Facultad de Ciencias

APLICACIÓN DE ALGORITMOS DE CLASIFICACIÓN PARA LA EVALUACIÓN DEL RATING CREDITICIO EN EMPRESAS PYMES CÁNTABRAS

(Application of classification algorithms for the assessment of credit rating in Cantabrian SMEs)

Trabajo de Fin de Máster
para acceder al

MÁSTER EN CIENCIA DE DATOS

Autor: Adriana Borroto Blanco

Director: Sixto Herrera García

Co-Director: Begoña Torre Olmo

Enero – 2024

Resumen

La aplicación de algoritmos estadísticos y de machine learning toman cada vez más relevancia en el sector bancario y financiero, ya que permiten realizar de una manera eficiente y automatizada distintas tareas que facilitan la toma de decisiones ágiles por parte de dichas organizaciones.

En la presente memoria se tiene como objetivo aplicar una serie de algoritmos de clasificación que permitan determinar el rating crediticio de las pequeñas y medianas empresas (pymes) cántabras, con el fin de brindar una herramienta útil a un organismo público financiero, que contribuya con la decisión de la concesión o no de créditos a las empresas solicitantes.

Para alcanzar los objetivos del presente trabajo se desarrolló un análisis previo de los datos, tras lo cual se aplicaron once algoritmos de clasificación, seleccionando el óptimo en base al mejor desempeño.

Como resultado del proyecto, se encontró que los algoritmos XGBoost, Random Forest, Adaptive Boosting, Gradient Boosting, KNN, SVM y redes neuronales obtuvieron una mayor precisión a la hora de clasificar correctamente a las empresas, mientras que LDA, regresión logística y redes bayesianas tuvieron un desempeño inferior. Finalmente, se escoge en base a su performance y precisión obtenida el modelo XGBoost para ser la base de clasificación de rating crediticio.

Palabras clave: Rating crediticio, algoritmos de clasificación, machine learning, calificación crediticia.

Abstract

The application of statistical algorithms and machine learning is becoming increasingly relevant in the banking and financial sector, as they allow for efficient and automated execution of various tasks that facilitate agile decision-making by these organizations.

The objective of this report is to apply a series of classification algorithms to determine the credit rating of small and medium-sized enterprises (SMEs) in Cantabria, in order to provide a useful tool to a public financial institution that contributes to the decision regarding the approval or rejection of credit applications from these companies.

To achieve the objectives of this study, a preliminary data analysis was conducted, and then, eleven classification algorithms were applied, selecting the optimal one based on the best performance.

As a result of the project, it was found that the XGBoost, Random Forest, Adaptive Boosting, Gradient Boosting, KNN, SVM, and neural networks algorithms achieved higher accuracy in correctly classifying the companies, while LDA, logistic regression and Bayesian networks performed poorly. Finally, based on its performance and obtained accuracy, the XGBoost model is chosen to serve as the basis for credit rating classification.

Keywords: Credit rating, classification algorithms, machine learning, credit assessment.

Tabla de Contenido

Tabla de Ilustraciones

Tabla de Tablas

Tabla de Ecuaciones

1 Introducción

1.1 Marco teórico

La demanda creciente de solicitudes crediticias por parte de empresas a instituciones financieras ha dado lugar a la búsqueda de métodos prácticos, automatizados y ágiles que faciliten la decisión al respecto de la concesión o no de créditos por parte de las instituciones financieras ya que actualmente, para la mayoría de los otorgamientos de dichos créditos se aplican técnicas de auditoría manual o estadísticas (Shi, Tse, Luo, D'Addona, & Pau, 2022).

El *credit scoring* es una metodología que es empleada desde hace más de 30 años por los organismos financieros, para predecir si un determinado cliente será o no capaz de hacer frente a una obligación financiera (Kritzinger & W. van Vuuren, 2018). Para esto se tienen en cuenta distintos factores: historial crediticio, historial de pagos, ingresos, deudas, etc. En particular, en las empresas, se hace una revisión de sus estados financieros y se seleccionan una serie de ratios a tener en cuenta (por ejemplo: Garantía). Dichos ratios se basan en la comparación de dos magnitudes significativas (por ejemplo: Garantía = Activo/Pasivo), lo que permite llevar a cabo mediciones financieras importantes (Ibarra Mares, 2006). En la ilustración 1 se exponen algunos ejemplos de ratios financieros básicos.

RATIO	FÓRMULA	ÓPTIMO
LIQUIDEZ	$\frac{\text{activo corriente}}{\text{pasivo corriente}}$	1,5 - 2
TESORERÍA	$\frac{\text{realizable} + \text{disponible}}{\text{pasivo corriente}}$	0,8 – 1,2
DISPONIBILIDAD	$\frac{\text{disponible}}{\text{pasivo corriente}}$	0,2 – 0,3
GARANTÍA	$\frac{\text{activo}}{\text{pasivo}}$	1,5 - 2
ENDEUDAMIENTO	$\frac{\text{pasivo}}{\text{pat neto} + \text{pasivo}}$	0,4 – 0,6
CALIDAD DE LA DEUDA	$\frac{\text{pasivo corriente}}{\text{pasivo}}$	0,2 – 0,5

Ilustración 1. Ejemplos de ratios financieros. **Fuente:** (Martínez Argudo, 2023)

Teniendo en cuenta los resultados de los ratios, a estos se les otorga un determinado valor, dando lugar a una puntuación final que engloba las contribuciones de todos los aspectos considerados. Cabe resaltar que tanto los ratios y los valores asignados están definidos por cada entidad financiera, en base a sus políticas de aprobación o denegación de créditos, límites de crédito, tasas de interés, etc. Por lo tanto, no existe un estándar definido. En este

sentido, si bien los algoritmos de aprendizaje deben ser calibrados para cada entidad financiera en base a sus parámetros, el proceso de aprendizaje y aplicación es, en principio, común y exportable a cualquier entidad.

Una vez se obtenga la puntuación, se sabrá la calidad crediticia del cliente pudiendo decidir si se otorga o no dicha financiación. Esto abre paso al concepto de rating crediticio, el cual es una calificación que aporta información sobre el nivel de riesgo crediticio y solvencia de una empresa (Reyes, 2021). Si bien existen actualmente distintas agencias de rating que se encargan de la valoración del riesgo crediticio para grandes compañías y países, cada entidad financiera puede diseñar su propio rating en base a los resultados obtenidos en el credit scoring (antes descrito) y clasificar a sus clientes en base a este.

Por lo tanto, dada la relevancia de los conceptos de credit scoring y rating crediticio, y su rol estratégico esencial para los organismos financieros, se han implementado progresivamente soluciones basadas en métodos estadísticos y de machine learning para su uso en este ámbito (Kozodoi, Jacob, & Lessmann, 2021). La diferencia principal entre estos algoritmos y las herramientas econométricas es que los primeros priorizan la búsqueda del poder predictivo, mientras que los últimos se fundamentan en modelos estructurales que exploran las relaciones causales entre las distintas variables (Alonso & Carbó, 2020).

La base de la econometría se fundamenta en el modelo de regresión y cómo este es capaz de explicar si dos o más variables son o no linealmente independientes (Buján Pérez, 2018). Del mismo modo, la econometría, se especializa principalmente en el análisis de series temporales, por tal razón dado el tipo del problema a tratar y las relaciones de los datos, se vuelve más conveniente la aplicación de modelos de machine learning, ya que son más flexibles en estas áreas.

Como anteriormente se ha abordado la importancia de utilizar modelos de machine learning, a continuación, se resume en base a estudios previos aplicados a credit scoring, qué algoritmos han dado mejores resultados para este tipo de problema de clasificación.

1.1.1 Modelos más utilizados en credit scoring

En base a estudios anteriores relacionados con el uso de algoritmos en análisis de riesgo crediticio, se ha comprobado que las técnicas de machine learning superan a las estadísticas, ya que logran automatizar el proceso y dar resultados satisfactorios en cuanto a precisión y a la clasificación correcta de las escalas de rating (Ekong, Akintola, & Kuboye, 2022).

Específicamente se ha demostrado qué técnicas como Support Vector Machine (SVM), Adaptive Boosting, Gradient Boosting y redes neuronales tienen rendimientos mayores que otros algoritmos (Fenerich, y otros, 2020). No obstante, se debe tener en cuenta que estos planteamientos son puntuales con

respecto a un conjunto de datos y el tipo de solución requerida, la cual, como se ha reflejado anteriormente, depende en gran medida del ámbito de aplicación, así como de la entidad financiera y su definición de los ratios de interés.

De acuerdo con lo anterior, en el presente trabajo se probarán las distintas aproximaciones identificadas en la literatura, con el propósito de seleccionar la(s) mejores que ayuden a clasificar correctamente a las empresas en base a los ratios definidos por la entidad financiera.

1.2 Motivación del estudio

Un organismo público financiero de Cantabria (cuyo nombre no se revela por temas de confidencialidad) desea mejorar y agilizar los procesos de otorgamiento de créditos para pymes cántabras.

Una vez definidos los conceptos previos en el marco teórico, los cuales son clave para comprender la naturaleza del problema, este estudio pretende seleccionar de entre distintos modelos, el que mejor desempeño tenga para realizar una clasificación múltiple del rating crediticio de empresas pymes cántabras.

1.3 Objetivos del estudio

En base a lo descrito anteriormente y a la motivación del estudio, se señalan los siguientes objetivos para el presente trabajo.

1.3.1 Objetivo principal

- Elegir el modelo con el mejor perfomance que permita determinar el rating crediticio de las pymes cántabras.

1.3.2 Objetivos específicos

- Describir todos los pasos seguidos para la extracción, procesado y análisis de los datos.

- Aplicar distintos algoritmos de clasificación múltiple al problema planteado.

- Mejorar los modelos seleccionados mediante el balanceo de muestras, técnicas de validación cruzada y tuneado de parámetros.

- Validar si estudios previos de machine learning aplicados a rating crediticio contemplan como modelos óptimos los obtenidos en el estudio.

1.3.3 Confidencialidad

Es importante tener en cuenta que el organismo financiero público al que se entregará el estudio ha dado sus políticas de rating crediticio como guía para la realización de los modelos y las mismas no pueden ser divulgadas. Por lo tanto, no se especificará en este trabajo la siguiente información:

- Ratios o indicadores financieros fijados para el credit scoring.
- Puntuación de ratios o indicadores financieros en base a la cifra dada.
- Nombre de los distintos ratings y sus cifras basadas en las puntuaciones del credit scoring.

De este modo, para asegurar la confidencialidad, las variables asociadas a los ratios financieros tomarán valores numéricos y se omitirá su nombre. Así mismo, la variable objetivo que toma cinco tipos de clasificación se discretizará.

2 Datos y métodos

2.1 Ratios y preprocesado de datos

Los datos de las pymes cántabras objeto de estudio son extraídos de la base de datos SABI (Sistema de Análisis de Balances Ibéricos). Esta base de datos suministra información de empresas españolas que presentan sus cuentas en los registros mercantiles. Los datos que se pueden encontrar son los siguientes: situación económica y financiera, tipo de actividad que realizan, sector, localización, forma jurídica, número de empleados, etc. Para el estudio que se va a realizar, se tendrá en cuenta la información financiera que está disponible tanto en el balance general, como en el estado de pérdidas y ganancias de cada empresa.

El filtro que se va a aplicar contiene todas las empresas cántabras activas durante los últimos tres años y con una facturación no mayor a 50 millones de euros. Después de realizar dicho filtro, se descargan los datos mediante una plantilla de Excel, la cual contiene 5379 empresas con dichas características.

Se debe tener en cuenta que para las variables del modelo se asignarán algunos ratios financieros para cada empresa (seleccionados por el organismo público financiero), los cuales se calculan a través de la información financiera disponible (véase, a modo de ejemplo, la Ilustración 1). Con estos indicadores, se establece una puntuación dependiendo de un rango previamente definido por las políticas dadas por el organismo público financiero. Finalmente, la suma total de todas esas puntuaciones permite establecer el rating de cada empresa (véase la explicación detallada de dichos conceptos en: 1.1 Marco teórico), el cual será la variable objetivo (target) de los modelos considerados.

Para efectuar el procesado de datos, cálculo de ratios y generación de modelos, se utilizará Python a través de Jupyter Notebook. A continuación, se detallan los pasos seguidos para el procesamiento de los datos:

- A todos los datos financieros numéricos se les eliminó los puntos separadores de miles para poder convertirlos posteriormente a enteros. (Ejemplo: 1.111.111 a 1111111)
- Se suprimieron todos los registros faltantes para calcular los ratios y la puntuación final. A estos registros no se les pudo asignar cero ya que algunos ratios obtienen puntuación máxima si son cero. Por lo tanto, no se tuvieron en cuenta con el fin de evitar generar modelos que no reflejen la realidad de la empresa correspondiente. Con esto, se redujo el número de empresas de 5379 a 1831 con toda la información disponible.
  Es importante aclarar que las 3548 empresas no utilizadas carecían, entre otros, de datos sobre sus deudas, siendo esta métrica esencial a la hora de un estudio crediticio. No obstante, se hizo un análisis de la situación de las mismas y se encontró que más del 50% no facturan más de 200.000 euros

y que tienen como máximo un empleado, lo que demuestra que son empresas muy pequeñas y, por lo tanto, no tan relevantes para el estudio.
- Se convirtieron los datos de tipo objeto a enteros y flotantes.
- Se calcularon los ratios (variables del modelo).
- Se asignó un scoring para cada ratio según las políticas definidas por el organismo público financiero.
- Se convirtieron los nuevos scorings a tipo flotante.
- Se calculó la puntuación total para cada empresa y se asignó el rating correspondiente (dato cualitativo).
- Se realizaron distintos gráficos para analizar los ratings por sector y tamaño de la empresa (dichos gráficos son de carácter confidencial).
- Se extrajeron las variables a utilizar para los modelos y se asignaron valores numéricos al rating (variable objetivo).

Como se explicó previamente, el objetivo del estudio es predecir el rating crediticio de la pymes cántabras a partir de variables financieras específicas (ratios). Debido a que existen cinco tipos de rating distintos, se trata de un problema de clasificación múltiple. Antes de construir los modelos, se realizó un análisis exploratorio de los datos que arrojó las siguientes conclusiones:

- Las variables que más se correlacionan con la variable objetivo son: Var10 (0.55), Var8 (0.44) y Var9 (0.43), sin embargo, ninguna supone una correlación mayor al 0.6. Dichas variables corresponden con ratios que miden en mayor medida la deuda financiera de las empresas.

Variables	Correlación con respecto a la variable objetivo
Var1	0.328921
Var2	0.294361
Var3	0.382209
Var4	0.329193
Var5	0.380626
Var6	0.332913
Var7	0.273740
Var8	0.434873
Var9	0.425956
Var10	0.547754
Var11	0.138467

Tabla 1. Correlación de variables con la variable objetivo. **Fuente:** Elaboración propia.

- La matriz de correlación (Ilustración 2) indicó que las variables no presentan una correlación muy alta entre sí. Lo anterior señala que los modelos lineales podrían ajustarse bien a los datos, ya que no existe una redundancia significativa entre las variables.

