UNIVERSIDAD EUROPEA DE MADRID

ESCUELA DE ARQUITECTURA, INGENIERÍA Y DISEÑO

MASTER UNIVERSITARIO EN ANÁLISIS DE DATOS MASIVOS

(BIG DATA)

TRABAJO FIN DE MÁSTER

SELECCIÓN DE LA MEJOR ESTRATEGIA DE APRENDIZAJE AUTOMÁTICO APLICABLE A CLASIFICACIÓN BINARIA DEL COMPORTAMIENTO CREDITICIO

JUAN PABLO CEVALLOS GUERRA

Dirigido por

Dr. OSCAR GARIBO ORTS

CURSO 2024-2025

TÍTULO: SELECCIÓN DE LA MEJOR ESTRATEGIA DE APRENDIZAJE AUTOMÁTICO APLICABLE A LA CLASIFICACIÓN BINARIA DEL COMPORTAMIENTO CREDITICIO

AUTOR: JUAN CEVALLOS GUERRA

TITULACIÓN: MASTER UNIVERSITARIO EN ANÁLISIS DE DATOS MASIVOS (BIG DATA)

DIRECTOR/ES DEL PROYECTO: DR. ÒSCAR GARIBO ORTS

FECHA: OCTUBRE DE 2025

RESUMEN

En un entorno financiero altamente competitivo y tecnológicamente dinámico, la capacidad de las instituciones para gestionar eficientemente el riesgo crediticio se ha vuelto crítica. Actualmente, el sistema HcRisk utiliza un modelo de regresión logística binaria (Logit) desarrollado en R para clasificar a los clientes como buenos o malos pagadores. Aunque este modelo ha mostrado eficacia durante más de una década, su arquitectura presenta limitaciones en términos de precisión, capacidad de adaptación, mantenimiento y escalabilidad frente a los avances en ciencia de datos y aprendizaje automático.

El presente proyecto tiene como objetivo principal reemplazar el modelo Logit por un modelo de aprendizaje automático (*machine learning*) más moderno, robusto y escalable, que permita una mejor aproximación al comportamiento crediticio real de los clientes. Para ello, serán seleccionados y evaluados al menos tres modelos del estado del arte como Random Forest, XGBoost y CatBoost, implementados mediante la plataforma Python y validados con datos históricos reales.

La metodología del proyecto combina análisis exploratorio de datos, preprocesamiento, entrenamiento supervisado, validación cruzada, comparación de métricas de rendimiento (como AUC-ROC y F1-score) y técnicas de interpretabilidad como SHAP para asegurar transparencia en la toma de decisiones. Asimismo, se contemplan criterios operativos como la facilidad de mantenimiento, compatibilidad tecnológica con el sistema HcRisk y capacidad de integración futura con plataformas analíticas.

El resultado esperado es la identificación e implementación de un modelo predictivo superior en desempeño y alineado con los requerimientos actuales del sector financiero. Además, el proyecto sienta las bases para la incorporación de modelos más complejos en otros procesos de evaluación, así como para el desarrollo de tableros de visualización, ampliación de variables de

comportamiento y mecanismos de reentrenamiento automatizado, consolidando un sistema de *scoring* crediticio moderno, confiable y sostenible.

Palabras clave:

Aprendizaje automático, *Scoring* crediticio, Modelo Logit, Clasificación binaria, Interpretabilidad, *Machine Learning* en Python

ABSTRACT

In a highly competitive and technologically dynamic financial environment, the ability of institutions to efficiently manage credit risk has become critical. Currently, the HcRisk system uses a binary logistic regression (Logit) model developed in R to classify clients as good or bad payers. Although this model has demonstrated effectiveness for over a decade, its architecture presents limitations in terms of accuracy, adaptability, maintenance, and scalability in comparison against recent advances in data science and machine learning.

The primary objective of this project is to replace the Logit model with a more modern, robust, and scalable machine learning model that offers a better approximation of clients' actual credit behavior. To achieve this, at least three state-of-the-art models like Random Forest, XGBoost, and CatBoost are going to be selected and evaluated. These models are going to be implemented using the Python platform and validated with real historical data.

The project's methodology combines exploratory data analysis, preprocessing, supervised training, cross-validation, performance metric comparison (such as AUC-ROC and F1-score), and interpretability techniques such as SHAP to ensure transparency in decision-making. Operational criteria are also considered, including ease of maintenance, technological compatibility with the HcRisk system, and future integration capabilities with analytical platforms.

The expected outcome is the identification and implementation of a predictive model that outperforms the current one and aligns with the evolving requirements of the financial sector. Furthermore, the project lays the groundwork for incorporating more complex models into other evaluation processes, as well as for developing visualization dashboards, expanding behavioral variables, and enabling automated retraining mechanisms consolidating a modern, reliable, and sustainable credit scoring system.

Keywords:

Automatic learming, Credit Scoring, Logit model, Binary Classification, Interpretability, Machine Learning in Python

AGRADECIMIENTOS

Quiero agradecer a mi amada esposa, Gabriela, y a mi madre Fabiola, por todo su amor y por todo su apoyo brindado durante mi carrera y el desarrollo del este proyecto.

Espero poderles recompensar todo aquello que me han regalado.

Quiero también hacer un agradecimiento póstumo a mi padre, quien me compartió su templanza y tenacidad para enfrentar cada obstáculo en la vida, así como con cariño y paciencia enseñarme a escuchar.

Locura es hacer la misma cosa una y otra vez esperando resultados diferentes.

- Albert Einstein -

TABLA RESUMEN

	DATOS
Nombre y apellidos:	Juan Pablo Cevallos Guerra
Título del proyecto:	Selección De La Mejor Estrategia De A.A. Aplicable A La Clasificación Binaria Del Comportamiento Crediticio
Directores del proyecto:	Dr Òscar Garibó Orts
El proyecto se ha realizado en colaboración de una empresa o a petición de una empresa:	SI
El proyecto ha implementado un producto:	NO
El proyecto ha consistido en el desarrollo de una investigación o innovación:	NO
Objetivo general del proyecto:	Sustituir el modelo Logit para clasificación de clientes buenos y malos por uno del estado del arte con mejores prestaciones.

Índice

RESUMEN4

ABSTRACT6

TABLA RESUMEN10

Capítulo 1. RESUMEN DEL PROYECTO17

1.1 Contexto y justificación17

1.2 Planteamiento del problema18

1.3 Objetivos del proyecto19

1.4 Resultados obtenidos19

1.5 Estructura de la memoria19

Capítulo 2. ANTECEDENTES / ESTADO DEL ARTE20

2.1 Acuerdos Internacionales con respecto a Riesgo De Crédito20

2.2 Construcción de modelos para calificación de riesgo de crédito26

2.3 Modelo Logit en el Sistema HCRISK27

2.4 Estado del arte en Aprendizaje Automático en torno al comportamiento crediticio**¡Error! Marcador no definido.**

2.5 Contexto y justificación35

2.6 Planteamiento del problema36

Capítulo 3. OBJETIVOS40

3.1 Objetivos generales40

3.2 Objetivos específicos40

3.3 Beneficios del proyecto41

Capítulo 4. DESARROLLO DEL PROYECTO43

4.1 Planificación del proyecto43

4.2 Descripción de la solución, metodologías y herramientas empleadas44

4.2.1 Arquitectura General del Sistema44

4.2.2 Metodología de Desarrollo KDD45

4.2.3 Herramientas y Tecnologías Empleadas**¡Error! Marcador no definido.**

4.2.4 Desarrollo del Modelo Predictivo48

4.2.5 Integración *Frontend* – *Backend***¡Error! Marcador no definido.**

4.2.6 Consideraciones sobre Seguridad y Regulación83

4.2.7 Futuras Mejoras y Extensibilidad84

4.3 Recursos requeridos84

4.4 Presupuesto85

4.5 Viabilidad86

4.6 Resultados del proyecto86

5 DISCUSIÓN88

6 CONCLUSIONES89

6.1 Conclusiones del trabajo89

6.2 Conclusiones personales89

7 FUTURAS LÍNEAS DE TRABAJO91

8 REFERENCIAS93

Fuentes del Estado del Arte del Aprendizaje Automático93

Fuentes del Acuerdo de Basilea (I, II, III)95

9 ANEXOS97

Índice de Figuras

Figura 1: Esquema del modelo conceptual simplificado:..... 36

Figura 2 Pantalla HcRisk Variables rangos y recodificación 37

Figura 3 HcRisk Políticas de calificación 38

Figura 4 Tareas de planificación del proyecto Parte 1 42

Figura 5 Tareas de planificación del proyecto Parte 2 42

Figura 6 Seleccionar datos de la Base de datos 51

Figura 7 Análisis estadístico de variables numéricas 52

Figura 8 Eliminación de constantes en los datos 52

Figura 9 Ejecución de Pivot sobre datos seleccionados 53

Figura 10 Eliminación de operaciones sin set de variables completo..... 54

Figura 11 Validación de valores faltantes 54

Figura 12 Cambio de tipo de variable correspondiente a su naturaleza 55

Figura 13 Verificación de cambio de tipo de dato 55

Figura 14 Eliminación de valores constantes 56

Figura 15 Correlación de variables..... 56

Figura 16 Eliminación de variables con correlación alta 57

Figura 17 Aplicación de transformaciones..... 58

Figura 18 División del conjunto de datos: Entrenamiento y validación..... 59

Figura 19 Definición de modelos para entrenamiento 60

Figura 20 Matriz de confusión..... 62

Figura 21 Ciclo de ejecución y valoración de modelos 63

Figura 22 Explicación del resultado de ejecución de modelos 64

Figura 23 Resultado Ejecución Logistic Regression 65

Figura 24 Resultado Ejecución Elastic Net 66

Figura 25 Resultado Ejecución Random Forest 67

Figura26 Resultado Ejecución XGBoost 68

Figura 27 Resultado Ejecución CatBoost 69

Figura 28 Comparativa de los modelos probados 70

Figura 29 Comparación de métricas de los modelos 70

Figura 30 Comparación de Validación Cruzada de modelos 71

Figura 31 Curva AUC de los modelos 72

Figura 32 Ejecución Smote – Oversampling 73

Figura 33 Entrenamiento SMOTE - Logistic Regression 74

Figura 34 Entrenamiento SMOTE - Elastic Net 75

Figura 35 Entrenamiento SMOTE - Random Forest 76

Figura 36 Entrenamiento SMOTE - XGBoost 77

Figura 37 Entrenamiento SMOTE - CatBoost 78

Figura 38 Comparativa de los modelos probados con SMOTE 79

Figura 39 Explicabilidad SHAP para CatBoost 81

Figura 40 Predicción del modelo CatBoost 82

Índice de Tablas

Tabla 1.- Comparación entre versiones de Acuerdos De Basilea 22

Tabla 2.- Categorización de la calificación de crédito externo 24

Tabla 3.- Principales Tablas involucradas en el modelo Logit 35

Tabla 4.- Variables originales numéricas 47

Tabla 5.- Variables originales cualitativas 49

Índice de Ecuaciones

Ecuación 1 Coeficiente de capital24

Ecuación 2 Pérdida Esperada25

Ecuación 3 Modelo Logit27

Ecuación 4 F1 Score62

Capítulo 1. RESUMEN DEL PROYECTO

1.1 Contexto y justificación

El sistema HcRisk es una solución tecnológica diseñada con una arquitectura de dos capas (cliente-servidor), donde el *frontend* permite que se realice el proceso del tratamiento de las variables, despliegue de informes del estado de las carteras de crédito, despliegue de indicadores de riesgo crediticio e indicadores del estado del modelo, adicionalmente tiene una interfaz para poder calcular los pesos de las variables y también una interfaz para permitir el ingreso de variables de un potencial cliente para determinar si es bueno o malo. La base de datos aloja tanto variables como datos de los clientes y sus operaciones tanto históricas como vigentes, además de conFiguraciones e indicadores de riesgo. Esta arquitectura, aunque funcional y eficaz durante años, está limitada en su capacidad de adaptarse a nuevos paradigmas analíticos.

Actualmente, el modelo Logit implementado en R ha sido envuelto como un servicio *backend* para responder a solicitudes de clasificación crediticia. Adicionalmente, se desarrolló una interfaz para realizar prospecciones mediante el ingreso manual de variables, lo que permite realizar simulaciones del comportamiento del *scoring* ante diferentes perfiles de cliente. Sin embargo, esta solución presenta limitaciones técnicas: el lenguaje base (R) no es óptimo para entornos de producción modernos, la integración con nuevas herramientas es compleja, y la escalabilidad para atender un mayor número de peticiones o reentrenar el modelo de forma continua es restringida. Estos desafíos justifican la transición hacia modelos implementados en Python, que ofrezcan mayor robustez, portabilidad y un ecosistema más amplio de herramientas de *machine learning* e integración web.

Dado el auge y avance de los modelos de aprendizaje automático en varios ámbitos de las actividades cotidianas de las personas y más aún en sectores como el financiero, surge la necesidad de actualizar el modelo de clasificación del sistema HcRisk (logit) que ya lleva funcionando por más de 10 años con mucha eficacia.

Este modelo se ha implementado en más de 5 Instituciones Financieras dentro de Ecuador por la Empresa HC Consultores con mucho éxito sin embargo la incorporación de un mejor modelo

fortalecería el proceso de toma de decisiones en gestión de riesgo crediticio mejorando la rentabilidad y sostenibilidad financiera de las instituciones.

1.2 Planteamiento del problema

El modelo actual de regresión logística, aunque útil durante más de una década, se basa en supuestos lineales que pueden limitar su capacidad predictiva cuando se enfrenta a conjuntos de datos de mayor complejidad, no linealidad o alta dimensionalidad. Además, su implementación en R dificulta la interoperabilidad con otros componentes del sistema, especialmente considerando que el ecosistema tecnológico moderno —incluyendo microservicios, APIs REST y *pipelines* de reentrenamiento automático— se encuentra mejor soportado en plataformas como Python y sus *frameworks* asociados.

En la práctica, las instituciones financieras que utilizan el sistema HcRisk enfrentan una creciente necesidad de respuestas más precisas, rápidas y adaptativas ante cambios en el comportamiento crediticio de los clientes. Además, la presión regulatoria y la necesidad de interpretabilidad hacen imperativo el uso de modelos que no solo sean más certeros, sino también comprensibles por los analistas y auditables por los entes reguladores. Por tanto, es indispensable adoptar un enfoque que contemple modelos más sofisticados de aprendizaje automático, combinados con herramientas de interpretación (SHAP), integrados de manera eficiente con la plataforma tecnológica de vanguardia.

En el entorno en que se vive actualmente las posibilidades del aprendizaje automático son bastante amplias y dado el éxito que el mismo tiene a nivel empresarial se tiene la necesidad de reemplazar el modelo Logit para clasificación de clientes buenos y malos por un modelo de aprendizaje automático acorde a las circunstancias actuales que aporta la tecnología. A pesar que el modelo Logit tiene ya algunos años de implementación, la puesta en marcha en una empresa nueva, que tiene sus propias variables y características de crédito propias, resulta en tiempos bastante altos que rondan los 2 a 3 meses.

1.3 Objetivos del proyecto

El objetivo principal de este proyecto es desarrollar e implementar un modelo de aprendizaje automático que reemplace al modelo de regresión logística binaria actualmente utilizado en el sistema HcRisk para la clasificación de clientes en buenos o malos pagadores, con el fin de mejorar la precisión predictiva, facilitar el mantenimiento y asegurar una mayor escalabilidad del sistema. Para ello, se evaluarán al menos tres modelos de *machine learning* desarrollados en Python, comparando su desempeño con el modelo Logit vigente, con miras a su integración en el sistema actual.

Los objetivos específicos se definen en el capítulo 3 de este documento.

1.4 Resultados obtenidos

Dado que los modelos empleados en el análisis comparativo tuvieron unos resultados bastante buenos, es interesante su aplicabilidad en el entorno empresarial. En cuanto a su implementación como reemplazo del modelo actual en R, es 100% recomendable ya que sus etapas de planificación, ejecución y evaluación se han realizado de forma ágil. Facilita mucho la implementación mucho la ventaja de que Python permite generar APIs que ejecutan lo requerido para las tareas de Preprocesamiento, Procesamiento, Verificación de datos y Puesta en marcha del mejor modelo.

1.5 Estructura de la memoria

El presente proyecto empieza con la presentación del proyecto en sí y el objetivo general, pasando luego por los antecedentes del proyecto y el estado del arte. Para luego profundizar en los objetivos y entrar en materia con el desarrollo del proyecto. Termina con una breve discusión, conclusiones y futuras líneas de trabajo.

Capítulo 2. ANTECEDENTES / ESTADO DEL ARTE

2.1 Acuerdos Internacionales con respecto a Riesgo De Crédito

2.1.1 Acuerdos de Basilea

El Acuerdo de Basilea es un conjunto de recomendaciones desarrolladas por el Comité de Supervisión Bancaria de Basilea (BCBS), que opera bajo el auspicio del Banco de Pagos Internacionales (BIS). Su objetivo es fortalecer la solidez y estabilidad del sistema financiero internacional, estableciendo normas para:

- **Medición y gestión de riesgos bancarios.**
- **AdEcuación del capital bancario.**

Estos acuerdos no son legalmente vinculantes, pero se adoptan como estándares regulatorios en muchos países, entre ellos Ecuador.

El primer acuerdo tuvo lugar en el año 1988, se le conoce actualmente como Basilea I, y tiene como objetivo introducir un requisito mínimo de capital para las instituciones financieras, basado principalmente en el riesgo de crédito.

Elementos clave (Basilea I):

- Introducción del concepto de capital mínimo del 8% sobre activos ponderados por riesgo (RWA).
- Los activos se clasificaban en grupos con ponderaciones fijas de riesgo: 0%, 20%, 50%, 100%.
- Simple y fácil de implementar, pero limitado en precisión y cobertura de riesgos (no incluía riesgo de mercado ni operacional).

Posteriormente, en el año 2004, se lleva a efecto Basilea II, que tiene como objetivo mejorar la sensibilidad al riesgo, incentivar una gestión más sofisticada**,** y ampliar el marco a riesgos de mercado y operacionales.

