UNIVERSIDAD TORCUATO DI TELLA

MASTER IN MANAGEMENT + ANALYTICS

ESCUELA DE NEGOCIOS

# Exploración y explotación en el análisis de riesgo de crédito

ALUMNO: ALDO ORLANDO ESCOBAR

DIRECTOR: PABLO ROCCATAGLIATA

AÑO: 2020

# EXPLORACIÓN Y EXPLOTACIÓN EN EL ANÁLISIS DE RIESGO DE CRÉDITO

## RESUMEN

El presente trabajo propone dos experimentos para automatizar el proceso de asignación de créditos bancarios utilizando un conjunto de datos desbalanceado con respecto al target. En el primer experimento se propone utilizar un algoritmo escalable y modular, **NGBoost**, para clasificar las observaciones en buenos y malos pagadores en base a una función objetivo dada por una matriz de costos (problema de tipo cost-sensitive). Una vez entrenado el modelo se realizarán predicciones y se efectuarán análisis de feature importance para decidir qué features ocultar para el posterior (segundo) ejercicio. Usando este conjunto de datos modificados, se analizará la performance en términos de beneficios de la compañía, con respecto al uso del scoring crediticio standard. Finalmente, se generará una nueva regla para asignar el crédito, considerando el scoring crediticio y la necesidad de dar crédito para poder aprender de los usuarios.

Palabras clave: scoring crediticio, machine learning, natural gradient boosting, problema de tipo cost-sensitive, thresholding, feature importance.

# EXPLORATION AND EXPLOITATION IN CREDIT RISK ANALYSIS

## ABSTRACT

The present work proposes two experiments to automate the process of bank credit allocation using an unbalanced data set. The first experiment uses an implementation of a scalable and modular algorithm, **NGBoost**, to classify observations as either good and bad payers based on an objective function given a cost matrix (cost-sensitive problem). Once the model has been trained a feature importance analysis will be carried out to decide which features to hide for the following (second) exercise. Using this modified data set, performance will then be studied, in terms of the company's benefits, with regard to the use of a standard credit scoring. Finally, a new rule will be generated to allocate credit, considering credit scoring and the need to give credit in order to learn from users.

Keywords: credit scoring, machine learning, natural gradient boosting, cost-sensitive problem, thresholding, feature importance.

# AGRADECIMIENTOS

En primer lugar, me gustaría agradecer a mi tutor, Mg. Pablo Roccatagliata, por su continuo apoyo durante mi estudio e investigación, por su motivación y entusiasmo, que fueron claves para darle vida a este presente trabajo.

Mi profundo agradecimiento hacia mis compañeros del Master in Management + Analytics (MiM) de la Universidad Torcuato di Tella, por permitirme compartir y debatir experiencias que no sólo me enriquecieron a hacer lo mejor desde el ámbito profesional y académico, sino también desde lo personal. También me gustaría darles las gracias a mis profesores de la maestría, que me brindaron las herramientas necesarias para que este trabajo fuera posible.

Un agradecimiento especial a mis compañeros de Mutt Data por el constante apoyo, en especial a Santiago Hernández, a Juan Pampliega y a Mateo de Monasterio, por su interés en este trabajo y por tomarse el tiempo en evaluarlo, proponiéndome nuevos desafíos que enriquecieron este contenido.

No puedo terminar sin decir lo agradecido que estoy con mi madre, Liliana Petracca, que siempre me apoyó y me guió a dar lo mejor de mí en todos los asuntos de la vida.

# ÍNDICE

# 1. INTRODUCCIÓN

## 1.1. DOMINIO

Los bancos juegan un papel crucial en las economías de mercado. Deciden quienes pueden obtener financiamiento y en qué términos. Para que funcionen los mercados y la sociedad, los individuos y las empresas necesitan acceso al crédito.