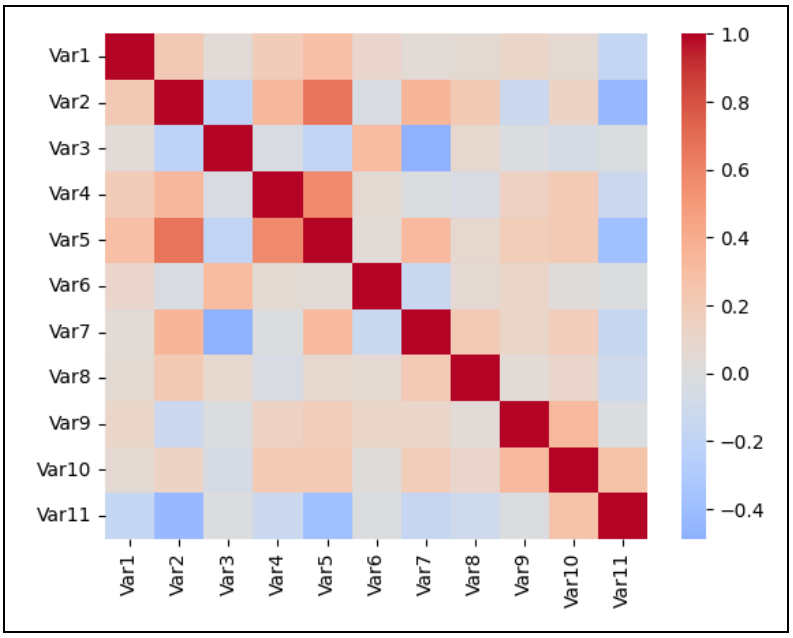

Ilustración 2. Matriz de correlación de variables. **Fuente:** Elaboración propia.

- El gráfico de cajas (Ilustración 3) mostró que las variables presentan rangos, tanto en media como en desviación estándar diferentes. Del mismo modo, todas ellas son variables no nulas, por lo tanto, se decidió estandarizarlas y redefinirlas en el rango [0,1] de manera que tengan una escala similar y evitar así que una variable tenga un impacto mayor que otra en el proceso de modelización. De esta forma, todas las variables quedan definidas en el rango [0,1] como se muestra en las Ilustraciones 4 y 5.

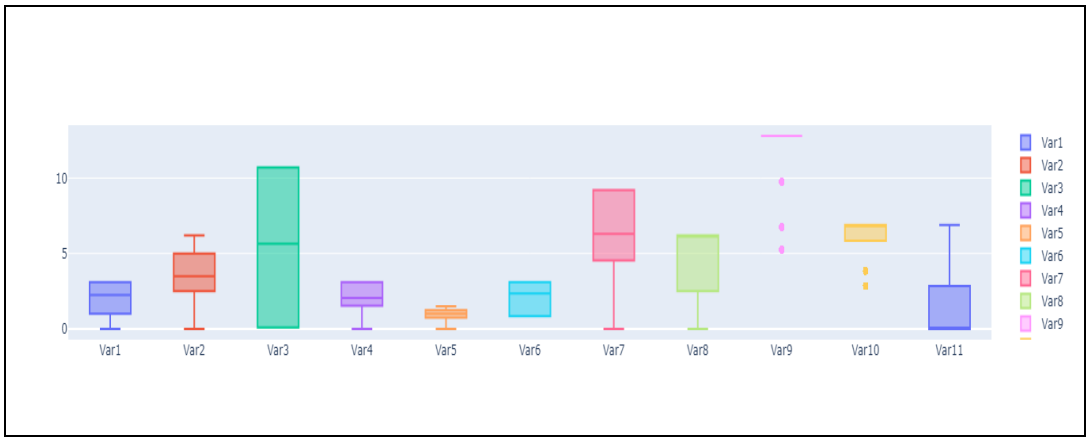

Ilustración 3. Diagrama de cajas. **Fuente:** Elaboración propia.

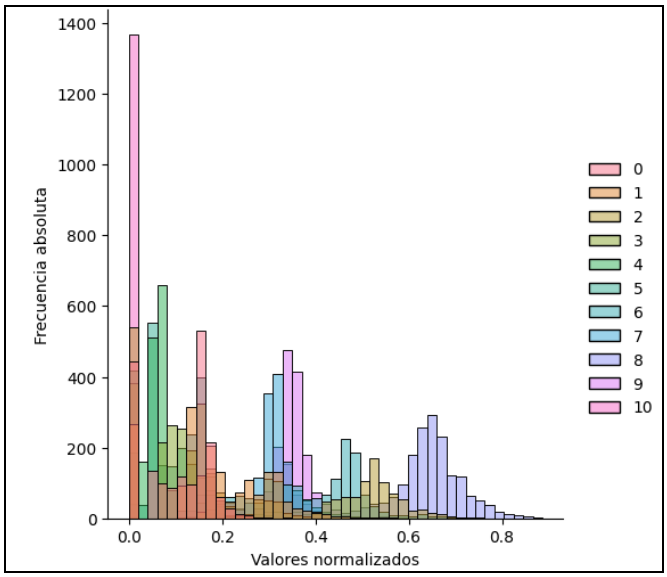

Ilustración 4. Variables normalizadas. **Fuente:** Elaboración propia.

- En la ilustración 5 se aprecia que las variables 9, 10 y 8 son las que más se asemejan a una distribución gaussiana y, casualmente, son las que más se correlacionan con la variable objetivo, tal como se resaltó anteriormente. Por otro lado, la variable 11 (Var11) es la que está más alejada de seguir una distribución normal.

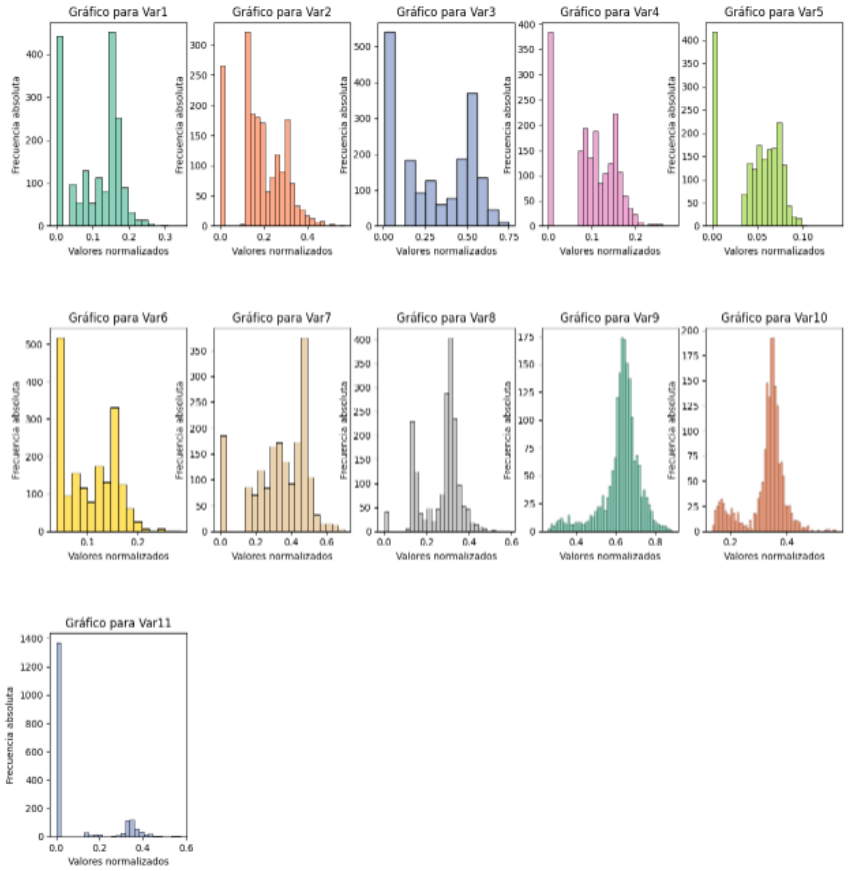

Ilustración 5. Distribución de los datos para cada variable. **Fuente:** Elaboración propia.

- Se verificó que las muestras para cada clase del rating están desbalanceadas (véase Ilustración 6).

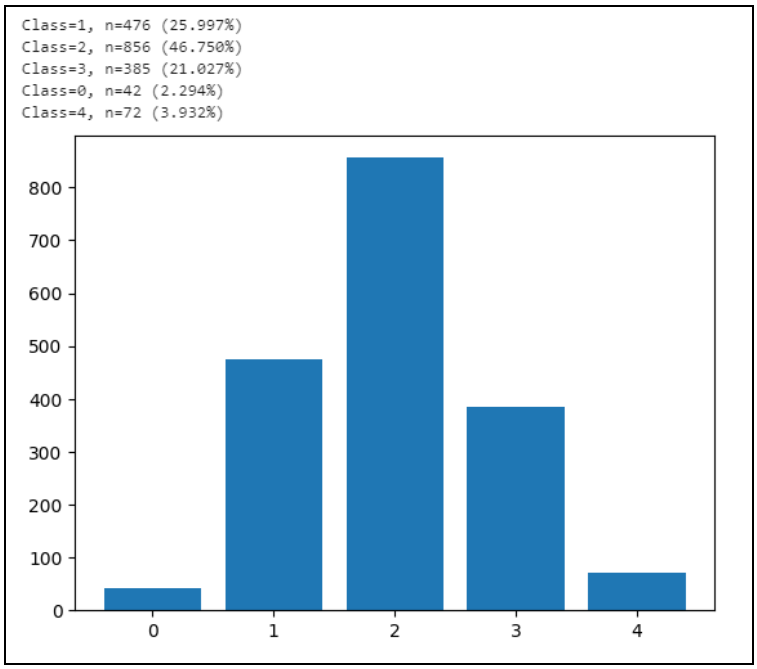

Ilustración 6. Número de muestras por clase (rating). **Fuente:** Elaboración propia.

De acuerdo con la Ilustración 6, la clase número 2 es la que tiene mayor número de muestras, con un total de 856, mientras que la clase 0, únicamente contiene 42 observaciones. Debido a esto es de vital importancia balancear el número de muestras para cada clase.

2.2 Técnicas y modelos utilizados

Una vez se ha realizado toda la etapa de procesamiento de la información, se comienza con la implementación de los modelos. Para este caso en particular, ya que se quiere conocer qué tipo de algoritmo se ajusta mejor al problema y a los datos, se han escogido once modelos distintos en base a la revisión bibliográfica realizada:

1. Discriminante Lineal (LDA).
2. Regresión Logística.
3. Árbol de decisión.
4. Random Forest.
5. Adaptive Boosting.
6. Gradient Boosting.
7. XGBoost.
8. KNN.
9. Support Vector Machine (SVM).
10. Redes Bayesianas.
11. Redes Neuronales (Neural networks).

Sin embargo, existen muchas dudas con respecto a cómo funcionan dichos modelos, ya que la mayoría tiende a utilizar técnicas matemáticas complejas. Por tal razón, se pretende explicar de manera sucinta cómo actúan los mismos, bajo la premisa de su aplicación al campo del credit scoring, basándose en modelos de clasificación múltiple. Lo anterior, es sumamente importante ya que la Comisión Europea remarca que los procesos relacionados con inteligencia artificial deben ser transparentes y tener la capacidad y el propósito de ser comunicados en abierto (European Commission, 2019).

Por otro lado, se abordan los conceptos de técnicas de validación cruzada y métricas utilizadas en los modelos de clasificación para evaluar su desempeño.

Finalmente, se plantea cómo manejar y equilibrar el número de muestras en un problema de clasificación múltiple, puesto que siempre hay alguna clase que tiene un mayor número de datos en relación con las otras. Por ende, el algoritmo tiende a aprender mejor de dicha clase, lo que genera un sesgo, haciendo que las clases con menos muestras (así estén clasificando incorrectamente) no penalicen tanto el valor de la precisión final.

2.2.1 Principales algoritmos utilizados en clasificación

A continuación, se plantean los principales algoritmos o modelos de clasificación, comenzando por los más sencillos hasta los más complejos (Shan & Nilsson, 2018) (Máster Data Science UC/UIMP):

2.2.1.1 Técnicas lineales

- **Discriminante lineal (LDA):** Esta es una de las técnicas populares de reducción de la dimensionalidad, en la cual se proyectan los datos a un espacio de menor dimensión, que permite maximizar tanto la varianza de los mismos, como de las distintas clases a separar (REDDY, y otros, 2020). De este modo, a diferencia de otras técnicas de reducción de la dimensión, LDA hace uso de la variable objetivo para obtener las nuevas direcciones que maximicen la separación de las clases. Por tanto, por un lado la reducción de la dimensión permite una disminución del coste computacional y, por otro lado, se obtiene un clasificador multivariado para las distintas clases consideradas.

  Si bien este método se aplica especialmente a datos con un alto número de variables, este estudio busca utilizar todas las técnicas de clasificación para su comparación. Se usará el algoritmo LinearDiscriminantAnalysis() disponible en la librería Sklearn de Python.

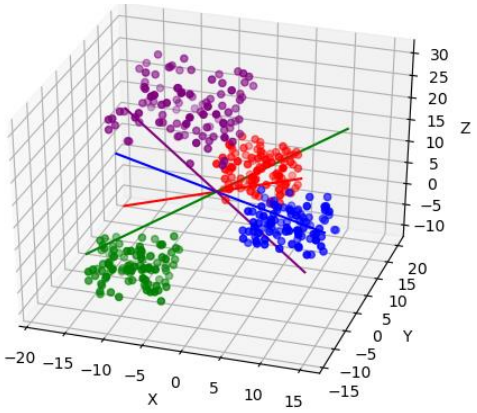

Ilustración 7. Ejemplo discriminante lineal (LDA). **Fuente:** (Wikipedia).

- **Regresión logística:** Este método modela la relación entre variables predictoras (X) y la variable de respuesta categórica (Y). A diferencia de la regresión lineal, esta busca predecir la probabilidad de pertenencia a cada clase.

  Esta técnica también se basa en la proyección lineal de los datos, buscando la maximización de la verosimilitud de los parámetros del modelo. Utiliza la función logit (Véase la Ecuación 1) para pasar de la proyección lineal a una escala de probabilidades (Levy & O'Malley, 2020). Para este estudio se aplicará el algoritmo LogisticRegression(), aclarando que el objetivo es multinomial, ya que la regresión logística por lo general se usa para problemas de clasificación binaria.

$$\textbf{(1)} \quad y = \beta_0 + \beta_1 x$$

$$\textbf{(2)} \quad P = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

Ecuación 1. Fórmula de regresión lineal (1) y regresión logística (función logit) (2). **Fuente:** (Saini, 2021).

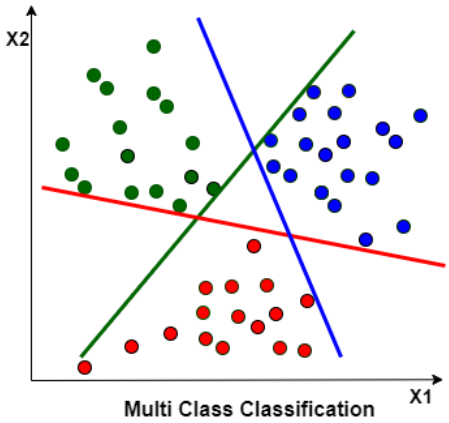

Ilustración 8. Ejemplo regresión logística. **Fuente:** (Gunjal, 2020).

2.2.1.2 Técnicas basadas en árboles

- **Árboles de clasificación:** Este algoritmo tiene una estructura similar a un árbol (véase Ilustración 9), dónde las observaciones se van dividiendo progresivamente en grupos homogéneos (a través de sus ramas) según algún criterio predefinido (por ejemplo Ecuación 2), basándose en una serie de reglas con respecto a los atributos o variables predictoras. De este modo, como se observa en la Ilustración 9, cada nodo representa a un atributo, cada rama representa el valor del atributo correspondiente al origen de la rama y cada hoja corresponde al valor de la clase final (Catalina Ortega, y otros, 2021).