Basilea II se fundamenta en 3 pilares:

PILAR I (Basilea II):

Requerimientos mínimos de capital. - Este es el núcleo de Basilea II. Requiere que las Instituciones Financieras mantengan capital mínimo por tres tipos de riesgos:

- **Riesgo de crédito**
- **Riesgo de mercado**
- **Riesgo operacional**

PILAR II:

Revisión Supervisora. - Requiere que los bancos tengan **procesos internos de evaluación del capital (ICAAP)**. Las autoridades supervisoras pueden **requerir capital adicional** si detectan riesgos no capturados por el Pilar I. Fomenta la **gestión proactiva de riesgos.**

PILAR III:

Disciplina de mercado. - Promueve la **transparencia** mediante requerimientos de divulgación de información sobre:

- Riesgos asumidos
- Políticas de gestión
- Capital disponible

Basilea III (2010)

Tras la crisis financiera de 2007-2008, se evidenció que los requerimientos de Basilea II no eran suficientes para enfrentar crisis de liquidez, apalancamiento excesivo ni el riesgo sistémico. Por ello, el Comité de Basilea lanzó Basilea III**.**

Principales mejoras de Basilea III:

- **Mayor calidad del capital:** Se exige mayor proporción de capital de más alta calidad (CET1 – *Common Equity Tier* 1). Introducción de buffers de conservación (2.5%) y contracíclicos (0-2.5%).
- **Gestión del apalancamiento:** Introduce el *Leverage Ratio* para limitar el endeudamiento, sin importar el riesgo ponderado.
- **Gestión de liquidez:** LCR (*Liquidity Coverage Ratio*), exige activos líquidos para sobrevivir 30 días de crisis. NSFR (*Net Stable Funding Ratio*)**,** promueve estabilidad en el financiamiento a largo plazo.
- **Cobertura de riesgo sistémico:** Exigencias adicionales para bancos de importancia sistémica mundial (G-SIBs)**.**

Tabla 1.- Comparación entre versiones de Acuerdos De Basilea

Elemento / Versión	Basilea I (1988)	Basilea II (2004)	Basilea III (2010 – presente)
Motivación principal	Unificar criterios de capital mínimo	Mejorar sensibilidad al riesgo	Prevenir crisis sistémicas tras la crisis financiera de 2007-2008
Tipo de riesgo cubierto	Solo riesgo de crédito	Crédito, mercado y operacional	Igual que Basilea II + riesgo de liquidez y riesgo sistémico
Metodología	Ponderaciones fijas	Métodos estándar e internos (IRB, AMA)	Métodos más conservadores y exigencias adicionales
Requerimientos de capital	8% sobre activos ponderados por riesgo	Igual, pero con mayor precisión en cálculo	Mayor calidad de capital, buffers adicionales, *ratios* más estrictas
Pilar I	Requerimiento mínimo de capital	Ampliado a más tipos de riesgo	Introduce nuevos requerimientos como el *Capital Conservation Buffer*
Pilar II	No	Revisión supervisora	Reforzado con énfasis en riesgo sistémico y planes de contingencia

Elemento / Versión	Basilea I (1988)	Basilea II (2004)	Basilea III (2010 – presente)
Pilar III	No	Transparencia y disciplina de mercado	Exigencias más detalladas en divulgación, comparabilidad y coherencia
Capital exigido	8%	8% (mínimo regulatorio)	8% + buffers: hasta 13% en algunos casos (incluyendo CET1 y buffers)
Liquidez	No	No	Sí: introduce ratios como LCR (*Liquidity Coverage Ratio*) y NSFR
Leverage Ratio	No	No	Sí: ratio de apalancamiento no ponderado (mínimo 3%)
Aplicación	Globalmente, pero simple	Requiere sofisticación técnica	Se aplica progresivamente desde 2013; más exigente, aún en adopción parcial

Retomando el Pilar I de Basilea II, para el riesgo de crédito, existen tres métodos de medición:

Estándar.- Esta basado en calificaciones externas (agencias de rating o bureaus de crédito). Es el más usado por bancos pequeños y medianos por su simplicidad.

IRB Básico.- En éste método de medición la Institución Financiera estima probabilidad de incumplimiento (PD), lo demás es estándar.

IRB Avanzado.- Aquí la Institución Financiera estima todos los componentes del riesgo como la probabilidad de incumplimiento (PD), la pérdida una vez que ocurre el incumplimiento de un pago (LGD) y la exposición al incumplimiento (EAD).

2.1.2 Método Estándar Para Riesgo De Crédito

Los activos se ponderan según el tipo de contraparte (gobierno, banco, empresa, persona natural).

Se utilizan calificaciones crediticias externas para asignar ponderaciones de riesgo. La Tabla 2 muestra un ejemplo de categorización de la calificación de crédito.

Tabla 2.- Categorización de la calificación de crédito externo

Rating de crédito externo	AAA a AA-	A+ a A-	BBB+ a BB-	Inferior a BB-	No Calificada
Ponderación de riesgo según B II	20%	50%	100%	150%	100%
Requerimiento de capital según B II	1.6%	4%	8%	12%	8%
Ponderación de riesgo según B I	100%	100%	100%	100%	100%
Requerimiento de capital según B I	8%	8%	8%	8%	8%

Por ejemplo, supongamos que una entidad solicita un crédito corporativo por 50 millones a una institución financiera con calificación BB-

El requerimiento de capital según Basilea II es del 8%, es decir 4 millones

La ponderación de riesgo para esta operación de crédito es del 150%

El coeficiente de capital (qk) corresponde a la Ecuación 1

$$qk = \frac{4}{\frac{50 * 150}{100}}$$

$$qk = 5.3\%$$

Ecuación 1 Coeficiente de capital

Para mantener el requerimiento mínimo del 8%, se requieren 6 de capital. Desarrollando la Ecuación 3, se obtiene lo siguiente:

$$qk = \frac{6}{\frac{50 * 150}{100}}$$

$$qk = 8\%$$

Por tanto, con el método estándar el requerimiento de capital se incrementa 1.5 veces.

2.1.3 Métodos basados en modelos internos (Básico)

En le IRB básico, la institución financiera puede emplear su propia estimación de probabilidad de incumplimiento (PD) en un horizonte de un año, así como la exposición al incumplimiento (EAD), para lo cual:

- La PD puede estar basada en la experiencia histórica e incluso en un modelo de calificación de crédito.
- La exposición al incumplimiento (EAD) es igual al valor nominal de la exposición, por ejemplo, el valor en el balance de un préstamo.
- La pérdida dado el incumplimiento (LGD) es dada o asumida por el supervisor.
- Las pérdidas esperadas (PE) sobre el incumplimiento se calculan mediante la Ecuación 2.

$$PE = PD * EAD * (1 - LGD)$$

Ecuación 2 Pérdida Esperada

, siendo (1 – LGD) la severidad de la pérdida.

2.1.4 Métodos basados en modelos internos (Avanzado)

La única diferencia con respecto al Básico es que la pérdida dado el incumplimiento (LGD) está basada en estimaciones propias de la Institución Financiera.

2.2 Construcción de modelos para calificación de riesgo de crédito

La Superintendencia De Bancos Y Seguros Del Ecuador provee una normativa que es la SBS 1897 en la que existen lineamientos para la construcción de modelos para la calificación de riesgo de crédito y se divide en 3 etapas:

- Otorgamiento
- Seguimiento y control
- Pérdidas esperadas

En la etapa de Otorgamiento se establece que deben implementarse sistemas de otorgamiento basado en *scoring* el cual cumpla con requerimientos mínimos como:

- Fundamentación estadística y matemática
- Ejecute una depuración de históricos y variables en el modelo.
- Considere pruebas de *Backtesting*, pruebas de correlación y discriminación de variables.
- Contemple seguridades tecnológicas que garanticen la integridad, confidencialidad y disponibilidad de la información.

En la etapa de Seguimiento y Control, y Pérdidas esperadas, se establece que se ejecuten pruebas de *Backtesting*, estabilidad de la población, significancia de los coeficientes, pruebas de correlación, discriminación de las variables.

La SBS también considera apartados en función del segmento de crédito al que se dirige una operación crediticia y son los siguientes:

- Créditos comerciales.
- Créditos de consumo.
- Pymes.
- Microcrédito.

El objetivo de la tesis se enfocará en el segmento de Crédito De Consumo, sin descartar la aplicabilidad de la metodología en los otros segmentos de crédito.

2.2.1 Objetivos del modelo de *scoring*

- Discriminar clientes buenos de los malos.
- Generar probabilidades categóricas o numéricas de incumplimiento (PI) para operaciones nuevas y existentes.
- Mejorar la gestión de las colocaciones.
- Mejorar la gestión de capital.

2.3 Modelo Logit en el Sistema HCRISK

El modelo implementado en el Sistema HcRisk para clasificación de clientes buenos y malos es el Modelo Logit, que es del tipo No Lineal.

En este modelo se define una variable Dependiente (VD) que va a ser definida en función de pesos asignados a variables independientes correspondientes a características socio demográficas, socio económicas, y de comportamiento financiero de los sujetos de crédito.

El Modelo Logit es una función de distribución logística como se muestra en la Ecuación 3.

$$Y = \frac{1}{1 + e^{-(c+\beta 1 * X1+\beta 2 * X2+\cdots)}}$$

Ecuación 3 Modelo Logit

Para la implementación de este modelo se siguió la siguiente metodología:

- Levantamiento de Datos Históricos

Primero, se identificaron variables de los sujetos de créditos que pudieran ser alimentadas por los oficiales o personal de la Institución Financiera. Luego, se verificó la maduración de los datos observando la recomendación de que los datos deben tener una antigüedad de mínimo un año para el segmento de crédito de consumo, también se verificó la consistencia de la información a través de análisis de frecuencias de las variables cualitativas.

Segundo, se identificaron datos incorrectos o no estandarizados, también se identificó la concentración por categorías y por último se realizó un análisis descriptivo de las variables numéricas, verificando comportamientos extraños de los datos (momentos de la media), así como también los mínimos y máximos.

- Variable dependiente u objetivo (*target*)

Se definió a la variable dependiente (VD) como dicotómica con los valores 0 para Buenos y 1 para Malos clientes. El valor respectivo a cada set de variables de cada sujeto de crédito se asignó en función de su valor de morosidad (Políticas de morosidad), es decir que si el cliente no tiene mora entonces es Bueno, y si el cliente tiene una mora mayor a 15 días el cliente es Malo.

- Se definió el tamaño de la muestra y se la seleccionó de forma aleatoria.

El sistema HcRisk tiene una ventana para realizar la selección de la muestra en donde se pueden definir los parámetros de selección.

- Se realizó el desarrollo estadístico del modelo:

En esta etapa se realiza un Análisis de correlación de variables y se crearon nuevas variables combinadas de tal forma que se asemeja al proceso automático de escalado y codificación que tiene la librería Scikit-learn de Python.

Como penúltimo paso, aplicando el modelo Logit con R, se calculan los pesos de las variables respetando políticas institucionales. Se seleccionaron las mejores variables y se prueban las probabilidades obtenidas frente a la muestra.

- Validación y Seguimiento:

Por último, en la etapa de validación y seguimiento, se verifica la capacidad de discriminación del modelo a través de indicadores Gini y K-S.

  - Se verifica el nivel de confianza.
  - Se verifica la estabilidad de la población (PSI).

2.4 Estado del arte en Aprendizaje Automático en torno al comportamiento crediticio

A continuación, se exponen técnicas y modelos actuales de aprendizaje automático utilizados en clasificación binaria y que pueden aplicarse a casos de análisis del comportamiento crediticio, para sustituir efectivamente el modelo Logit clásico. Estos modelos tienen en común su capacidad para detectar relaciones complejas**,** mejorar la precisión predictiva y, en algunos casos, mantener la interpretabilidad, lo cual es crucial en entornos regulados como el financiero.

2.4.1 Árboles de decisión y similares

Los árboles de decisión son algoritmos que particionan el espacio de datos en regiones jerárquicas, permitiendo modelar relaciones no lineales e interacciones entre variables de forma automática. Capturan relaciones no lineales y efectos de interacción automáticamente.

2.4.1.1 Random Forest

Conjunto de múltiples árboles entrenados con subconjuntos aleatorios del *dataset* y de las variables. **(**Breiman, L. 2001). Es un modelo robusto frente al sobreajuste, fácil de interpretar parcialmente. Puede aplicarse con éxito en *scoring* crediticio donde se busca un balance entre precisión e interpretabilidad. Es robusto ante datos ruidosos y multicolinealidad, puede manejar relaciones no lineales entre variables, es fácil de interpretar (importancia de variables) y generalmente mejora el rendimiento frente a la regresión logística.

2.4.1.2 Gradient Boosting Machines (XGBoost, LightGBM, CatBoost)

Construyen árboles secuenciales, donde cada uno corrige los errores del anterior. Altamente efectivo en tareas de clasificación binaria con datos estructurados tabulares (Chen, T., & Guestrin, C. 2016); XGBoost y LightGBM se usan ampliamente en competencias de ciencia de datos. Mientras que CatBoost es especialmente útil con variables categóricas sin preprocesamiento (Prokhorenkova, L. et al. 2018). Son altamente eficiente para clasificación binaria, manejan bien los valores faltantes y relaciones complejas, además son muy usados en sistemas financieros por su precisión y capacidad de interpretación parcial (con SHAP).

2.4.1.3 Máquinas de Vector Soporte (SVM)

Se basan en la idea de maximizar el margen entre clases. Usan kernels para proyectar los datos a espacios de mayor dimensión donde la separación es más fácil (Cortes, C., & Vapnik, V. 1995). Alcanza alta precisión en conjuntos complejos y de gran tamaño. Los kernels pueden capturar relaciones no lineales. Sin embargo, son más costosas computacionalmente y menos interpretables (Xia, Y., Liu, C., Li, Y., & Liu, N. 2017)

2.4.1.4 Redes neuronales multicapa (MLP)

Son aproximadores universales para modelar cualquier función si tienen suficiente profundidad y datos de entrenamiento normalizados (Hornik, K., Stinchcombe, M., & White, H. 1989).

Tiene flexibilidad para capturar patrones complejos, pero tienden al sobreajuste. No son usadas con frecuencia en el ámbito financiero debido a la dificultad de interpretación aunque son cada vez más viables si se combinan con técnicas explicativas (Yeh, I.-C., & Lien, C.-H. 2009).

2.4.1.5 Modelos explicables (*Explainable* AI)

La inteligencia artificial explicable (XAI) es un conjunto de procesos y métodos que permiten a los usuarios humanos comprender y confiar en los resultados creados por algoritmos de *machine learning*. La IA explicable se utiliza para describir un modelo de IA, así como su impacto previsto y también sus posibles sesgos. (Dhanesh Ramachandram)

2.4.1.6 Logit Penalizado (ElasticNet)

Es un modelo de regresión lineal que normaliza el vector de coeficientes con las normas L1 (Lasso) y L2 (Ridge). Esto permite generar un modelo en el que solo algunos de los coeficientes sean no nulos. Hace una selección automática de variables y es más robusto que el modelo Logit clásico (Zou, H., & Hastie, T. 2005).

2.4.1.7 Generalized Additive Models (GAM)

Permiten relaciones no lineales, pero aditivas. No modelan interacciones entre variables (Hastie, T., & Tibshirani, R. 1986). Son bien aceptados en entornos regulados por su balance entre flexibilidad e interpretabilidad (Dhanesh Ramachandram , et al 2025).

2.4.1.8 SHAP (SHapley Additive exPlanations)

Basado en teoría de juegos. Atribuye a cada característica una contribución al resultado y permite un modo de caja negra o caja blanca. Es muy utilizado en banca y salud por su capacidad de generar confianza en decisiones algorítmicas (Lundberg, S. M., & Lee, S.-I. 2017).

2.4.1.9 LIME

Genera modelos locales lineales para explicar predicciones individuales. Permiten interpretar modelos complejos, lo cual es crucial en sectores regulados como el financiero, es más simple que SHAP pero menos robusto (Ribeiro, M. T., Singh, S., & Guestrin, C. 2016).

2.4.1.10 TabNet

Es un modelo moderno dentro de las Redes Neuronales interpretables que aplica atención secuencial sobre las variables, permitiendo explicar qué atributos guían la decisión. Está diseñado para trabajar con datos tabulares (Shijie Wang, Xueyong Zhang, 2025).

2.4.1.11 **AutoML *frameworks***

Aceleran la experimentación, ideal para prototipado rápido. Facilitan la automatización del proceso de modelado: selección del modelo, ajuste de hiperparámetros, validación cruzada y ensamblado. Son útiles en etapas exploratorias y para validar que no se está dejando pasar un modelo más efectivo por otro (Zöller, M.-A., & Huber, M. F. 2021).

Auto-sklearn. – Basado en scikit-learn, usa optimización bayesiana.

TPOT. – Evoluciona Pipelines mediante algoritmos genéticos.

H2O AutoML. - Automatiza la comparación y selección del mejor modelo y metaparámetros. Es una herramienta robusta de nivel industrial, fácil de integrar con entornos empresariales.

2.4.2 Herramientas para permitir reentrenamiento del modelo

Objetivo: Que usuarios del sistema HcRisk puedan reentrenar el modelo con nuevos datos sin necesidad de conocimientos avanzados.

2.4.2.1 Lenguaje y entorno

Para el desarrollo, pruebas y puesta en marcha del modelo se usará **Python** con la ayuda de sus librerías scikit-learn, catboost, xgboost para modelado, pandas para persistencia de datos y de modelos, así como para limpieza y preprocesamiento, y la librería Flask para exponer una API REST que permita llamar a los procesos desde la interfaz web.

2.4.2.2 Arquitectura propuesta

Pipeline de entrenamiento: Se realiza primero la lectura de datos a través de una conexión a la base de datos. Como segundo paso, se realiza el preprocesamiento. El tercer paso es el entrenamiento del modelo seleccionado. Y, por último, el paso 4 consiste en la evaluación y almacenamiento del nuevo modelo.
Además, desde la interfaz web habrá un botón que lanza el reentrenamiento llamando al API respectivo.

2.4.2.3 Métricas

A través de la interfaz web se harán llamadas a las APIs que devolverán las métricas clave como AUC, F1-score, precisión y sensibilidad.

2.5 Contexto y justificación

El sistema HcRisk, en su conFiguración actual, emplea un modelo estadístico desarrollado en R, que es ejecutado desde el *backend* por medio de un servicio web expuesto bajo un modelo de arquitectura de dos capas. La capa de presentación permite el ingreso de variables por parte de los usuarios —principalmente oficiales de crédito o analistas de riesgo— quienes pueden realizar simulaciones o cálculos en tiempo real para nuevos solicitantes. La lógica de negocio se encuentra encapsulada en un componente de servidor que consulta el modelo Logit y devuelve el resultado.

Esta implementación, aunque funcional, no está optimizada para una evolución continua del modelo, ya que cada cambio implica una reconFiguración manual, regeneración del servicio y verificación de dependencias de R. Además, los procesos de reentrenamiento requieren intervención técnica especializada. En contraste, migrar a una solución basada en Python no solo permite aprovechar *frameworks* modernos (como scikit-learn, XGBoost, SHAP), sino también facilitar el despliegue de modelos mediante APIs ligeras (como Flask), integrables fácilmente con el *frontend* desarrollado en Java.

La ventaja del desarrollo de aplicaciones modularmente permite que soluciones que tienen enfoques diferentes o plataformas diferentes se puedan integrar para maximizar su aplicabilidad y dar al usuario final herramientas fáciles de usar que le ahorrarán tiempo y dinero en su trabajo diario. Es así que con este proyecto el usuario podrá realizar las evaluaciones de clientes de una manera ágil, también podrá monitorear el desempeño del modelo y solicitar al experto el reentrenamiento sin que sea necesario compartir toda la base de datos si no sólo el conjunto de datos que sea necesario.