Los algoritmos de calificación crediticia, que estiman la probabilidad de incumplimiento de pago, son el método que utilizan los bancos para determinar si se debe otorgar o no un préstamo.[1]
Para mitigar el impacto del riesgo crediticio y tomar decisiones más objetivas y precisas las instituciones financieras utilizan scoring de crédito para asistir en las decisiones de asignación de crédito. El objetivo de la calificación crediticia es clasificar qué clientes potenciales probablemente incumplirán una obligación financiera que fue otorgada en función al historial financiero del cliente y con esa información, decidir si aprobar o rechazar el préstamo. Esta herramienta se ha convertido en una práctica estándar en la industria financiera para predecir y controlar las carteras de préstamos.[2]

En el análisis de crédito un enfoque tradicional es otorgar productos de acuerdo con un scoring de riesgo estático, es decir, todos los errores de clasificación son igualmente costosos. En muchos casos, se utiliza la métrica AUC que pondera en igual magnitud los errores de clasificación. En la industria del crédito se considera un error más costoso darle crédito a un mal cliente en relación a no darle crédito a un buen cliente. Otorgar dicho crédito a alguien que delinque, ocasionaría un gasto del crédito asignado y, por otro lado, no asignarle un crédito a alguien que es buen pagador produciría perder ese interés por el monto crediticio. En ese caso, podríamos modelar un problema binario de clasificación donde nuestra función objetivo podría ser el accuracy con ponderaciones asimétricas para los distintos tipos de errores de clasificación o bien, considerar una métrica de negocio que contemple los costos de asignación en casos donde el prestatario es o no moroso. Por lo que, entonces, el banco deberá clasificar a sus clientes en base a un criterio a definir mediante una serie de reglas.

[1] *Descripción de la competencia de Kaggle a tratar en el presente trabajo* https://www.kaggle.com/c/GiveMeSomeCredit
[2] *Anderson, R. (2007). The Credit Scoring Toolkit : Theory and Practice for Retail Credit Risk Management and Decision Automation. Oxford University Press.*

Para muchos bancos online que apuntan a un segmento tradicionalmente no bancarizado este enfoque no incorpora de forma adecuada la necesidad de aprender sobre los clientes. Tampoco es satisfactorio para ofrecer productos con límites dinámicos en el tiempo como en las propuestas de valor de firmas como Brex[3][4] o Zero[5]. Adicionalmente estos procesos de scoring generalmente no toman en cuenta el riesgo de las predicciones puntuales en los modelos de scoring, las cuales pueden tener distinto grado de incertidumbre según las características de cada observación. [6]

Entre otras consideraciones, además de la capacidad predictiva, en muchos casos de clasificación, los datos pueden contener implícitamente características sensibles (tales como sexo, origen étnico, orientación sexual, etc) de un individuo. Sería importante garantizar el Fairness del algoritmo utilizado. Por ejemplo, la probabilidad de asignar el crédito en el caso del género, debe ser igual para ambos, es decir, que no debería estar correlacionada al sexo.[7]

## 1.2. PROBLEMA

En este contexto tenemos entonces dos problemas. El primero consiste en mejorar la metodología tradicional de análisis crediticio que utiliza la métrica AUC. Esta métrica evalúa diferentes clasificadores entrenados sobre una misma familia de distribuciones probabilísticas, lo que en algunas situaciones produce resultados inconsistentes, dado que, los costos de los errores de clasificación no deberían ser asimétricos. Este problema es particularmente grave cuando se comparan clasificadores sobre diferentes conjuntos de prueba.[8]

Como segundo problema, tendremos un caso donde la función objetivo del negocio no está alineada con la función objetivo de los modelos de riesgo. Idealmente nos gustaría aprender más sobre los clientes respecto a los cuáles nuestro modelo de scoring presente mayor incertidumbre en sus predicciones. Al mismo tiempo, nos gustaría poder explorar esos casos, dándoles crédito para luego ver cómo resultan en su cumplimiento de pago, lo cual puede resultar costoso.

---

[3] *Artículo sobre límites de crédito dinámicos (Febrero 2020)* https://medium.com/the-asian-edge/is-a-dynamic-credit-limit-the-future-of-credit-cards-4a5263f6961e

[4] *Brochure oficial de Brex* https://brex.com/Hubspot_Brex_Finance_to_Scale.pdf

[5] *Sección FAQ de la página oficial de Zero* https://zero.app/faq

[6] *S. Lessmann, B. Baesens, Hsin-Vonn Seow, Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research (2015)*

[7] *S. Barocas, M. Hardt, A. Narayanan, Fairness and machine learning, Limitations and Opportunities.* https://fairmlbook.org/

[8] *W. Verbeke, B. Baesens, C. Bravo, Profit-Driven Business Analytics (2018), Capítulo 6.*

Este problema presenta entonces algunas similitudes con el multi-armed bandit problem[9]., con la complejidad adicional de que ahora nuestra medida de incertidumbre estará dada por alguna técnica que trate de cuantificar el riesgo en cada una de las predicciones individuales de nuestros modelos de riesgo.

