression (acceso: 30/05/2024).

IBM. *¿Qué son los clasificadores Naive Bayes?* https://www.ibm.com/es-es/topics/naive-bayes (acceso: 30/05/2024).

La morosidad del crédito bancario en España. https://www.aebanca.es/noticias/articulos/la-morosidad-del-credito-bancario-en-espana/ (acceso: 30/05/2024).

La morosidad en España en 2022. https://www.tas-consultoria.com/blog-es/la-morosidad-en-espana-en-2022/ (acceso: 30/05/2024).

The Machine Learning Life Cycle Explained. https://www.datacamp.com/es/blog/machine-learning-lifecycle-explained (acceso: 19/04/2024).

Mani, C. *How Is Big Data Analytics Using Machine Learning?* https://www.forbes.com/sites/forbestechcouncil/2020/10/20/how-is-big-data-analytics-using-machine-learning/?sh=68b9c25c71d2 (acceso: 30/04/2024).

Mascareñas, J. (2012). *Monografías de Juan Mascareñas sobre Finanzas Corporativas nº 39.* ISSN: 1988-1878. http://dx.doi.org/10.2139/ssrn.2310565

Metwalli, S. A. *How to Choose the Right Machine Learning Algorithm for Your Application.* https://towardsdatascience.com/how-to-choose-the-right-machine-learning-algorithm-for-your-application-1e36c32400b9 (acceso: 30/05/2024).

Moore, D. S., & McCabe, G. P. (2005). *Introduction to the Practice of Statistics.* W.H. Freeman & Company.

Observatorio de Morosidad 2T 2021. https://cepyme.es/storage/2022/03/Observatorio-de-morosidad_2do-trimestre_2021-1.pdf (acceso: 30/05/2024).

Pérez, G. R. *Poco interés y una confianza mal entendida: por qué los españoles apenas aprueban en conocimientos financieros.* https://elpais.com/economia/2023-11-19/poco-interes-y-una-confianza-mal-entendida-por-que-los-espanoles-apenas-aprueban-en-conocimientos-financieros.html (acceso: 30/05/2024).

Prabhakaran, S. *MICE imputation – How to predict missing values using machine learning in Python.* https://www.machinelearningplus.com/machine-learning/mice-imputation/ (acceso: 30/05/2024).

Qué es la morosidad y qué consecuencias tiene. https://www.bbva.com/es/salud-financiera/que-es-la-morosidad-y-que-consecuencias-tiene/ (acceso: 30/04/2024).

Rebala, G., Ravi, A., & Churiwala, S. (2019). Machine Learning Definition and Basics. En *An Introduction to Machine Learning.* Springer International Publishing. https://doi.org/10.1007/978-3-030-15729-6_1

Rego, N. *¿En qué punto se encuentra España en materia de préstamos dudosos (NPL)?* https://www.gloval.es/prestamos-dudosos-npls-en-espana-2022/ (acceso: 30/05/2024).

Riesgo del crédito: Qué es y por qué importa. https://www.sas.com/es_es/insights/risk-management/credit-risk-management.html (acceso: 30/04/2024).

Royston, P. (s.f.). Multiple imputation of missing values. https://doi.org/https://doi.org/10.1177/1536867X0400400301 (acceso: 30/05/2024).

Segal, T. *5 Cs of Credit: What They Are, How They're Used, and Which Is Most Important.* https://www.investopedia.com/terms/f/five-c-credit.asp (acceso: 19/04/2024).

Shirin. (2017). *Dealing with unbalanced data in machine learning.* https://www.r-bloggers.com/2017/04/dealing-with-unbalanced-data-in-machine-learning/ (acceso: 14/06/2024).

Silva, J. M. C. (2021). *Una herramienta para afinar criterios de decisión.* https://www.debatesiesa.com/una-herramienta-para-afinar-criterios-de-decision/ (acceso: 14/04/2024).

Solutions, M. (2009). *Análisis de la morosidad en base a factores macroeconómicos*. https://www.managementsolutions.com/sites/default/files/publicaciones/esp/Morosidad.pdf

Torres, R., & Fernández, M. J. *La economía española, de la pandemia a la crisis energética*. https://www.funcas.es/revista/la-economia-espanola-durante-la-pandemia-resumenes-ejecutivos/#:~:text=La%20econom%C3%ADa%20espa%C3%B1ola%2C%20de%20la%20pandemia%20a%20la%20crisis%20energ%C3%A9tica&text=El%20PIB%20registr%C3%B3%20un%20descenso,de%20movilidad%2C%20como%20el%20turismo. (acceso: 30/05/2024).

Training data vs Testing data. https://www.geeksforgeeks.org/training-data-vs-testing-data/ (acceso: 31/05/2024).

Valores perdidos: MCAR, MAR y MNAR. https://www.datanalytics.com/2012/06/28/valores-perdidos-mcar-mar-y-mnar/ (acceso: 30/05/2024).

Wikipedia. (2024a). Criterio de información de Akaike — Wikipedia, La enciclopedia libre [[Internet; descargado 3-enero-2024]]. https://es.wikipedia.org/w/index.php?title=Criterio_de_informaci%C3%B3n_de_Akaike&oldid=156586167

Wikipedia. (2024b). Descenso del gradiente — Wikipedia, La enciclopedia libre [[Internet; descargado 20-abril-2024]]. https://es.wikipedia.org/w/index.php?title=Descenso_del_gradiente&oldid=159559902

Wikipedia. (2024c). Métodos de Montecarlo basados en cadenas de Markov — Wikipedia, La enciclopedia libre [[Internet; descargado 8-junio-2024]]. https://es.wikipedia.org/w/index.php?title=M%C3%A9todos_de_Montecarlo_basados_en_cadenas_de_Markov&oldid=160616664

Wikipedia. *Data preprocessing*. https://en.wikipedia.org/w/index.php?title=Data_preprocessing&oldid=1225758329 (acceso: 31/05/2024).

Wikipedia. *Validación cruzada*. https://es.wikipedia.org/wiki/Validaci%C3%B3n_cruzada (acceso: 31/05/2024).

Anexo

Código y salidas de R

kNN

```
library(MLmetrics)
library(dplyr)
library(ggplot2)
library(tidyverse)
library(pROC)
library(caret)
library(reshape)
library(doParallel)

cl <- makePSOCKcluster(12)
registerDoParallel(cl)

model_knn <- train(data.credit_rating ~., data = dataTrain, method = "knn", trControl = ctrl, preProcess = c("nzv", "BoxCox"), tuneLength = 10)
predicciones_knn = predict(model_knn, newdata = dataTest)
confusion = confusionMatrix(data = predicciones_knn, reference = dataTest$data.credit_rating)

model_knn_up <- train(Class ~., data = up_train, method = "knn", trControl = ctrl, preProcess = c("nzv", "BoxCox"), tuneLength = 10)
predicciones_knn_up = predict(model_knn_up, newdata = dataTest)
confusion_up = confusionMatrix(data = predicciones_knn_up, reference = dataTest$data.credit_rating)

model_knn_down <- train(Class ~., data = down_train, method = "knn", trControl = ctrl, preProcess = c("nzv", "BoxCox"), tuneLength = 10)
predicciones_knn_down = predict(model_knn_down, newdata = dataTest)
confusion_down = confusionMatrix(data = predicciones_knn_down, reference = dataTest$data.credit_rating)

model_knn_smote <- train(class ~., data = smote_train, method = "knn", trControl = ctrl, preProcess = c("nzv", "BoxCox"), tuneLength = 10)
predicciones_knn_smote = predict(model_knn_smote, newdata = dataTest)
confusion_smote = confusionMatrix(data = predicciones_knn_smote, reference = dataTest$data.credit_rating)

model_knn_rose <- train(rose_train.credit_rating ~., data = rose_train, method = "knn", trControl = ctrl, preProcess = c("center", "scale"), tuneLength = 10)
predicciones_knn_rose = predict(model_knn_rose, newdata = dataTest)
confusion_rose = confusionMatrix(data = predicciones_knn_rose, reference = dataTest$data.credit_rating)
```

```
32
33 byClass <- rbind(confusion =round(confusion$byClass, 3),
34                  confusion_up =round(confusion_up$byClass,3),
35                  confusion_down =round(confusion_down$byClass,3),
36                  confusion_smote =round(confusion_smote$byClass,3),
37                  confusion_rose =round(confusion_rose$byClass,3))
38
39 overall <- rbind(confusion =round(confusion$overall[1:2], 3),
40                  confusion_up =round(confusion_up$overall[1:2],3),
41                  confusion_down =round(confusion_down$overall[1:2],3),
42                  confusion_smote =round(confusion_smote$overall[1:2],3),
43                  confusion_rose =round(confusion_rose$overall[1:2],3))
44 Table2 <- t(cbind(byClass, overall))
45 colnames(Table2) = c("kNN", "kNN - Oversampling", "kNN - Undersampling", "kNN -
   ↪  SMOTE", "kNN - ROSE" )
46
47 mat.heatmap.m = melt(Table2)
48
49 ggplot(data = data.frame(mat.heatmap.m),aes(x = X2, y = X1, fill= value)) +
50   geom_tile(color = "white")+
51   geom_text(aes(label = value))+
52   scale_fill_gradient2(low = "red", high = "blue", mid = "white",
53                        midpoint = 0.5, limit = c(0,1), space = "Lab",
54                        name="") +
55   theme_minimal()+
56   theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank())+
57   scale_x_discrete(position = "top") +
58   labs(x = "", y="")
59
60 par(mar=c(5.1,4.1,6,2.1), xpd=F, pty = "s")
61 roc(as.numeric(dataTest$data.credit_rating), as.numeric(predicciones_knn), plot =
   ↪  TRUE, legacy.axes = TRUE,
62     percent = TRUE, xlab = "Porcentaje Falsos positivos",
63     ylab = "Porcentaje verdaderos postivios", col = "black", lwd = 2,
64     print.auc = TRUE, print.auc.x = 50, print.auc.y = 22, xlim=c(100, 0), ylim =
       ↪  c(0,100))
65 plot.roc(as.numeric(dataTest$data.credit_rating), as.numeric(predicciones_knn_up),
   ↪  percent=TRUE, col="red",lwd= 2,
66          print.auc =TRUE, add=TRUE, print.auc.x = 50, print.auc.y = 17)
67 plot.roc(as.numeric(dataTest$data.credit_rating), as.numeric(predicciones_knn_down),
   ↪  percent=TRUE,lwd= 2,
68          print.auc =TRUE, add=TRUE, col = "blue", print.auc.x = 50, print.auc.y =
            ↪  12.5)
69 plot.roc(dataTest$data.credit_rating, as.numeric(predicciones_knn_rose),
   ↪  percent=TRUE,
70          print.auc = TRUE,add = TRUE, col = "orange", print.auc.x = 50, print.auc.y =
            ↪  8)
71 plot.roc(as.numeric(dataTest$data.credit_rating), as.numeric(predicciones_knn_smote),
   ↪  percent=TRUE, col="green4",lwd= 2,
```

```
        print.auc =TRUE, add=TRUE, print.auc.x = 50, print.auc.y = 3)
legend("bottomright", c("Desbalanceado", "Oversampling", "Undersampling", "ROSE",
↪  "SMOTE"), lty=1,
       col = c("black", "red", "blue","orange","green4"), bty="n", lwd = 2)
> model_knn_rose
k-Nearest Neighbors

1000 samples
  58 predictor
   2 classes: 'Malo', 'Bueno'

Pre-processing: centered (58), scaled (58)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 900, 900, 901, 899, 900, 899, ...
Resampling results across tuning parameters:

  k   logLoss    AUC        prAUC      Accuracy   Kappa      F1
   5  1.3633647  0.8270371  0.5082651  0.7419894  0.4838522  0.7466298
   7  1.0062464  0.8063604  0.6242139  0.7383661  0.4767114  0.7405744
   9  0.7831216  0.7946717  0.6688358  0.7280419  0.4560099  0.7314160
  11  0.5856958  0.7934356  0.7028200  0.7216513  0.4433255  0.7218100
  13  0.5795159  0.7921899  0.7193069  0.7212647  0.4425720  0.7221499
  15  0.5849893  0.7881875  0.7281737  0.7112539  0.4225233  0.7113077
  17  0.5882955  0.7874060  0.7342409  0.7112604  0.4225904  0.7097189
  19  0.5688921  0.7854438  0.7352090  0.7122804  0.4246380  0.7104418
  21  0.5638162  0.7811153  0.7385617  0.7036098  0.4073929  0.7007943
  23  0.5678589  0.7773994  0.7363935  0.6989662  0.3981640  0.6944954
  Sensitivity  Specificity  Pos_Pred_Value  Neg_Pred_Value  Precision
  0.7536471    0.7301769    0.7424949       0.7453584       0.7424949
  0.7399869    0.7367755    0.7434999       0.7361419       0.7434999
  0.7347582    0.7212381    0.7308263       0.7286734       0.7308263
  0.7167582    0.7266259    0.7301385       0.7170370       0.7301385
  0.7187190    0.7239592    0.7288927       0.7173945       0.7288927
  0.7060784    0.7165170    0.7193953       0.7062328       0.7193953
  0.7007712    0.7219184    0.7211023       0.7040923       0.7211023
  0.7001830    0.7245714    0.7235725       0.7040965       0.7235725
  0.6869804    0.7206259    0.7177021       0.6928610       0.7177021
  0.6784575    0.7199320    0.7138945       0.6875481       0.7138945
  Recall     Detection_Rate  Balanced_Accuracy
  0.7536471  0.3806170       0.7419120
  0.7399869  0.3736935       0.7383812
  0.7347582  0.3710430       0.7279981
  0.7167582  0.3619857       0.7216920
  0.7187190  0.3629493       0.7213391
  0.7060784  0.3566154       0.7112977
  0.7007712  0.3539418       0.7113448
  0.7001830  0.3536384       0.7123772
  0.6869804  0.3469379       0.7038031
```

```
  0.6784575  0.3426342       0.6991947

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 5.
```