2.6 Planteamiento del problema

El modelo Logit, siendo un modelo lineal, tiene limitaciones intrínsecas para capturar relaciones complejas entre variables, particularmente cuando se manejan conjuntos de datos con fuerte multicolinealidad, no linealidades evidentes, o interacciones entre atributos que afectan de manera combinada el comportamiento de los clientes. Además, la interpretación de sus coeficientes, aunque directa, puede inducir a errores si no se contextualiza con pruebas de significancia y validación.

En la operatividad del sistema HcRisk, estas limitaciones se traducen en una menor capacidad de discriminación entre clientes (*score* medio), así como una mayor dependencia de ingeniería de características manual y de validaciones estadísticas exógenas. También se identifica una carga operacional significativa en el mantenimiento del modelo, ya que su entorno en R impone barreras para la actualización ágil del modelo, su seguimiento en producción o su integración con *pipelines* automatizados. Por tanto, existe una necesidad apremiante de contar con un modelo más flexible, interpretable y eficiente en ejecución.

2.6.1 Descripción de Tablas y procesos de HcRisk

Las Tablas principales involucradas en el modelamiento de la solución con el modelo Logit son las siguientes:

Tabla 3.- Principales Tablas involucradas en el modelo Logit

Tabla	Descripción
Cge_cliente	Atributos de los clientes actuales y potenciales de la institución
Rc_credito	Atributos de las operaciones crediticias
Rc_estado_cuenta	Atributos de las cuotas que se van pagando de cada operación

Rc_variables	Variables o *features* de los clientes y operaciones
Rc_cliente_variable	Almacena los valores que toman las variables por cada cliente y operación
Rc_rango_variable	Almacena los posibles valores que puede tener una variable cualitativa
Rc_recodif_variable	Almacena los posibles valores que puede tener una variable recodificada
Rc_tipo_cartera	Atributos de los tipos de cartera de la institución
Rc_politica	Almacena las políticas para calificar a las operaciones como buenas o malas en función de los pagos almacenados en rc_estado_cuenta
Rc_zona	Contiene los límites para determinar si la probabilidad de incumplimiento cae en un Rango como Aprobado, Rechazado o Comité (que indica que la solicitud de crédito se somete a una evaluación personalizada)

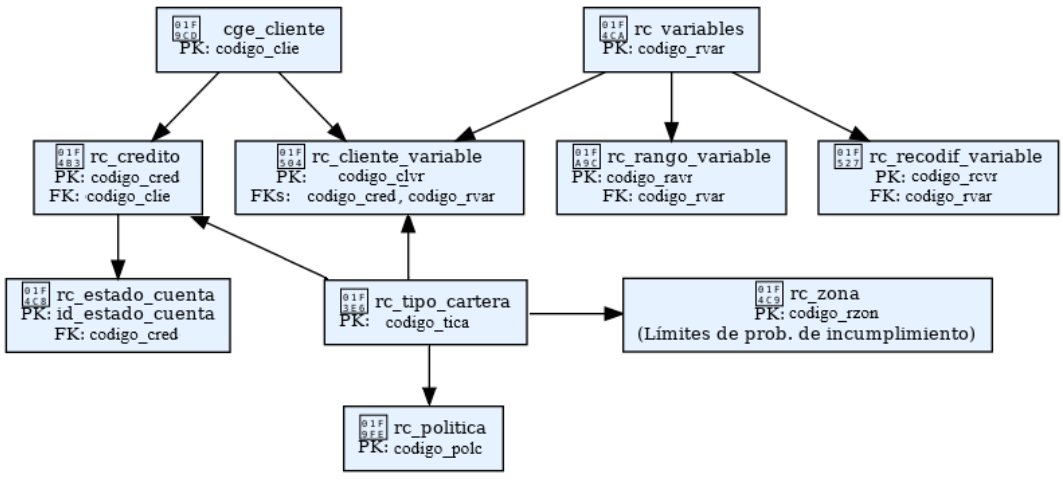


Figura 1: Esquema del modelo conceptual simplificado:

El Sistema HcRisk tiene una ventana sobre la cual se realiza el ingreso de variables o *features* y en caso de que sean cualitativas permite el ingreso del rango que les corresponde.

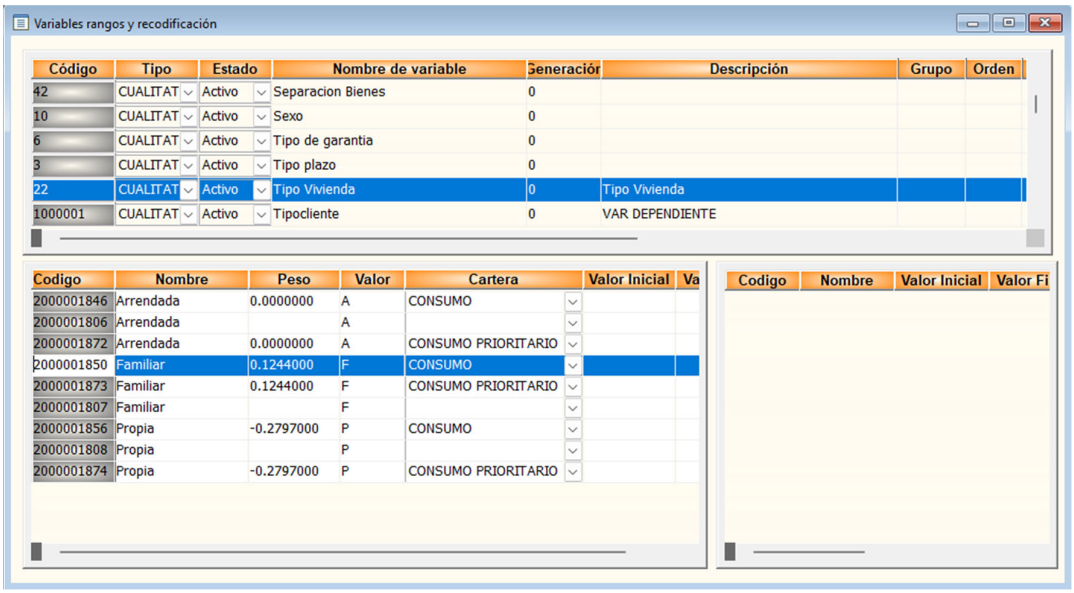


Figura 2 Pantalla HcRisk Variables rangos y recodificación

En la Figura 2 se puede apreciar la variable Tipo Vivienda con sus posibles valores que puede tomar (A, F, P) así como también los pesos calculados para dichos valores. Dicho peso fue realizado por el modelo logit implementado en el sistema HcRisk.

El sistema HcRisk utiliza un sistema de políticas para poder calificar a las operaciones como buenas o malas basado en los días impagos de las cuotas y el porcentaje o cantidad de incumplimientos en los pagos periódicos como se ve en la figura 3.

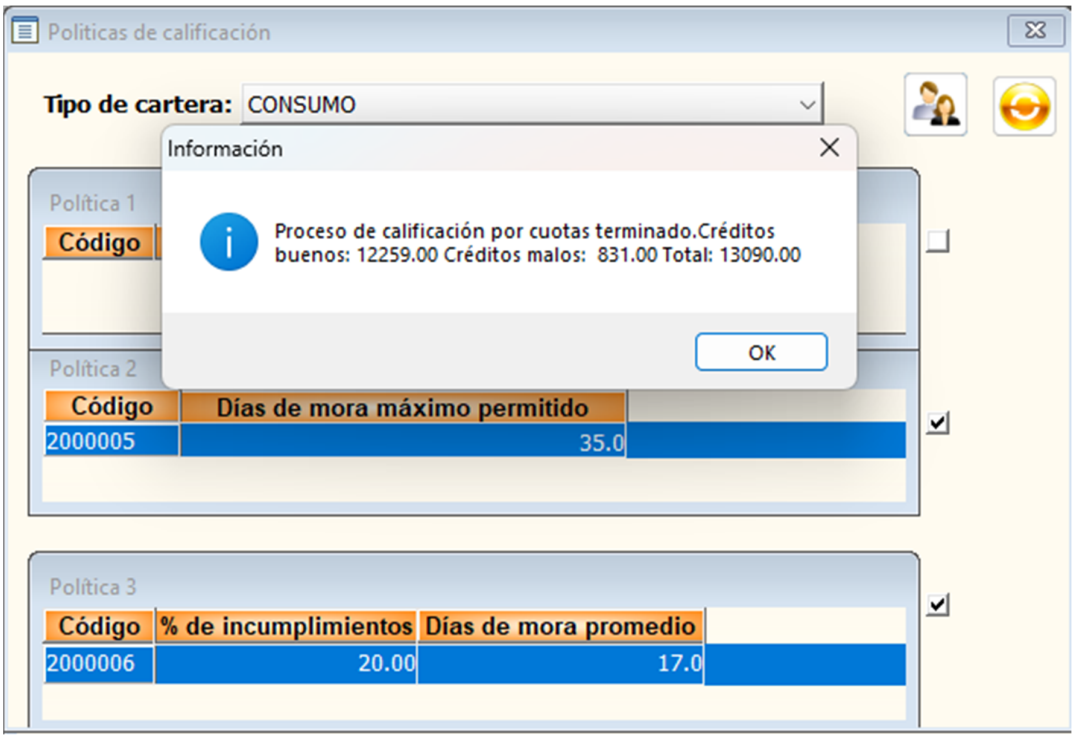


Figura 3 HcRisk Políticas de calificación

En la Figura 3 se puede observar que para que una operación se califique como mala la fecha de pago debe sobrepasar la fecha de vencimiento en 35 días y además haber incumplido los pagos de las cuotas un 20% de veces y tener en promedio 17 días de mora en cada cuota incumplida. Además, luego de realizar el proceso de calificación a operaciones desde el año 2021 hasta el año 2024 se encuentra que hay 12,259 operaciones buenas y 831 operaciones malas dando un total de 13,090 operaciones. Lo cual nos da un conjunto de datos desbalanceado.

Capítulo 3. OBJETIVOS

3.1 Objetivos generales

El objetivo general de este proyecto es desarrollar e implementar un modelo de aprendizaje automático que reemplace al modelo de regresión logística binaria actualmente utilizado en el sistema HcRisk para la clasificación de clientes en buenos o malos pagadores.

Con dicho modelo se busca mejorar la precisión predictiva, facilitar el mantenimiento y asegurar una mayor escalabilidad del sistema. Para ello, se evaluarán al menos tres modelos de *machine learning* desarrollados en Python, comparando su desempeño con el modelo Logit vigente, con miras a su integración en el sistema actual.

.

3.2 Objetivos específicos

- Analizar el modelo actual de regresión logística (Logit) implementado en el sistema HcRisk, identificando sus limitaciones en cuanto a precisión, interpretabilidad, escalabilidad y mantenimiento.
- Seleccionar al menos tres modelos de aprendizaje automático modernos aplicables a la clasificación binaria, considerando criterios de rendimiento, interpretabilidad y adEcuación a datos tabulares financieros.
- Implementar los modelos seleccionados en un entorno de desarrollo en Python, utilizando bibliotecas especializadas en aprendizaje automático y asegurando la reproducibilidad del proceso de entrenamiento y evaluación.
- Comparar el rendimiento de los modelos propuestos respecto al modelo Logit, utilizando métricas de evaluación como AUC-ROC, F1-score, precisión y sensibilidad, además de técnicas de validación cruzada.

- Evaluar la interpretabilidad y facilidad de mantenimiento de cada modelo, mediante herramientas como SHAP o LIME, y estimar su viabilidad operativa dentro del sistema HcRisk.
- Recomendar el modelo más adecuado para ser implementado como reemplazo del modelo Logit, justificando la decisión en función de resultados técnicos, escalabilidad futura y adaptabilidad al entorno financiero actual.
- Implementar dicho modelo en un entorno de Pruebas previo a la puesta en Producción.

3.3 Beneficios del proyecto

3.3.1 Beneficios Técnicos

Uno de los beneficios técnicos del proyecto es que mejora en la precisión predictiva del sistema de *scoring* crediticio, gracias a la implementación de modelos de aprendizaje automático más avanzados que capturan relaciones no lineales y complejas entre las variables.

Otro beneficio inherente a la implementación de modelos de aprendizaje automático es la reducción del sesgo y la mejora en la detección de patrones asociados al comportamiento de pago, lo cual permite decisiones más informadas en la evaluación de riesgo crediticio.

Al aplicar técnicas modernas de interpretabilidad (*Explainable AI*) como SHAP, se brinda tanto al usuario, entidades de control y al cliente potencial de transparencia en el proceso de clasificación, lo cual determina el alto grado de confianza en las decisiones crediticias en entornos regulados como el financiero.

Otro beneficio es que con la automatización y estandarización del pipeline de entrenamiento y validación de modelos en Python se facilitan las futuras actualizaciones y recalibraciones del sistema.

3.3.2 Beneficios Operativos

Como se mencionó en el punto anterior la facilidad de calibración y actualización del modelo aventaja mucho al presente proyecto respecto a la implementación en R que está codificado en el programa sin posibilidad de modificación alguna porque se encuentra compilado.

En el transcurso normal de la vida de la institución financiera, habrá nuevas variables que aumentarán la complejidad de la clasificación crediticia. Este tipo de retos no es mayor inconveniente para los modelos de aprendizaje automático ya que cuentan con técnicas para determinar si las nuevas variables son útiles para el modelo.

Otro gran beneficio del proyecto para una institución financiera es la reducción de errores de clasificación (falsos positivos y falsos negativos), ya que de otra manera se impactaría directamente en la calidad de la cartera de clientes reduciendo la rentabilidad de los productos crediticios.

3.3.3 Beneficios Estratégicos

El fortalecimiento del proceso de toma de decisiones en la gestión de riesgo crediticio es un gran beneficio estratégico que mejora la rentabilidad y sostenibilidad financiera de la institución.

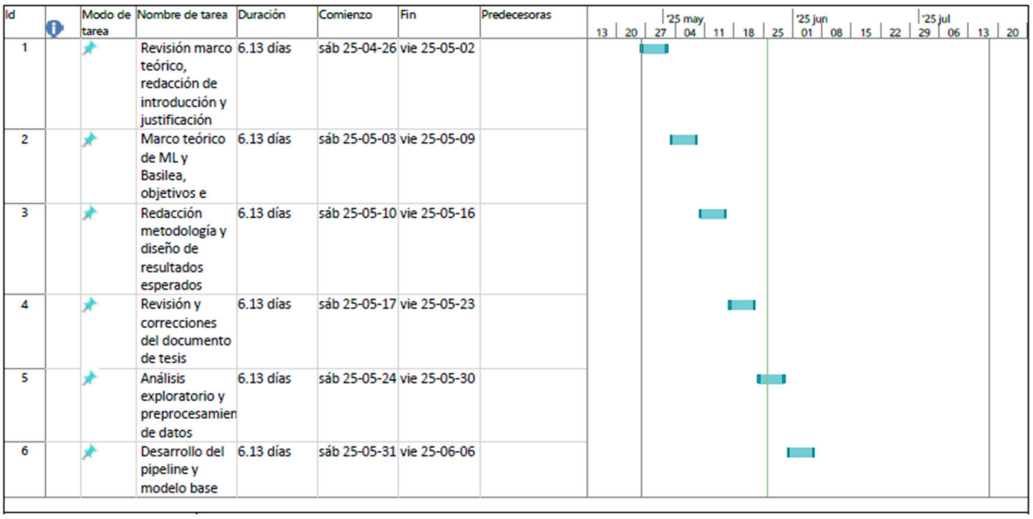
Debido al entorno tecnológico que rodea a las instituciones financieras, la incorporación de aplicaciones apoyadas en aprendizaje automático sobre entornos web harán que evolucionen al mismo ritmo de los cambios actuales.

En la misma dirección de lo antes mencionado, se crea la posibilidad de extender la metodología a otros modelos internos de la institución financiera, generando un efecto multiplicador hacia una modernización vanguardista.

Capítulo 4. DESARROLLO DEL PROYECTO

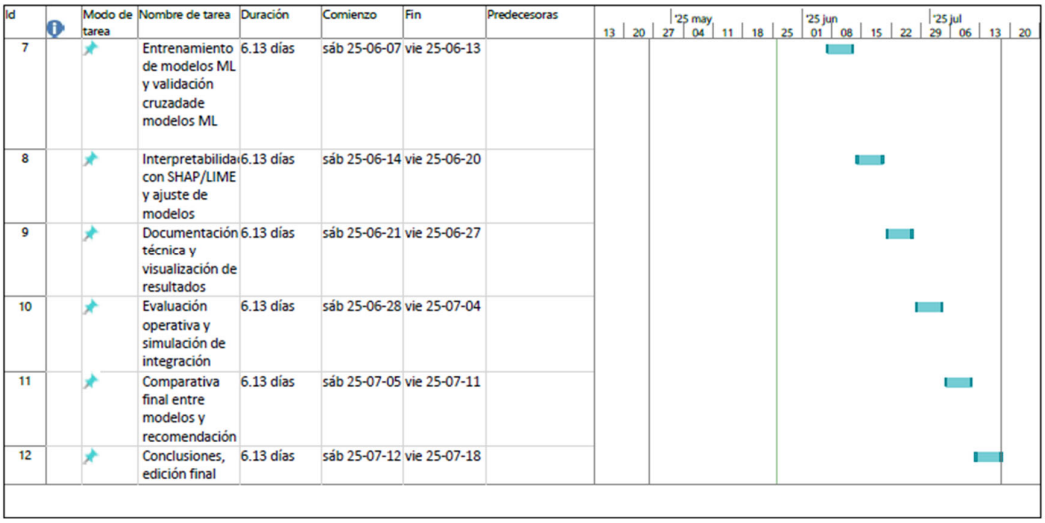
4.1 Planificación del proyecto

El presente proyecto se ha planificado para que sea desarrollado en un lapso de 2 meses de acuerdo a la distribución temporal de las tareas que se detallan en las Figuras 4 y 5.



Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1		Revisión marco teórico, redacción de introducción y justificación	6.13 días	sáb 25-04-26	vie 25-05-02	
2		Marco teórico de ML y Basilea, objetivos e	6.13 días	sáb 25-05-03	vie 25-05-09	
3		Redacción metodología y diseño de resultados esperados	6.13 días	sáb 25-05-10	vie 25-05-16	
4		Revisión y correcciones del documento de tesis	6.13 días	sáb 25-05-17	vie 25-05-23	
5		Análisis exploratorio y preprocesamien de datos	6.13 días	sáb 25-05-24	vie 25-05-30	
6		Desarrollo del pipeline y modelo base	6.13 días	sáb 25-05-31	vie 25-06-06	

Figura 4 Tareas de planificación del proyecto Parte 1



Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
7		Entrenamiento de modelos ML y validación cruzadade modelos ML	6.13 días	sáb 25-06-07	vie 25-06-13	
8		Interpretabilida con SHAP/LIME y ajuste de modelos	6.13 días	sáb 25-06-14	vie 25-06-20	
9		Documentación técnica y visualización de resultados	6.13 días	sáb 25-06-21	vie 25-06-27	
10		Evaluación operativa y simulación de integración	6.13 días	sáb 25-06-28	vie 25-07-04	
11		Comparativa final entre modelos y recomendación	6.13 días	sáb 25-07-05	vie 25-07-11	
12		Conclusiones, edición final	6.13 días	sáb 25-07-12	vie 25-07-18	

Figura 5 Tareas de planificación del proyecto Parte 2

4.2 Descripción de la solución, metodologías y herramientas empleadas

A continuación, se describe en detalle la solución técnica propuesta para el desarrollo del modelo predictivo de clasificación binaria del comportamiento crediticio. La implementación se realizará utilizando tecnologías modernas y estandarizadas, con énfasis en el uso de Python 3.11 como lenguaje principal, SQL Server como base de datos relacional, y Java junto con las librerías PrimeFaces y JPA para la integración de interfaces gráficas y llamadas a los modelos desde el *frontend*.