  Las reglas de división son utilizadas para medir la pureza del nodo de un árbol durante el proceso de partición. Generalmente, en el caso de clasificación se consideran el índice Gini (basado en la probabilidad de que una instancia al azar no esté correctamente clasificada) o la entropía (*Entropy,* medida que refleja la impureza o desorden en un conjunto de datos), cuyas respectivas fórmulas se pueden apreciar en la Ecuación 2.

Gini Index $$I_G = 1 - \sum_{j=1}^{c} p_j^2$$

Entropy $$I_H = -\sum_{j=1}^{c} p_j log_2(p_j)$$

P_j: Proporción de muestras que pertenecen a una determinada clase para un nodo particular.

Ecuación 2. Fórmula del índice Gini y Entropy. **Fuente:** (Kali Galarnyk, 2019)

Este tipo de modelo es muy sencillo de entender e interpretar visualmente. Otra ventaja que posee es que no necesita de muestras grandes para obtener resultados competitivos. Del mismo modo, en ocasiones se utilizan de preproceso para otros modelos ya que permite seleccionar únicamente los atributos o predictores implicados en el árbol, eliminando el resto en modelos más complejos. Para el ajuste de los árboles, en este estudio se usará el algoritmo DecisionTreeClassifier() implementado en la librería de Python Sklearn.

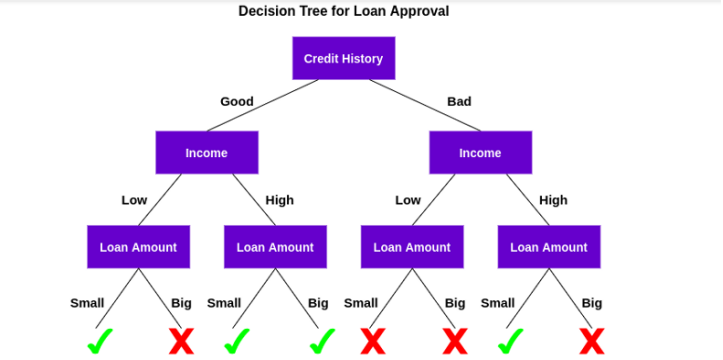

Ilustración 9. Ejemplo árbol de clasificación. **Fuente:** (Sharma, 2020)

Si bien los árboles de clasificación tienen facilidad de interpretación visual, no siempre estos modelos ofrecen la mejor solución al problema planteado ya que tienden al sobreajuste; de ahí que, una aproximación estándar, es tratar de agregar muchos árboles en un mismo modelo (métodos basados en ensembles), de modo que se mejore su capacidad predictiva y se evite incurrir en el sobreajuste. El aprendizaje basado en ensembles se fundamenta en generar muchos modelos débiles, los cuales son frecuentemente árboles de decisión (aunque no son los únicos modelos utilizados) y que se combinan con el fin de crear un modelo fuerte (Ilustración 10).

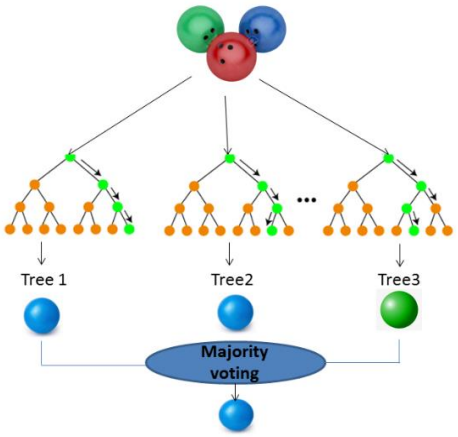

Ilustración 10. Ejemplo método ensemble. **Fuente:** (Narang, 2023)

- **Random Forest:** Es un tipo de método ensemble basado en desarrollar árboles paralelos independientes, los cuales se agregan para obtener el modelo final (Bagging: Bootstrap aggregation). Para asegurar la independencia de cada árbol con respecto al resto, random forest realiza, para cada uno de ellos, una selección aleatoria tanto de los atributos/predictores implicados en el árbol, como de la muestra considerada para el aprendizaje del árbol a través de técnicas de remuestreo Bootstrap (se eligen aleatoriamente ejemplos de la muestra original, en donde algunos de ellos se repiten y otros quedan fuera). Una

vez se obtienen los distintos árboles independientes, estos se combinan para obtener la predicción final, en el caso de clasificación a través de la clase con mayor voto y para regresión mediante el promedio.

El riesgo de sobreajuste de cada árbol se previene precisamente por la combinación de los mismos (Catalina Ortega, y otros, 2021). Si bien al ser un modelo tipo ensemble es más robusto que el árbol de decisión, su interpretación visual no es tan sencilla. Sin embargo, en este caso se puede considerar el parámetro de importancia de las variables, el cual describe el descenso del error del modelo cuando se considera cada variable. En este estudio se usará RandomForestClassifier() implementado en la librería de Python Sklearn.

- **Adaptive Boosting:** A diferencia del random forest que utiliza la técnica Bagging (Bootstrap Aggregating), este algoritmo usa el enfoque Boosting, en el cual inicialmente a todas las instancias de los datos se les asignan pesos iguales (no se generan muestras aleatorias), para luego, iterativamente crear modelos débiles que van ajustando sus pesos en base a las instancias clasificadas incorrectamente en iteraciones previas (Schmitt, 2022).

  Es decir, cada modelo se ajusta teniendo en cuenta los errores cometidos en los modelos anteriores, logrando así, mejorar su capacidad de aprendizaje. Finalmente, se combinan todos los modelos débiles ponderándolos, dándole mayor peso a aquellos con mejor desempeño. En este trabajo se usará el algoritmo AdaBoostClassifier() implementado en la librería Sklearn de Python.

- **Gradient Boosting:** Tiene gran parecido con el algoritmo Adaptive Boosting, la diferencia es que se van generando modelos que se ajustan en función de los errores previos cometidos. Es decir, en vez de enfocarse en las instancias clasificadas incorrectamente, su objetivo es minimizar la función de pérdida y, por lo tanto, actualiza los modelos mediante el descenso de gradiente y no a través de la asignación de pesos.

  La principal ventaja del Gradient Boosting frente al Adaptive Boosting, es que se pueden aplicar una gran cantidad de funciones de pérdida (Schmitt, 2022). Se usará para el presente estudio GradientBoostingClassifier() implementado en la librería Sklearn de Python.

- **XGBoost (Extreme Gradient Boosting):** Es una implementación del Gradient Boosting a través de un algoritmo que mejora a este en los siguientes términos:

- Regularización: Incluye esta técnica para prevenir el sobreajuste.
- Valores faltantes: A diferencia del Gradient Boosting, el XGBoost es capaz de aprender automáticamente cómo manejar los valores faltantes sin necesidad de especificación previa.
- Optimización y escalabilidad: Utiliza técnicas de paralelización y optimización, permitiendo el manejo de un gran volumen de datos, mientras se agiliza el proceso de entrenamiento.
- Variables categóricas: Establece directamente variables categóricas sin necesidad de convertirlas previamente a numéricas.
- Stop early: Da la posibilidad de detener el proceso de entrenamiento si no se observa mejora en las métricas de validación.

Este algoritmo: XGBClassifier() se aplicará en el estudio a través de la librería xgboost en Python.

2.2.1.3Técnicas no lineales

- **KNN (K-Nearest Neighbors):** El algoritmo KNN se basa en clasificar una nueva etiqueta de acuerdo con en el número de vecinos cercanos (K). Es decir, los puntos más próximos a esa etiqueta demarcarán a qué clase pertenecerá, esto se determina mediante la distancia de la nueva etiqueta con todas las instancias de entrenamiento. Por lo tanto, este algoritmo se fundamenta en el número de vecinos (K), el cual debe ajustarse adecuadamente y en la distancia considerada, si bien en el caso de variables continuas la distancia estándar es la euclídea.

  Debe notarse que un valor muy bajo de K puede llevar al sobreajuste de los datos, mientras que un valor más alto puede llevar a sesgos, de ahí que su elección debe ser adecuada.

  Es importante resaltar, que es una técnica de aprendizaje no paramétrica, por lo cual, si se agrega una cantidad sustancial de datos o si el espacio de trabajo tiene una dimensionalidad muy alta, el modelo no se comporta adecuadamente. Se utilizará el algoritmo KNeighborsClassifier() implementado en la librería Sklearn de Python.

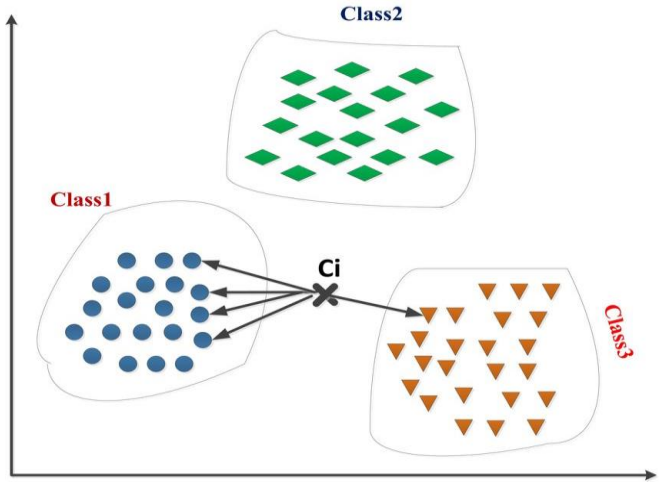

Ilustración 11. Ejemplo KNN. **Fuente:** (Khan, y otros, 2018).

- **SVM (Support Vector Machines):** Este algoritmo consiste en llevar los datos a un espacio de alta dimensionalidad, en el cual las distintas clases son linealmente separables o, al menos, exista un hiperplano que maximiza la distancia mínima entre los datos de ambas clases (Ilustración 12), dando lugar a un problema de optimización lineal con restricciones.

  Los puntos que se encuentran más cerca del hiperplano serán los vectores soporte y estos demarcan la superficie de decisión (véase Ilustración 13). Si estos puntos cambian, el hiperplano se modifica. Para estimar los vectores soporte se utiliza el llamado “kernel trick” o el truco del kernel, donde a través del cálculo del producto interno de los mismos, se evita conocer exactamente las coordenadas de los vectores en el espacio de alta dimensionalidad, logrando reducir así costes computacionales (Priya, 2023). Así mismo el truco kernel permite introducir no linealidades en los modelos.

  Para este estudio se utilizará SVC() Support Vector Classifier de la librería Sklearn de Python.

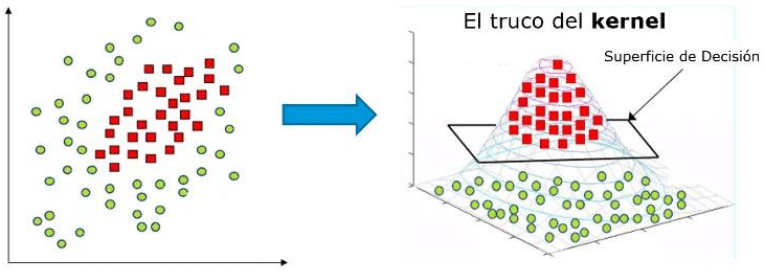

Ilustración 12. Truco kernel. **Fuente:** (Martínez Heras, 2019)

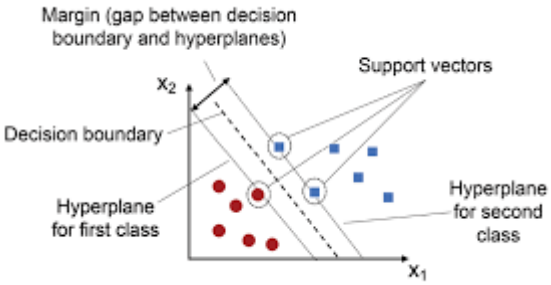

Ilustración 13. Ejemplo SVM. **Fuente:** (Kumar, 2023)

- **Redes neuronales:** Las redes neuronales son modelos basados en la estructura y el funcionamiento del cerebro, en donde un conjunto de nodos (neuronas) interconectados trabajan en paralelo para resolver un problema no lineal. Una red neuronal se organiza en capas (véase Ilustración 14):

  - ❖ Capa de entrada: Es la que recibe los datos iniciales (capa amarilla).
  - ❖ Capas ocultas: Son las que realizan operaciones complejas de extracción de información y descubrimiento de patrones en los datos (capas verdes).
  - ❖ Capa de salida: Genera la predicción final de la red (capa roja).

  La red neuronal se entrena de forma que, mediante los datos de entrada, se ejemplifican los resultados que se esperan obtener y por medio de las conexiones de las neuronas, funciones de activación y el ajuste de pesos, logran mejorar su capacidad predictiva. Notar que, tanto por el tamaño muestral como por la naturaleza del problema de este trabajo, no se han considerado arquitecturas más complejas como redes convolucionales o recurrentes.

Para este estudio debido a su naturaleza y su cantidad de datos se usará la red neuronal básica, la cual se encuentra en Python mediante la librería Tensorflow.keras.

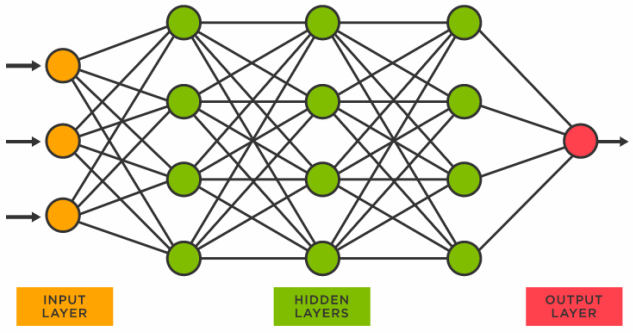

Ilustración 14. Ejemplo redes neuronales. **Fuente:** (TIBCO).

2.2.1.4 Métodos bayesianos

- **Redes Bayesianas:** Son modelos probabilísticos basados en grafos, que buscan aprender de forma eficiente la función de probabilidad (densidad) conjunta de todas las variables consideradas en el estudio. Para ello, considera grafos dirigidos y acíclicos, en donde los nodos representan las variables y las aristas las relaciones de dependencia/independencia probabilística entre ellas.

  La estructura de la red refleja las dependencias/independencias probabilísticas entre las variables del modelo, dando lugar a una factorización de la función de probabilidad (densidad) conjunta y, por tanto, reduciendo el número de parámetros a ajustar, permitiendo inferir la probabilidad de una variable dada la evidencia de otras. Para el problema que se está tratando se utilizará el algoritmo GaussianNB() implementado en la librería de Python Sklearn, el cual asume que las variables tienen una distribución gaussiana.

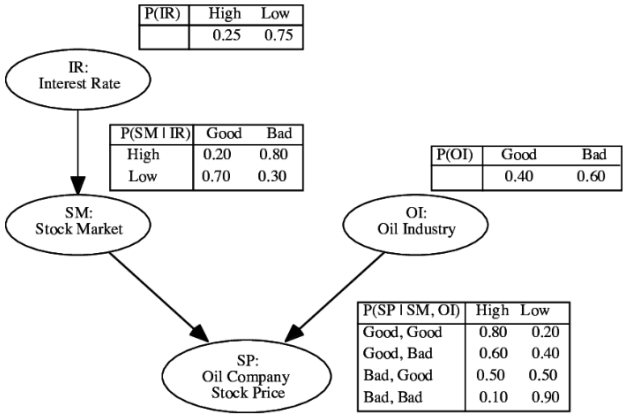

Ilustración 15. Ejemplo redes bayesianas. **Fuente:** (Shenoy & Shenoy, 2003)

2.2.2 Validación cruzada y métricas utilizadas en los modelos de clasificación

La validación cruzada tiene un uso fundamental en machine learning, ya que permite evaluar el sobreajuste de los modelos y, por tanto, obtener modelos con mejor capacidad de generalización. Para ello, la muestra se divide en el conjunto de entrenamiento y en el de validación, ambos independientes, siendo esta división la que define la validación cruzada considerada con sus ventajas e inconvenientes.