xgBoost

```
library(MLmetrics)
library(dplyr)
library(ggplot2)
library(tidyverse)
library(pROC)
library(caret)
library(reshape)
library(doParallel)

cl <- makePSOCKcluster(12)
registerDoParallel(cl)

model_xgboost <- train(data.credit_rating ~., data = dataTrain, method = "xgbDART",
↪  trControl = ctrl, preProcess = c("nzv", "BoxCox"), tuneLength = 2)
predicciones_xgboost = predict(model_xgboost, newdata = dataTest)
confusion = confusionMatrix(data = predicciones_xgboost, reference =
↪  dataTest$data.credit_rating)

model_xgboost_up <- train(Class ~., data = up_train, method = "xgbDART", trControl =
↪  ctrl, preProcess = c("nzv", "BoxCox"), tuneLength = 2)
predicciones_xgboost_up = predict(model_xgboost_up, newdata = dataTest)
confusion_up = confusionMatrix(data = predicciones_xgboost_up, reference =
↪  dataTest$data.credit_rating)

model_xgboost_down <- train(Class ~., data = down_train, method = "xgbDART",
↪  trControl = ctrl, preProcess = c("nzv", "BoxCox"), tuneLength = 2)
predicciones_xgboost_down = predict(model_xgboost_down, newdata = dataTest)
confusion_down = confusionMatrix(data = predicciones_xgboost_down, reference =
↪  dataTest$data.credit_rating)

model_xgboost_smote <- train(class ~., data = smote_train, method = "xgbDART",
↪  trControl = ctrl, preProcess = c("nzv", "BoxCox"), tuneLength = 2)
predicciones_xgboost_smote = predict(model_xgboost_smote, newdata = dataTest)
confusion_smote = confusionMatrix(data = predicciones_xgboost_smote, reference =
↪  dataTest$data.credit_rating)

model_xgboost_rose <- train(rose_train.credit_rating ~., data = rose_train, method =
↪  "xgbDART", trControl = ctrl, preProcess = c("center", "scale"), tuneLength = )
```

```
predicciones_xgboost_rose = predict(model_xgboost_rose, newdata = dataTest)
confusion_rose = confusionMatrix(data = predicciones_xgboost_rose, reference = dataTest$data.credit_rating)

byClass <- rbind(confusion =round(confusion$byClass, 3),
                 confusion_up =round(confusion_up$byClass,3),
                 confusion_down =round(confusion_down$byClass,3),
                 confusion_smote =round(confusion_smote$byClass,3),
                 confusion_rose =round(confusion_rose$byClass,3))

overall <- rbind(confusion =round(confusion$overall[1:2], 3),
                 confusion_up =round(confusion_up$overall[1:2],3),
                 confusion_down =round(confusion_down$overall[1:2],3),
                 confusion_smote =round(confusion_smote$overall[1:2],3),
                 confusion_rose =round(confusion_rose$overall[1:2],3))
Table2 <- t(cbind(byClass, overall))
colnames(Table2) = c("xgBoost", "xgBoost - Oversampling", "xgBoost - Undersampling", "xgBoost - SMOTE", "xgBoost - ROSE" )

mat.heatmap.m = melt(Table2)

ggplot(data = data.frame(mat.heatmap.m),aes(x = X2, y = X1, fill= value)) +
  geom_tile(color = "white")+
  geom_text(aes(label = value))+
  scale_fill_gradient2(low = "red", high = "blue", mid = "white",
                       midpoint = 0.5, limit = c(0,1), space = "Lab",
                       name="") +
  theme_minimal()+
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank())+
  scale_x_discrete(position = "top") +
  labs(x = "", y="")

par(mar=c(5.1,4.1,6,2.1), xpd=F, pty = "s")
roc(as.numeric(dataTest$data.credit_rating), as.numeric(predicciones_xgboost), plot = TRUE, legacy.axes = TRUE,
    percent = TRUE, xlab = "Porcentaje Falsos positivos",
    ylab = "Porcentaje verdaderos postivios", col = "black", lwd = 2,
    print.auc = TRUE, print.auc.x = 50, print.auc.y = 22, xlim=c(100, 0), ylim = c(0,100))
plot.roc(as.numeric(dataTest$data.credit_rating), as.numeric(predicciones_xgboost_up), percent=TRUE, col="red",lwd= 2,
         print.auc =TRUE, add=TRUE, print.auc.x = 50, print.auc.y = 17)
plot.roc(as.numeric(dataTest$data.credit_rating), as.numeric(predicciones_xgboost_down), percent=TRUE,lwd= 2,
         print.auc =TRUE, add=TRUE, col = "blue", print.auc.x = 50, print.auc.y = 12.5)
plot.roc(dataTest$data.credit_rating, as.numeric(predicciones_xgboost_rose), percent=TRUE,
```

```
        print.auc = TRUE,add = TRUE, col = "orange", print.auc.x = 50, print.auc.y = 8)
plot.roc(as.numeric(dataTest$data.credit_rating), as.numeric(predicciones_xgboost_smote), percent=TRUE, col="green4",lwd= 2,
        print.auc =TRUE, add=TRUE, print.auc.x = 50, print.auc.y = 3)
legend("bottomright", c("Desbalanceado", "Oversampling", "Undersampling", "ROSE", "SMOTE"), lty=1,
       col = c("black", "red", "blue","orange","green4"), bty="n", lwd = 2)

> model_xgboost_rose
eXtreme Gradient Boosting

1000 samples
  58 predictor
   2 classes: 'Malo', 'Bueno'

Pre-processing: centered (58), scaled (58)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 899, 900, 900, 900, 900, 900, ...
Resampling results across tuning parameters:

  eta  rate_drop  skip_drop  colsample_bytree  logLoss    AUC
  0.3  0.01       0.05       0.6               0.5578009  0.7892226
  0.3  0.01       0.05       0.8               0.5599100  0.7846302
  0.3  0.01       0.95       0.6               0.5578421  0.7877014
  0.3  0.01       0.95       0.8               0.5578218  0.7861828
  0.3  0.50       0.05       0.6               0.6267253  0.7263794
  0.3  0.50       0.05       0.8               0.6268595  0.7257764
  0.3  0.50       0.95       0.6               0.5600888  0.7836091
  0.3  0.50       0.95       0.8               0.5597259  0.7844452
  0.4  0.01       0.05       0.6               0.5566692  0.7871183
  0.4  0.01       0.05       0.8               0.5562023  0.7852443
  0.4  0.01       0.95       0.6               0.5595624  0.7841599
  0.4  0.01       0.95       0.8               0.5557415  0.7876561
  0.4  0.50       0.05       0.6               0.6200760  0.7371723
  0.4  0.50       0.05       0.8               0.6191829  0.7311078
  0.4  0.50       0.95       0.6               0.5534045  0.7907254
  0.4  0.50       0.95       0.8               0.5552455  0.7879137
  prAUC      Accuracy   Kappa      F1         Sensitivity  Specificity
  0.7707821  0.7109795  0.4215889  0.7194043  0.7372941    0.6840408
  0.7673314  0.6989819  0.3977582  0.7054000  0.7176471    0.6799864
  0.7676424  0.7136165  0.4269502  0.7210318  0.7353464    0.6914558
  0.7660747  0.7085664  0.4167509  0.7171264  0.7345882    0.6819320
  0.6991950  0.6633365  0.3249885  0.7014717  0.7861961    0.5380000
  0.7040582  0.6583529  0.3149457  0.6967233  0.7795033    0.5346667
  0.7642980  0.7032895  0.4064263  0.7089612  0.7181830    0.6881769
  0.7643578  0.7030156  0.4057474  0.7113257  0.7261830    0.6794286
  0.7657592  0.7132864  0.4264501  0.7173081  0.7241438    0.7022449
```

```
  0.7652279  0.7159598  0.4317142  0.7214334  0.7314248    0.7001769
  0.7628602  0.7083225  0.4164806  0.7124990  0.7194902    0.6968844
  0.7674227  0.7089427  0.4176787  0.7137274  0.7202353    0.6973605
  0.7072419  0.6732704  0.3454828  0.7002392  0.7570458    0.5879048
  0.7014235  0.6696736  0.3377895  0.7030404  0.7761961    0.5608435
  0.7692303  0.7059523  0.4116724  0.7133671  0.7268235    0.6847347
  0.7677112  0.7109726  0.4218925  0.7134981  0.7182222    0.7036327
  Pos_Pred_Value  Neg_Pred_Value  Precision  Recall     Detection_Rate
  0.7052739       0.7217700       0.7052739  0.7372941  0.3723461
  0.6967372       0.7053506       0.6967372  0.7176471  0.3623588
  0.7097576       0.7211839       0.7097576  0.7353464  0.3713295
  0.7029807       0.7185158       0.7029807  0.7345882  0.3709595
  0.6357675       0.7158384       0.6357675  0.7861961  0.3970443
  0.6328017       0.7070908       0.6328017  0.7795033  0.3936872
  0.7034019       0.7076507       0.7034019  0.7181830  0.3626558
  0.7002954       0.7103240       0.7002954  0.7261830  0.3666992
  0.7147657       0.7170433       0.7147657  0.7241438  0.3656456
  0.7146757       0.7217815       0.7146757  0.7314248  0.3693324
  0.7082506       0.7119658       0.7082506  0.7194902  0.3633419
  0.7106040       0.7112024       0.7106040  0.7202353  0.3637155
  0.6539911       0.7055419       0.6539911  0.7570458  0.3823067
  0.6445483       0.7133169       0.6445483  0.7761961  0.3920238
  0.7034181       0.7129008       0.7034181  0.7268235  0.3670222
  0.7131454       0.7142433       0.7131454  0.7182222  0.3626485
  Balanced_Accuracy
  0.7106675
  0.6988167
  0.7134011
  0.7082601
  0.6620980
  0.6570850
  0.7031799
  0.7028058
  0.7131943
  0.7158009
  0.7081873
  0.7087979
  0.6724753
  0.6685198
  0.7057791
  0.7109274

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

```
 1, eta = 0.4, gamma = 0, subsample = 0.5, colsample_bytree =
 0.8, rate_drop = 0.01, skip_drop = 0.05 and min_child_weight = 1.