4.2.1 Arquitectura General del Sistema

La arquitectura del sistema se basa en un modelo cliente-servidor de aplicaciones-repositorio de datos (conocido como 3 capas) que permite una separación clara entre la lógica de presentación, la lógica de negocio y el acceso a los datos. A continuación, se describen las capas del sistema:

1. **Capa de Presentación (*Frontend*):**

El *frontend* se desarrolla en el lenguaje Java con la librería PrimeFaces, usando tecnologías web estándar como HTML5, CSS3 y JavaScript. Este permite al usuario interactuar con el sistema mediante formularios, y reportes. Incluye pantallas específicas para: Carga manual de variables de nuevos clientes, ejecución del modelo de *scoring, v*isualización de resultados (probabilidad de incumplimiento, categorización como buen o mal pagador). También incluye reportes para el monitoreo de métricas clave del modelo (Ej. AUC-ROC, F1-score, etc.).

2. **Capa de Lógica de Negocio (*Backend*):**

La capa de la lógica se implementa en Java para las interfaces de conexión con la base de datos, así como para la integración con los componentes gráficos. Por otro lado, también involucra la programación en Python de las APIs y del modelo predictivo en sí mismo.

3. **Capa de Datos:**

La capa de datos o de persistencia consiste en la Base de datos relacional MariaDB, utilizada para almacenar el historial de operaciones crediticias, las variables utilizadas por el modelo, los resultados de ejecuciones anteriores del modelo y configuraci del sistema y parámetros de validación.

4.2.2 Metodología de Desarrollo KDD

El desarrollo del proyecto sigue una metodología KDD que se centra en 5 fases que son:

- Recopilación de datos
- Selección, limpieza y transformación
- Minería de datos
- Interpretación de los modelos obtenidos
- Evaluación de los modelos obtenidos

Esta metodología se la integrará con una Agile de tal forma de realizar iteraciones cortas enfocadas en resolver problemas específicos y entregar valor incremental. Se estructuran las fases principales del desarrollo como sigue:

1. **Análisis de Requisitos:**
    - Identificación de requisitos funcionales y no funcionales.
    - Definición de variables críticas para el modelo.
    - Revisión de políticas institucionales y regulaciones financieras (como los Acuerdos de Basilea).
2. **Diseño de Solución:**
    - Diseño de la arquitectura general del sistema.
    - Definición de interfaces entre componentes (*frontend* y *backend*).
    - Especificación de bases de datos y diagramas entidad-relación.

3. **Desarrollo:**
    - Implementación del *pipeline* de preprocesamiento de datos.
    - Desarrollo e integración de modelos de aprendizaje automático.
    - Construcción de APIs RESTful para la comunicación entre capas.
    - Desarrollo de interfaces gráficas con Java.
4. **Pruebas:**
    - Pruebas unitarias y de integración de cada componente.
    - Validación estadística del modelo (*cross-validation*, *backtesting*).
    - Pruebas de rendimiento y escalabilidad del sistema.
5. **Implementación y Despliegue:**
    - Despliegue en entorno de pruebas y posteriormente en producción.
    - Documentación técnica y de usuario.
6. **Monitoreo y Mantenimiento:**
    - Monitorización de desempeño del modelo.
    - Actualización periódica del modelo y reentrenamiento automatizado.

4.2.3 Herramientas y Tecnologías Empleadas

Para el desarrollo y puesta en marcha del proyecto se requieren lenguajes de programación específicos, entornos de programación para cada lenguaje, servidor de aplicaciones compatibles y servidor de base de datos, pero de fácil acceso a través de la web.

4.2.3.1 Lenguajes de programación

- **Python 3.11:** Para el desarrollo del modelo predictivo y *pipeline* de datos.
- **Java 8:** Para el desarrollo de la capa de lógica de negocio y servicios web.
- **JavaScript:** Para interacciones dinámicas en el *frontend*.

4.2.3.2 ***Frameworks* y Librerías**

Python:

- **Scikit-learn:** Para modelado estadístico, validación cruzada y evaluación de métricas.
- **XGBoost / CatBoost :** Implementación de los modelos avanzados de aprendizaje automático.
- **Pandas & NumPy:** Manipulación y transformación de datos.
- **SHAP :** Técnicas de interpretabilidad del modelo.
- **Flask:** Creación de APIs RESTful para la exposición del modelo, que son programas que actúan como interfaz entre la pantalla de usuario y la aplicación de Python.

4.2.3.3 **Java:**

- **PrimeFaces:** Plataforma Java para construcción de interfaces web ricas.
- **JPA (Java Persistence API):** Mapeo objeto-relacional para acceder a la base de datos. Es una interfaz entre el modelo de datos de la base de datos y el programa Java.
- **Jboss Wildfly:** servidor de aplicaciones para interactuar con la base de datos, la aplicación web Java y los servicios web (APIs Restful) de Python.

4.2.3.4 Base de Datos:

- **MariaDB:** Como motor relacional para almacenamiento estructurado de datos históricos y resultados del modelo.

4.2.3.5 Herramientas Adicionales:

- **Jupyter Notebook:** Entorno interactivo para análisis exploratorio de datos y prototipado de modelos.
- **Postman:** Pruebas de APIs RESTful.
- **Power BI:** (Opcional) Para generación de informes analíticos posteriores.

4.2.4 Desarrollo del Modelo Predictivo

Se genera un proyecto en PyCharm con un *script* para probar el modelado el cual obtiene los datos desde la base de datos MariaDB. Empezamos con el análisis exploratorio.

Variables con datos

En la base de datos existe la Tabla rc_Variables que contiene el catálogo de variables en el que constan tanto Variables originales y Variables recodificadas, y ésta se relaciona con la Tabla rc_cliente_variable que contiene los valores de cada variable para cada operación crediticia.

Las Variables originales numéricas con datos se muestran en la Tabla 4:

Tabla 4.- Variables originales numéricas

Nombre de variable	Descripción	Operaciones Por Variable
Activos Totales	Activos Totales	11,420
AMORTIZADOCREDITO	Amortizado crédito (saldo del crédito / Monto concedido)	13,012
Antiguedad Laboral	Años trabajados	11,420
Cargas Familiares	Cargas Familiares	11,420

Cred Sis Finan corto plazo	Crédito en el Sistema financiero CP	11,420
Cred Sis Finan largo plazo	Crédito en el Sistema financiero LP	11,420
Creditos casas comerciales	Número de operaciones en comercios	11,420
Creditos prestamistas	Número de operaciones prestamistas	11,420
Cuota Estimada Buro	Cuota de acuerdo al Buró de crédito	11,420
CUOTANOPAGA3MESES	Numero de cuotas no pagadas a la fecha (últimos 3 meses) corto plazo	13,012
CUOTANOPAGA6MESES	Numero de cuotas no pagadas a la fecha (últimos 6meses)	13,013
CUOTAPAGADAVTOTALCREDITO	Sum del valor de las cuotas pagadas por el cliente /Suma total de los créditos activos del cliente	13,012
DEUDAINDVENCIDACOOP	deudas indirectas vencidas dentro de la cooperativa	13,012
Dias mora promedio credanter	Días de mora en operaciones anteriores	11,420
Edad	Edad del cliente	11,420
Gastos	Gastos incurridos por el cliente	11,420
INDCARTERAVENCIDA	Indicador de Cartera Vencida) Sumatoria de cuotas vencidas / Saldo del crédito	13,013
Ingreso	Ingreso promedio mensual	11,420
Ingreso Familiar	Ingreso global de la familia	11,420
Ingreso Neto	Ingreso Neto del cliente	11,420
Ingresos	Ingresos Totales del cliente	13,012
INGSALDOCREDIOSEG	Ingreso saldo crédito segmento	13,012
INTERESMORA	intereses x mora (todos los créditos)	13,013
Mayor monto atrasado buro	Valor más alto registrado como atraso según Buró	11,420
Mayor plazo atrasado Buro	Tiempo más alto registrado como atraso según Buró	11,420
MONTOCREDITOSEG	Monto de la operación	13,012
MORA	Mora incurrida de la operación	13,012
Num incump cred anteriores	Número de incumplimientos anteriores	11,420
Numero cred anteriores	Cantidad de operaciones anteriores	11,420
Numero de Cuotas	Número de cuotas que tiene la operación	11,420
NUMEROGARANTIAS	Número de Garantías	13,012
NUMOPVIGENTESCOOP	Cantidad de operaciones vigentes dentro de la Institución	13,013
Nuncuotacurso	Número de cuota en curso	13,012
ORIGEN	Origen de fondos	13,012
Otros creditos corto plazo	Otras operaciones CP	11,420
Otros creditos largo plazo	Otras operaciones LP	11,420

Otros Ingresos	Otros ingresos que registra el cliente	11,420
Pasivos Totales	Pasivos que tiene el cliente	11,420
Patrimonio Socio	Patrimonio que tiene el cliente	11,420
Plazo meses	Plazo meses	11,420
PROVISIONMONTO	Monto de la provisión	13,012
PROVISIONPORC	Porcentaje de la provisión	13,012
Saldo Promedio	Saldo Promedio de cuenta bancaria	11,420
SALDOCREDITOSEG	Saldo crédito seguimiento	13,012
SALDOCUENTAAHORROSEG	Saldo cuenta ahorro seguimiento	13,012
SUMATOTALCREDITOSCLIE	Suma total de los créditos activos del cliente	13,011
SUMCUOTAVENCIDA	Suma cuota vencida	13,012
sumtotcuotas	Suma de todas las cuotas (total que paga al mes de todos créditos que tenga el cliente)	13,011
Tiempo de residencia	Tiempo de residencia	11,420
Valor Cuota	Valor Cuota	11,420
Valor Cuota ingreso	Valor de cuota inicial	11,420
Valor garantias	Valor de las garantías entregadas por el cliente	11,420

Las Variables originales cualitativas con datos en muestran en la Tabla 5

Tabla 5.- Variables originales cualitativas

Nombre de variable	**Descripción**	**Operaciones Por Variable**
Actividad Economica	Código de actividad económica principal a la que se dedica el cliente	11,420
Actividad Secundaria	Código de actividad económica secundaria a la que se dedica el cliente	11,420
CALIFICACION	Calificación	13,012
Destino del Credito	Destino de los fondos	11,420
Detalle Destino	Detalle del destino	11,420
Estado	Estado Nuevo	11,420
Estado Civil	Estado Civil	11,420
ESTADO CIVIL	Nuevo estado Civil	13,012
Localidad	Ubicación del cliente	11,420
Ocupacion profesion	Profesión del cliente	11,420

Peor Calificacion	Peor calificación del cliente	11,420
Producto	Nombre del Producto de crédito	11,420
Recibe SPN	Recibe SPN	11,420
Relacion Dependencia	Trabaja como empleado Si o No	11,420
Separacion Bienes	Está casado con separación de Bienes Si o No	11,420
Sexo	Masculino o Femenino	11,420
Tipo de garantia	Garantía real o no real	11,420
Tipo plazo	Tipo de plazo de la operación	11,420
Tipo Vivienda	Tipo Vivienda	11,420
Tipocliente	**Variable Dependiente**	**13,090**

Como se puede observar, la variable objetivo es tipo categórica y se llama Tipocliente. Además, se observa que existen al menos 11,420 operaciones con las que se puede realizar el modelado.

Dado que hay variables con más de 11,420 operaciones, se procede a analizar esos casos y se encuentra que son variables de seguimiento que varían mensualmente. Por lo tanto, se tomará para el modelado el cambio más reciente de cada caso.

Se desarrolla en Python una API (Interfaz de Programación de Aplicaciones) que permitirá interactuar entre la interfaz Java (*frontend*) y el modelado de ML.

4.2.4.1 Pipeline de procesamiento de datos

Se empieza por la creación de un archivo *jupyter notebook* que tendrá todos los scripts necesarios para el modelamiento y a partir de éste se tomarán extractos para generar las APIs

- **Carga de Datos. -** El proceso de carga de datos es el primer paso de todo lo que conlleva el modelamiento. Los datos son extraídos directamente desde la base de datos MariaDB a través de una consulta a la Tabla rc_cliente_variable (Figura 6) y se almacena en *DataFrames* de Pandas para su manipulación. Realizado este paso vemos en la Figura 6 que se han recuperado 850,000 filas.

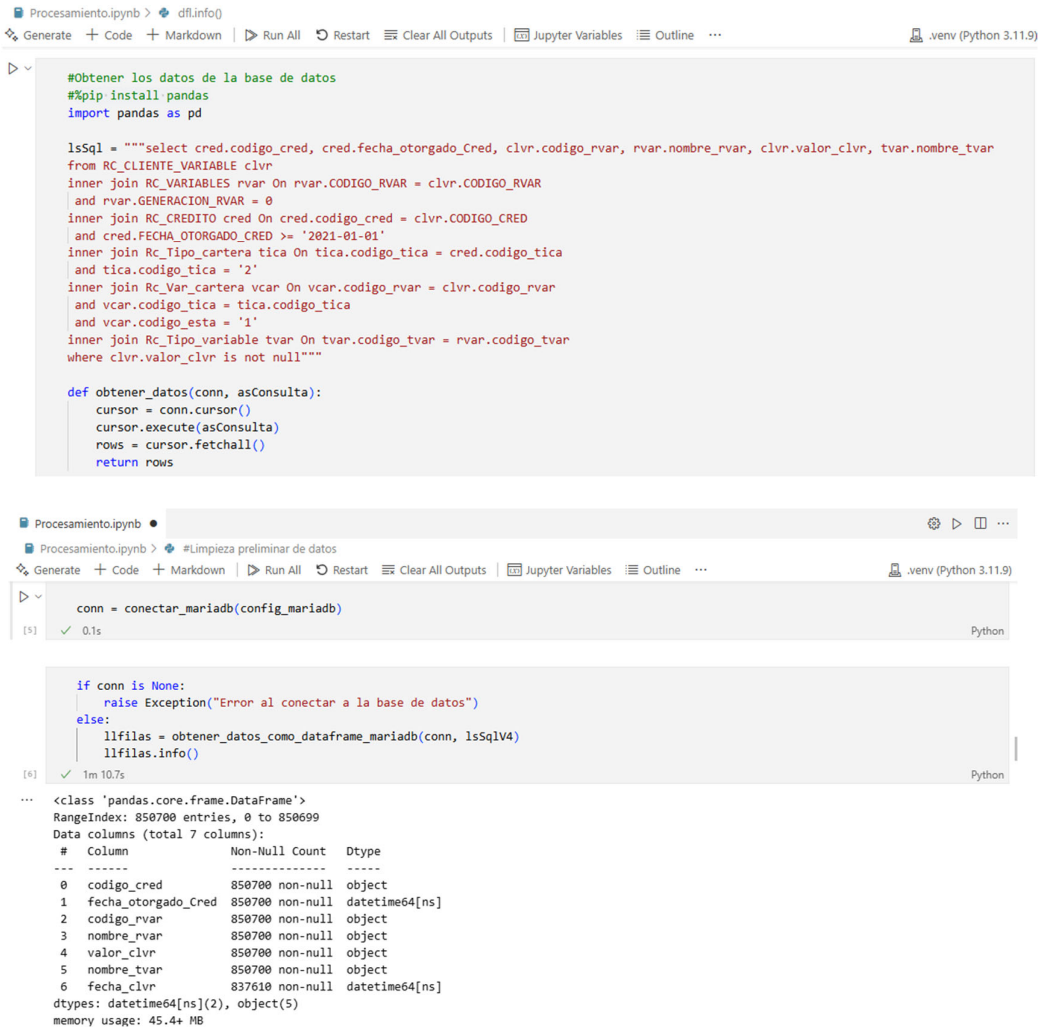


Figura 6 Seleccionar datos de la Base de datos

- **Limpieza de Datos. –** La limpieza de datos es el siguiente paso dentro del modelamiento. Aquí se da tratamiento a los valores nulos y atípicos, eliminación de columnas irrelevantes para el modelo, eliminación de columnas redundantes, así como también podría realizarse una transformación para variables categóricas o el escalado o normalización de variables numéricas. En la Figura 7 se observa un informe descriptivo de las variables numéricas del conjunto de datos.

Haciendo un análisis exploratorio de las variables se encuentra que hay variables que no aportan valor al modelo porque son constantes (Figura 8).

```
# Análisis exploratorio de variables
varNumericas = df.select_dtypes(include=["float64", "Int64"])
varCategoricas = df.select_dtypes(include=["object"])
print("Análisis exploratorio de variables")
print("Variables numéricas:")
print(varNumericas.describe())
print("Variables categóricas:")
```

✓ 0.1s

```
Análisis exploratorio de variables
Variables numéricas:
nombre_rvar  AMORTIZADOCREDITO  Activos Totales  Antiguedad Laboral  \
count             11420.000000     1.142000e+04             11420.0
mean                 35.051437     4.514528e+04           10.362434
std                  29.100429     5.937649e+04            8.152172
min                   0.000000     0.000000e+00                 0.0
25%                   8.847500     1.519264e+04                 5.0
50%                  30.975000     3.065000e+04                 9.0
75%                  56.480000     5.800000e+04                13.0
max                  99.990000     1.622000e+06                59.0

nombre_rvar  CUOTANOPAGA3MESES  CUOTANOPAGA6MESES  CUOTAPAGADAVTOTALCREDITO  \
count             11420.000000       11420.000000              11420.000000
mean                  0.095709           0.151401                  1.298589
std                   0.483586           0.827923                  2.506347
min                   0.000000           0.000000                  0.000000
25%                   0.000000           0.000000                  0.405625
```

Figura 7 Análisis estadístico de variables numéricas

```
# Encuentra columnas numéricas constantes
constantes = [col for col in df.select_dtypes(include="number").columns if df[col].nunique() == 1]

print("Variables numéricas constantes encontradas:")
print(constantes)
# Eliminar columnas constantes numéricas
df_reducido = df.drop(columns=constantes)

constantes = [col for col in df.select_dtypes(include="object").columns if df[col].nunique() == 1]
print("Variables categóricas constantes encontradas:")
print(constantes)
df_reducido = df.drop(columns=constantes)
df_reducido.info()
```

✓ 0.1s Python

```
Variables numéricas constantes encontradas:
['Cred Sis Finan corto plazo', 'Cred Sis Finan largo plazo', 'Creditos casas comerciales', 'Creditos prestamistas', 'Cuota Estimada
Buro', 'Mayor monto atrasado buro', 'Mayor plazo atrasado Buro', 'Otros creditos corto plazo', 'Otros creditos largo plazo']
Variables categóricas constantes encontradas:
['ORIGEN', 'Relacion Dependencia', 'Separacion Bienes']
<class 'pandas.core.frame.DataFrame'>
Index: 11420 entries, 442 to 13133
Data columns (total 69 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   codigo_cred                 11420 non-null  object
 1   AMORTIZADOCREDITO           11420 non-null  float64
 2   Actividad Economica         11420 non-null  object
 3   Actividad Secundaria        11420 non-null  object
 4   Activos Totales             11420 non-null  float64
```

Figura 8 Eliminación de constantes en los datos

En el análisis exploratorio se encuentra que efectivamente hay variables con valor constante y se procede a eliminarlas del modelo como lo muestra la Figura 8.