Las técnicas de validación cruzada más utilizadas son: Hold-out y K-fold. La primera divide el conjunto de datos en dos partes: entrenamiento y test. No obstante, para obtener resultados más confiables, este proceso se debe repetir varias veces y generar nuevas particiones en los conjuntos de entrenamiento y test. Sin embargo, con esta aproximación los resultados de una realización a

otra pueden variar significativamente dando lugar a una gran incertidumbre sobre el error a considerar finalmente.

Para evitar lo anterior, se utiliza la segunda técnica: K-fold, ya que se particionan los datos en K subconjuntos disjuntos (usualmente K = 10). De este modo, para cada subconjunto se entrena el modelo con el resto de los subconjuntos, usando el subconjunto como conjunto de validación y repitiendo el proceso para todos los subconjuntos. En este caso, se obtiene un valor predicho para toda la muestra a través de modelos entrenados con muestras independientes, de modo que al promediar los errores obtenemos un único valor, el cual aproxima mejor el error del modelo y presenta menos variabilidad que el Hold-out. Por lo tanto, esta técnica presenta mayor robustez que la primera.

Además de las técnicas de validación cruzada, es importante definir métricas que evalúen el desempeño de los modelos, en el caso de clasificación se utilizan principalmente:

- **Accuracy o precisión:** Es la medida empleada para determinar el rendimiento de un modelo, basada en el número de predicciones correctas sobre el número total de predicciones.
- **Matriz de confusión:** También es una métrica utilizada para evaluar el rendimiento de un modelo, no obstante, permite comparar de forma visual las predicciones realizadas por el algoritmo con los valores reales de las clases. Se muestran cuatro resultados (Visa, Ralescu, Ramsay, & Esther, 2011):
  - ❖ Verdaderos Positivos (True Positives): Número de casos en donde se predijo correctamente la clase positiva.
  - ❖ Verdaderos Negativos (True Negatives): Número de casos en donde se predijo correctamente la clase negativa.
  - ❖ Falsos Positivos (False Positives): Número de casos en donde se predijo incorrectamente una clase positiva, siendo negativa.
  - ❖ Falsos Negativos (False Negatives): Número de casos en donde se predijo incorrectamente una clase negativa, siendo positiva.

  También se debe tener en cuenta estos conceptos:

  - ❖ Precisión: Número de casos en donde se predijo correctamente la clase positiva frente a todos los casos clasificados como positivos.
  - ❖ Sensibilidad (Recall): Número de casos en donde se predijo correctamente la clase positiva frente a todos los casos que realmente son positivos.

  En la ilustración 16 se muestra una matriz de confusión:

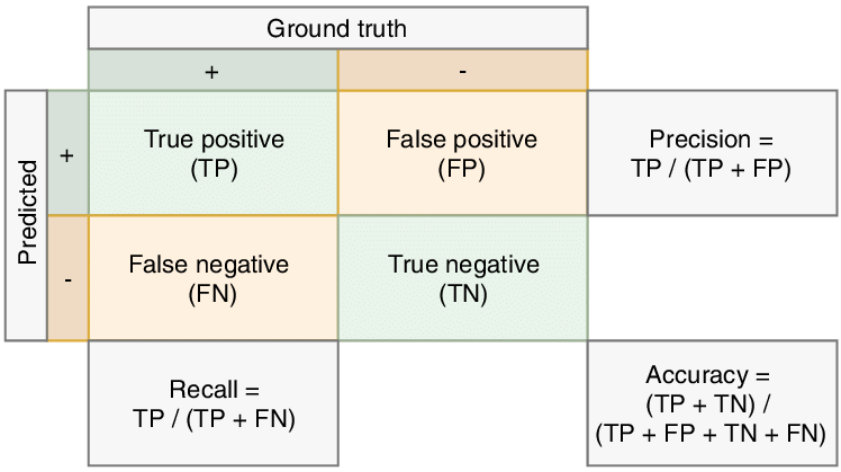

Ilustración 16. Matriz de confusión y métricas de evaluación. **Fuente:** (Jacobsen, 2019)

- **Curva ROC o AUC (Área bajo la curva):** Es una representación gráfica que muestra la Sensibilidad o Recall (explicada en la matriz de confusión) y la tasa de Falsos positivos, en la medida que varía el umbral de decisión del modelo. Un valor de ROC cercano a 1 indica un buen rendimiento del modelo, mientras que un valor cercano a 0.5 indica un rendimiento aleatorio. En la ilustración 17 se muestra un ejemplo:

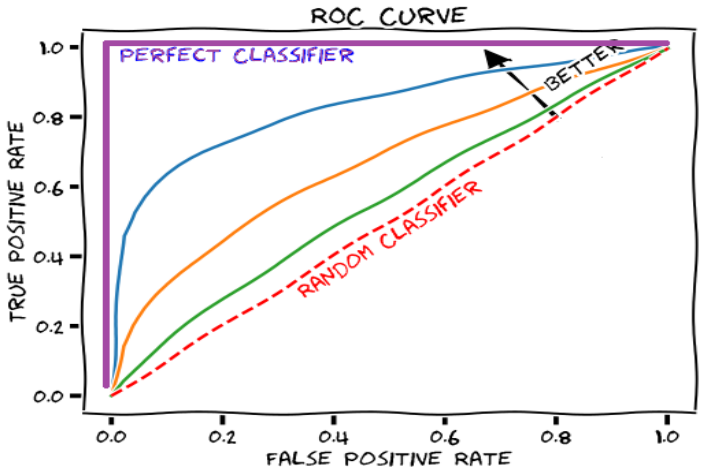

Ilustración 17. Ejemplo de Curva ROC. **Fuente:** (Draelos, 2019)

2.2.3Técnicas para balancear las muestras en problemas de clasificación en credit scoring

El problema de desbalanceo de muestras cobra cada vez más relevancia en las evaluaciones crediticias, ya que, en el mundo económico real, las empresas con mayor riesgo crediticio componen un porcentaje mucho menor que las compañías con menor riesgo (Sun, Lang, Fujita, & Li, 2017).

En la literatura se han propuesto distintos algoritmos para balancear las muestras (Zheng, y otros, 2022):

- **Sobremuestreo (Oversampling):** Se aumenta el número de muestras de las clases minoritarias y se nivela con la clase con mayor número de datos.
- **Submuestreo (Undersampling):** Se disminuye el número de muestras de las clases mayoritarias y se nivela con la clase con menor número de datos.
- **Método hibrido:** Combinación de sobremuestreo y submuestreo.

En Python los algoritmos más utilizados para sobremuestreo y submuestreo son SMOTE (Synthetic Minority Oversampling Technique) y Undersampling respectivamente. La técnica SMOTE divide las muestras de las clases minoritarias en tres categorías: segura, peligrosa y ruido (las categorías se basan en el radio de los vecinos cercanos de las clases mayoritarias) en donde la categoría peligrosa es la seleccionada para generar nuevas muestras (Bao & Yang, 2023). Por otro lado, Undersampling se basa en reducir las muestras de las clases mayoritarias ya sea aleatoriamente o teniendo en cuenta los vecinos más cercanos de las clases minoritarias (Okazaki, Okazaki, Asamoto, & Chun, 2020). Si bien estos métodos tienen sus ventajas y desventajas, dependerá del tipo del problema que se trate y qué tan bien se ajusten estas técnicas a los datos.

Con respecto a los problemas relacionados con riesgo crediticio, se ha comprobado en la bibliografía que mediante la técnica SMOTE los resultados de los modelos mejoran sustancialmente, también junto con validación cruzada K-fold (ALAM, y otros, 2020), la selección de métodos basados en ensembles (Sun, Lang, Fujita, & Li, 2017) o KNN (Melo Junior, Nardini, Renso, Trani, & Macedo, 2020).

3 Resultados

3.1 Selección de técnicas de balanceo de muestras y validación cruzada

Como primera aproximación se utilizó la técnica de validación cruzada Hold-out y se realizaron pruebas tanto para oversampling como undersampling, aquí se observa la precisión obtenida para cada modelo:

Modelo	Accuracy sin Oversampling (%)	Accuracy con Oversampling (%)	Resultado
LDA	68	66	Disminuye
Logistic Regression	45	66	Aumenta
Decision Tree	83	91	Aumenta
Random Forest	89	96	Aumenta
Adaptive Boosting	57	56	Disminuye
Gradient Boosting	91	95	Aumenta
XGBoost	91	96	Aumenta
KNN	75	91	Aumenta
SVM	71	84	Aumenta
Naive Bayes	64	66	Aumenta

Tabla 2. Comparación modelos con y sin oversampling. **Fuente:** Elaboración propia.

Modelo	Accuracy sin Undersampling (%)	Accuracy con Undersamplig (%)	Resultado
LDA	68	64	Disminuye
Logistic Regression	45	46	Aumenta
Decision Tree	83	70	Disminuye
Random Forest	89	70	Disminuye
Adaptive Boosting	57	50	Disminuye
Gradient Boosting	91	73	Disminuye
XGBoost	91	73	Disminuye
KNN	75	49	Disminuye
SVM	71	60	Disminuye
Naive Bayes	64	55	Disminuye

Tabla 3. Comparación modelos con y sin undersampling. **Fuente:** Elaboración propia.

Tal como se demuestra en las Tablas 2 y 3, la técnica de oversampling mejora notablemente la precisión de los modelos, exceptuando a dos (por pocos puntos porcentuales), mientras que el undersampling empeora, en mayor o menor grado, todos los modelos menos uno. Es más, las excepciones

señaladas presentan cambios poco significativos y, por lo tanto, se decide aplicar el método de oversampling para balancear de muestras.

Posteriormente, luego de haber seleccionado oversampling, se procede a comparar los métodos de validación cruzada Hold-out y K-fold:

Modelo	Accuracy Oversampling y Hold-out (%)	Accuracy Oversampling y K-fold (%)	Resultado
LDA	66	66	No hay cambio
Logistic Regression	66	62	Disminuye
Decision Tree	91	92	Aumenta
Random Forest	96	96	No hay cambio
Adaptive Boosting	56	55	Disminuye
Gradient Boosting	95	96	Aumenta
XGBoost	96	97	Aumenta
KNN	91	92	Aumenta
SVM	84	86	Aumenta
Naive Bayes	66	68	Aumenta

Tabla 4 Comparación modelos Hold-out y K-fold. **Fuente:** Elaboración propia.

Si bien la diferencia no es considerable, la mayoría de los modelos mejoran su desempeño con K-fold y al ser una técnica más estable y robusta (véase apartado 2.2.2), será la seleccionada. Finalmente, luego de haber elegido las técnicas oversampling y K-fold, se procede al tuneado de parámetros que se irán detallando en el siguiente apartado.

3.2 Tuneado de parámetros para cada modelo

En la librería Sklearn de Python se encuentra la función GridSearchCV, que permite hacer una búsqueda exhaustiva de todos los parámetros que se desean tunear en los modelos, los cuales se van combinando durante la fase de entrenamiento mediante validación cruzada. Se recuerda que la validación cruzada elegida corresponde a K-fold, con un K = 10 (valor estándar).

La selección adecuada de los parámetros puede contribuir a mejorar el desempeño de los modelos y su capacidad de generalización. A continuación, se describe para cada modelo los parámetros seleccionados a tunear:

- **Discriminante lineal (LDA):** Para el algoritmo LDA se tunearon los siguientes parámetros:

  - **Solver:** Corresponde al método utilizado para hallar los coeficientes del discriminante lineal. Para este estudio, teniendo en cuenta el número de muestras y características, se probaron: “lsqr” (Método de mínimos cuadrados) y "eigen" (Método que calcula los autovalores y autovectores de la matriz de dispersión dentro de las clases).

- **Shrinkage:** Determina si se aplica un término de regularización para mejorar la estimación de las matrices de dispersión. Se probaron los valores: 0.01, 0.05, 0.1, 0.5, 0.9, 1, “None” (que significa que no se debe aplicar regularización) y “auto” (Utiliza la estimación de regularización empírica mediante el método de Ledoit-Wolf).

Como resultado se obtuvo: solver: “lsqr” y shrinkage: None. El accuracy alcanzado fue del 66%, lo que implica que el tuneado de parámetros no mejoró el desempeño del modelo. A continuación, se muestra la matriz de confusión y la curva ROC:

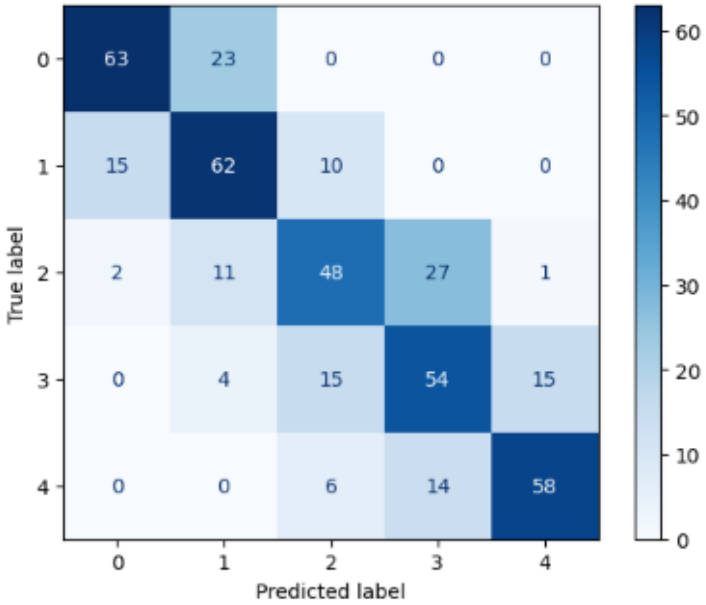

Ilustración 18. Matriz de confusión modelo LDA. **Fuente:** Elaboración propia.

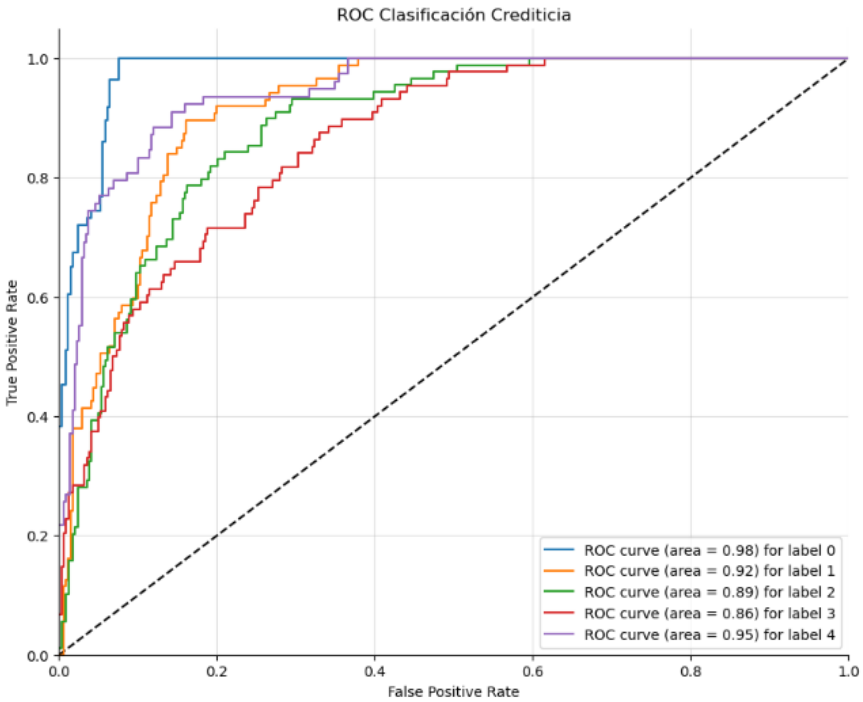

Ilustración 19. Curva ROC modelo LDA. **Fuente:** Elaboración propia.