> model_xgboost_rose$finalModel$tuneValue
   nrounds max_depth eta gamma subsample colsample_bytree rate_drop
10      50         1 0.4     0       0.5              0.8      0.01
   skip_drop min_child_weight
10      0.05                1
```

Random Forest

```
library(MLmetrics)
library(dplyr)
library(ggplot2)
library(tidyverse)
library(pROC)
library(caret)
library(reshape)
library(doParallel)

cl <- makePSOCKcluster(12)
registerDoParallel(cl)

model_rf <- train(data.credit_rating ~., data = dataTrain, method = "rf", trControl =
↪  ctrl, preProcess = c("nzv", "BoxCox"), tuneLength = 10)
predicciones_rf = predict(model_rf, newdata = dataTest)
confusion = confusionMatrix(data = predicciones_rf, reference =
↪  dataTest$data.credit_rating)

model_rf_up <- train(Class ~., data = up_train, method = "rf", trControl = ctrl,
↪  preProcess = c("nzv", "BoxCox"), tuneLength = 10)
predicciones_rf_up = predict(model_rf_up, newdata = dataTest)
confusion_up = confusionMatrix(data = predicciones_rf_up, reference =
↪  dataTest$data.credit_rating)

model_rf_down <- train(Class ~., data = down_train, method = "rf", trControl = ctrl,
↪  preProcess = c("nzv", "BoxCox"), tuneLength = 10)
predicciones_rf_down = predict(model_rf_down, newdata = dataTest)
confusion_down = confusionMatrix(data = predicciones_rf_down, reference =
↪  dataTest$data.credit_rating)

model_rf_smote <- train(class ~., data = smote_train, method = "rf", trControl =
↪  ctrl, preProcess = c("nzv", "BoxCox"), tuneLength = 10)
predicciones_rf_smote = predict(model_rf_smote, newdata = dataTest)
```

```
confusion_smote = confusionMatrix(data = predicciones_rf_smote, reference = dataTest$data.credit_rating)

model_rf_rose <- train(rose_train.credit_rating ~., data = rose_train, method = "rf", trControl = ctrl, preProcess = c("center", "scale"), tuneLength = 10)
predicciones_rf_rose = predict(model_rf_rose, newdata = dataTest)
confusion_rose = confusionMatrix(data = predicciones_rf_rose, reference = dataTest$data.credit_rating)

byClass <- rbind(confusion =round(confusion$byClass, 3),
                 confusion_up =round(confusion_up$byClass,3),
                 confusion_down =round(confusion_down$byClass,3),
                 confusion_smote =round(confusion_smote$byClass,3),
                 confusion_rose =round(confusion_rose$byClass,3))

overall <- rbind(confusion =round(confusion$overall[1:2], 3),
                 confusion_up =round(confusion_up$overall[1:2],3),
                 confusion_down =round(confusion_down$overall[1:2],3),
                 confusion_smote =round(confusion_smote$overall[1:2],3),
                 confusion_rose =round(confusion_rose$overall[1:2],3))
Table2 <- t(cbind(byClass, overall))
colnames(Table2) = c("RF", "RF - Oversampling", "RF - Undersampling", "RF - SMOTE", "RF - ROSE" )

mat.heatmap.m = melt(Table2)

ggplot(data = data.frame(mat.heatmap.m),aes(x = X2, y = X1, fill= value)) +
  geom_tile(color = "white")+
  geom_text(aes(label = value))+
  scale_fill_gradient2(low = "red", high = "blue", mid = "white",
                       midpoint = 0.5, limit = c(0,1), space = "Lab",
                       name="") +
  theme_minimal()+
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank())+
  scale_x_discrete(position = "top") +
  labs(x = "", y="")

par(mar=c(5.1,4.1,6,2.1), xpd=F, pty = "s")
roc(as.numeric(dataTest$data.credit_rating), as.numeric(predicciones_rf), plot = TRUE, legacy.axes = TRUE,
    percent = TRUE, xlab = "Porcentaje Falsos positivos",
    ylab = "Porcentaje verdaderos postivios", col = "black", lwd = 2,
    print.auc = TRUE, print.auc.x = 50, print.auc.y = 22, xlim=c(100, 0), ylim = c(0,100))
plot.roc(as.numeric(dataTest$data.credit_rating), as.numeric(predicciones_rf_up), percent=TRUE, col="red",lwd= 2,
         print.auc =TRUE, add=TRUE, print.auc.x = 50, print.auc.y = 17)
plot.roc(as.numeric(dataTest$data.credit_rating), as.numeric(predicciones_rf_down), percent=TRUE,lwd= 2,
```

```
        print.auc =TRUE, add=TRUE, col = "blue", print.auc.x = 50, print.auc.y = 12.5)
plot.roc(dataTest$data.credit_rating, as.numeric(predicciones_rf_rose), percent=TRUE,
        print.auc = TRUE,add = TRUE, col = "orange", print.auc.x = 50, print.auc.y = 8)
plot.roc(as.numeric(dataTest$data.credit_rating), as.numeric(predicciones_rf_smote), percent=TRUE, col="green4",lwd= 2,
        print.auc =TRUE, add=TRUE, print.auc.x = 50, print.auc.y = 3)
legend("bottomright", c("Desbalanceado", "Oversampling", "Undersampling", "ROSE", "SMOTE"), lty=1,
       col = c("black", "red", "blue","orange","green4"), bty="n", lwd = 2)

stopCluster(cl)
> model_rf_rose
Random Forest

1000 samples
  58 predictor
   2 classes: 'Malo', 'Bueno'

Pre-processing: centered (58), scaled (58)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 899, 901, 900, 900, 899, 900, ...
Resampling results across tuning parameters:

  mtry  logLoss    AUC        prAUC      Accuracy   Kappa
   2    0.4649516  0.9244400  0.9025823  0.8343770  0.6681846
   8    0.4254412  0.9305596  0.9091102  0.8479908  0.6955502
  14    0.4308991  0.9179715  0.8966478  0.8363135  0.6721735
  20    0.4345766  0.9119564  0.8900175  0.8329468  0.6653921
  26    0.4364932  0.9068475  0.8844138  0.8253131  0.6501125
  33    0.4385082  0.9030930  0.8804220  0.8256529  0.6508407
  39    0.4413979  0.8990692  0.8771281  0.8149792  0.6294397
  45    0.4443302  0.8952079  0.8730362  0.8162957  0.6320616
  51    0.4457992  0.8935575  0.8712681  0.8062921  0.6120269
  58    0.4495838  0.8875012  0.8652828  0.8053187  0.6100863
  F1         Sensitivity  Specificity  Pos_Pred_Value  Neg_Pred_Value
  0.8460677  0.8987712    0.7684898    0.8015011       0.8834649
  0.8565164  0.8981830    0.7965306    0.8199701       0.8863596
  0.8455065  0.8875948    0.7837551    0.8088944       0.8744749
  0.8430100  0.8889020    0.7755918    0.8029489       0.8743549
  0.8361790  0.8830719    0.7661905    0.7956209       0.8670771
  0.8361355  0.8804183    0.7696327    0.7971022       0.8641865
  0.8258126  0.8691895    0.7594694    0.7878600       0.8523454
  0.8277346  0.8738039    0.7574422    0.7873770       0.8558341
  0.8183224  0.8645490    0.7466803    0.7778769       0.8452617
  0.8175292  0.8645752    0.7447075    0.7764609       0.8452062
  Precision  Recall     Detection_Rate  Balanced_Accuracy
```

```
  0.8015011  0.8987712  0.4539449       0.8336305
  0.8199701  0.8981830  0.4536583       0.8473568
  0.8088944  0.8875948  0.4483247       0.8356749
  0.8029489  0.8889020  0.4489881       0.8322469
  0.7956209  0.8830719  0.4460180       0.8246312
  0.7971022  0.8804183  0.4446845       0.8250255
  0.7878600  0.8691895  0.4390111       0.8143295
  0.7873770  0.8738039  0.4413345       0.8156230
  0.7778769  0.8645490  0.4366643       0.8056146
  0.7764609  0.8645752  0.4366843       0.8046413

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 8.
```

Árboles de decisión

```
library(MLmetrics)
library(dplyr)
library(ggplot2)
library(tidyverse)
library(pROC)
library(caret)
library(reshape)
library(doParallel)

cl <- makePSOCKcluster(12)
registerDoParallel(cl)

model_cart <- train(data.credit_rating ~., data = dataTrain, method = "rpart",
↪  trControl = ctrl, preProcess = c("nzv", "BoxCox"), tuneLength = 10)
predicciones_cart = predict(model_cart, newdata = dataTest)
confusion = confusionMatrix(data = predicciones_cart, reference =
↪  dataTest$data.credit_rating)

model_cart_up <- train(Class ~., data = up_train, method = "rpart", trControl = ctrl,
↪  preProcess = c("nzv", "BoxCox"), tuneLength = 10)
predicciones_cart_up = predict(model_cart_up, newdata = dataTest)
confusion_up = confusionMatrix(data = predicciones_cart_up, reference =
↪  dataTest$data.credit_rating)

model_cart_down <- train(Class ~., data = down_train, method = "rpart", trControl =
↪  ctrl, preProcess = c("nzv", "BoxCox"), tuneLength = 10)
predicciones_cart_down = predict(model_cart_down, newdata = dataTest)
confusion_down = confusionMatrix(data = predicciones_cart_down, reference =
↪  dataTest$data.credit_rating)
```