Se procede a convertir el resultado a un formato tabular (Figura 9) no sin antes ejecutar una instrucción para almacenar el valor 0 a variables cuantitativas sin valor recuperado desde la base de datos.

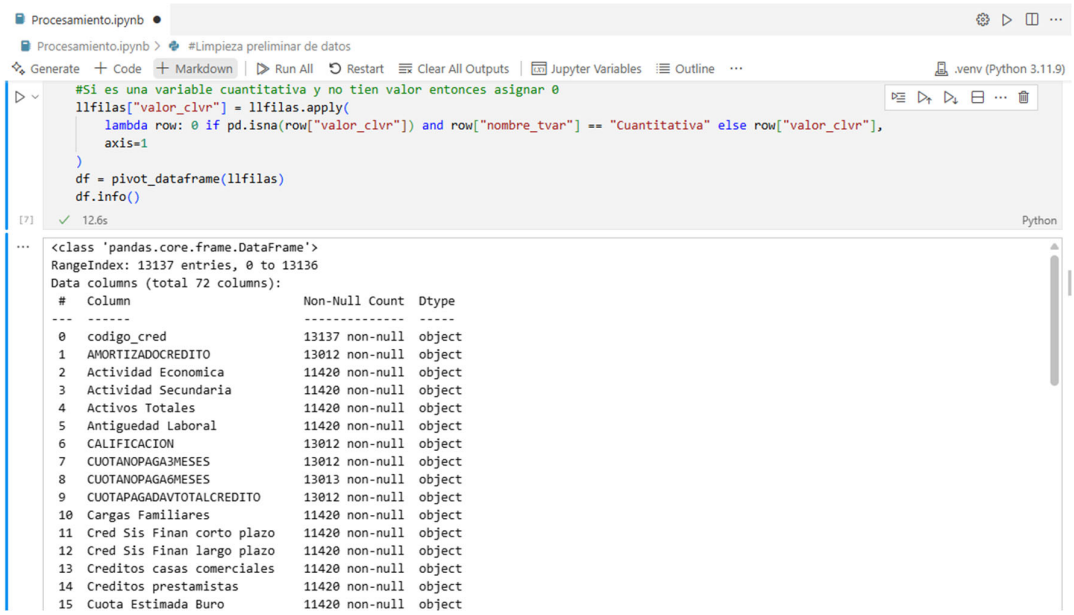


Figura 9 Ejecución de Pivot sobre datos seleccionados

Como se puede apreciar en la Figura 9, para cada fila del listado de columnas hay la cantidad datos no nulos. Dichos valores no son los mismos para todas las columnas, lo cual indica que hay valores faltantes en ciertos casos, por lo que se procede a realizar una limpieza e imputación de datos. Cabe mencionar que en la sección 2.6.1 se indicó que el Sistema HcRisk calificó, según las políticas, a 13,090 operaciones de crédito. Sin embargo, en la Figura 8 se observa que 1,620 operaciones no tienen todas las variables requeridas para modelar. En consecuencia, se limpia de *dataframe* para eliminar casos que no tienen el set de variables completo, como se muestra en la Figura 10. Por último, en la Figura 11 se aprecia el script para verificar que no hay valores faltantes.

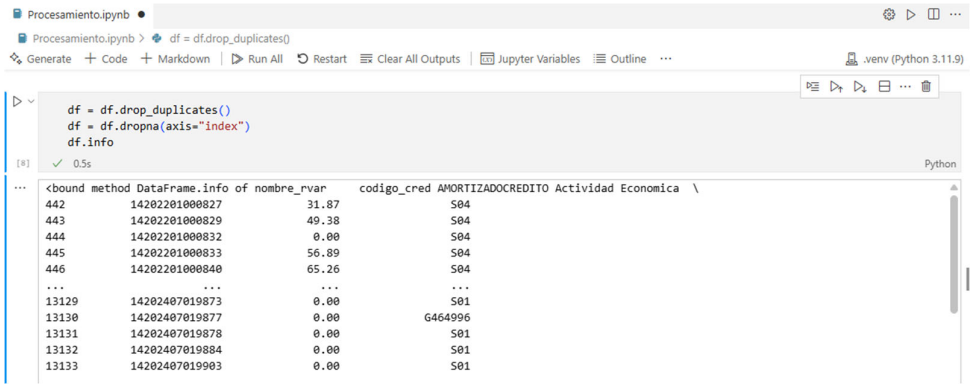


Figura 10 Eliminación de operaciones sin set de variables completo

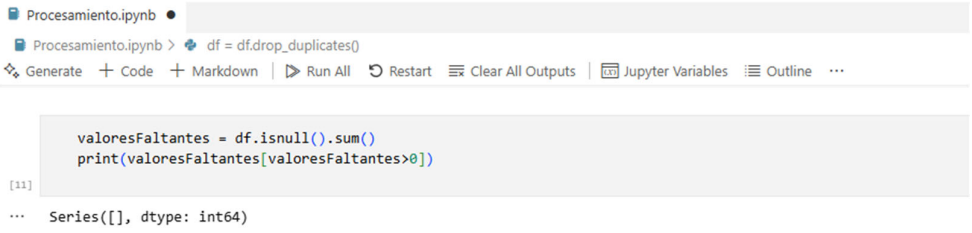


Figura 11 Validación de valores faltantes

En la Figura 11, se observa que ya no existen filas sin valores nulos y se procede a cambiar el tipo de datos por defecto de las columnas del *dataframe* a los tipos de dato correspondiente a cada variable según su naturaleza (Figura 12). En la Figura 13 se evidencia que ahora el conjunto de datos tiene columnas con un tipo de dato definido correctamente.

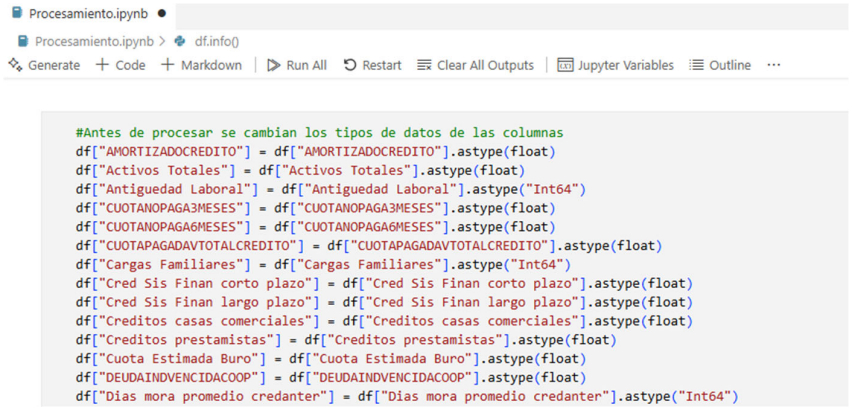


Figura 12 Cambio de tipo de variable correspondiente a su naturaleza

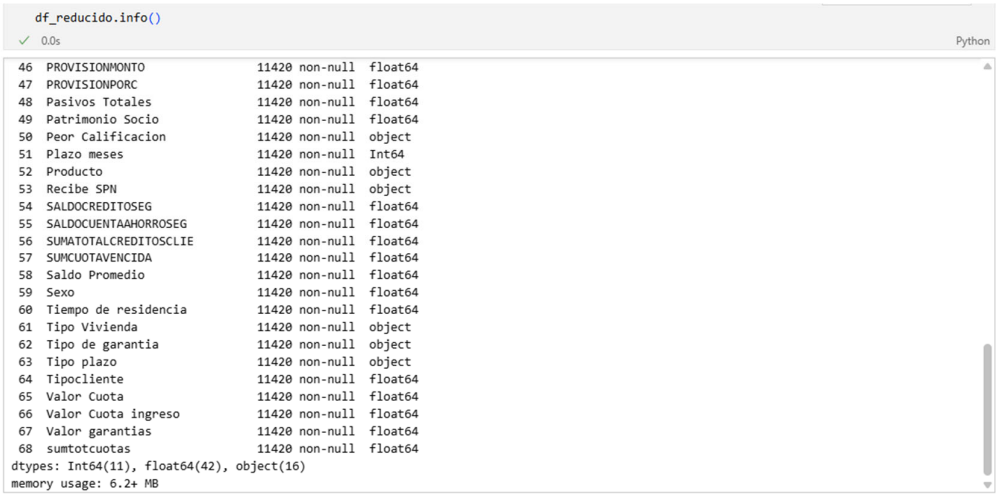


Figura 13 Verificación de cambio de tipo de dato

Posteriormente se realiza un análisis estadístico de los datos y se encuentra que hay algunas variables que tienen un valor constante que no van a ser de utilidad dentro de los modelos de *machine learning.* La Figura 14 muestra el código empleado para eliminar las variables con valor constante.

```python
# Encuentra columnas numéricas constantes
constantes = [col for col in df.select_dtypes(include="number").columns if df[col].nunique() == 1]

print("Variables numéricas constantes encontradas:")
print(constantes)
# Eliminar columnas constantes numéricas
df_reducido = df.drop(columns=constantes)

constantes = [col for col in df.select_dtypes(include="object").columns if df[col].nunique() == 1]
print("Variables categóricas constantes encontradas:")
print(constantes)
df_reducido = df.drop(columns=constantes)
df_reducido.info()
```

0.1s Python

```
Variables numéricas constantes encontradas:
['Cred Sis Finan corto plazo', 'Cred Sis Finan largo plazo', 'Creditos casas comerciales', 'Creditos prestamistas', 'Cuota Estimada Buro', 'Mayor monto atrasado buro', 'Mayor plazo atrasado Buro', 'Otros creditos corto plazo', 'Otros creditos largo plazo']
Variables categóricas constantes encontradas:
['ORIGEN', 'Relacion Dependencia', 'Separacion Bienes']
```

Figura 14 Eliminación de valores constantes

Posteriormente se realiza un análisis de correlación entre las variables y se obtiene la Figura 15

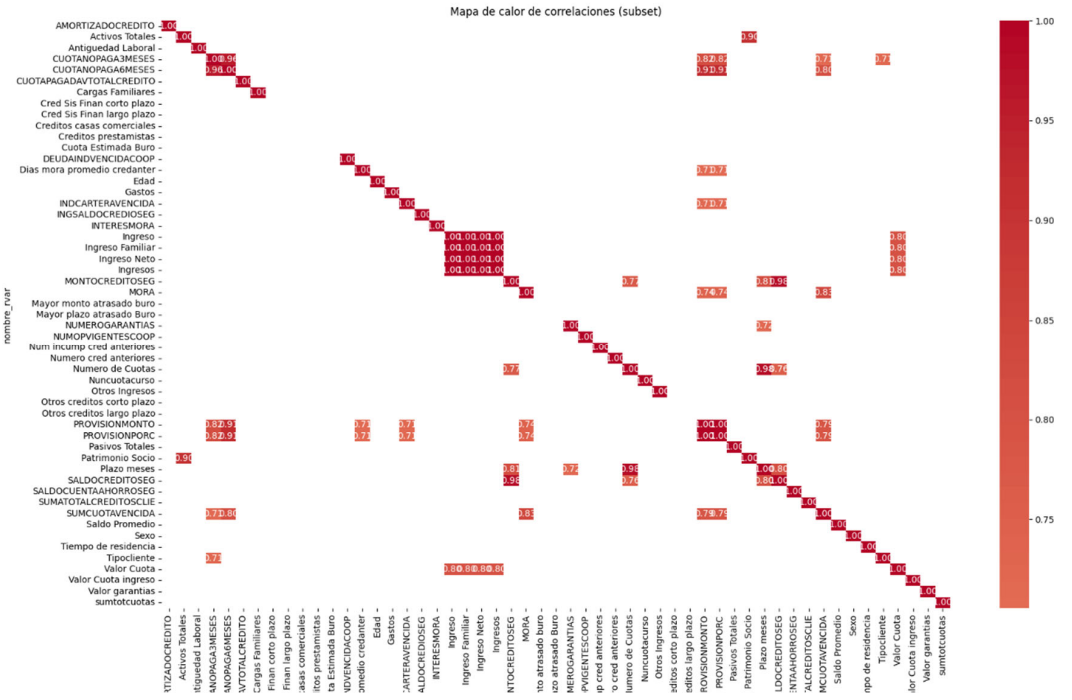


Figura 15 Correlación de variables

De este análisis se proceden a eliminar las variables con una correlación mayor a 0.9 como se muestra en la Figura 16.

Figura 16 Eliminación de variables con correlación alta

```
colsConCorrelacionAlta = set()
corrLimite = 0.9
#Ciclo para iterar sobre la matriz de correlación y encontrar pares de variables con alta correlación
for i in range(len(corr.columns)):
    for j in range(i):
        if abs(corr.iloc[i, j]) > corrLimite:
            colname = corr.columns[i]
            colsConCorrelacionAlta.add(colname) #Agregar la segunda columna del par a eliminar

print("Columnas con alta correlación para eliminar:", colsConCorrelacionAlta)
#Eliminar variables con correlación alta
df_reducido = df_reducido.drop(columns=list(colsConCorrelacionAlta))
print("DataFrame original:", df.shape)
print("DataFrame reducido:", df_reducido.shape)
```

✓ 0.0s Python

```
Columnas con alta correlación para eliminar: {'CUOTANOPAGA6MESES', 'Ingreso Familiar', 'Ingresos', 'Plazo meses', 'SALDOCREDITOSEG',
'Patrimonio Socio', 'Ingreso Neto', 'PROVISIONPORC', 'PROVISIONMONTO'}
DataFrame original: (11420, 72)
DataFrame reducido: (11420, 60)
```

De esta forma resulta que existen 60 variables disponibles para el modelamiento con 11,420 casos reales.

Transformación de Variables. – En el paso de transformación de variable se prepara un pipeline donde se aplicarán de técnicas de codificación (*One-Hot Encoding, Label Encoding*) y también emplearán de técnicas de escalamiento e imputación (StandardScaler, SimpleImputer). Este pipeline será utilizado más tarde, durante la ejecución de los modelos. Además, en la Figura 17 se observa que se elimina a la variable objetivo del conjunto de datos.

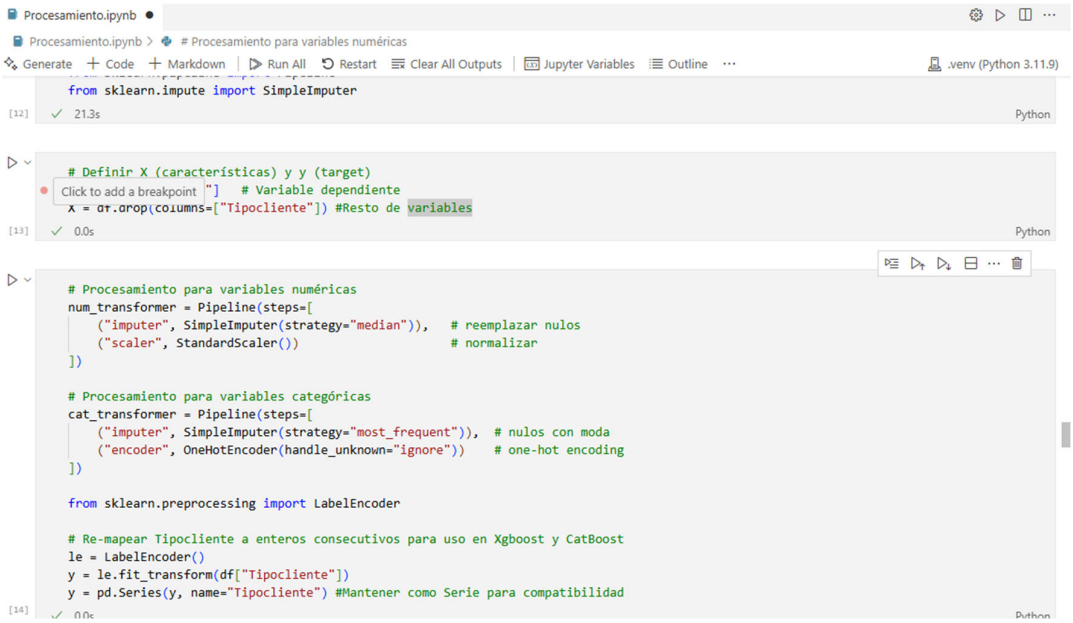
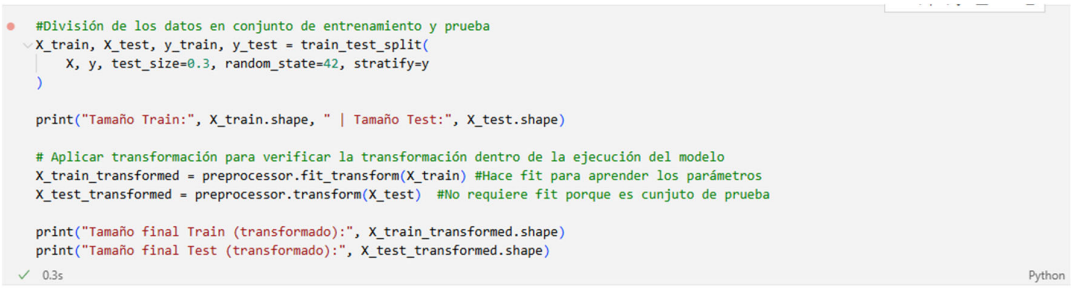


Figura 17 Aplicación de transformaciones

En la Figura 17 también se observa la codificación de la variable dependiente u objetivo (Tipocliente) para que adquiera una forma de 1 o 0 de tal forma que sea compatible con los métodos de *machine learning*. El valor Bueno se transforma a 0 y Malo se transforma a 1.

- **División de Datos. –** El paso división de datos trata de separar una parte del conjunto de datos para que sea destinada al entrenamiento del modelo y la otra parte será destinada a las pruebas.