Los resultados anteriores, se pueden atribuir a que esta técnica está diseñada para reducir la dimensionalidad (alto número de variables) y a que asume que la relación de las variables y las clases son lineales, factores que no se ajustan a los datos de este estudio.

- **Regresión logística:** A continuación, se muestran los parámetros aplicados al algoritmo de Regresión Logística:

  - **Penalty:** Tipo de regularización a aplicar: “L1”, “L2”, “elasticnet” (combinación de la regularización L1 y L2) o “None” (No se aplica regularización).

  - **C:** Determina la cantidad de regularización aplicada. Valores de C más pequeños generan mayor penalización y viceversa. Para este estudio se tomaron los siguientes valores para tunear: 0.01, 0.1, 1, 10, 100, 1000.

  - **Solver:** Este parámetro es el mismo explicado en LDA. Se tunearon dos tipos de algoritmos de optimización, los cuales son los únicos que se pueden aplicar para un problema de clasificación múltiple: "lbfgs"(Usa algoritmo de optimización Broyden-Fletcher-Goldfarb-Shanno (BFGS)) y "newton-cg" (Utiliza el método de Newton-Conjugate Gradient para optimizar los coeficientes del modelo).

  - **Multi_class:** Mediante este parámetro se aclara que se va a realizar un problema de clasificación múltiple: “multinomial”.

  - **Max_iter:** Representa el número de iteraciones máximas para alcanzar la convergencia del algoritmo, se tunearon hasta 1000 iteraciones.

Al tunear estos parámetros se obtuvieron los siguientes resultados: penalty: “None”, C: 0.01 y solver: “lbfgs”. El accuracy mejoró a través del tuneado alcanzando un 74% de precisión con respecto a un 62% previo. A continuación, se muestra la matriz de confusión y la curva ROC:

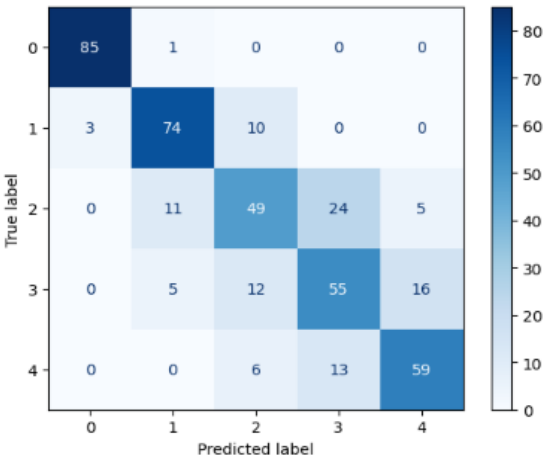

Ilustración 20. Matriz de confusión modelo Logistic Regression. **Fuente:** Elaboración propia.

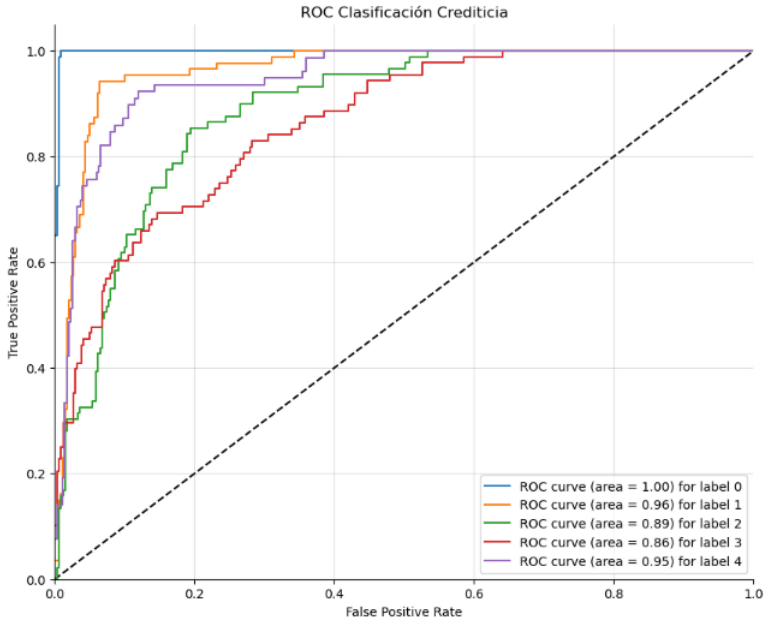

Ilustración 21. Curva ROC modelo Logistic Regression. **Fuente:** Elaboración propia.

Los resultados anteriores se pueden atribuir, al igual que en el algoritmo LDA, a que se asume que la relación de las variables y las clases son lineales, demostrando que no es así para este estudio.

- **Árboles de clasificación:** Los parámetros tuneados son los siguientes:

  - **Max_depth:** Especifica la profundidad máxima del árbol, es decir, controla la cantidad de niveles que el árbol puede tener. Se tuneó un barrido de 1 hasta 20 niveles.

  - **Min_samples_leaf:** Determina el número mínimo de muestras requeridas para formar un nodo hoja. Se tunearon de 1 hasta 10 muestras.

  - **Criterion:** Este parámetro especifica la función utilizada para evaluar la calidad de una división. Se tuneó “Gini” y “entropy” (Véase estos conceptos en el apartado 2.2.1.2).

  - **Min_samples_split:** Representa el número mínimo de muestras requeridas para dividir un nodo interno. Se tunearon desde 2 hasta 10 muestras.

  - **Max_features:** Este parámetro controla el número máximo de características que se consideran para cada división. Están: "sqrt" (raíz cuadrada del número total de características), "log2" (logaritmo en base 2 del número total de características) y “None” (todas las características).

Los resultados obtenidos fueron: “criterion”: entropy, “max_depth”: 9, “max_features”: None, “min_samples_leaf”: 1, “min_samples_split”: 2. Al tunear los parámetros, el modelo no mejoró sustancialmente, sin embargo, tiene un accuracy del 92%, un resultado muy superior con respecto a las técnicas lineales. A continuación, se muestra la matriz de confusión y la curva ROC:

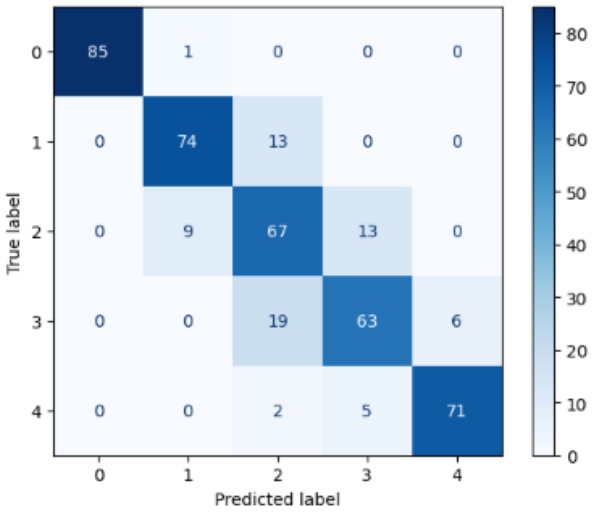

Ilustración 22. Matriz de confusión modelo Decision Tree. **Fuente:** Elaboración propia.

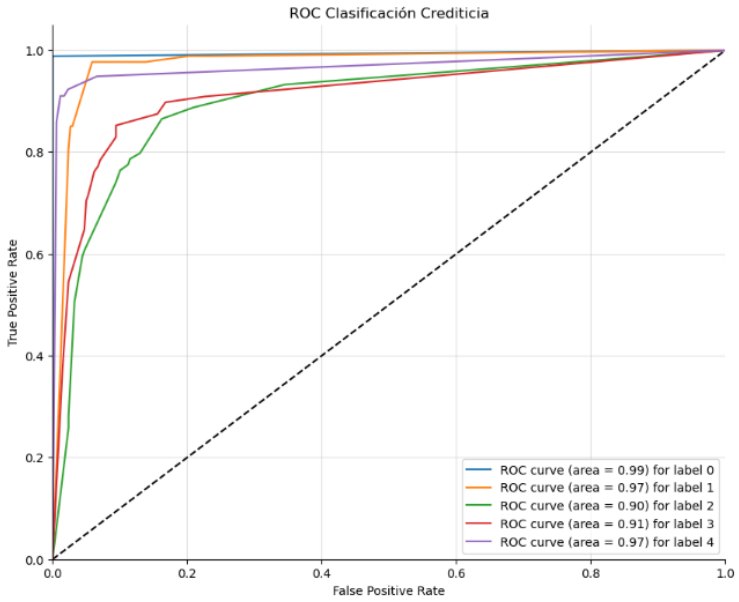

Ilustración 23. Curva ROC modelo Decision Tree. **Fuente:** Elaboración propia.

Las variables más importantes del algoritmo corresponden a Var9, Var8 y Var2, dos de estas (Var9 y Var8) coinciden con las variables que más se correlacionan con la variable objetivo (Véase apartado 2.1), lo que indica que estas tienen mayor poder predictivo sobre las clases.

- **Random Forest:** Para el algoritmo Random Forest se tunearon los mismos parámetros del árbol de clasificación más un ítem nuevo:

- **N_estimators:** Este parámetro representa el número de árboles que se construyen en el algoritmo. Se tunearon 100, 150, 200, 250, 300, 350, 400, 450 y 500 árboles.

Los resultados obtenidos fueron: “criterion”: entropy, “max_depth”: 12, “max_features”: “log2”, 'min_samples_leaf': 1, 'min_samples_split': 2, “n_estimators”: 150. Al tunear los parámetros, el modelo no mejoró de una forma importante, sin embargo, tiene un accuracy del 96%, un resultado superior al lado de las técnicas lineales y del árbol de decisión. A continuación, se muestra la matriz de confusión y la curva ROC:

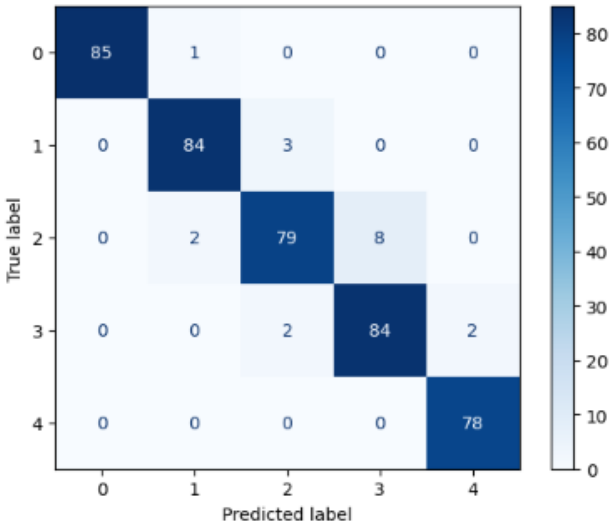

Ilustración 24. Matriz de confusión modelo Random Forest. **Fuente:** Elaboración propia.

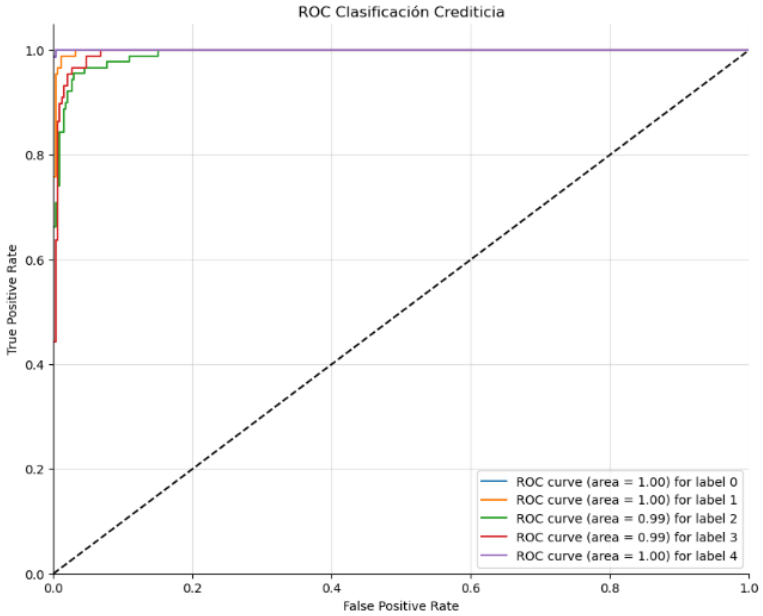

Ilustración 25. Curva ROC modelo Random Forest. **Fuente:** Elaboración propia.

Al igual que en el árbol de decisión, las variables más importantes del algoritmo corresponden a Var9, Var8 y Var2, ratificando que las mismas tienen mayor poder predictivo sobre la variable objetivo.

- **Adaptive Boosting:** Este algoritmo al igual que Random Forest tiene parámetros similares, no obstante en n_estimators se tunearon hasta 50 árboles. El resto de los parámetros son los siguientes:

  - **Algorithm:** Se selecciona el algoritmo utilizado para el cálculo de los pesos de las muestras en cada iteración. Están: “SAMME” (los pesos son iguales para todas las muestras) y “SAMME.R” (aplica el uso de probabilidades de clasificación para calcular los pesos).

  - **Learning_rate:** Controla la contribución de cada árbol en la combinación final del modelo. Se tunearon estas tasas de aprendizaje: 0.01, 0.05, 0.1, 0.5, 1.

Los resultados obtenidos fueron: “n_estimators”: 41, “algorithm”: “SAMME”, “max_depth”: 8 y “learning_rate”: 1. Al tunear los parámetros, el modelo mejoró pasando de un 55% a un 96% de precisión, obteniendo un resultado similar al algoritmo Random Forest. A continuación, se muestra la matriz de confusión y la curva ROC:

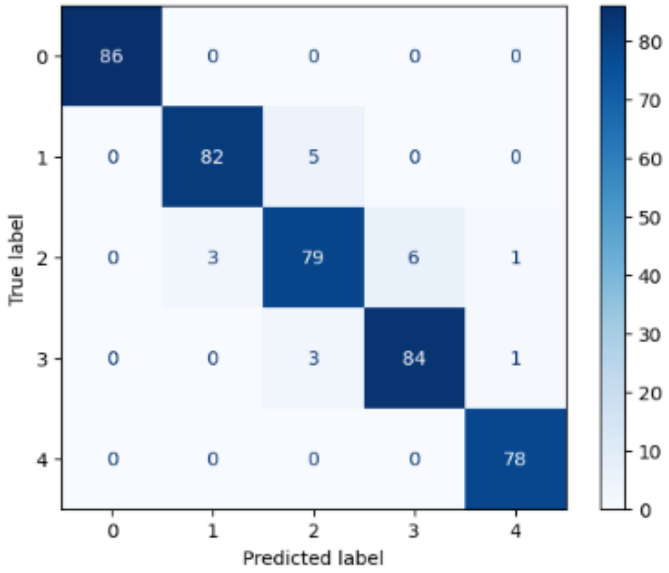

Ilustración 26. Matriz de confusión modelo Adaptive Boosting. **Fuente:** Elaboración propia.