```
24
25 model_cart_smote <- train(class ~., data = smote_train, method = "rpart", trControl =
   ↪  ctrl, preProcess = c("nzv", "BoxCox"), tuneLength = 10)
26 predicciones_cart_smote = predict(model_cart_smote, newdata = dataTest)
27 confusion_smote = confusionMatrix(data = predicciones_cart_smote, reference =
   ↪  dataTest$data.credit_rating)
28
29 model_cart_rose <- train(rose_train.credit_rating ~., data = rose_train, method =
   ↪  "rpart", trControl = ctrl, preProcess = c("nzv", "BoxCox"), tuneLength = 10)
30 predicciones_cart_rose = predict(model_cart_rose, newdata = dataTest)
31 confusion_rose = confusionMatrix(data = predicciones_cart_rose, reference =
   ↪  dataTest$data.credit_rating)
32
33 byClass <- rbind(confusion =round(confusion$byClass, 3),
34                  confusion_up =round(confusion_up$byClass,3),
35                  confusion_down =round(confusion_down$byClass,3),
36                  confusion_smote =round(confusion_smote$byClass,3),
37                  confusion_rose =round(confusion_rose$byClass,3))
38
39 overall <- rbind(confusion =round(confusion$overall[1:2], 3),
40                  confusion_up =round(confusion_up$overall[1:2],3),
41                  confusion_down =round(confusion_down$overall[1:2],3),
42                  confusion_smote =round(confusion_smote$overall[1:2],3),
43                  confusion_rose =round(confusion_rose$overall[1:2],3))
44 Table2 <- t(cbind(byClass, overall))
45 colnames(Table2) = c("CART", "CART - Oversampling", "CART - Undersampling", "CART -
   ↪  SMOTE", "CART - ROSE" )
46
47 mat.heatmap.m = melt(Table2)
48
49 ggplot(data = data.frame(mat.heatmap.m),aes(x = X2, y = X1, fill= value)) +
50   geom_tile(color = "white")+
51   geom_text(aes(label = value))+
52   scale_fill_gradient2(low = "red", high = "blue", mid = "white",
53                        midpoint = 0.5, limit = c(0,1), space = "Lab",
54                        name="") +
55   theme_minimal()+
56   theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank())+
57   scale_x_discrete(position = "top") +
58   labs(x = "", y="")
59
60
61 roc_rose = roc(dataTest$data.credit_rating, as.numeric(predicciones_cart_rose),
   ↪  percent=TRUE,
62     print.auc = TRUE)
63 roc_smote =roc(as.numeric(dataTest$data.credit_rating),
   ↪  as.numeric(predicciones_cart_smote), percent=TRUE, col="#4daf4a",lwd= 2,
64          print.auc =TRUE, add=TRUE)
```

```
roc_up =roc(as.numeric(dataTest$data.credit_rating), as.numeric(predicciones_cart_up), percent=TRUE, col="red",lwd= 2,
        print.auc =TRUE, add=TRUE)
roc_down =roc(as.numeric(dataTest$data.credit_rating), as.numeric(predicciones_cart_down), percent=TRUE, col="yellow2",lwd= 2,
        print.auc =TRUE, add=TRUE)
roc_base =roc(as.numeric(dataTest$data.credit_rating), as.numeric(predicciones_cart), percent=TRUE, col="pink",lwd= 2,
        print.auc =TRUE, add=TRUE)
roc_list = list (roc_base,roc_up,roc_down, roc_smote,roc_rose)

ggroc(roc_list)+
  scale_color_discrete(name ="",labels=c("Desbalanceado", "Oversampling", "Undersampling", "SMOTE", "ROSE"))

par(mar=c(5.1,4.1,6,2.1), xpd=F, pty = "s")
roc(as.numeric(dataTest$data.credit_rating), as.numeric(predicciones_cart), plot = TRUE, legacy.axes = TRUE,
    percent = TRUE, xlab = "Porcentaje Falsos positivos",
    ylab = "Porcentaje verdaderos postivios", col = "black", lwd = 2,
    print.auc = TRUE, print.auc.x = 40, print.auc.y = 22, xlim=c(100, 0), ylim = c(0,100))
plot.roc(as.numeric(dataTest$data.credit_rating), as.numeric(predicciones_cart_up), percent=TRUE, col="red",lwd= 2,
        print.auc =TRUE, add=TRUE, print.auc.x = 40, print.auc.y = 17)
plot.roc(as.numeric(dataTest$data.credit_rating), as.numeric(predicciones_cart_down), percent=TRUE,lwd= 2,
        print.auc =TRUE, add=TRUE, col = "blue", print.auc.x = 40, print.auc.y = 12.5)
plot.roc(dataTest$data.credit_rating, as.numeric(predicciones_cart_rose), percent=TRUE,
        print.auc = TRUE,add = TRUE, col = "orange", print.auc.x = 40, print.auc.y = 8)
plot.roc(as.numeric(dataTest$data.credit_rating), as.numeric(predicciones_cart_smote), percent=TRUE, col="green4",lwd= 2,
        print.auc =TRUE, add=TRUE, print.auc.x = 40, print.auc.y = 3)
legend("bottomright", c("Desbalanceado", "Oversampling", "Undersampling", "ROSE", "SMOTE"), lty=1,
       col = c("black", "red", "blue","orange","green4"), bty="n", lwd = 2)

> model_cart_rose
CART

1000 samples
  58 predictor
   2 classes: 'Malo', 'Bueno'

Pre-processing: Box-Cox transformation (4), remove (6)
```

```
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 899, 901, 899, 900, 901, 899, ...
Resampling results across tuning parameters:

  cp           logLoss    AUC        prAUC      Accuracy   Kappa
  0.005050505  0.9631414  0.7314109  0.6527054  0.6877167  0.3751802
  0.006060606  0.8733246  0.7244964  0.6518925  0.6877302  0.3750981
  0.008080808  0.7398694  0.7150712  0.6401399  0.6844099  0.3683412
  0.009427609  0.7206351  0.7061219  0.6158696  0.6777660  0.3550393
  0.012121212  0.6673823  0.7084788  0.5645928  0.6740824  0.3476762
  0.014141414  0.6612666  0.7060428  0.5189389  0.6760489  0.3515271
  0.018181818  0.6259074  0.6846392  0.3950417  0.6750559  0.3485426
  0.034343434  0.6372198  0.6528651  0.3024090  0.6573987  0.3128554
  0.038383838  0.6436386  0.6480126  0.2638371  0.6537419  0.3058002
  0.303030303  0.6772960  0.5644830  0.1171062  0.5679163  0.1292358
  F1         Sensitivity  Specificity  Pos_Pred_Value  Neg_Pred_Value
  0.6934612  0.7044314    0.6706122    0.6861572       0.6926638
  0.6959750  0.7123399    0.6625714    0.6835169       0.6956775
  0.6973165  0.7216340    0.6465034    0.6787680       0.6966097
  0.6922826  0.7197647    0.6350884    0.6710092       0.6916507
  0.6877689  0.7130458    0.6344626    0.6693524       0.6870649
  0.6919234  0.7236340    0.6277007    0.6703178       0.6952017
  0.7133047  0.8074641    0.5402449    0.6442963       0.7462891
  0.7057570  0.8174641    0.4944898    0.6241367       0.7361240
  0.6982969  0.7950196    0.5100544    0.6255413       0.7131502
  0.6742904  0.8860000    0.2429660    0.5536597       0.6829090
  Precision  Recall     Detection_Rate  Balanced_Accuracy
  0.6861572  0.7044314  0.3557488       0.6875218
  0.6835169  0.7123399  0.3597257       0.6874556
  0.6787680  0.7216340  0.3643958       0.6840687
  0.6710092  0.7197647  0.3634189       0.6774266
  0.6693524  0.7130458  0.3600653       0.6737542
  0.6703178  0.7236340  0.3653823       0.6756673
  0.6442963  0.8074641  0.4077139       0.6738545
  0.6241367  0.8174641  0.4127522       0.6559769
  0.6255413  0.7950196  0.4014116       0.6525370
  0.5536597  0.8860000  0.4474709       0.5644830

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.006060606.

> model_cart_rose$finalModel
n= 1000

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

   1) root 1000 495 Malo (0.50500000 0.49500000)
```

```
148      2) account_status.A14< 0.5 654 247 Malo (0.62232416 0.37767584)
149        4) guarantors.A103< 0.5 605 213 Malo (0.64793388 0.35206612)
150          8) credit_amount>=6608.027 103  17 Malo (0.83495146 0.16504854) *
151          9) credit_amount< 6608.027 502 196 Malo (0.60956175 0.39043825)
152           18) installment_rate>=2.92656 311  97 Malo (0.68810289 0.31189711)
153             36) credit_amount< 851.6458 79  11 Malo (0.86075949 0.13924051) *
154             37) credit_amount>=851.6458 232  86 Malo (0.62931034 0.37068966)
155               74) months>=42.43661 21   1 Malo (0.95238095 0.04761905) *
156               75) months< 42.43661 211  85 Malo (0.59715640 0.40284360)
157                150) personal_status.A93< 0.5 97  27 Malo (0.72164948 0.27835052)
158                  300) residence>=1.061837 82  17 Malo (0.79268293 0.20731707) *
159                  301) residence< 1.061837 15   5 Bueno (0.33333333 0.66666667) *
160                151) personal_status.A93>=0.5 114  56 Bueno (0.49122807 0.50877193)
161                  302) savings.A61>=0.5 74  30 Malo (0.59459459 0.40540541)
162                    604) credit_history.A34< 0.5 57  18 Malo (0.68421053 0.31578947) *
163                    605) credit_history.A34>=0.5 17   5 Bueno (0.29411765 0.70588235)
                       ↪  *
164                  303) savings.A61< 0.5 40  12 Bueno (0.30000000 0.70000000) *
165             19) installment_rate< 2.92656 191  92 Bueno (0.48167539 0.51832461)
166               38) age< 4.445901 38   9 Malo (0.76315789 0.23684211) *
167               39) age>=4.445901 153  63 Bueno (0.41176471 0.58823529)
168                 78) purpose.A412>=0.5 7   0 Malo (1.00000000 0.00000000) *
169                 79) purpose.A412< 0.5 146  56 Bueno (0.38356164 0.61643836)
170                  158) months>=28.92655 21   7 Malo (0.66666667 0.33333333) *
171                  159) months< 28.92655 125  42 Bueno (0.33600000 0.66400000)
172                    318) dependents>=0.09165931 50  24 Malo (0.52000000 0.48000000)
173                      636) property.A124>=0.5 9   1 Malo (0.88888889 0.11111111) *
174                      637) property.A124< 0.5 41  18 Bueno (0.43902439 0.56097561)
175                       1274) residence< 1.763762 13   3 Malo (0.76923077 0.23076923) *
176                       1275) residence>=1.763762 28   8 Bueno (0.28571429 0.71428571) *
177                    319) dependents< 0.09165931 75  16 Bueno (0.21333333 0.78666667) *
178          5) guarantors.A103>=0.5 49  15 Bueno (0.30612245 0.69387755)
179           10) other_installments.A141>=0.5 12   3 Malo (0.75000000 0.25000000) *
180           11) other_installments.A141< 0.5 37   6 Bueno (0.16216216 0.83783784)
181             22) months>=37.28321 8   2 Malo (0.75000000 0.25000000) *
182             23) months< 37.28321 29   0 Bueno (0.00000000 1.00000000) *
183      3) account_status.A14>=0.5 346  98 Bueno (0.28323699 0.71676301)
184        6) other_installments.A143< 0.5 67  25 Malo (0.62686567 0.37313433)
185         12) installment_rate>=1.889341 52  14 Malo (0.73076923 0.26923077)
186           24) purpose.A43< 0.5 40   6 Malo (0.85000000 0.15000000) *
187           25) purpose.A43>=0.5 12   4 Bueno (0.33333333 0.66666667) *
188         13) installment_rate< 1.889341 15   4 Bueno (0.26666667 0.73333333) *
189        7) other_installments.A143>=0.5 279  56 Bueno (0.20071685 0.79928315)
190         14) employment.A72>=0.5 36  18 Malo (0.50000000 0.50000000)
191           28) credit_cards>=0.2570882 20   4 Malo (0.80000000 0.20000000) *
192           29) credit_cards< 0.2570882 16   2 Bueno (0.12500000 0.87500000) *
193         15) employment.A72< 0.5 243  38 Bueno (0.15637860 0.84362140) *
```