Para este proyecto se dividirán los datos en conjuntos de entrenamiento y validación en una proporción 70 - 30 (70% para datos de entrenamiento -30% para datos de prueba). En la Figura 18 se verifica que para el Conjunto de datos de entrenamiento hay 7,994 elementos y para el conjunto de datos de pruebas hay 3,426 elementos. Sin embargo, la primera fila del resultado sólo tiene las 68 variables originales, mientras que las siguientes dos filas siguientes muestran 8611 columnas resultantes de la transformación.



```python
#División de los datos en conjunto de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y
)

print("Tamaño Train:", X_train.shape, " | Tamaño Test:", X_test.shape)

# Aplicar transformación para verificar la transformación dentro de la ejecución del modelo
X_train_transformed = preprocessor.fit_transform(X_train) #Hace fit para aprender los parámetros
X_test_transformed = preprocessor.transform(X_test)  #No requiere fit porque es cunjuto de prueba

print("Tamaño final Train (transformado):", X_train_transformed.shape)
print("Tamaño final Test (transformado):", X_test_transformed.shape)
```

0.3s Python

```
Tamaño Train: (7994, 59)  | Tamaño Test: (3426, 59)
Tamaño final Train (transformado): (7994, 8602)
Tamaño final Test (transformado): (3426, 8602)
```

Figura 18 División del conjunto de datos: Entrenamiento y validación

4.2.4.2 Entrenamiento y evaluación del Modelo

Uso de bibliotecas de *machine learning* para entrenar modelos como Random Forest, XGBoost y CatBoost. En la Figura 19 se puede observar la definición de los modelos que se entrenarán y se evaluará su efectividad con la validación cruzada.

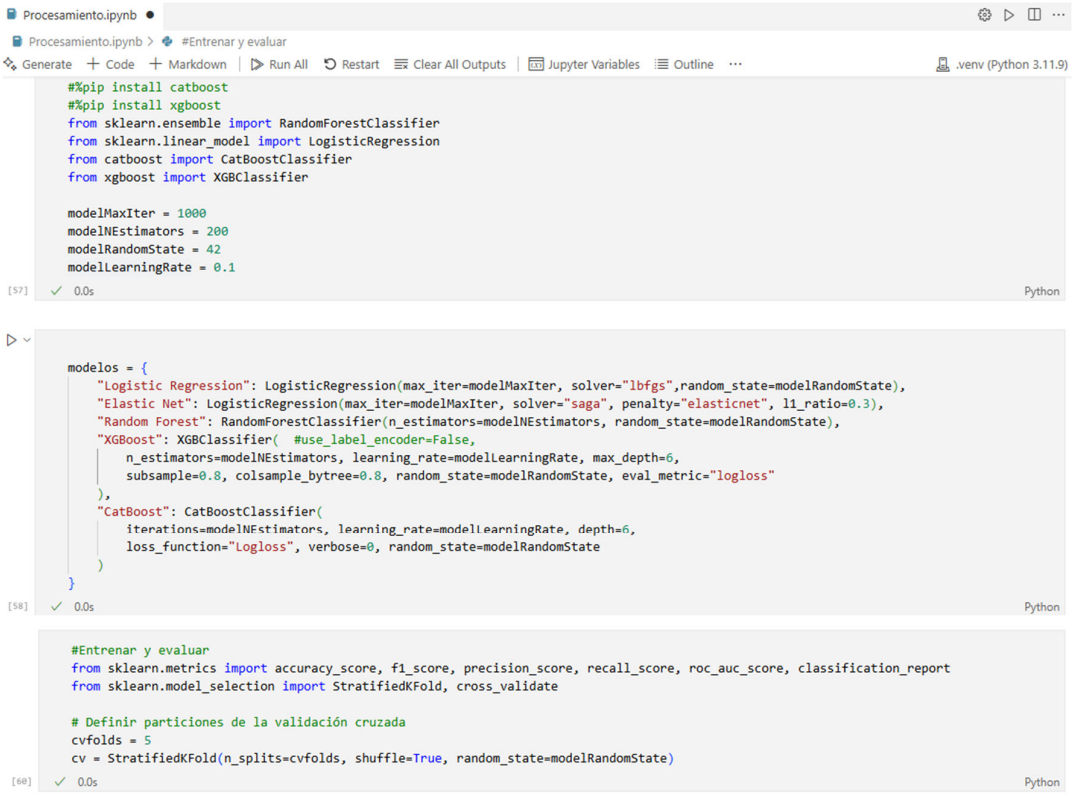


Figura 19 Definición de modelos para entrenamiento

En la Figura 19 tenemos el ciclo de código de Python para poder ejecutar cada modelo y efectuar una evaluación. Sin embargo, para poder comprender los datos de la Figura anterior se explicarán los conceptos alrededor de las métricas que se evalúan en el bucle (Documentación Librería Sklearn Python).

Accuracy. – Es una métrica común usada para evaluar el desempeño de los modelos de clasificación. Mide la proporción de instancias correctamente clasificadas (Positivas y Negativas) entre el total de instancias (Falsas y Verdaderas). Es una métrica de calidad para modelos genéricos, pero no muy confiable en conjuntos de datos desbalanceados.

F1 *Score*. - Es una métrica que se usa para evaluar el desempeño de un modelo de clasificación. Es particularmente útil cuando se trata con conjuntos de datos desbalanceados. Representa la media armónica de la *precision* y *recall.* La Ecuación 4 muestra su definición.

$$F1 = \frac{2 * (precision * recall)}{precision * recall}$$

Ecuación 4 F1 Score

Recall. – Llamada también sensibilidad, es la tasa de las predicciones verdadero positivas (TP) contra el número total de instancias positivas reales (tanto los verdadero positivos como los falsos negativos, TP + FN). Un hipotético valor cercano a 1 indicaría que el modelo está prediciendo con éxito. Resulta buen indicador en conjuntos de datos desbalanceados.

Precision. – Es la tasa de las predicciones verdadero positivas (TP) contra el número total de predicciones positivas (tanto los verdaderos positivos como los falsos positivos, TP + FP). La precisión mejora a medida que disminuyen los falsos positivos, mientras que la recuperación mejora cuando disminuyen los falsos negativos.

Roc_Auc.- ROC es la tasa de verdaderos positivos contra la tasa de falsos positivos ejecutados en todos los umbrales posibles. AUC representa la probabilidad de que el modelo eligiendo un

resultado al azar, clasifique un positivo más alto que el negativo. Un modelo perfecto indicaría un AUC cercano a 1.

La Figura 20 muestra la matriz de confusión que permite comprender gráficamente el significado de los parámetros de la Ecuación 4.

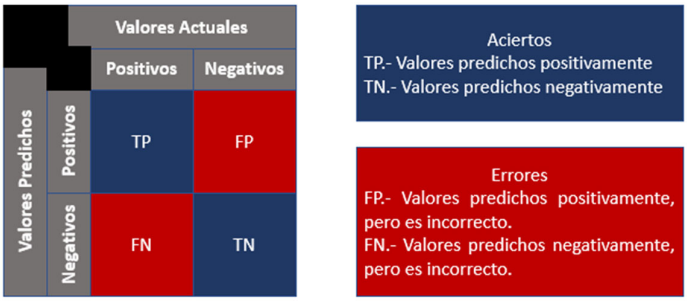


Figura 20 Matriz de confusión

```python
#Variable para alojar los resultados de las métricas
resultados = {}

for nombre, modelo in modelos.items():
    clf = Pipeline(steps=[("preprocessor", preprocessor),
                          ("classifier", modelo)])

    print(f"\n Entrenando modelo: {nombre}...")
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    y_proba = clf.predict_proba(X_test)[:,1] if hasattr(clf, "predict_proba") else None

    # Métricas
    acc = accuracy_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred, average="weighted")
    prec = precision_score(y_test, y_pred, average="weighted")
    rec = recall_score(y_test, y_pred, average="weighted")
    auc = roc_auc_score(y_test, y_proba) if y_proba is not None and len(y.unique()) == 2 else None

    resultados[nombre] = {
        "Accuracy": acc,
        "F1": f1,
        "Precision": prec,
        "Recall": rec,
        "ROC-AUC": auc
    }

    print(" Reporte en conjunto de pruebas:")
    print(classification_report(y_test, y_pred))

    #  Validación cruzada
    scoring = {
        "accuracy": "accuracy",
        "f1": "f1_weighted",
        "precision": "precision_weighted",
        "recall": "recall_weighted",
        "roc_auc": "roc_auc" if len(y.unique()) == 2 else "accuracy"
    }

    scores = cross_validate(clf, X, y, cv=cv, scoring=scoring, n_jobs=-1)

    # Promedio de los folds
    mean_scores = {metric: scores[f"test_{metric}"].mean() for metric in scoring.keys()}

    resultados[nombre].update({f"{m} (cv)": v for m, v in mean_scores.items()})

    print(f" Resultados Validación Cruzada (promedio {cvfolds} folds):")
    for m, v in mean_scores.items():
        print(f"  {m}: {v:.4f}")
```

✓ 4m 44.5s Python

Figura 21 Ciclo de ejecución y valoración de modelos

Una vez ejecutado el proceso se mostrarán resultados similares a los de la Figura23, además de una matriz de confusión que resultará de gran ayuda para verificar los resultados que se descartan o que se aprueban.

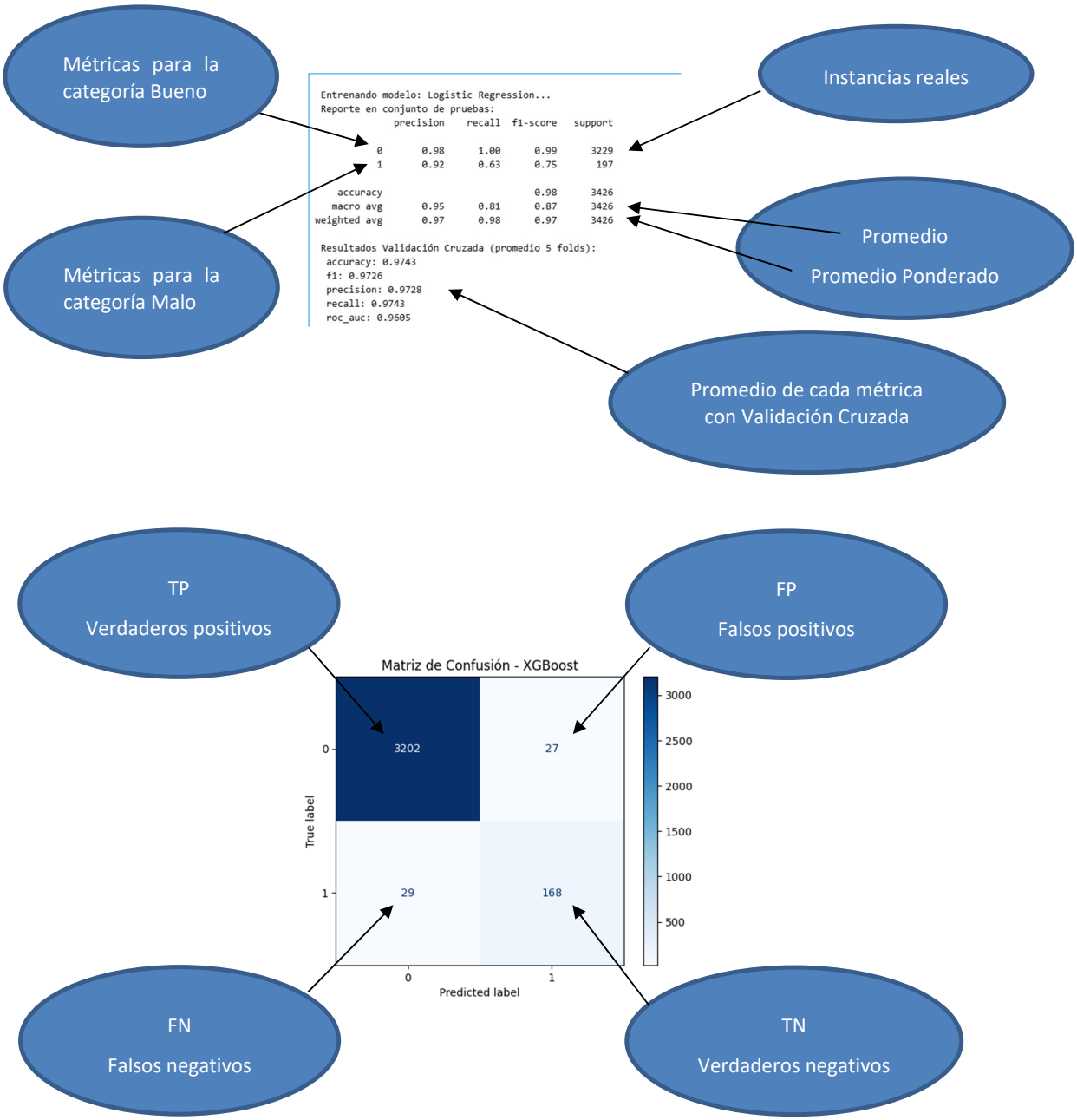


Figura 22 Explicación del resultado de ejecución de modelos

4.2.4.3 EJECUCION DE MODELOS

```
Entrenando modelo: Logistic Regression...
Reporte en conjunto de pruebas:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      3229
           1       0.92      0.63      0.75       197

    accuracy                           0.98      3426
   macro avg       0.95      0.81      0.87      3426
weighted avg       0.97      0.98      0.97      3426

Resultados Validación Cruzada (promedio 5 folds):
 accuracy: 0.9743
 f1: 0.9726
 precision: 0.9728
 recall: 0.9743
 roc_auc: 0.9605
```

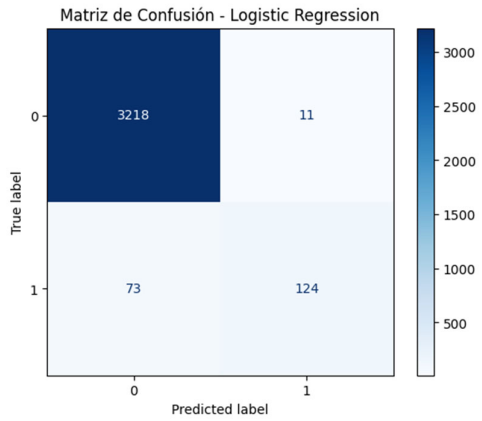


Figura 23 Resultado Ejecución Logistic Regression

De acuerdo a la Figura 23 se puede ver una sensibilidad de 0.81 de promedio, sin embargo, para la clase 1 tiene un 0.63 que coincide con lo indicado por la matriz de confusión en donde se ve

el cuadrante FN que se dejan pasar 73 casos que son malos clientes. y niega 11 casos que son buenos clientes.

```
 Entrenando modelo: Elastic Net...
c:\Xware\Python\TFMhcrisk\.venv\Lib\site-packages\sklearn\linear_
means the coef_ did not converge
  warnings.warn(
 Reporte en conjunto de pruebas:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      3229
           1       0.92      0.62      0.74       197

    accuracy                           0.98      3426
   macro avg       0.95      0.81      0.87      3426
weighted avg       0.97      0.98      0.97      3426

 Resultados Validación Cruzada (promedio 5 folds):
  accuracy: 0.9750
  f1: 0.9734
  precision: 0.9736
  recall: 0.9750
  roc_auc: 0.9626
```

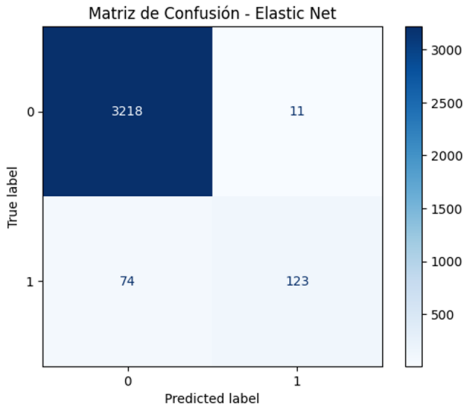


Figura 24 Resultado Ejecución Elastic Net

De acuerdo a la Figura 24 se puede ver una sensibilidad de 0.81 de promedio, sin embargo, para la clase 1 tiene un 0.62 que coincide con lo indicado por la matriz de confusión en donde se ve el cuadrante FN que se dejan pasar 74 casos que son malos clientes y niega 11 casos que son buenos clientes.

```
Entrenando modelo: Random Forest...
Reporte en conjunto de pruebas:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      3229
           1       0.95      0.64      0.77       197

    accuracy                           0.98      3426
   macro avg       0.97      0.82      0.88      3426
weighted avg       0.98      0.98      0.98      3426

Resultados Validación Cruzada (promedio 5 folds):
 accuracy: 0.9787
 f1: 0.9771
 precision: 0.9780
 recall: 0.9787
 roc_auc: 0.9873
```

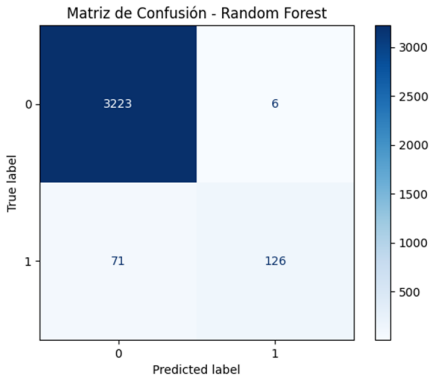


Figura 25 Resultado Ejecución Random Forest

De acuerdo a la Figura 25 se puede ver una sensibilidad de 0.82 de promedio, sin embargo, para la clase 1 tiene un 0.64 que coincide con lo indicado por la matriz de confusión en donde se ve el cuadrante FN que se dejan pasar 71 casos que son malos clientes y niega 6 casos que son buenos clientes. Aquí se nota una mejoría en la evaluación de los casos.

```
Entrenando modelo: XGBoost...
Reporte en conjunto de pruebas:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      3229
           1       0.90      0.83      0.86       197

    accuracy                           0.98      3426
   macro avg       0.95      0.91      0.93      3426
weighted avg       0.98      0.98      0.98      3426

Resultados Validación Cruzada (promedio 5 folds):
 accuracy: 0.9850
 f1: 0.9848
 precision: 0.9847
 recall: 0.9850
 roc_auc: 0.9950
```

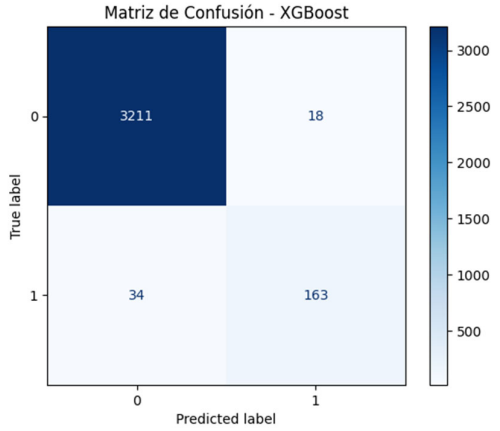


Figura26 Resultado Ejecución XGBoost

De acuerdo a la Figura 26 se puede ver una sensibilidad de 0.91 de promedio, sin embargo, para la clase 1 tiene un 0.83, lo cual mejora los resultados anteriores y esto se respalda en que en la matriz de confusión en donde se ve el cuadrante FN que se dejan pasar 34 casos que son malos clientes y niega 18 casos que son buenos clientes. Aquí se nota una mejoría en la valoración de Falsos Negativos.

```
Entrenando modelo: CatBoost...
Reporte en conjunto de pruebas:
              precision    recall  f1-score   support

           0       0.99      1.00      0.99      3229
           1       0.94      0.76      0.84       197

    accuracy                           0.98      3426
   macro avg       0.96      0.88      0.92      3426
weighted avg       0.98      0.98      0.98      3426

Resultados Validación Cruzada (promedio 5 folds):
 accuracy: 0.9847
 f1: 0.9842
 precision: 0.9842
 recall: 0.9847
 roc auc: 0.9942
```

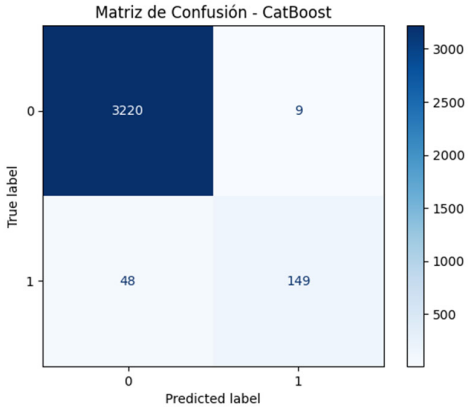


Figura 27 Resultado Ejecución CatBoost

De acuerdo a la Figura 27 se puede ver una sensibilidad de 0.88 de promedio, sin embargo, para la clase 1 tiene un 0.76, lo cual es mejor que el resultado anterior (XGBoost) y esto se respalda en que en la matriz de confusión en donde se ve el cuadrante FN que se dejan pasar 48 casos que son malos clientes y niega 9 casos que son buenos clientes.