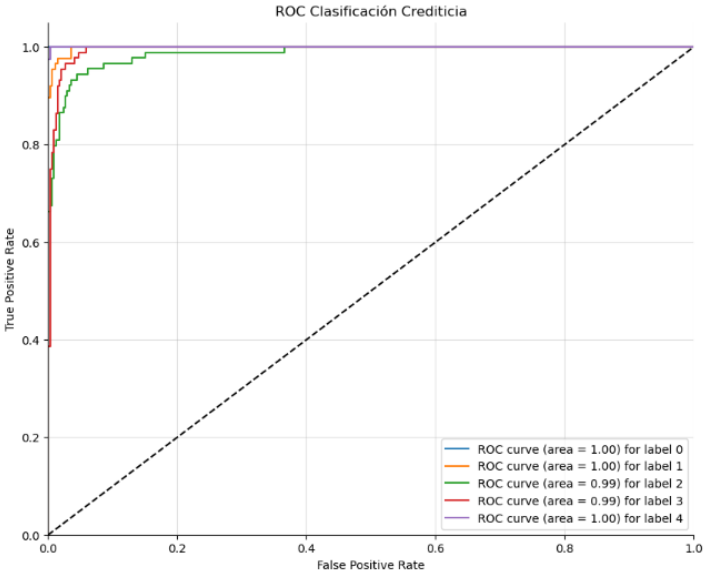

Ilustración 27.Curva ROC modelo Adaptive Boosting. **Fuente:** Elaboración propia.

- **Gradient Boosting:** En este algoritmo se tunearon parámetros iguales que en Random Forest y Adaptive Boosting, tales como: min_samples_leaf, min_samples_split, learning_rates y n_estimators.

Los resultados obtenidos fueron: “n_estimators”: 67, “learning_rate”: 0.5, 'min_samples_leaf': 1 y 'min_samples_split': 2. Al tunear los parámetros, el modelo no mejoró de manera importante, sin embargo, tiene un accuracy del 96%, un resultado similar al de Random Forest y Adaptive Boosting. A continuación, se muestra la matriz de confusión y la curva ROC:

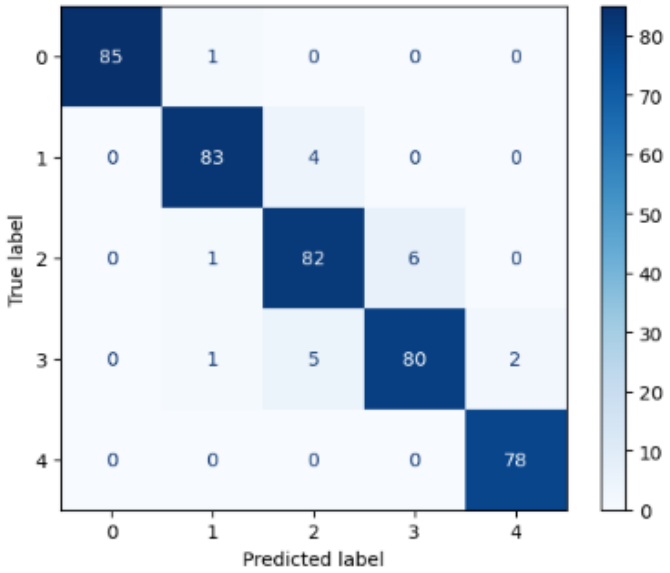

Ilustración 28. Matriz de confusión modelo Gradient Boosting. **Fuente:** Elaboración propia.

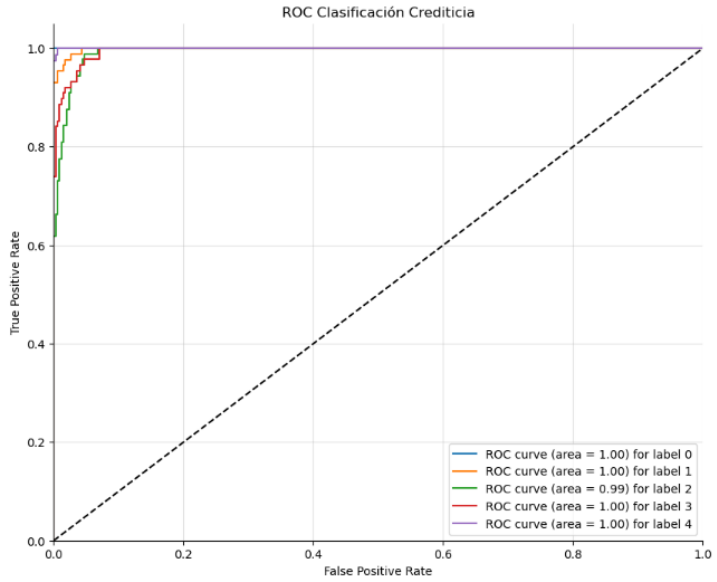

Ilustración 29.Curva ROC modelo Gradient Boosting. **Fuente:** Elaboración propia.

- **XGBoost (Extreme Gradient Boosting):** En este algoritmo se tunearon parámetros iguales que en el Gradient Boosting, tales como: min_samples_leaf, learning_rates y n_estimators. También se le aclara al modelo que se está tratando con un problema de clasificación múltiple y que se tienen 5 clases: objective = 'multi:softmax', num_class = 5.

Los resultados obtenidos fueron: “n_estimators”: 91, “learning_rate”: 0.5 y 'min_samples_leaf': 1. Al tunear los parámetros, el modelo no mejoró sustancialmente, sin embargo, tiene un accuracy del 97%, un resultado superior con respecto a los modelos tipo ensemble. A continuación, se muestra la matriz de confusión y la curva ROC:

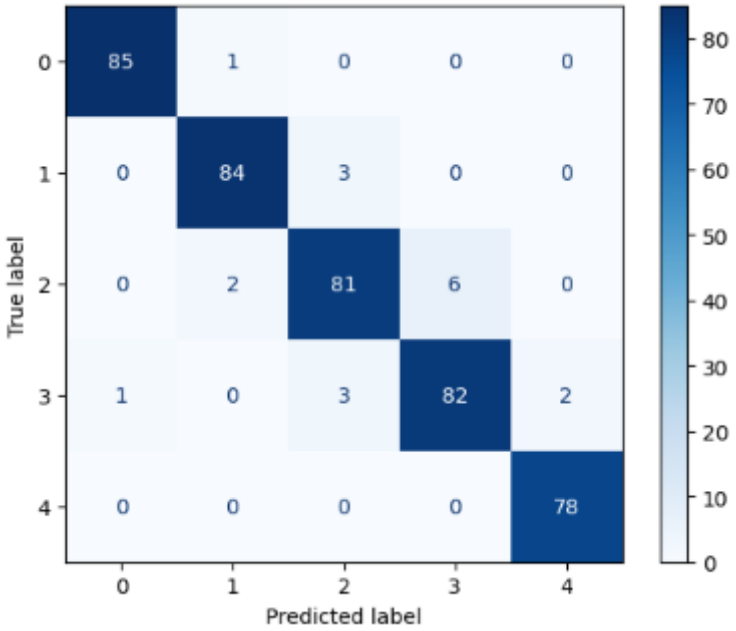

Ilustración 30.Matriz de confusión modelo XGBoost. **Fuente:** Elaboración propia.

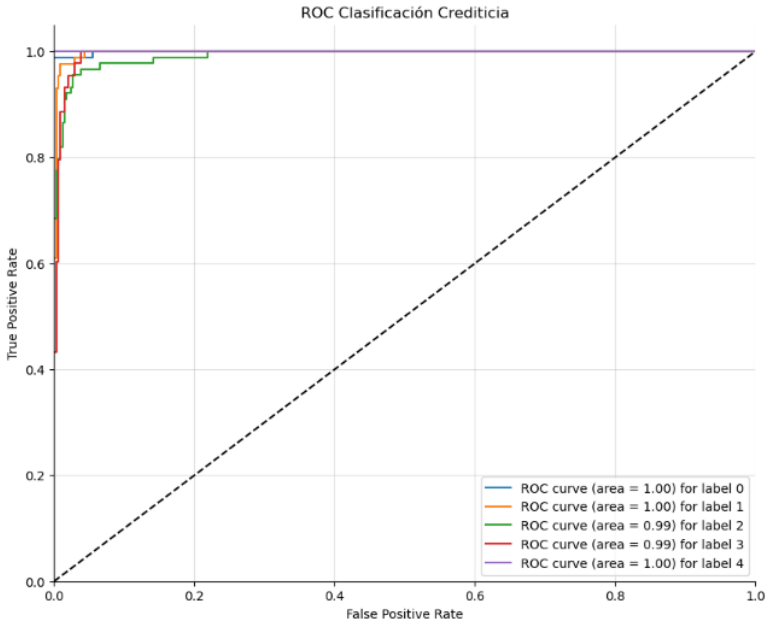

Ilustración 31.Curva ROC modelo XGBoost. **Fuente:** Elaboración propia.

- **KNN (K-Nearest Neighbors):** Para este algoritmo se tunearon los siguientes parámetros:

  - **N_neighbors:** Indica el número de vecinos cercanos a tener en cuenta para clasificar la nueva etiqueta. Se hace un barrido desde 1 hasta 20 vecinos cercanos.

  - **Weights:** Representa como se ponderan los vecinos cercanos en la clasificación. Se encuentra: “uniform” (los vecinos tienen el mismo peso) y “distance” (los vecinos más cercanos tienen un peso mayor y contribuyen más a la clasificación).

  - **P:** Es el parámetro que selecciona el tipo de distancia a utilizar. Están: 1 (corresponde con la distancia de Manhattan) y 2 (distancia euclidiana).

Los resultados obtenidos fueron: “n_neighbors”: 1, “weights”: “uniform” y “p”: 1. Al tunear los parámetros, el modelo mejoró pasando del 92% al 96% de precisión, un resultado similar a Random Forest, Adaptive Boosting y Gradient Boosting. A continuación, se muestra la matriz de confusión, se debe tener en cuenta que al dar un resultado de un único vecino cercano, no se pueden calcular las probabilidades basadas en las distintas clases, por lo tanto no hay una curva ROC.

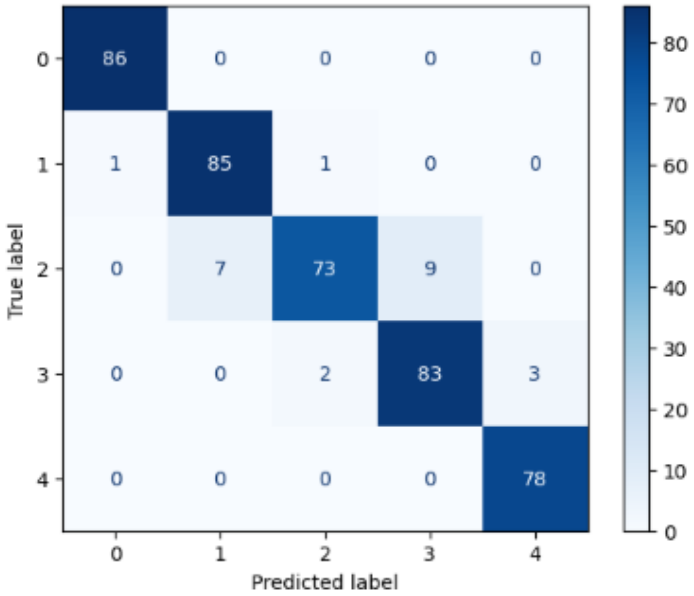

Ilustración 32. Matriz de confusión modelo KNN. **Fuente:** Elaboración propia.

- **SVM (Support Vector Machines):** A continuación, se muestran los parámetros tuneados:

  - **C:** Se probaron los siguientes valores de regularización (véase apartado de regresión logística): 0.01, 0.1, 1, 10, 100.

  - **Kernel:** Indica el tipo de kernel a utilizar: “linear” (kernel lineal), “poly”: (kernel polinómico) y “rbf” (kernel Gaussiano).

  - **Gamma:** Controla la influencia de cada muestra en la fase de entrenamiento. Un valor pequeño de gamma indica una frontera de decisión más suave y un valor grande, sugiere una frontera de decisión ajustada (sobreajuste). Se tomaron los siguientes valores: 0.01, 0.1, 1, 10.

Los resultados obtenidos fueron: “c”: 100, “kernel”: rbf y “gamma”: 10. Al tunear los parámetros, el modelo mejoró sustancialmente, pasando de 86% al 96% de accuracy, un resultado similar a los árboles de decisión, Random Forest, Gradient Boosting y KNN. A continuación, se muestra la matriz de confusión y la curva ROC:

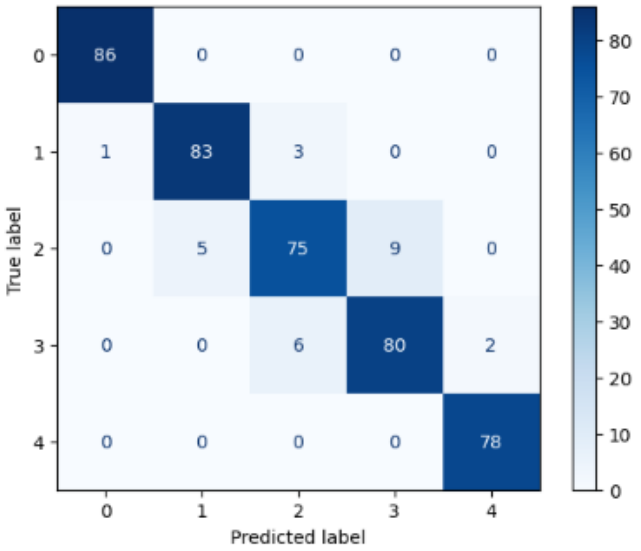

Ilustración 33. Matriz de confusión modelo SVM. **Fuente:** Elaboración propia.

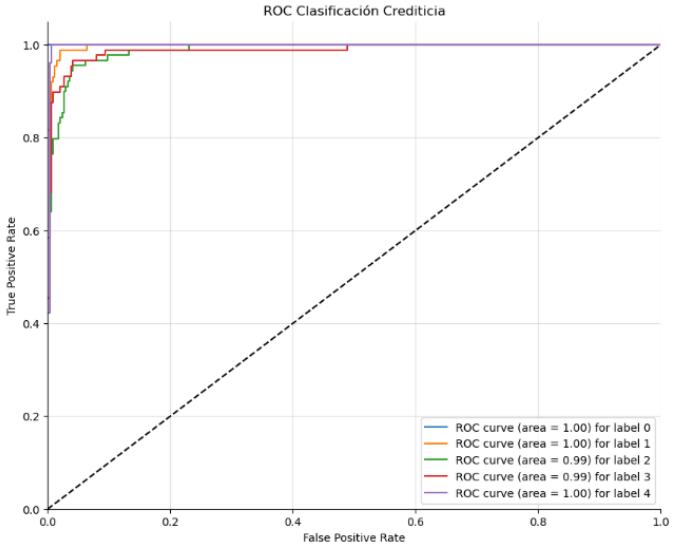

Ilustración 34. Curva ROC modelo SVM. **Fuente:** Elaboración propia.

- **Redes Bayesianas:** En el algoritmo Naive Bayes solo existen dos parámetros a tunear. Sin embargo, solo se hará para uno, ya que el otro establece las probabilidades a priori de las clases. Por lo tanto, se tunea únicamente “var_smoothing”, el cual permite controlar el suavizado aplicado a las varianzas de las características.