```
194 >
```

SVM

```
1  ##############################Lineal#####################################
2  library(MLmetrics)
3  library(dplyr)
4  library(ggplot2)
5  library(tidyverse)
6  library(pROC)
7  library(caret)
8  library(reshape)
9  library(doParallel)
10
11 cl <- makePSOCKcluster(12)
12 registerDoParallel(cl)
13
14 model_svm <- train(data.credit_rating ~., data = dataTrain, method = "svmLinear",
   ↪  trControl = ctrl, preProcess = c("nzv", "BoxCox"), tuneLength = 10)
15 predicciones_svm = predict(model_svm, newdata = dataTest)
16 confusion = confusionMatrix(data = predicciones_svm, reference =
   ↪  dataTest$data.credit_rating)
17
18 model_svm_up <- train(Class ~., data = up_train, method = "svmLinear", trControl =
   ↪  ctrl, preProcess = c("nzv", "BoxCox"), tuneLength = 10)
19 predicciones_svm_up = predict(model_svm_up, newdata = dataTest)
20 confusion_up = confusionMatrix(data = predicciones_svm_up, reference =
   ↪  dataTest$data.credit_rating)
21
22 model_svm_down <- train(Class ~., data = down_train, method = "svmLinear", trControl
   ↪  = ctrl, preProcess = c("nzv", "BoxCox"), tuneLength = 10)
23 predicciones_svm_down = predict(model_svm_down, newdata = dataTest)
24 confusion_down = confusionMatrix(data = predicciones_svm_down, reference =
   ↪  dataTest$data.credit_rating)
25
26 model_svm_smote <- train(class ~., data = smote_train, method = "svmLinear",
   ↪  trControl = ctrl, preProcess = c("nzv", "BoxCox"), tuneLength = 10)
27 predicciones_svm_smote = predict(model_svm_smote, newdata = dataTest)
28 confusion_smote = confusionMatrix(data = predicciones_svm_smote, reference =
   ↪  dataTest$data.credit_rating)
29
30 model_svm_rose <- train(rose_train.credit_rating ~., data = rose_train, method =
   ↪  "svmLinear", trControl = ctrl, preProcess = c("center", "scale"), tuneLength =
   ↪  10)
```

```
predicciones_svm_rose = predict(model_svm_rose, newdata = dataTest)
confusion_rose = confusionMatrix(data = predicciones_svm_rose, reference = dataTest$data.credit_rating)

byClass <- rbind(confusion =round(confusion$byClass, 3),
                 confusion_up =round(confusion_up$byClass,3),
                 confusion_down =round(confusion_down$byClass,3),
                 confusion_smote =round(confusion_smote$byClass,3),
                 confusion_rose =round(confusion_rose$byClass,3))

overall <- rbind(confusion =round(confusion$overall[1:2], 3),
                 confusion_up =round(confusion_up$overall[1:2],3),
                 confusion_down =round(confusion_down$overall[1:2],3),
                 confusion_smote =round(confusion_smote$overall[1:2],3),
                 confusion_rose =round(confusion_rose$overall[1:2],3))
Table2 <- t(cbind(byClass, overall))
colnames(Table2) = c("SVM_l", "SVM_l - Oversampling", "SVM_l - Undersampling", "SVM_l - SMOTE", "SVM_l - ROSE" )

mat.heatmap.m = melt(Table2)

ggplot(data = data.frame(mat.heatmap.m),aes(x = X2, y = X1, fill= value)) +
  geom_tile(color = "white")+
  geom_text(aes(label = value))+
  scale_fill_gradient2(low = "red", high = "blue", mid = "white",
                       midpoint = 0.5, limit = c(0,1), space = "Lab",
                       name="") +
  theme_minimal()+
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank())+
  scale_x_discrete(position = "top") +
  labs(x = "", y="")

par(mar=c(5.1,4.1,6,2.1), xpd=F, pty = "s")
roc(as.numeric(dataTest$data.credit_rating), as.numeric(predicciones_svm), plot = TRUE, legacy.axes = TRUE,
    percent = TRUE, xlab = "Porcentaje Falsos positivos",
    ylab = "Porcentaje verdaderos postivios", col = "black", lwd = 2,
    print.auc = TRUE, print.auc.x = 50, print.auc.y = 22, xlim=c(100, 0), ylim = c(0,100))
plot.roc(as.numeric(dataTest$data.credit_rating), as.numeric(predicciones_svm_up), percent=TRUE, col="red",lwd= 2,
         print.auc =TRUE, add=TRUE, print.auc.x = 50, print.auc.y = 17)
plot.roc(as.numeric(dataTest$data.credit_rating), as.numeric(predicciones_svm_down), percent=TRUE,lwd= 2,
         print.auc =TRUE, add=TRUE, col = "blue", print.auc.x = 50, print.auc.y = 12.5)
plot.roc(dataTest$data.credit_rating, as.numeric(predicciones_svm_rose), percent=TRUE,
```

```
        print.auc = TRUE,add = TRUE, col = "orange", print.auc.x = 50, print.auc.y = 8)
plot.roc(as.numeric(dataTest$data.credit_rating), as.numeric(predicciones_svm_smote), percent=TRUE, col="green4",lwd= 2,
        print.auc =TRUE, add=TRUE, print.auc.x = 50, print.auc.y = 3)
legend("bottomright", c("Desbalanceado", "Oversampling", "Undersampling", "ROSE", "SMOTE"), lty=1,
       col = c("black", "red", "blue","orange","green4"), bty="n", lwd = 2)
> model_svm_rose
Support Vector Machines with Linear Kernel

1000 samples
  58 predictor
   2 classes: 'Malo', 'Bueno'

Pre-processing: centered (58), scaled (58)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 901, 900, 901, 900, 900, 899, ...
Resampling results:

  logLoss    AUC        prAUC      Accuracy  Kappa      F1
  0.5600087  0.7918205  0.7630842  0.711343  0.4225612  0.7178235
  Sensitivity  Specificity  Pos_Pred_Value  Neg_Pred_Value  Precision
  0.7287059    0.6937823    0.7102102       0.7162288       0.7102102
  Recall     Detection_Rate  Balanced_Accuracy
  0.7287059  0.3680253       0.7112441

Tuning parameter 'C' was held constant at a value of 1

##############################Radial#################################
library(MLmetrics)
library(dplyr)
library(ggplot2)
library(tidyverse)
library(pROC)
library(caret)
library(reshape)
library(doParallel)

cl <- makePSOCKcluster(12)
registerDoParallel(cl)

model_svmRadial <- train(data.credit_rating ~., data = dataTrain, method = "svmRadial", trControl = ctrl, preProcess = c("nzv", "BoxCox"), tuneLength = 10)
predicciones_svmRadial = predict(model_svmRadial, newdata = dataTest)
confusion = confusionMatrix(data = predicciones_svmRadial, reference = dataTest$data.credit_rating)

```

```
model_svmRadial_up <- train(Class ~., data = up_train, method = "svmRadial", trControl = ctrl, preProcess = c("nzv", "BoxCox"), tuneLength = 10)
predicciones_svmRadial_up = predict(model_svmRadial_up, newdata = dataTest)
confusion_up = confusionMatrix(data = predicciones_svmRadial_up, reference = dataTest$data.credit_rating)

model_svmRadial_down <- train(Class ~., data = down_train, method = "svmRadial", trControl = ctrl, preProcess = c("nzv", "BoxCox"), tuneLength = 10)
predicciones_svmRadial_down = predict(model_svmRadial_down, newdata = dataTest)
confusion_down = confusionMatrix(data = predicciones_svmRadial_down, reference = dataTest$data.credit_rating)

model_svmRadial_smote <- train(class ~., data = smote_train, method = "svmRadial", trControl = ctrl, preProcess = c("nzv", "BoxCox"), tuneLength = 10)
predicciones_svmRadial_smote = predict(model_svmRadial_smote, newdata = dataTest)
confusion_smote = confusionMatrix(data = predicciones_svmRadial_smote, reference = dataTest$data.credit_rating)

model_svmRadial_rose <- train(rose_train.credit_rating ~., data = rose_train, method = "svmRadial", trControl = ctrl, preProcess = c("center", "scale"), tuneLength = 10)
predicciones_svmRadial_rose = predict(model_svmRadial_rose, newdata = dataTest)
confusion_rose = confusionMatrix(data = predicciones_svmRadial_rose, reference = dataTest$data.credit_rating)

byClass <- rbind(confusion =round(confusion$byClass, 3),
                 confusion_up =round(confusion_up$byClass,3),
                 confusion_down =round(confusion_down$byClass,3),
                 confusion_smote =round(confusion_smote$byClass,3),
                 confusion_rose =round(confusion_rose$byClass,3))

overall <- rbind(confusion =round(confusion$overall[1:2], 3),
                 confusion_up =round(confusion_up$overall[1:2],3),
                 confusion_down =round(confusion_down$overall[1:2],3),
                 confusion_smote =round(confusion_smote$overall[1:2],3),
                 confusion_rose =round(confusion_rose$overall[1:2],3))
Table2 <- t(cbind(byClass, overall))
colnames(Table2) = c("SVM_r", "SVM_r - Oversampling", "SVM_r - Undersampling", "SVM_r - SMOTE", "SVM_r - ROSE" )

mat.heatmap.m = melt(Table2)

ggplot(data = data.frame(mat.heatmap.m),aes(x = X2, y = X1, fill= value)) +
  geom_tile(color = "white")+
  geom_text(aes(label = value))+
  scale_fill_gradient2(low = "red", high = "blue", mid = "white",
                       midpoint = 0.5, limit = c(0,1), space = "Lab",
                       name="") +
```

```
  theme_minimal()+
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank())+
  scale_x_discrete(position = "top") +
  labs(x = "", y="")

par(mar=c(5.1,4.1,6,2.1), xpd=F, pty = "s")
roc(as.numeric(dataTest$data.credit_rating), as.numeric(predicciones_svmRadial), plot = TRUE, legacy.axes = TRUE,
    percent = TRUE, xlab = "Porcentaje Falsos positivos",
    ylab = "Porcentaje verdaderos postivios", col = "black", lwd = 2,
    print.auc = TRUE, print.auc.x = 50, print.auc.y = 22, xlim=c(100, 0), ylim = c(0,100))
plot.roc(as.numeric(dataTest$data.credit_rating),
as.numeric(predicciones_svmRadial_up), percent=TRUE, col="red",lwd= 2,
         print.auc =TRUE, add=TRUE, print.auc.x = 50, print.auc.y = 17)
plot.roc(as.numeric(dataTest$data.credit_rating),
as.numeric(predicciones_svmRadial_down), percent=TRUE,lwd= 2,
         print.auc =TRUE, add=TRUE, col = "blue", print.auc.x = 50, print.auc.y = 12.5)
plot.roc(dataTest$data.credit_rating, as.numeric(predicciones_svmRadial_rose),
as.numeric(predicciones_svmRadial_rose), percent=TRUE,
         print.auc = TRUE,add = TRUE, col = "orange", print.auc.x = 50, print.auc.y = 8)
plot.roc(as.numeric(dataTest$data.credit_rating),
as.numeric(predicciones_svmRadial_smote), percent=TRUE, col="green4",lwd= 2,
         print.auc =TRUE, add=TRUE, print.auc.x = 50, print.auc.y = 3)
legend("bottomright", c("Desbalanceado", "Oversampling", "Undersampling", "ROSE",
"SMOTE"), lty=1,
       col = c("black", "red", "blue","orange","green4"), bty="n", lwd = 2)