Uniendo los resultados de todos los modelos se observa la Tabla de la Figura 28

Comparación de modelos:

	Accuracy	F1	Precision	Recall	ROC-AUC
Logistic Regression	0.975482	0.973309	0.974408	0.975482	0.954642
Elastic Net	0.975190	0.972948	0.974094	0.975190	0.957597
Random Forest	0.977525	0.975417	0.977071	0.977525	0.984508
XGBoost	0.984822	0.984519	0.984406	0.984822	0.995373
CatBoost	0.983363	0.982499	0.982881	0.983363	0.993739

	accuracy (cv)	f1 (cv)	precision (cv)	recall (cv)	roc_auc (cv)
Logistic Regression	0.974343	0.972594	0.972791	0.974343	0.960479
Elastic Net	0.975044	0.973364	0.973628	0.975044	0.962626
Random Forest	0.978722	0.977135	0.977956	0.978722	0.987289
XGBoost	0.985026	0.984787	0.984697	0.985026	0.994967
CatBoost	0.984676	0.984163	0.984212	0.984676	0.994219

Figura 28 Comparativa de los modelos probados

Con esas premisas se puede concluir que el modelo con mejor desempeño es XGBoost, sin perjudicar en mayor medida a CatBoost.

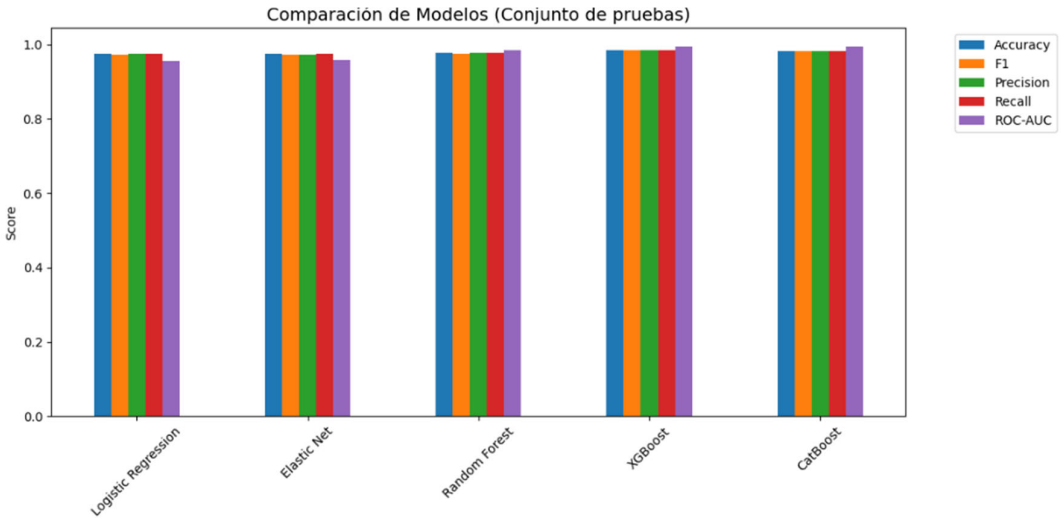


Figura 29 Comparación de métricas de los modelos

La Figura 29 tiene una representación en barras de los valores de las métricas por cada modelo. Y se aprecia que el mejor ROC_AUC lo tienen los modelos XGBoost y CatBoost.

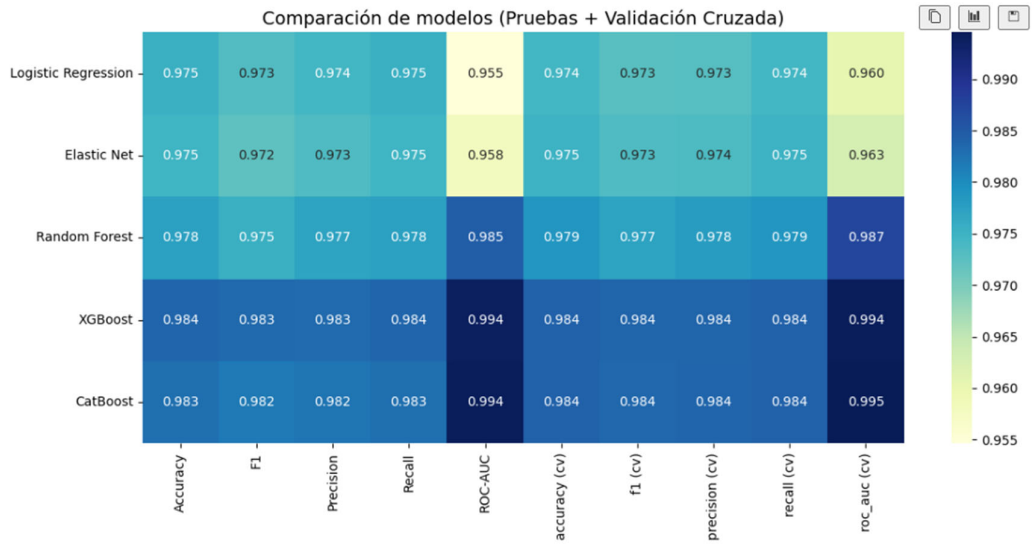


Figura 30 Comparación de Validación Cruzada de modelos

La Figura 30 muestra un mapa de calor con la validación cruzada indicando que el modelo XGBoost tiene el mejor puntaje. La Figura 29 muestra el área bajo la curva de cada modelo respaldando lo dicho anteriormente.

La Figura 31 muestra el gráfico del área bajo la curva y se aprecia que tanto CatBoost como XGBoost son los modelos con mayor área.

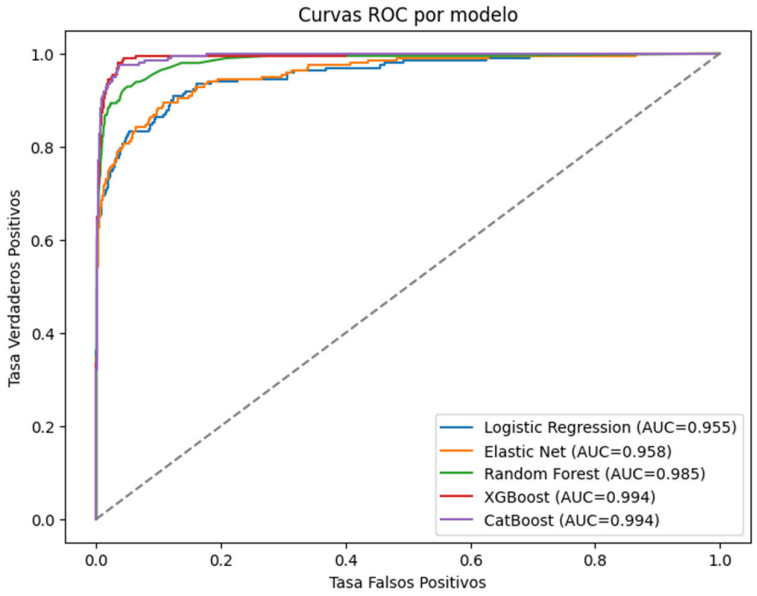


Figura 31 Curva AUC de los modelos

Balanceo de datos con Oversampling

Como se mencionó en un capítulo anterior, los datos se encuentran desbalanceados por tanto se aplicará SMOTE para balancear los datos y verificar lo concluido anteriormente ejecutando un nuevo script de Python (Figura 32).

```
#Oversampling
#Dado que la variable objetivo está desbalanceada se aplica oversampling para balancear las clases
from imblearn.over_sampling import SMOTE
smote = SMOTE(random_state=modelRandomState)
X_resampled, y_resampled = smote.fit_resample(X_train_transformed, y_train)

print("Distribución original de clases en y_train:", y_train.value_counts(normalize=True))
print("Distribución después de SMOTE:", pd.Series(y_resampled).value_counts(normalize=True))
```

✓ 0.1s

```
Distribución original de clases en y_train: Tipocliente
0    0.942332
1    0.057668
Name: proportion, dtype: float64
Distribución después de SMOTE: Tipocliente
0    0.5
1    0.5
Name: proportion, dtype: float64
```

Figura 32 Ejecución Smote – Oversampling

A continuación, se muestran las Figuras correspondientes a la ejecución de un script que crea un *pipeline* similar al *pipeline* inicial, pero añade SMOTE para aplicar *Over-sampling* al conjunto de datos para aumentar los casos de la clase minoritaria.

```
Entrenando modelo Logistic Regression con SMOTE ...
              precision    recall  f1-score   support

           0       0.98      0.98      0.98      3229
           1       0.68      0.73      0.70       197

    accuracy                           0.96      3426
   macro avg       0.83      0.85      0.84      3426
weighted avg       0.97      0.96      0.97      3426

 Resultados Validación Cruzada (promedio 5 folds):
  accuracy: 0.9651
  f1: 0.9666
  precision: 0.9690
  recall: 0.9651
  roc_auc: 0.9596
```

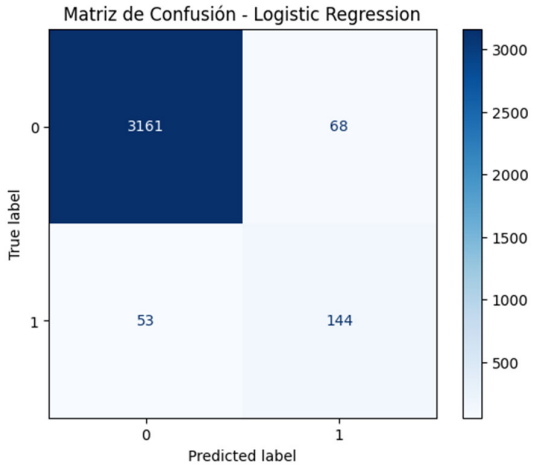


Figura 33 Entrenamiento SMOTE - Logistic Regression

De acuerdo a la Figura 33 se puede ver una sensibilidad de 0.85 de promedio, sin embargo, para la clase 1 tiene un 0.73 que coincide con lo indicado por la matriz de confusión en donde se ve que el cuadrante FN se dejan pasar 53 casos que son malos clientes. y niega 68 casos que son buenos clientes. Estos datos convergen en una mejora del modelo con respecto a su versión inicial pero no lo suficiente.

```
Entrenando modelo Elastic Net con SMOTE ...
  warnings.warn(
c:\Xware\Python\TFMhcrisk\.venv\Lib\site-packages\skl
means the coef_ did not converge
  warnings.warn(
              precision    recall  f1-score   support

           0       0.99      0.97      0.98      3229
           1       0.59      0.79      0.68       197

    accuracy                           0.96      3426
   macro avg       0.79      0.88      0.83      3426
weighted avg       0.96      0.96      0.96      3426

 Resultados Validación Cruzada (promedio 5 folds):
  accuracy: 0.9563
  f1: 0.9599
  precision: 0.9663
  recall: 0.9563
```

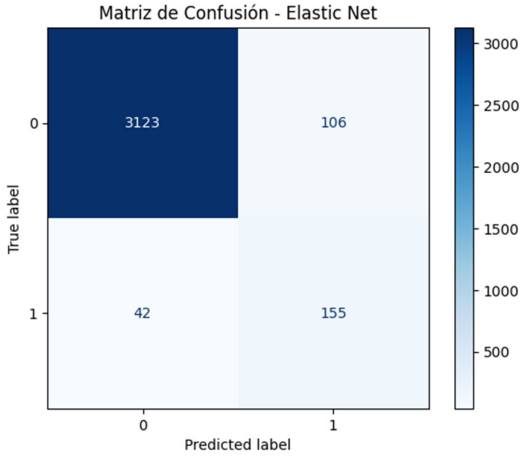


Figura 34 Entrenamiento SMOTE - Elastic Net

De acuerdo a la Figura 34 se puede ver una sensibilidad de 0.88 de promedio, sin embargo, para la clase 1 tiene un 0.79. En cambio, la matriz de confusión se ve que el cuadrante FN se dejan pasar 42 casos que son malos clientes y niega 106 casos que son buenos clientes. Estos datos convergen en una mejora del modelo con respecto a su versión inicial pero no lo suficiente.

```
Entrenando modelo Random Forest con SMOTE ...
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      3229
           1       0.90      0.72      0.80       197

    accuracy                           0.98      3426
   macro avg       0.94      0.86      0.89      3426
weighted avg       0.98      0.98      0.98      3426

 Resultados Validación Cruzada (promedio 5 folds):
  accuracy: 0.9806
  f1: 0.9798
  precision: 0.9798
  recall: 0.9806
  roc_auc: 0.9897
```

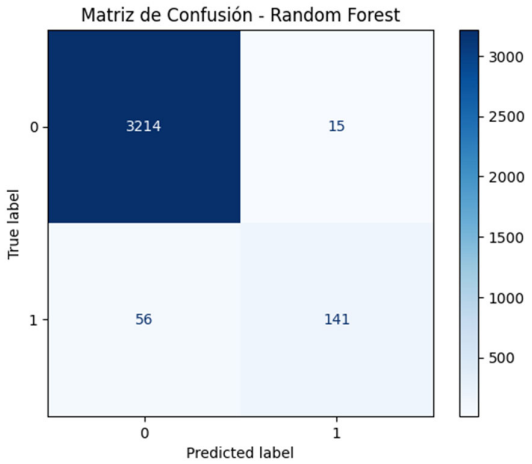


Figura 35 Entrenamiento SMOTE - Random Forest

De acuerdo a la Figura 35 se puede ver una sensibilidad de 0.86 de promedio, sin embargo, para la clase 1 tiene un 0.72 que coincide con lo indicado por la matriz de confusión en donde se ve el cuadrante FN que se dejan pasar 56 casos que son malos clientes y niega 15 casos que son buenos clientes. Aquí se nota una mejoría en la evaluación de los casos contra la versión anterior del mismo modelo.

```
Entrenando modelo XGBoost con SMOTE ...
  bst.update(dtrain, iteration=i, fobj=obj)
c:\Xware\Python\TFMhcrisk\.venv\Lib\site-packages\xgbc
runner\_work\xgboost\xgboost\src\learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      3229
           1       0.86      0.85      0.86       197

    accuracy                           0.98      3426
   macro avg       0.93      0.92      0.92      3426
weighted avg       0.98      0.98      0.98      3426

 Resultados Validación Cruzada (promedio 5 folds):
  accuracy: 0.9842
  f1: 0.9842
  precision: 0.9843
  recall: 0.9842
  roc auc: 0.9940
```

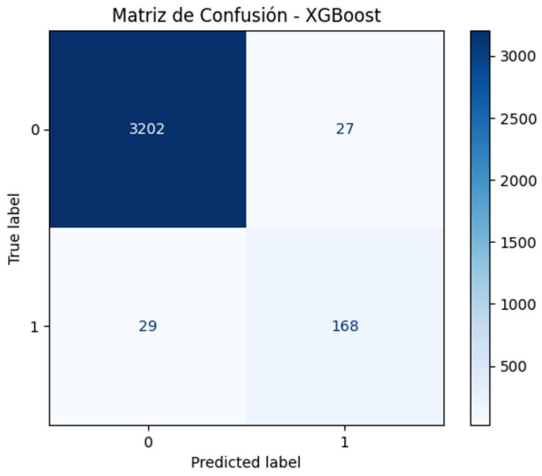


Figura 36 Entrenamiento SMOTE - XGBoost

De acuerdo a la Figura 36 se puede ver una sensibilidad de 0.92 de promedio, sin embargo, para la clase 1 tiene un 0.85, lo cual mejora los resultados contra la versión anterior del modelo. En la matriz de confusión se dejan pasar 29 casos que son malos clientes y niega 27 casos que son buenos clientes. Aquí se nota que el área bajo la curva es mayor al resto de modelos.

```
Entrenando modelo CatBoost con SMOTE ...
              precision    recall  f1-score   support

           0       1.00      0.99      0.99      3229
           1       0.80      0.93      0.86       197

    accuracy                           0.98      3426
   macro avg       0.90      0.96      0.92      3426
weighted avg       0.98      0.98      0.98      3426

Resultados Validación Cruzada (promedio 5 folds):
 accuracy: 0.9792
 f1: 0.9802
 precision: 0.9822
 recall: 0.9792
 roc_auc: 0.9951
```

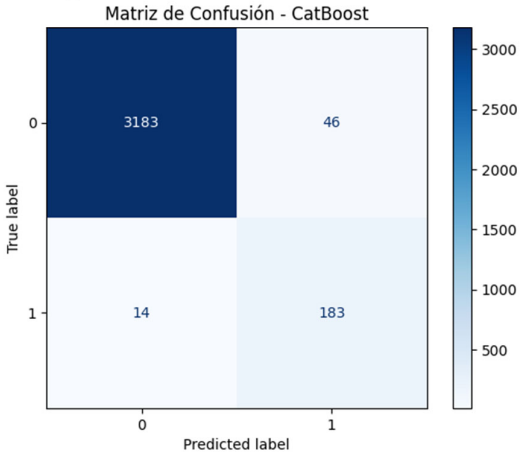


Figura 37 Entrenamiento SMOTE - CatBoost

De acuerdo a la Figura 37 se puede ver una sensibilidad de 0.96 de promedio, sin embargo, para la clase 1 tiene un 0.93, lo cual mejora los resultados contra todos los modelos anteriores. En la matriz de confusión se dejan pasar 14 casos que son malos clientes y niega 46 casos que son buenos clientes. Aquí se nota que el área bajo la curva es mucho mayor al resto de modelos. Por lo tanto, el modelo CatBoost tiene mejores puntajes que el resto de modelos.