Los resultados obtenidos fueron: “var_smoothing”: 0.0005336699231206307. Al tunear los parámetros, el modelo no mejoró, obteniendo un accuracy del 69%. Este algoritmo asume que las variables son independientes dada la variable objetivo y que los datos siguen una distribución gaussiana, por lo tanto, este modelo no se ajusta eficientemente al problema tratado, de modo que habría que probar una red bayesiana sin restricciones y con distribuciones más ajustadas a las de las variables consideradas. A continuación, se muestra la matriz de confusión y la curva ROC:

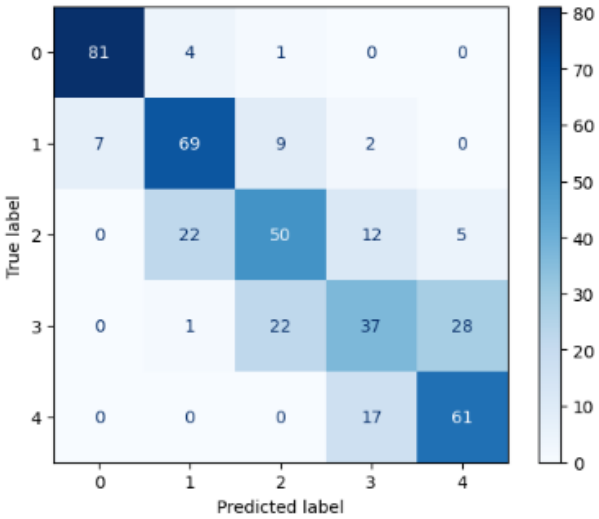

Ilustración 35. Matriz de confusión modelo Redes Bayesianas. **Fuente:** Elaboración propia.

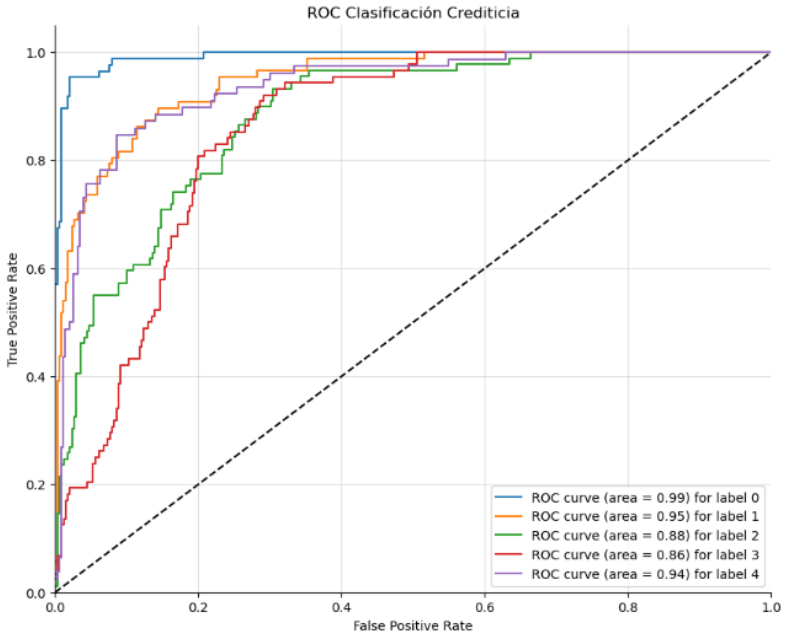

Ilustración 36. Curva ROC modelo Redes Bayesianas. **Fuente:** Elaboración propia.

- **Redes neuronales:** En las redes neuronales no se utiliza el algoritmo GridSearchCV, los parámetros se tienen que ir probando y viendo cómo se puede mejorar el desempeño del modelo. Cabe recordar que como se dijo en el apartado 2.2.1.3, se utilizan redes neuronales básicas con capas densas, lo cual se selecciona de acuerdo con el problema a tratar y su complejidad. Los parámetros seleccionados son los siguientes:

  - ❖ **Capa de entrada**: Una capa densa de 10 neuronas con una función de activación relu (esta función evita el problema del desvanecimiento del gradiente y además es más eficiente computacionalmente).

  - ❖ **Capas ocultas**: Se añadieron dos capas densas ocultas de: 30 y 20 neuronas respectivamente, todas con función de activación relu.

❖ **Capa de salida**: Finalmente, se agrega una capa densa de salida con 5 neuronas (clases a clasificar) y la función de activación softmax (permite clasificar varias categorías o clases).

❖ **Función de pérdida (loss):** El objetivo es minimizar la función de pérdida entre las probabilidades reales y las que da el modelo, a través del ajuste de pesos en la red. Se seleccionó como función de pérdida categorical_crossentropy que corresponde para problemas de clasificación múltiple.

❖ **Optimizador:** El optimizador elegido es Adam, el cual se utiliza para ajustar los pesos y sesgos del modelo durante la fase de entrenamiento, este combina los algoritmos RMSprop y el momentum para actualizar los parámetros del modelo.

❖ **Épocas:** Se seleccionaron 100 épocas.

Se obtuvo una precisión del 95% para el modelo de redes neuronales. A continuación se muestra la matriz de confusión y la curva ROC:

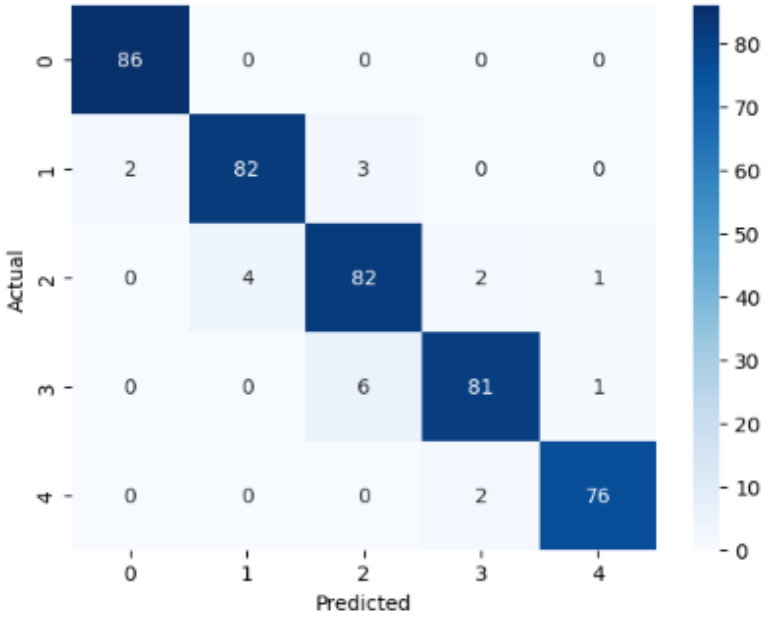

Ilustración 37. Matriz de confusión modelo Redes Neuronales. **Fuente:** Elaboración propia.

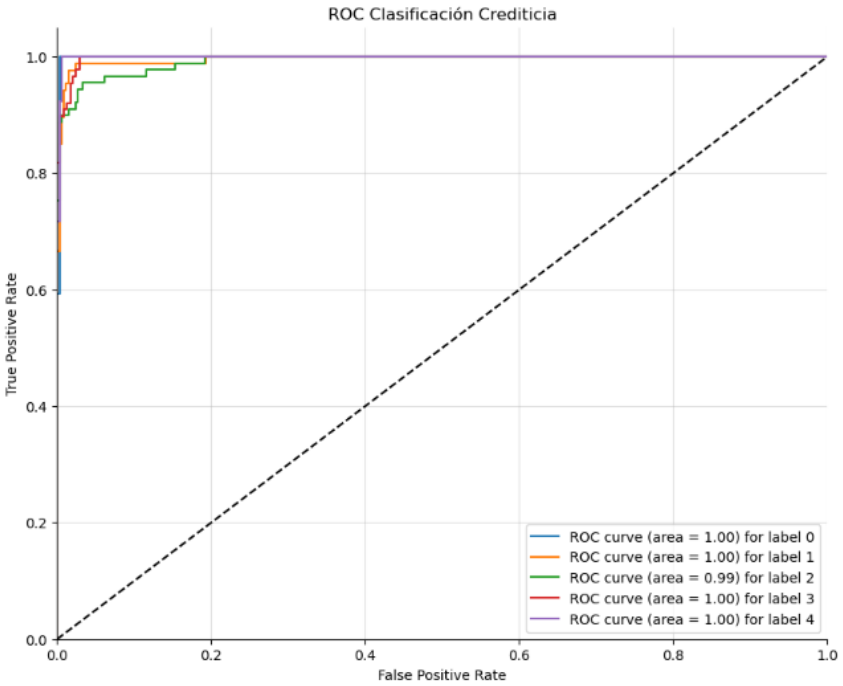

Ilustración 38. Curva ROC modelo Redes Neuronales. **Fuente:** Elaboración propia.

En la Tabla 5 se puede apreciar como mejoraron los modelos con el tuneado de parámetros:

Modelo	Accuracy Oversampling y K-fold sin tuneado (%)	Accuracy Oversampling y K-fold con tuneado (%)	Resultado
LDA	66	66	No hay cambio
Logistic Regression	62	74	Aumenta
Decision Tree	92	92	No hay cambio
Random Forest	96	96	No hay cambio
Adaptive Boosting	55	96	Aumenta
Gradient Boosting	96	96	No hay cambio
XGBoost	97	97	No hay cambio
KNN	92	96	Aumenta
SVM	86	96	Aumenta
Naive Bayes	69	69	No hay cambio

Tabla 5. Comparación de modelos con y sin tuneado de parámetros. **Fuente:** Elaboración propia.

Como se muestra en la Tabla 5, aunque algunos modelos no aportan cambios y otros aumentan ligeramente, sí se alcanza una mejor precisión en la regresión logística, adaptive boosting, KNN y SVM.

A continuación se ilustra de forma general el rendimiento de los modelos:

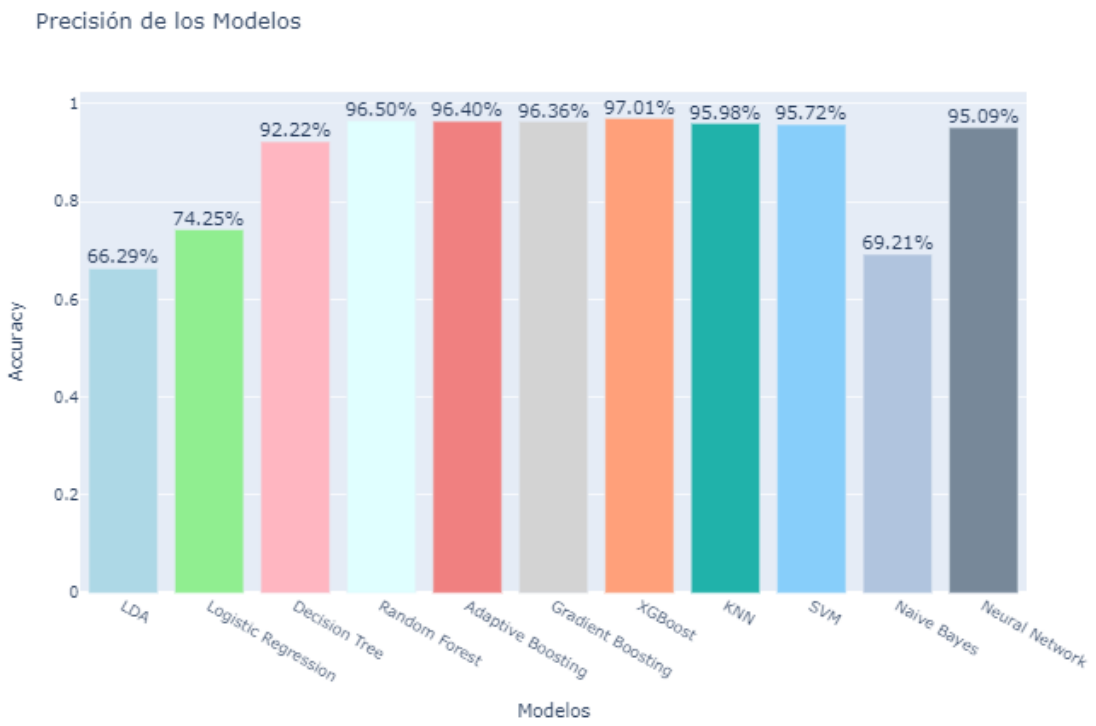

Ilustración 39. Precisión de los modelos con tuneado de parámetros, oversampling y K-fold. **Fuente:** Elaboración propia.

Un factor importante para analizar es qué tan bien los modelos se encuentran clasificando las distintas clases, esto se aprecia desde la Ilustración 40 a la 44 y en la Tabla 6:

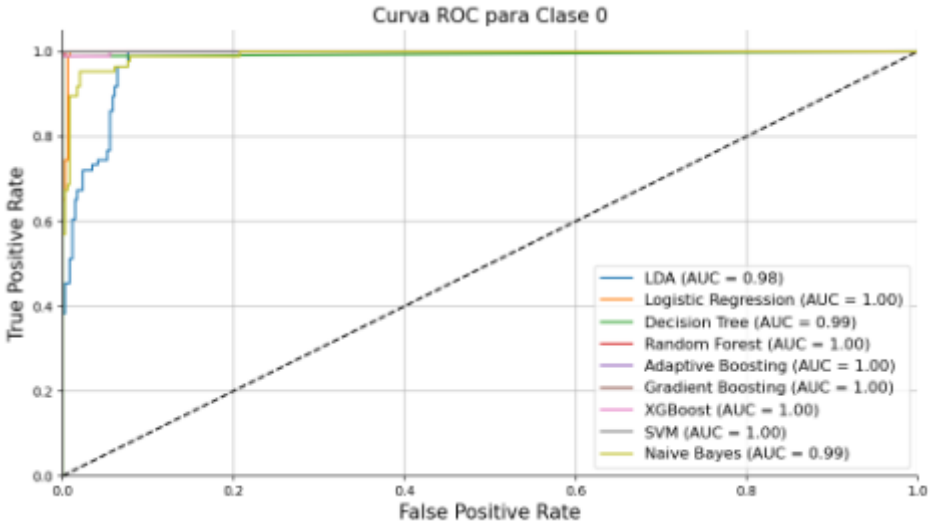

Ilustración 40. Comparación de modelos para la clase 0. **Fuente:** Elaboración propia.

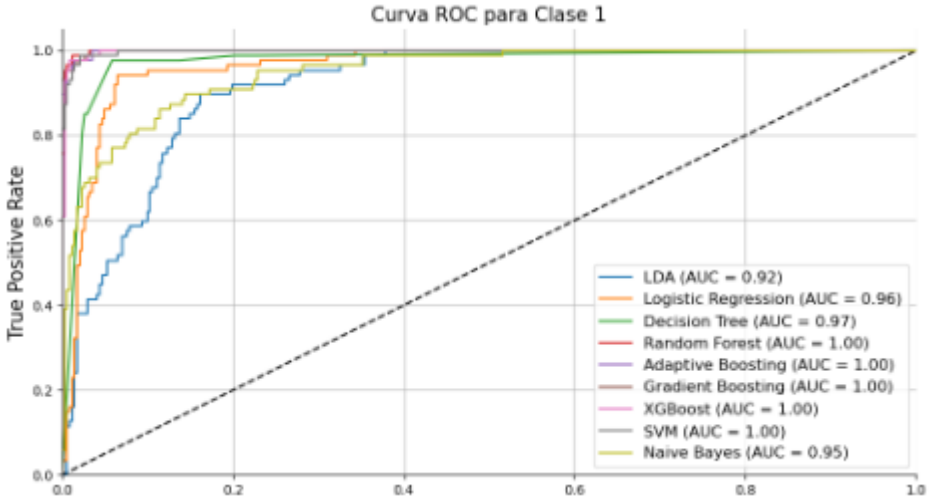

Ilustración 41. Comparación de modelos para la clase 1. **Fuente:** Elaboración propia.