> model_svmRadial_rose
Support Vector Machines with Radial Basis Function Kernel

1000 samples
  58 predictor
   2 classes: 'Malo', 'Bueno'

Pre-processing: centered (58), scaled (58)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 900, 900, 900, 899, 901, 900, ...
Resampling results across tuning parameters:

  C       logLoss    AUC        prAUC      Accuracy   Kappa
    0.25  0.4961253  0.8412397  0.8192402  0.7516233  0.5026556
    0.50  0.4648643  0.8647894  0.8401289  0.7819951  0.5634882
    1.00  0.4228602  0.8921576  0.8675517  0.8163035  0.6323121
    2.00  0.3830259  0.9133938  0.8905446  0.8356612  0.6711393
    4.00  0.3469917  0.9283938  0.9026394  0.8553092  0.7104576
```

```
    8.00  0.3254801  0.9367652  0.9085474  0.8749568  0.7497853
   16.00  0.3247428  0.9363430  0.9067132  0.8776469  0.7551929
   32.00  0.3330292  0.9329084  0.9024204  0.8756466  0.7512087
   64.00  0.3355280  0.9318077  0.9006495  0.8756400  0.7511689
  128.00  0.3367959  0.9302208  0.8983084  0.8736432  0.7471587
  F1         Sensitivity  Specificity  Pos_Pred_Value  Neg_Pred_Value
  0.7637868  0.7991373    0.7029524    0.7345309       0.7793884
  0.7931029  0.8302222    0.7326667    0.7618651       0.8131874
  0.8224813  0.8447843    0.7871429    0.8046578       0.8366578
  0.8405043  0.8574510    0.8134558    0.8270280       0.8511059
  0.8592030  0.8753203    0.8348707    0.8467773       0.8711325
  0.8786524  0.8984575    0.8510612    0.8624601       0.8946790
  0.8806145  0.8958562    0.8591429    0.8683350       0.8926347
  0.8784461  0.8912288    0.8598367    0.8681911       0.8877722
  0.8788225  0.8944837    0.8564490    0.8656365       0.8903522
  0.8773938  0.8978039    0.8490476    0.8595876       0.8928452
  Precision  Recall     Detection_Rate  Balanced_Accuracy
  0.7345309  0.7991373  0.4036382       0.7510448
  0.7618651  0.8302222  0.4193393       0.7814444
  0.8046578  0.8447843  0.4266729       0.8159636
  0.8270280  0.8574510  0.4330233       0.8354534
  0.8467773  0.8753203  0.4420472       0.8550955
  0.8624601  0.8984575  0.4536944       0.8747594
  0.8683350  0.8958562  0.4523642       0.8774995
  0.8681911  0.8912288  0.4500274       0.8755327
  0.8656365  0.8944837  0.4516874       0.8754663
  0.8595876  0.8978039  0.4533676       0.8734258

Tuning parameter 'sigma' was held constant at a value of 0.01055634
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were sigma = 0.01055634 and C = 16.
##############################Polinomico###################################
library(MLmetrics)
library(dplyr)
library(ggplot2)
library(tidyverse)
library(pROC)
library(caret)
library(reshape)
library(doParallel)

cl <- makePSOCKcluster(12)
registerDoParallel(cl)

model_svmPoly <- train(data.credit_rating ~., data = dataTrain, method = "svmPoly",
↪  trControl = ctrl, preProcess = c("nzv", "BoxCox"), tuneLength = 2)
predicciones_svmPoly = predict(model_svmPoly, newdata = dataTest)
confusion = confusionMatrix(data = predicciones_svmPoly, reference =
↪  dataTest$data.credit_rating)
```

```
238
239 model_svmPoly_up <- train(Class ~., data = up_train, method = "svmPoly", trControl =
    ↪  ctrl, preProcess = c("nzv", "BoxCox"), tuneLength = 2)
240 predicciones_svmPoly_up = predict(model_svmPoly_up, newdata = dataTest)
241 confusion_up = confusionMatrix(data = predicciones_svmPoly_up, reference =
    ↪  dataTest$data.credit_rating)
242
243 model_svmPoly_down <- train(Class ~., data = down_train, method = "svmPoly",
    ↪  trControl = ctrl, preProcess = c("nzv", "BoxCox"), tuneLength = 2)
244 predicciones_svmPoly_down = predict(model_svmPoly_down, newdata = dataTest)
245 confusion_down = confusionMatrix(data = predicciones_svmPoly_down, reference =
    ↪  dataTest$data.credit_rating)
246
247 model_svmPoly_smote <- train(class ~., data = smote_train, method = "svmPoly",
    ↪  trControl = ctrl, preProcess = c("nzv", "BoxCox"), tuneLength = 2)
248 predicciones_svmPoly_smote = predict(model_svmPoly_smote, newdata = dataTest)
249 confusion_smote = confusionMatrix(data = predicciones_svmPoly_smote, reference =
    ↪  dataTest$data.credit_rating)
250
251 model_svmPoly_rose <- train(rose_train.credit_rating ~., data = rose_train, method =
    ↪  "svmPoly", trControl = ctrl, preProcess = c("center", "scale"), tuneLength = 2)
252 predicciones_svmPoly_rose = predict(model_svmPoly_rose, newdata = dataTest)
253 confusion_rose = confusionMatrix(data = predicciones_svmPoly_rose, reference =
    ↪  dataTest$data.credit_rating)
254
255 byClass <- rbind(confusion =round(confusion$byClass, 3),
256                  confusion_up =round(confusion_up$byClass,3),
257                  confusion_down =round(confusion_down$byClass,3),
258                  confusion_smote =round(confusion_smote$byClass,3),
259                  confusion_rose =round(confusion_rose$byClass,3))
260
261 overall <- rbind(confusion =round(confusion$overall[1:2], 3),
262                  confusion_up =round(confusion_up$overall[1:2],3),
263                  confusion_down =round(confusion_down$overall[1:2],3),
264                  confusion_smote =round(confusion_smote$overall[1:2],3),
265                  confusion_rose =round(confusion_rose$overall[1:2],3))
266 Table2 <- t(cbind(byClass, overall))
267 colnames(Table2) = c("SVM_p", "SVM_p - Oversampling", "SVM_p - Undersampling", "SVM_p
    ↪  - SMOTE", "SVM_p - ROSE" )
268
269 mat.heatmap.m = melt(Table2)
270
271 ggplot(data = data.frame(mat.heatmap.m),aes(x = X2, y = X1, fill= value)) +
272   geom_tile(color = "white")+
273   geom_text(aes(label = value))+
274   scale_fill_gradient2(low = "red", high = "blue", mid = "white",
275                        midpoint = 0.5, limit = c(0,1), space = "Lab",
276                        name="") +
```

```
  theme_minimal()+
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank())+
  scale_x_discrete(position = "top") +
  labs(x = "", y="")

par(mar=c(5.1,4.1,6,2.1), xpd=F, pty = "s")
roc(as.numeric(dataTest$data.credit_rating), as.numeric(predicciones_svmPoly), plot = TRUE, legacy.axes = TRUE,
    percent = TRUE, xlab = "Porcentaje Falsos positivos",
    ylab = "Porcentaje verdaderos postivios", col = "black", lwd = 2,
    print.auc = TRUE, print.auc.x = 50, print.auc.y = 22, xlim=c(100, 0), ylim = c(0,100))
plot.roc(as.numeric(dataTest$data.credit_rating),
as.numeric(predicciones_svmPoly_up), percent=TRUE, col="red",lwd= 2,
         print.auc =TRUE, add=TRUE, print.auc.x = 50, print.auc.y = 17)
plot.roc(as.numeric(dataTest$data.credit_rating),
as.numeric(predicciones_svmPoly_down), percent=TRUE,lwd= 2,
         print.auc =TRUE, add=TRUE, col = "blue", print.auc.x = 50, print.auc.y = 12.5)
plot.roc(dataTest$data.credit_rating, as.numeric(predicciones_svmPoly_rose),
as.numeric(predicciones_svmPoly_rose), percent=TRUE,
         print.auc = TRUE,add = TRUE, col = "orange", print.auc.x = 50, print.auc.y = 8)
plot.roc(as.numeric(dataTest$data.credit_rating),
as.numeric(predicciones_svmPoly_smote), percent=TRUE, col="green4",lwd= 2,
         print.auc =TRUE, add=TRUE, print.auc.x = 50, print.auc.y = 3)
legend("bottomright", c("Desbalanceado", "Oversampling", "Undersampling", "ROSE",
"SMOTE"), lty=1,
       col = c("black", "red", "blue","orange","green4"), bty="n", lwd = 2)

> model_svmPoly_rose
Support Vector Machines with Polynomial Kernel

1000 samples
  58 predictor
   2 classes: 'Malo', 'Bueno'

Pre-processing: centered (58), scaled (58)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 900, 900, 901, 899, 901, 899, ...
Resampling results across tuning parameters:

  degree  scale  C     logLoss    AUC        prAUC      Accuracy
  1       0.001  0.25  0.6557284  0.7825374  0.7596565  0.6641487
  1       0.001  0.50  0.6072190  0.7843372  0.7615295  0.6784763
  1       0.010  0.25  0.5552251  0.7903340  0.7651406  0.7147047
  1       0.010  0.50  0.5528451  0.7939561  0.7676427  0.7114079
  2       0.001  0.25  0.6033971  0.7853732  0.7626444  0.6828266
```

```
  2       0.001  0.50  0.5605768  0.7862585  0.7636294  0.7130447
  2       0.010  0.25  0.5194069  0.8235162  0.7975030  0.7347193
  2       0.010  0.50  0.4982574  0.8412853  0.8138076  0.7550135
  Kappa      F1         Sensitivity  Specificity  Pos_Pred_Value
  0.3310497  0.5793507  0.4623791    0.8699592    0.7856188
  0.3584512  0.6323913  0.5533464    0.8059456    0.7474526
  0.4289278  0.7248320  0.7486275    0.6799728    0.7055233
  0.4223741  0.7206766  0.7414248    0.6806531    0.7039891
  0.3671134  0.6373405  0.5586405    0.8093061    0.7521662
  0.4256300  0.7226799  0.7453333    0.6799728    0.7047178
  0.4689725  0.7455892  0.7730719    0.6955374    0.7223891
  0.5095517  0.7666192  0.7981046    0.7110340    0.7400227
  Neg_Pred_Value  Precision  Recall     Detection_Rate
  0.6145581       0.7856188  0.4623791  0.2334824
  0.6409532       0.7474526  0.5533464  0.2794784
  0.7305692       0.7055233  0.7486275  0.3780467
  0.7248553       0.7039891  0.7414248  0.3744300
  0.6450305       0.7521662  0.5586405  0.2821551
  0.7283579       0.7047178  0.7453333  0.3764067
  0.7537435       0.7223891  0.7730719  0.3904075
  0.7782198       0.7400227  0.7981046  0.4030283
  Balanced_Accuracy
  0.6661691
  0.6796460
  0.7143001
  0.7110389
  0.6839733
  0.7126531
  0.7343047
  0.7545693

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were degree = 2, scale = 0.01 and
 C = 0.5.