Comparación final de modelos con SMOTE

	accuracy	f1	precision	recall	roc_auc
Logistic Regression	0.965149	0.966627	0.968969	0.965149	0.959560
Elastic Net	0.956305	0.959868	0.966335	0.956305	0.961951
Random Forest	0.980560	0.979842	0.979761	0.980560	0.989697
XGBoost	0.984238	0.984244	0.984316	0.984238	0.993994
CatBoost	0.979247	0.980208	0.982239	0.979247	0.995068

Figura 38 Comparativa de los modelos probados con SMOTE

En la Figura 36 se puede observar que, si bien el *accuracy* de CatBoost no es el mejor, el indicador roc_auc es el mayor de todos indicando que la proporción de aciertos es mejor y esto es confirmado por la matriz de confusión como se indicó en la observación anterior respectiva. Negar 46 casos es menos costoso para la institución que dejar aceptar 16 casos de malos pagadores, finalmente a esos 46 casos se los podría someter a una validación posterior ya sea por un comité o por alguna otra Tabla que considere ciertos patrones que lo califiquen como apto o no apto para un crédito.

Continuando con la Explicabilidad del modelo, se implementa SHAP al proceso.

```
from catboost import CatBoostClassifier

print("SHAP para CatBoost\n")
clf = modelos['CatBoost']
clf.fit(X_test_transformed, y_test)

explainer_xgb = shap.TreeExplainer(clf)
shap_values_xgb = explainer_xgb.shap_values(X_test_transformed)

print("Importancia de características (CatBoost):")
shap.summary_plot(shap_values_xgb, X_test_transformed, feature_names=feature_names )
```

✓ 17.1s

```
SHAP para CatBoost

Importancia de características (CatBoost):
```

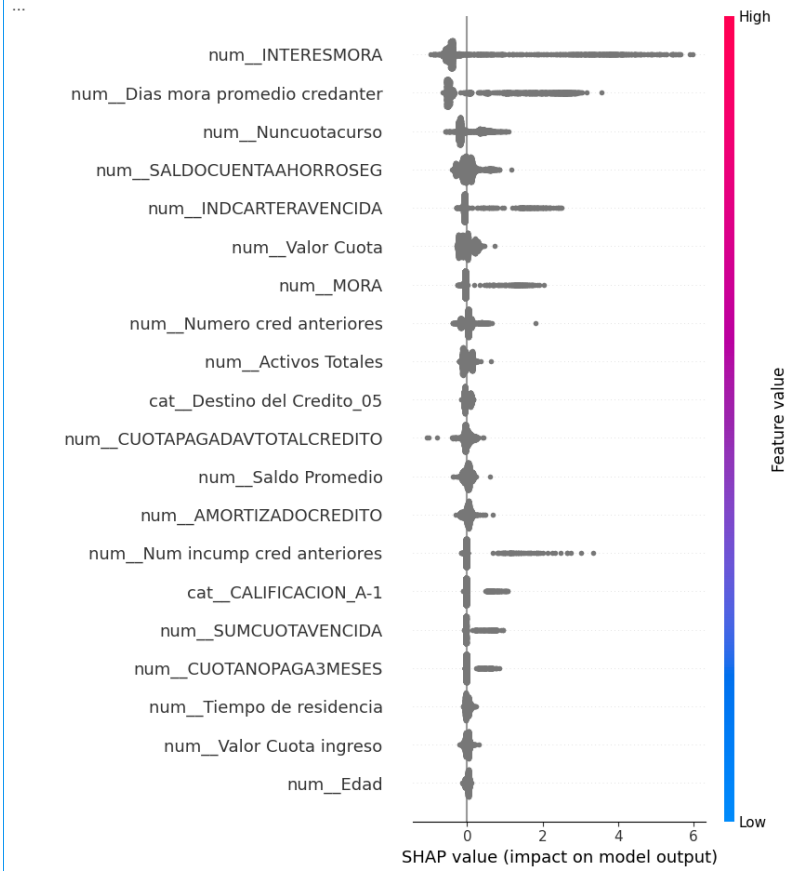


Figura 39 Explicabilidad SHAP para CatBoost

De la Figura 39 se observa que las variables que se mencionan son las que participan activamente en la clasificación de los clientes como buenos o malos pagadores.

Finalmente se prueba un Conjunto de datos de 3 operaciones con datos aleatorios para verificar el modelo CatBoost.

```
clf = Pipeline(steps=[("preprocessor", preprocessor),
                      ("classifier", modelos['CatBoost'])])
y_pred = clf.predict(df_scoring)
print("Predicciones en el set de scoring:")
print(y_pred)
```

✓ 0.1s Open 'y_pred' in Data Wrangler

```
Predicciones en el set de scoring:
[0 0 0]
```

Figura 40 Predicción del modelo CatBoost

4.2.5 Integración *Frontend – Backend*

4.2.5.1 Comunicación entre Java y Python

- Se utiliza una API RESTful desarrollada en Python (con Flask) que expone *endpoints* para:
  - Cargar datos desde MariaDB.
  - Ejecutar el modelo predictivo.
  - Devolver resultados procesados al *frontend*.
- Desde Java (*backend*), se realizan llamadas HTTP (GET/POST) a estos *endpoints*, pasando los datos necesarios como JSON o CSV.

4.2.5.2 Ejemplo de flujo de trabajo:

1. Un usuario ingresa datos de un nuevo cliente en una pantalla HTML desarrollada en Java.
2. Al hacer clic en "Evaluar", se envían los datos al *backend* Java.
3. Java llama a un *endpoint* de la API REST desarrollada en Python.
4. Python ejecuta el modelo, devuelve la probabilidad de incumplimiento y la clasificación (buen o mal pagador).
5. Java recibe la respuesta y muestra los resultados en una nueva pantalla.

4.2.6 Consideraciones sobre Seguridad y Regulación

- **Confidencialidad de datos:** Uso de conexiones seguras (HTTPS), cifrado de datos sensibles y control de acceso basado en roles.
- **Auditoría:** Registro de todas las ejecuciones del modelo, incluyendo parámetros, métricas y usuarios involucrados.
- **Cumplimiento regulatorio:** Alineación con Acuerdos de Basilea y normativas locales sobre riesgo crediticio y transparencia de modelos.

4.2.7 Futuras Mejoras y Extensibilidad

- **Automatización de reentrenamiento:** Implementación de ciclos periódicos de actualización del modelo según cambios en patrones de datos.
- **Escalabilidad:** Uso de contenedores Docker y orquestación con Kubernetes para manejar mayor volumen de peticiones.
- **Integración con sistemas externos:** Conexión con plataformas de monitoreo de riesgos, CRM, o sistemas de gestión de carteras.
- **Expansión a otros segmentos de crédito:** Adaptación del modelo a créditos comerciales, hipotecarios o PYMEs.

4.3 Recursos requeridos

Recursos Humanos

- Ingeniero de Datos, programador
- Experto ML

Recursos Tecnológicos

- Computadora AMD EPYC, 8GB RAM, 500GB Disco duro -> Servidor de base de datos
- Computadora AMD EPYC, 8GB RAM, 500GB Disco duro -> Servidor de aplicaciones
- Computadora Intel I5, 16GB RAM, 1TB Disco duro -> Desarrollo de modelos y programación
- Infraestructura de red LAN/Internet: router 4 puertos Gigabit ethernet

- Software Propietario Windows server, Windows 10
- Software de libre uso: Java, Python, MariaDB, MySql Workbench, Eclipse, PyCharm, Visual Studio Code

4.4 Presupuesto

Tipo de coste	**Valor**	**Comentarios**
Horas de trabajo en el proyecto	240	Ingeniero de datos, programador
	40	Experto ML
Equipo técnico utilizado	746€	2 servidores virtuales Computadora AMD EPYC, 8GB RAM, 500GB Disco duro. Plan anual.
	315€	Computadora Intel I5, 16GB RAM, 1TB Disco duro -> Desarrollo de modelos y programación
	270€	Infraestructura de red LAN/Internet: *router* 4 puertos Gigabit ethernet. Plan anual
Software utilizado	0€	Windows Server 2022 Standard
	6.95€	Windows 11Pro
	0€	Java – Lenguaje de programación para interfaz web
	0€	Python – Lenguaje de programación para la el servicio web y modelo ML

	0€	MariaDB – Base de datos
	0€	MySql Workbench – Interfaz para administrar y consultar la base de datos MariaDB
	0€	IDE Eclipse – Interfaz de desarrollo con Java
	0€	Waitress, JbossWildfly – Servidores de aplicaciones web
	0€	PyCharm – Interfaz de Desarrollo del servicio web en Python
	0€	Visual Studio Code – Interfaz de desarrollo del servicio web en Python para pruebas del modelo de ML

4.5 Viabilidad

El proyecto es totalmente viable ya que no representa una inversión económica muy alta y el tiempo de desarrollo es relativamente corto. Por otro lado, el beneficio que se obtendrá será bastante alto y tiene grandes posibilidades de que se expanda para realizar otros modelos como de seguimiento y control.

.

4.6 Resultados del proyecto

Dado que los modelos empleados en el análisis comparativo tuvieron unos resultados bastante buenos, es interesante su aplicabilidad en el entorno empresarial. En cuanto a su implementación como reemplazo del modelo actual en R, es 100% recomendable ya

que sus etapas de planificación, ejecución y evaluación se han realizado de forma ágil. Facilita mucho la implementación mucho la ventaja de que Python permite generar APIs que ejecutan lo requerido para las tareas de Preprocesamiento, Procesamiento, Verificación de datos y Puesta en marcha del mejor modelo.

5 DISCUSIÓN

6 CONCLUSIONES

6.1 Conclusiones del trabajo

El modelo actual de regresión logística Logit, si bien tiene un buen desempeño, su implementación en R y en Power Builder significan un retraso tanto a nivel tecnológico como en tiempo de implementación en nuevas instituciones bancarias. El presente proyecto demuestra que la integración de modelos de aprendizaje automático a los sistemas financieros es totalmente viable y de gran utilidad que deviene en ahorro de tiempo y por ende ahorro de dinero.

Se seleccionaron 4 modelos para realizar la comparación y pruebas de su desempeño identificando que XGBoost tuvo el mejor puntaje en la primera versión del modelado, pero aplicando over-sampling con SHAP el modelo CatBoost tuvo el mejor desempeño. Los modelos basados en árboles permiten una mayor explicabilidad del resultado de las clasificaciones y son bastante rápidos de ejecutar.

La matriz de confusión es una herramienta que ha resultado de mucha utilidad para poder determinar la viabilidad y selección de un modelo frente a otro, ya que permite verificar objetivamente los casos que el modelo pasa por alto y los que acierta correctamente.

Se realizó la implementación de los modelos seleccionados de una forma modular para garantizar su reproducibilidad en cualquier entorno y para su adaptación en los APIs de integración con la interfaz web.

6.2 Conclusiones personales

A través de este proyecto he podido profundizar mucho más en el aprendizaje automático, desarrollo de APIs en Python, integración de software con Java y también sobre el desarrollo de documentos técnicos.

Ha sido una experiencia muy enriquecedora que me motiva para continuar adquiriendo más conocimientos sobre Aprendizaje automático y sobre el lenguaje Python, y que gracias a su alta

versatilidad y a su gran cantidad de librerías destinadas a muy variados tópicos, me resulta muy divertido el trabajar.

7 FUTURAS LÍNEAS DE TRABAJO

Se identifican diversas oportunidades de mejora y expansión que pueden ser consideradas como futuras líneas de investigación y desarrollo:

7.1.1 Mejora en la calidad de los datos y depuración automatizada

Se podría realizar la implementación de *pipelines* de preprocesamiento automatizado que identifiquen y corrijan *outliers*, datos faltantes, errores de codificación y duplicados.

También se podrían implementar rutinas de auditoría y trazabilidad de datos para fortalecer la confiabilidad del modelo en entornos regulados.

Sería de gran utilidad la aplicación de técnicas de *feature engineering* dinámico, con actualización periódica de variables derivadas según el comportamiento real de los clientes.

7.1.2 Ampliación de indicadores de comportamiento y nuevas fuentes de datos

Resulta interesante la integración de indicadores adicionales de ejecución financiera y comportamiento transaccional, como recurrencia de pagos, uso de canales digitales, o interacción con productos financieros complementarios.

Se podría mejorar el proyecto con la inclusión de datos alternativos o no tradicionales (*open banking*, redes sociales, comportamiento digital) siempre que el marco legal y ético lo permita.

Con lo anterior se implementaría la evaluación del uso de datos temporales o series de tiempo para modelos secuenciales de comportamiento de los clientes.

7.1.3 Ciclos de reentrenamiento y validación continua del modelo

La normativa financiera indica el tiempo sobre el cual los modelos deben ser evaluados pero el establecimiento de una política de actualización periódica del modelo en función del cambio en

los patrones de datos resulta interesante para hacer dinámica la adaptación del proyecto a la realidad cambiante.

También se podría incorporar procedimientos de aprendizaje online o incremental para adaptar el modelo sin necesidad de reentrenar desde cero (Modelo de clasificación por seguimiento).

7.1.4 Aplicación a otros productos o segmentos de crédito

La adaptación del modelo a otras categorías o productos como crédito comercial, microcrédito, o crédito hipotecario resulta interesante.

Se podría implementar una segmentación de modelos por tipo de cliente (personas naturales vs jurídicas, nuevos vs antiguos).

Por último se podría experimentar con modelos multiclasificadores que incluyan no solo “buen o mal pagador” sino niveles de riesgo que le den al usuario una ventana sobre la cual evaluar la concesión de un crédito.

8 REFERENCIAS

Fuentes del Estado del Arte del Aprendizaje Automático

1. Breiman, L. (2001). *"Random Forests." Machine Learning*. Este trabajo seminal introduce el algoritmo *Random Forest* y demuestra su superioridad frente a modelos lineales tradicionales para tareas de clasificación y regresión.
2. Prokhorenkova, L. et al. (2018). "*CatBoost: unbiased boosting with categorical features*." NeurIPS. Introduce CatBoost y demuestra su rendimiento superior en *datasets* con variables categóricas, como los típicos en el análisis crediticio.
3. Chen, T., & Guestrin, C. (2016). "*XGBoost: A Scalable Tree Boosting System*." KDD. Este *paper* presenta XGBoost, el cual ha demostrado rendimiento superior en múltiples competencias de clasificación.
4. Cortes, C., & Vapnik, V. (1995). *Support-vector networks. Machine Learning*, 20(3), 273–297. *Paper* fundador de SVM, establece las bases matemáticas y de generalización.
5. Xia, Y., Liu, C., Li, Y., & Liu, N. (2017). *A boosted decision tree approach using Bayesian hyper-parameter optimization for credit scoring. Expert Systems with Applications*, 78, 225–241. Comparación entre SVM y otros modelos en clasificación binaria financiera.
6. Hornik, K., Stinchcombe, M., & White, H. (1989). *Multilayer feedforward networks are universal approximators*. *Neural Networks*, 2(5), 359–366. Demuestra teóricamente que las MLP pueden aproximar cualquier función continua.
7. Yeh, I.-C., & Lien, C.-H. (2009). The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. Expert Systems with Applications, 36(2), 2473–2480. Estudio comparativo en scoring crediticio con redes neuronales.
8. Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why Should I Trust You?" Explaining the Predictions of Any Classifier. KDD. Describe LIME, una herramienta de explicabilidad local útil para explicar predicciones individuales en modelos complejos.
9. Lundberg, S. M., & Lee, S.-I. (2017). "A Unified Approach to Interpreting Model Predictions." NeurIPS. Este artículo establece las bases de SHAP, una técnica interpretativa esencial para entornos regulados como el financiero.

10. Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 67(2), 301–320. Introducción del Elastic Net como combinación de L1 y L2.
11. Dhanesh Ramachandram , et al, 2025, *Transparent AI: The Case for Interpretability and Explainability* .- https://arxiv-org.translate.goog/html/2507.23535v1?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc
12. Shijie Wang, Xueyong Zhang, 2025, *Credit Rating Model Based on Improved TabNet* .- https://www.mdpi.com/2227-7390/13/9/1473
13. Zöller, M.-A., & Huber, M. F. (2021). *Benchmark and Survey of Automated Machine Learning Frameworks. Journal of Artificial Intelligence Research*, 70, 409–472. Comparación empírica entre frameworks (TPOT, Auto-sklearn, H2O).
14. **Hastie, Tibshirani & Friedman (2009)** – The Elements of Statistical Learning
    - URL: https://web.stanford.edu/~hastie/ElemStatLearn/
15. **Goodfellow, Bengio & Courville (2016)** – Deep Learning
    - URL: https://www.deeplearningbook.org/
16. **Scikit-learn Documentation**
    - URL: https://scikit-learn.org/stable/documentation.html
17. **XGBoost, LightGBM y CatBoost official docs**
    - https://xgboost.readthedocs.io/
    - https://lightgbm.readthedocs.io/
    - https://catboost.ai/docs/
18. **(Medium)** Artículos de divulgación técnica sobre comparativas de modelos y pipelines modernos.
    - https://medium.com/@data.science.enthusiast/machine-learning-model-selection-c571324f31e
    - https://medium.com/accelerated-analyst/when-to-use-logistic-regression-vs-deep-learning-for-business-problems-589359d49600
    - https://www.peerbits.com/blog/choose-right-machine-learning-algorithm-guide.html

- https://machinelearningmastery.com/practical-guide-choosing-right-algorithm-your-problem/

Fuentes del Acuerdo de Basilea (I, II, III)

1. **Banco de Pagos Internacionales (BIS) – Basel Committee on Banking Supervision**
   - Principal organismo emisor de los Acuerdos de Basilea.
   - URL: https://www.bis.org/bcbs/
2. **Documento original de Basilea I (1988)**
   - International Convergence of Capital Measurement and Capital Standards
   - URL: https://www.bis.org/publ/bcbsc111.htm
3. **Basilea II: Framework (2004)**
   - International Convergence of Capital Measurement and Capital Standards: A Revised Framework
   - URL: https://www.bis.org/publ/bcbs107.htm
4. **Basilea III: *Reforms* (2010-2017)**
   - Basel III: A global regulatory framework for more resilient banks and banking systems
   - URL: https://www.bis.org/bcbs/basel3.htm
5. **European Banking Authority (EBA) y Financial Stability Board (FSB)**
   - Complementos regulatorios sobre implementación y evaluación del cumplimiento.
   - https://www.eba.europa.eu
   - https://www.fsb.org
6. **Libros especializados**:
   - **Hull, John C.** – Risk Management and Financial Institutions
     - Fuente clave sobre métodos estándar de riesgo de crédito y modelos regulatorios.
7. **Material universitario y bancario**:
   - Manuales del **Banco Central Europeo**, Banco de España y documentos académicos en cursos de gestión de riesgos financieros.

8. Modelo Logit
    - https://economipedia.com/definiciones/modelo-logit.html
    - https://e-archivo.uc3m.es/rest/api/core/bitstreams/3e6d3870-a72d-4a1e-8748-8b053fffc2b6/content

9 ANEXOS

Sirven para incluir documentación complementaria (planos, circuitos, código, ficheros de conFiguración, especificaciones técnicas y hojas de características, fichas explicativas, resultado de encuestas, reglamentación y normativas requeridas, etc.).

[PÁGINA INTENCIONADAMENTE EN BLANCO]