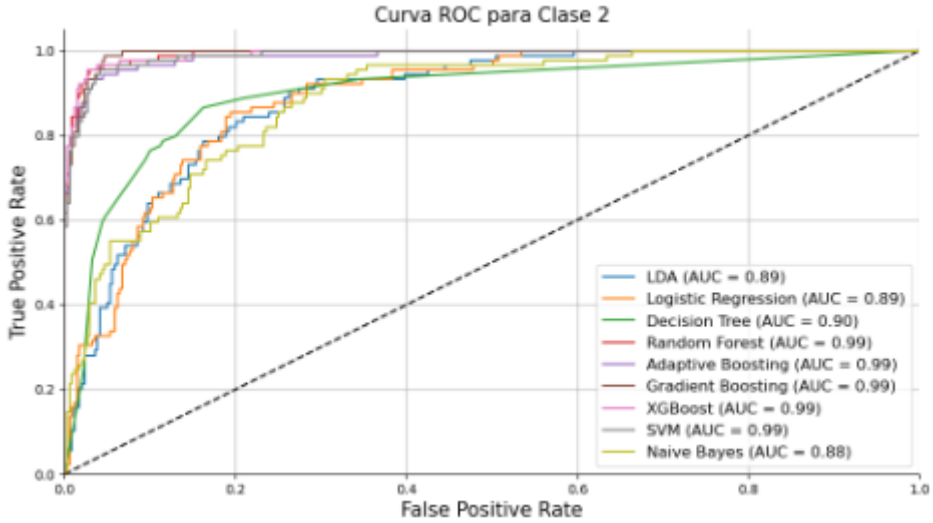

Ilustración 42. Comparación de modelos para la clase 2. **Fuente:** Elaboración propia.

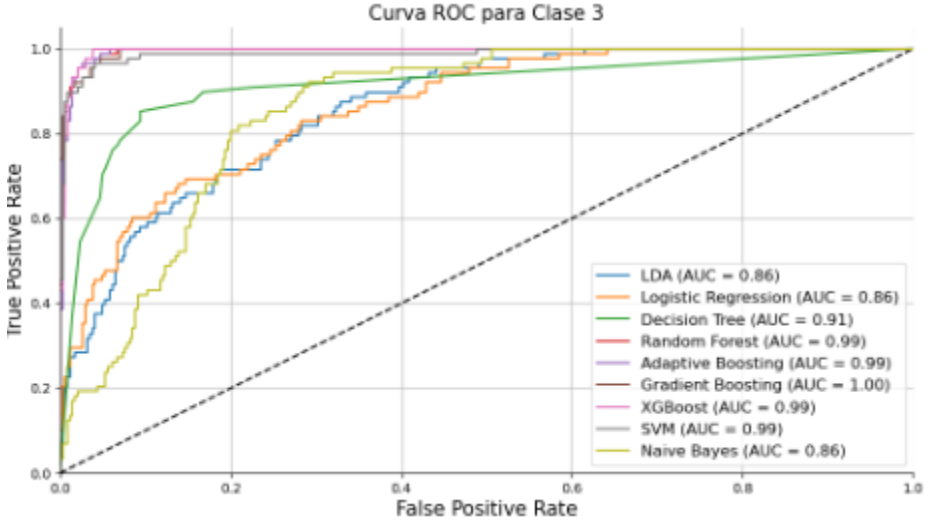

Ilustración 43. Comparación de modelos para la clase 3. **Fuente:** Elaboración propia.

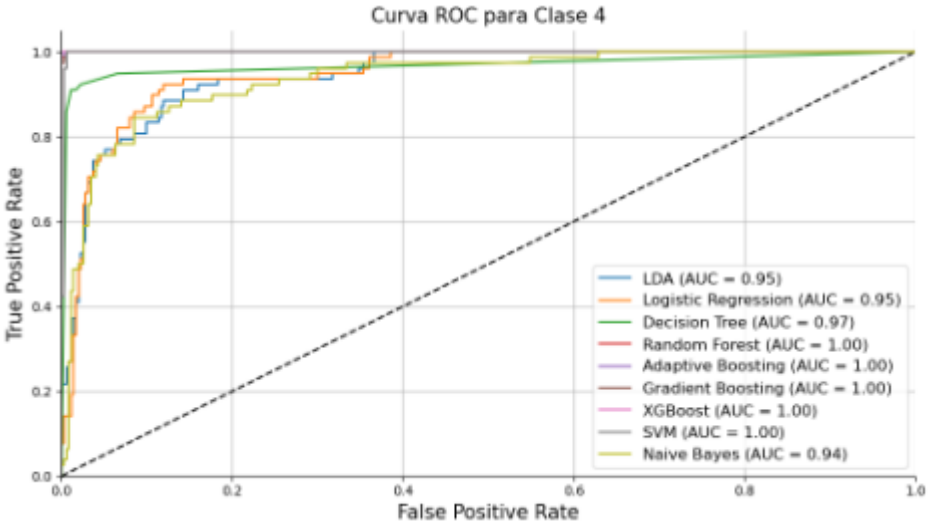

Ilustración 44. Comparación de modelos para la clase 4. **Fuente:** Elaboración propia.

Modelo	Accuracy Clase 0 (%)	Accuracy Clase 1 (%)	Accuracy Clase 2 (%)	Accuracy Clase 3 (%)	Accuracy Clase 4 (%)
LDA	98	92	89	86	95
Logistic Regression	100	96	89	86	95
Decision Tree	99	97	90	91	97
Random Forest	100	100	99	99	100
Adaptive Boosting	100	100	99	99	100
Gradient Boosting	100	100	99	100	100
XGBoost	100	100	99	99	100
SVM	100	100	99	99	100
Naive Bayes	99	95	88	86	94

Tabla 6. Comparación de modelos para las distintas clases. **Fuente:** Elaboración propia.

Como se observa en los gráficos anteriores, la clase 0 es la que mejor están prediciendo los modelos, mientras que la clase 3 es la que tiene mayor error en su clasificación. Es importante resaltar que los modelos con mayor precisión clasifican bien en general para las cinco clases, las diferencias más significativas se observan en los modelos con un bajo desempeño como LDA, Regresión logística y Redes Bayesianas.

4 Conclusiones

- Los once modelos seleccionados, en su mayoría, mejoran su desempeño aplicando la técnica de oversampling para el problema de balanceo de muestras.

- Las técnicas de validación cruzada Hold-out y K-fold dan resultados muy parecidos en la precisión de los modelos, no obstante, K-fold supera ligeramente a Hold-out en algunos modelos y al ser una técnica más robusta, fue la seleccionada.

- El tuneado de parámetros mantuvo un desempeño similar a los parámetros por defecto en los modelos, exceptuando a la regresión logística, adaptive boosting, KNN y SVM.

- Desde el inicio se comprobó cómo los algoritmos LDA, regresión logística y redes bayesianas no eran los óptimos para el problema planteado. Esto es debido a que modelos como LDA y regresión logística asumen cierta linealidad sobre los datos, condición que no se cumple para este caso. Por otro lado, las redes bayesianas asumen independencia de las variables predictoras dada la variable objetivo y una distribución normal de las mismas, afirmación que tampoco se cumple.

- Se observa cómo los métodos basados en árboles, especialmente los ensemble presentan buenos resultados en los modelos.

- Métodos como KNN y SVM si bien tenían resultados satisfactorios, al realizar el tuneado de parámetros llegaron a ubicarse dentro de los mejores.

- El modelo de redes neuronales se trabajó mediante capas densas acorde con el tipo de problema y se hicieron distintas pruebas en cuanto al tuneado de parámetros hasta alcanzar un resultado satisfactorio.

- El modelo con mejor desempeño es XGBoost el cual alcanza un 97% de precisión en los datos de test, seguido por: Random Forest, Adaptive Boosting, Gradient Boosting, KNN, SVM y redes neuronales.

- Se valida de acuerdo con los resultados obtenidos en este estudio y estudios previos, que efectivamente, para los problemas de clasificación de rating crediticio, los modelos tipo ensemble, SVM y redes neuronales funcionan correctamente y generan buenos resultados.

5 Discusión y trabajos futuros

Una de las principales cuestiones que se ha hecho evidente en el trabajo, es la poca cantidad de datos para la realización de los modelos. Esto es debido a que se está teniendo en cuenta pymes, que no siempre disponen ni comparten de información financiera. Por eso, de 5379 con información disponible, se pasó a 1831 empresas con todos los datos necesarios para calcular los ratios, es decir, las variables de los modelos.

Una desventaja de tener una muestra limitada de datos es la imposibilidad de aplicar modelos más complejos como técnicas de Deep Learning o redes bayesianas basadas en distribuciones no gaussianas.

Si se lograsen obtener más datos, los modelos tendrían una mayor capacidad de aprendizaje y se aplicarían algoritmos más complejos, por lo tanto, se haría un análisis de sensibilidad del comportamiento de los modelos actuales frente a estos nuevos.

6 Referencias

Zheng, M., Wang, F., Hu, X., Miao, Y., Cao, H., & Tang, M. (2022). A Method for Analyzing the Performance Impact of Imbalanced Binary Data on Machine Learning Models. *axioms*, 19.

ALAM, T., SHAUKAT, K., HAMEED, I., LUO, S., SARWAR, M., SHABBIR, S., . . . KHUSHI, M. (2020). An Investigation of Credit Card Default Prediction in the Imbalanced Datasets. *IEEE*, 26.

Alonso, A., & Carbó, J. (2020). *Machine learning in credit risk: Measuring the dilemma.* Banco de España Eurosistema.

Bao, Y., & Yang, S. (2023). Two Novel SMOTE Methods for Solving Imbalanced Classification Problems. *IEEE*, 11.

Buján Pérez, A. (18 de Mayo de 2018). *Enciclopedia Financiera*. Obtenido de www.enciclopediafinanciera.com/estadistica/econometria.htm

Catalina Ortega, C. A., Mariscal, M. A., Boulagouas, W., Herrera, S., Espinosa, J. M., & García-Herrero, S. (2021). Effects of Mobile Phone Use on Driving Performance: An Experimental Study ofWorkload and Traffic Violations. *International Journal of Environmental Research and Public Health*, 22.

Draelos, R. (23 de Febrero de 2019). *Glass Box*. Obtenido de Measuring Performance: AUC (AUROC): https://glassboxmedicine.com/2019/02/23/measuring-performance-auc-auroc/

Ekong, R. E., Akintola, K. G., & Kuboye, B. M. (2022). Development of Credit Scoring Model for Borrowers Using Machine Learning Techniques. *ResearchGate*, 11.

European Commission. (2019). *Ethics guidelines for trustworthy AI.*

Fenerich, A., Arns Steiner, M., Steiner Neto, P., Tochetto, E., Tsutsumi, D., Medeiros Assef, F., & Samways dos Santos, B. (2020). Use of machine learning techniques in bank credit risk analysis. *Scipedia*, 10.

Gunjal, S. (2020). *Logistic Regression from scratch with Python - Quality Tech Tutorials.* Obtenido de https://satishgunjal.com/binary_lr/

Ibarra Mares, A. (2006). Una perspectiva sobre la evolución en la utilización de las razones financieras o ratios. *Pensamiento & Gestión*, 39.

Jacobsen, R. H. (2019). Confusion matrix and evaluation metrics. *ResearchGate*, 1.

Kali Galarnyk, M. J. (2019). *Quora*. Obtenido de www.quora.com/Are-gini-index-entropy-or-classification-error-measures-causing-any-difference-on-Decision-Tree-classification

Khan, S., Ali, H., Ullah, Z., Minallah, N., Maqsood, S., & Hafeez, A. (2018). KNN and ANN-based Recognition of Handwritten Pashto Letters using Zoning Features. *ResearchGate*, 9.

Kozodoi, N., Jacob, J., & Lessmann, S. (2021). Fairness in credit scoring: Assessment, implementation and profit implications. *European Journal of Operational Research*, 12.

Kritzinger, N., & W. van Vuuren, G. (2018). An optimised credit scorecard to enhance cut-off score determination. *South African Journal of Economic and Management Sciences*, 15.

Kumar, A. (27 de Marzo de 2023). *Data Analytics*. Obtenido de www.vitalflux.com/classification-model-svm-classifier-python-example/

Levy, J. J., & O'Malley, A. J. (2020). Don't dismiss logistic regression: the case for sensible extraction of interactions in the era of machine learning. *BMC Medical Research Methodology*, 15.

Martínez Argudo, J. (20 de Febrero de 2023). *ECONOSUBLIME*. Obtenido de http://www.econosublime.com/2020/02/resumen-ratios-financieros-medidas-correctoras-liquidez-solvencia-exceso-escasez.html

Martínez Heras, J. (28 de Mayo de 2019). *IArtificial.net.* Obtenido de https://www.iartificial.net/maquinas-de-vectores-de-soporte-svm/

Máster Data Science UC/UIMP. (s.f.). *Clases de Estadística para Data Science y Minería de Datos [Diapositivas de PowerPoint]*. Obtenido de Moodle Unican.

Melo Junior, L., Nardini, F., Renso, C., Trani, R., & Macedo, J. (2020). A novel approach to define the local region of dynamic selection techniques in imbalanced credit scoring problems. *ELSEVIER*, 16.

Narang, M. (18 de Abril de 2023). *Shiksha online.* Obtenido de https://www.shiksha.com/online-courses/articles/random-forest-algorithm-python-code/

Okazaki, Y., Okazaki, S., Asamoto, S., & Chun, P.-j. (2020). Undersampling Strategy for Machine-learned Deterioration Regression. *Journal of Advanced Concrete Technology*, 14.

Priya, B. (10 de Julio de 2023). *KDnuggets*. Obtenido de A Gentle Introduction to Support Vector Machines: www.kdnuggets.com/2023/07/gentle-introduction-support-vector-machines.html

REDDY, G. T., KUMAR REDDY, M. P., LAKSHMANNA, K., KALURI, R., SINGH RAJPUT, D., SRIVASTAVA, G., & BAKER, T. (2020). Analysis of Dimensionality Reduction Techniques on Big Data. *IEEE Access*, 13.

Reyes, P. (2021). *BBVA.* Obtenido de https://www.bbva.com/es/que-es-un-rating/

Saini, A. (3 de Agosto de 2021). *Analytics Vidhya*. Obtenido de www.analyticsvidhya.com/blog/2021/08/conceptual-understanding-of-logistic-regression-for-data-science-beginners/

Schmitt, M. (2022). *Deep Learning vs. Gradient Boosting: Benchmarking state-of-the-art machine learning algorithms for credit scoring.* Oxford: Department of Computer Science.

Shan, Q., & Nilsson, M. (2018). Credit risk analysis with machine learning techniques in peer-to-peer lending market. 42.

Sharma, A. (12 de Mayo de 2020). *Analytics Vidhya.* Obtenido de https://www.analyticsvidhya.com/blog/2020/05/decision-tree-vs-random-forest-algorithm/

Shenoy, P., & Shenoy, C. (2003). Bayesian Network Models of Portfolio Risk and Return. *ReasearchGate*, 16.

Shi, S., Tse, R., Luo, W., D'Addona, S., & Pau, G. (2022). Machine learning-driven credit risk: a systemic review. *Springer*, 13.

Sun, J., Lang, J., Fujita, H., & Li, H. (2017). Imbalanced enterprise credit evaluation with DTE-SBD: Decision tree ensemble based on SMOTE and bagging with differentiated sam pling rates. *ELSEVIER*, 91.

TIBCO. (s.f.). *TIBCO.* Obtenido de https://www.tibco.com/reference-center/what-is-a-neural-network

Visa, S., Ralescu, A., Ramsay, B., & Esther, V. d. (2011). Confusion Matrix-based Feature Selection. *ResearchGate*, 9.

Wikipedia. (s.f.). *Wikipedia.* Obtenido de https://en.wikipedia.org/wiki/Linear_discriminant_analysis

Zheng, M., Wang, F., Hu, X., Miao, Y., Cao, H., & Tang, M. (2022). A Method for Analyzing the Performance Impact of Imbalanced. *axioms*, 19.