> model_svmRadial_rose$finalModel
Support Vector Machine object of class "ksvm"

SV type: C-svc  (classification)
 parameter : cost C = 16

Gaussian Radial Basis kernel function.
 Hyperparameter : sigma =  0.0105563433163171

Number of Support Vectors : 522

Objective Function Value : -1425.475
Training error : 0.004
```

```
Probability model included.
```

Naïve Bayes

```
library(MLmetrics)
library(dplyr)
library(ggplot2)
library(tidyverse)
library(pROC)
library(caret)
library(reshape)
library(doParallel)

cl <- makePSOCKcluster(12)
registerDoParallel(cl)

model_nb <- train(data.credit_rating ~., data = dataTrain, method = "nb", trControl = ctrl, preProcess = c("nzv", "BoxCox"), tuneLength = 10)
predicciones_nb = predict(model_nb, newdata = dataTest)
confusion = confusionMatrix(data = predicciones_nb, reference = dataTest$data.credit_rating)

model_nb_up <- train(Class ~., data = up_train, method = "nb", trControl = ctrl, preProcess = c("nzv", "BoxCox"), tuneLength = 10)
predicciones_nb_up = predict(model_nb_up, newdata = dataTest)
confusion_up = confusionMatrix(data = predicciones_nb_up, reference = dataTest$data.credit_rating)

model_nb_down <- train(Class ~., data = down_train, method = "nb", trControl = ctrl, preProcess = c("nzv", "BoxCox"), tuneLength = 10)
predicciones_nb_down = predict(model_nb_down, newdata = dataTest)
confusion_down = confusionMatrix(data = predicciones_nb_down, reference = dataTest$data.credit_rating)

model_nb_smote <- train(class ~., data = smote_train, method = "nb", trControl = ctrl, preProcess = c("nzv", "BoxCox"), tuneLength = 10)
predicciones_nb_smote = predict(model_nb_smote, newdata = dataTest)
confusion_smote = confusionMatrix(data = predicciones_nb_smote, reference = dataTest$data.credit_rating)

model_nb_rose <- train(rose_train.credit_rating ~., data = rose_train, method = "nb", trControl = ctrl, preProcess = c("nzv", "BoxCox"), tuneLength = 10)
predicciones_nb_rose = predict(model_nb_rose, newdata = dataTest)
confusion_rose = confusionMatrix(data = predicciones_nb_rose, reference = dataTest$data.credit_rating)
```

```
32
33 byClass <- rbind(confusion =round(confusion$byClass, 3),
34                  confusion_up =round(confusion_up$byClass,3),
35                  confusion_down =round(confusion_down$byClass,3),
36                  confusion_smote =round(confusion_smote$byClass,3),
37                  confusion_rose =round(confusion_rose$byClass,3))
38
39 overall <- rbind(confusion =round(confusion$overall[1:2], 3),
40                  confusion_up =round(confusion_up$overall[1:2],3),
41                  confusion_down =round(confusion_down$overall[1:2],3),
42                  confusion_smote =round(confusion_smote$overall[1:2],3),
43                  confusion_rose =round(confusion_rose$overall[1:2],3))
44 Table2 <- t(cbind(byClass, overall))
45 colnames(Table2) = c("NB", "NB - Oversampling", "NB - Undersampling", "NB - SMOTE",
   ↪  "NB - ROSE" )
46
47 mat.heatmap.m = melt(Table2)
48
49 ggplot(data = data.frame(mat.heatmap.m),aes(x = X2, y = X1, fill= value)) +
50   geom_tile(color = "white")+
51   geom_text(aes(label = value))+
52   scale_fill_gradient2(low = "red", high = "blue", mid = "white",
53                        midpoint = 0.5, limit = c(0,1), space = "Lab",
54                        name="") +
55   theme_minimal()+
56   theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank())+
57   scale_x_discrete(position = "top") +
58   labs(x = "", y="")
59
60 par(mar=c(5.1,4.1,6,2.1), xpd=F, pty = "s")
61 roc(as.numeric(dataTest$data.credit_rating), as.numeric(predicciones_nb), plot =
   ↪  TRUE, legacy.axes = TRUE,
62     percent = TRUE, xlab = "Porcentaje Falsos positivos",
63     ylab = "Porcentaje verdaderos postivios", col = "black", lwd = 2,
64     print.auc = TRUE, print.auc.x = 40, print.auc.y = 22, xlim=c(100, 0), ylim =
       ↪  c(0,100))
65 plot.roc(as.numeric(dataTest$data.credit_rating), as.numeric(predicciones_nb_up),
   ↪  percent=TRUE, col="red",lwd= 2,
66          print.auc =TRUE, add=TRUE, print.auc.x = 40, print.auc.y = 17)
67 plot.roc(as.numeric(dataTest$data.credit_rating), as.numeric(predicciones_nb_down),
   ↪  percent=TRUE,lwd= 2,
68          print.auc =TRUE, add=TRUE, col = "blue", print.auc.x = 40, print.auc.y =
            ↪  12.5)
69 plot.roc(dataTest$data.credit_rating, as.numeric(predicciones_nb_rose), percent=TRUE,
70          print.auc = TRUE,add = TRUE, col = "orange", print.auc.x = 40, print.auc.y =
            ↪  8)
71 plot.roc(as.numeric(dataTest$data.credit_rating), as.numeric(predicciones_nb_smote),
   ↪  percent=TRUE, col="green4",lwd= 2,
```

```
        print.auc =TRUE, add=TRUE, print.auc.x = 40, print.auc.y = 3)
legend("bottomright", c("Desbalanceado", "Oversampling", "Undersampling", "ROSE",
↪  "SMOTE"), lty=1,
       col = c("black", "red", "blue","orange","green4"), bty="n", lwd = 2)

> model_nb_rose
Naive Bayes

1000 samples
  58 predictor
   2 classes: 'Malo', 'Bueno'

Pre-processing: Box-Cox transformation (4), remove (6)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 899, 900, 901, 900, 899, 901, ...
Resampling results across tuning parameters:

  usekernel  logLoss    AUC        prAUC      Accuracy   Kappa
  FALSE      1.2811162  0.7673483  0.7301134  0.7143345  0.4285979
   TRUE      0.6223375  0.7973993  0.7729521  0.7146243  0.4298760
  F1         Sensitivity  Specificity  Pos_Pred_Value  Neg_Pred_Value
  0.7167916  0.7215294    0.7070340    0.7160995       0.7174402
  0.6965405  0.6558954    0.7744354    0.7488288       0.6918639
  Precision  Recall     Detection_Rate  Balanced_Accuracy
  0.7160995  0.7215294  0.3643327       0.7142817
  0.7488288  0.6558954  0.3312840       0.7151654

Tuning parameter 'fL' was held constant at a value of 0

Tuning parameter 'adjust' was held constant at a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were fL = 0, usekernel = TRUE
 and adjust = 1.
```

Logit

```
library(MLmetrics)
library(dplyr)
library(ggplot2)
library(tidyverse)
library(pROC)
library(caret)
library(reshape)
library(doParallel)
```

```
9
10  cl <- makePSOCKcluster(12)
11  registerDoParallel(cl)
12
13  model_logit <- train(data.credit_rating ~., data = dataTrain, method = "glmStepAIC",
    ↪  trControl = ctrl, preProcess = c("nzv", "BoxCox"), tuneLength = 10)
14  predicciones_logit = predict(model_logit, newdata = dataTest)
15  confusion = confusionMatrix(data = predicciones_logit, reference =
    ↪  dataTest$data.credit_rating)
16
17  model_logit_up <- train(Class ~., data = up_train, method = "glmStepAIC", trControl =
    ↪  ctrl, preProcess = c("nzv", "BoxCox"), tuneLength = 10)
18  predicciones_logit_up = predict(model_logit_up, newdata = dataTest)
19  confusion_up = confusionMatrix(data = predicciones_logit_up, reference =
    ↪  dataTest$data.credit_rating)
20
21  model_logit_down <- train(Class ~., data = down_train, method = "glmStepAIC",
    ↪  trControl = ctrl, preProcess = c("nzv", "BoxCox"), tuneLength = 10)
22  predicciones_logit_down = predict(model_logit_down, newdata = dataTest)
23  confusion_down = confusionMatrix(data = predicciones_logit_down, reference =
    ↪  dataTest$data.credit_rating)
24
25  model_logit_smote <- train(class ~., data = smote_train, method = "glmStepAIC",
    ↪  trControl = ctrl, preProcess = c("nzv", "BoxCox"), tuneLength = 10)
26  predicciones_logit_smote = predict(model_logit_smote, newdata = dataTest)
27  confusion_smote = confusionMatrix(data = predicciones_logit_smote, reference =
    ↪  dataTest$data.credit_rating)
28
29  model_logit_rose <- train(rose_train.credit_rating ~., data = rose_train, method =
    ↪  "glmStepAIC", trControl = ctrl, preProcess = c("center", "scale"), tuneLength =
    ↪  10)
30  predicciones_logit_rose = predict(model_logit_rose, newdata = dataTest)
31  confusion_rose = confusionMatrix(data = predicciones_logit_rose, reference =
    ↪  dataTest$data.credit_rating)
32
33  byClass <- rbind(confusion =round(confusion$byClass, 3),
34                   confusion_up =round(confusion_up$byClass,3),
35                   confusion_down =round(confusion_down$byClass,3),
36                   confusion_smote =round(confusion_smote$byClass,3),
37                   confusion_rose =round(confusion_rose$byClass,3))
38
39  overall <- rbind(confusion =round(confusion$overall[1:2], 3),
40                   confusion_up =round(confusion_up$overall[1:2],3),
41                   confusion_down =round(confusion_down$overall[1:2],3),
42                   confusion_smote =round(confusion_smote$overall[1:2],3),
43                   confusion_rose =round(confusion_rose$overall[1:2],3))
44  Table2 <- t(cbind(byClass, overall))
45  colnames(Table2) = c("Logit", "Logit - Oversampling", "Logit - Undersampling", "Logit
    ↪  - SMOTE", "Logit - ROSE" )
```

```

mat.heatmap.m = melt(Table2)

ggplot(data = data.frame(mat.heatmap.m),aes(x = X2, y = X1, fill= value)) +
  geom_tile(color = "white")+
  geom_text(aes(label = value))+
  scale_fill_gradient2(low = "red", high = "blue", mid = "white",
                       midpoint = 0.5, limit = c(0,1), space = "Lab",
                       name="") +
  theme_minimal()+
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank())+
  scale_x_discrete(position = "top") +
  labs(x = "", y="")

par(mar=c(5.1,4.1,6,2.1), xpd=F, pty = "s")
roc(as.numeric(dataTest$data.credit_rating), as.numeric(predicciones_logit), plot =
↪  TRUE, legacy.axes = TRUE,
    percent = TRUE, xlab = "Porcentaje Falsos positivos",
    ylab = "Porcentaje verdaderos postivios", col = "black", lwd = 2,
    print.auc = TRUE, print.auc.x = 50, print.auc.y = 22, xlim=c(100, 0), ylim =
    ↪  c(0,100))
plot.roc(as.numeric(dataTest$data.credit_rating), as.numeric(predicciones_logit_up),
↪  percent=TRUE, col="red",lwd= 2,
         print.auc =TRUE, add=TRUE, print.auc.x = 50, print.auc.y = 17)
plot.roc(as.numeric(dataTest$data.credit_rating),
↪  as.numeric(predicciones_logit_down), percent=TRUE,lwd= 2,
         print.auc =TRUE, add=TRUE, col = "blue", print.auc.x = 50, print.auc.y =
         ↪  12.5)
plot.roc(dataTest$data.credit_rating, as.numeric(predicciones_logit_rose),
↪  percent=TRUE,
         print.auc = TRUE,add = TRUE, col = "orange", print.auc.x = 50, print.auc.y =
         ↪  8)
plot.roc(as.numeric(dataTest$data.credit_rating),
↪  as.numeric(predicciones_logit_smote), percent=TRUE, col="green4",lwd= 2,
         print.auc =TRUE, add=TRUE, print.auc.x = 50, print.auc.y = 3)
legend("bottomright", c("Desbalanceado", "Oversampling", "Undersampling", "ROSE",
↪  "SMOTE"), lty=1,
       col = c("black", "red", "blue","orange","green4"), bty="n", lwd = 2)
> model_logit_rose
Generalized Linear Model with Stepwise Feature Selection

1000 samples
  58 predictor
   2 classes: 'Malo', 'Bueno'

Pre-processing: centered (58), scaled (58)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 900, 901, 900, 899, 899, 901, ...
```

```
Resampling results:

  logLoss    AUC        prAUC      Accuracy   Kappa      F1
  0.5714662  0.7848961  0.7579242  0.7055857  0.4110145  0.7092728
  Sensitivity  Specificity  Pos_Pred_Value  Neg_Pred_Value  Precision
  0.7120654    0.6988844    0.7086914       0.7049599       0.7086914
  Recall     Detection_Rate  Balanced_Accuracy
  0.7120654  0.3596117       0.7054749

> model_logit_rose$finalModel$coefficients
           (Intercept)      account_status.A11      account_status.A12
           -0.01400108             -0.80334426             -0.52573203
     account_status.A13                  months      credit_history.A30
           -0.13814018             -0.25873065             -0.19535152
     credit_history.A31      credit_history.A32             purpose.A41
           -0.24797793             -0.23443899              0.47894368
          purpose.A410            purpose.A411            purpose.A412
            0.17369935              0.40211330             -0.14584551
           purpose.A43           credit_amount            savings.A64
            0.35265227             -0.43916414              0.15242419
         employment.A72          employment.A74        installment_rate
           -0.15590241              0.50048592             -0.29070550
    personal_status.A91     personal_status.A92         guarantors.A101
           -0.14496955             -0.21400701             -0.36755894
        guarantors.A102           property.A121           property.A123
           -0.42115796              0.19104383             -0.14020147
                   age other_installments.A141            housing.A151
            0.14783332             -0.33178382             -0.25916624
           credit_cards              phone.A191     foreign_worker.A201
           -0.15907851             -0.13159458             -0.19099701

Coefficients:
                      Estimate Std. Error z value Pr(>|z|)
(Intercept)           -0.01400    0.07789  -0.180 0.857350
account_status.A11    -0.80334    0.09812  -8.187 2.67e-16 ***
account_status.A12    -0.52573    0.09281  -5.665 1.47e-08 ***
account_status.A13    -0.13814    0.07771  -1.778 0.075472 .
months                -0.25873    0.09065  -2.854 0.004315 **
credit_history.A30    -0.19535    0.09658  -2.023 0.043101 *
credit_history.A31    -0.24798    0.08953  -2.770 0.005609 **
credit_history.A32    -0.23444    0.10531  -2.226 0.026005 *
purpose.A41            0.47894    0.08868   5.401 6.64e-08 ***
purpose.A410           0.17370    0.08517   2.039 0.041418 *
purpose.A411           0.40211    0.09024   4.456 8.35e-06 ***
purpose.A412          -0.14585    0.08562  -1.703 0.088500 .
purpose.A43            0.35265    0.09197   3.834 0.000126 ***
credit_amount         -0.43916    0.10387  -4.228 2.36e-05 ***
savings.A64            0.15242    0.08840   1.724 0.084670 .
```

```
employment.A72          -0.15590    0.08272  -1.885 0.059470 .
employment.A74           0.50049    0.08743   5.724 1.04e-08 ***
installment_rate        -0.29071    0.08333  -3.488 0.000486 ***
personal_status.A91     -0.14497    0.08023  -1.807 0.070757 .
personal_status.A92     -0.21401    0.08408  -2.545 0.010916 *
guarantors.A101         -0.36756    0.11497  -3.197 0.001389 **
guarantors.A102         -0.42116    0.11383  -3.700 0.000216 ***
property.A121            0.19104    0.09090   2.102 0.035575 *
property.A123           -0.14020    0.09114  -1.538 0.123979
age                      0.14783    0.08413   1.757 0.078882 .
other_installments.A141 -0.33178    0.08143  -4.074 4.61e-05 ***
housing.A151            -0.25917    0.08402  -3.084 0.002039 **
credit_cards            -0.15908    0.09749  -1.632 0.102728
phone.A191              -0.13159    0.08300  -1.586 0.112844
foreign_worker.A201     -0.19100    0.09719  -1.965 0.049402 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1386.2  on 999  degrees of freedom
Residual deviance: 1018.9  on 970  degrees of freedom
AIC: 1078.9

Number of Fisher Scoring iterations: 5
```

Gráfico CART

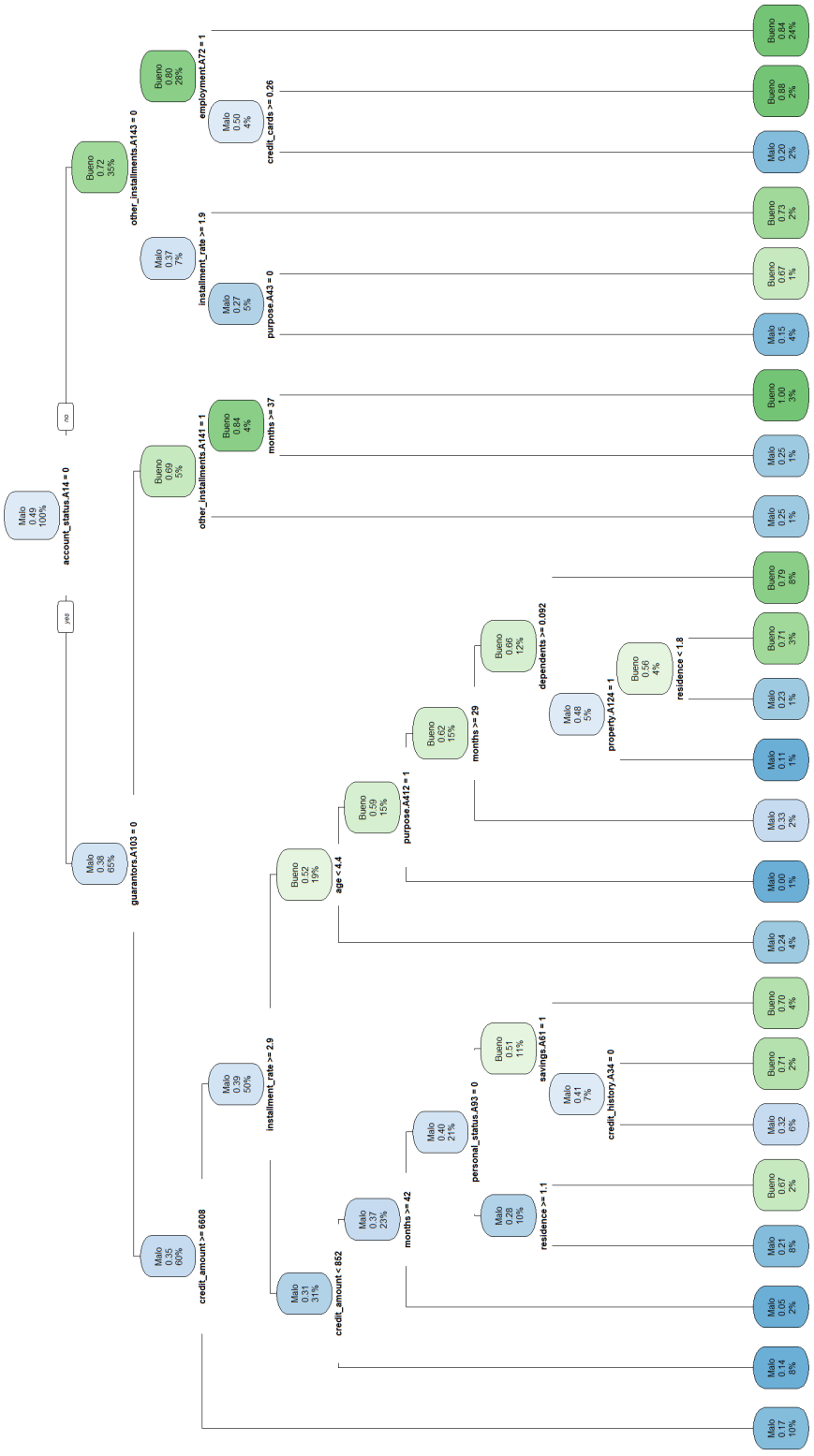


Figura 3.36: Árbol de decisión del modelo final utilizando ROSE

Curvas ROC adicionales

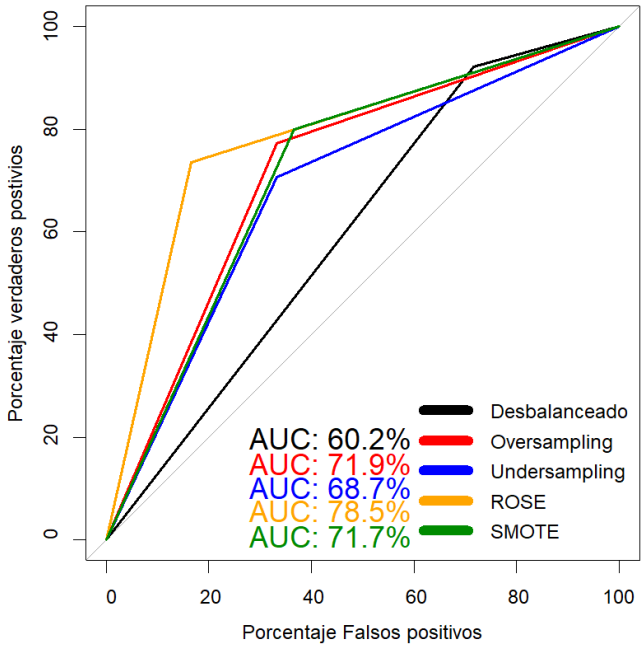


Figura 3.37: Curva ROC del algoritmo SVM usando un kernel lineal

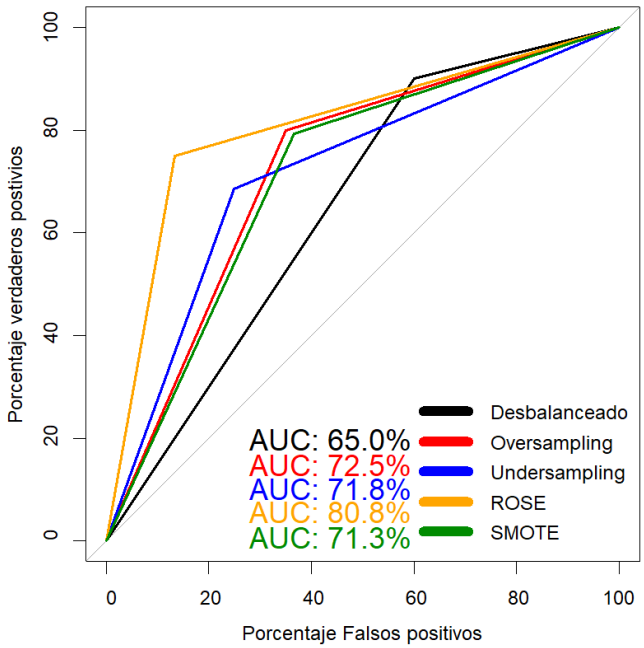


Figura 3.38: Curva ROC del algoritmo SVM usando un kernel polinómico