## 1.3. OBJETIVO

La propuesta es utilizar **NGBoost** como algoritmo de predicción probabilística para medir el riesgo de nuestras predicciones y en base a esa medida se definirá una función objetivo con dos componentes:

- Explotación: asignar bien el crédito en base a la información disponible.
- Exploración: asignar crédito a usuarios dónde hay mucha incertidumbre en las predicciones de manera de poder aprender sobre ellos.

## 1.4. OUTPUT ESPERADO DEL PROYECTO

El alcance del output esperado contempla un conjunto de reglas para la toma de decisiones en la asignación de productos de crédito. En el armado de estas reglas se utilizará el output del algoritmo **NGBoost**, que consiste en una distribución de probabilidad sobre los labels de la variable target. Esta distribución nos permite cuantificar la incertidumbre de nuestras predicciones.

El tomador de decisión, que corresponde a un responsable del área de asignación de créditos, recibe un conjunto de criterios para determinar cómo será la asignación de estos productos teniendo en cuenta el balance óptimo entre exploración (aprender sobre nuevos clientes) y explotación (darle crédito a clientes que son buenos prospectos, según la probabilidad asignada por el algoritmo). Consideraremos entonces que el banco tiene como política no invertir más de un cierto porcentaje para aplicar el segundo experimento (tomando como referencia el primer experimento), dicha métrica será explicada con mayor profundidad al final del segundo experimento (Para más información respecto a esta métrica: secciones *DEFINIENDO PORCENTAJE DE INVERSIÓN DEL SEGUNDO EXPERIMENTO* y

[9] *Problema en el que un conjunto limitado de recursos debe ser asignado entre las opciones que compiten en una manera que maximiza su ganancia esperada, cuando las propiedades de cada opción se conocen sólo parcialmente en el momento de la asignación, y se pueden entender mejor a medida que pasa el tiempo o al asignar recursos a la opción.*

*EVALUACIÓN DE LA INVERSIÓN* del presente trabajo). Entonces, este balance óptimo estará dado por cuatro criterios:

- Primer criterio: El menor costo (**CostoSegundoExperimento**) calculado en base a las predicciones del algoritmo.
- Segundo criterio: El mayor porcentaje de inversión posible (considerando como techo el porcentaje impuesto como política del banco).
- Tercer criterio: Mayor cantidad de préstamos asignados (y menor costo del segundo experimento)
- Cuarto criterio: Menor costo de aprendizaje (**learning_cost**)

Como detallaremos más adelante en el segundo experimento, contaremos con dos hiper parámetros a la hora de evaluar la inversión del segundo experimento, los cuales son *alpha* y *threshold*. *Alpha* cuantifica de alguna manera el costo que el banco tiene que incurrir sobre esa observación con incertidumbre (definida dentro de una fórmula de costo de aprendizaje). *Threshold* se refiere al umbral sobre el cual el algoritmo de clasificación recomendará si conviene otorgar o no el crédito.
A modo de ejemplo, para el mejor modelo de segundo experimento, se realizarán cómputos para los distintos alphas y thresholds. La salida de este proceso tiene un formato tabular definido bajo este formato:

- alpha: Constante a asignar a cada observación con incertidumbre para determinar el learning_cost.
- threshold: Umbral sobre el cual el algoritmo de clasificación recomendará si conviene otorgar o no el crédito.
- count_zero: Cantidad de préstamos asignados, es decir, aquellas observaciones que fueron etiquetadas por el algoritmo de clasificación como **Responsible (**$\hat{y} = 0$**)**.
- count_one: Cantidad de préstamos no asignados, es decir, aquellas observaciones que fueron etiquetadas por el algoritmo de clasificación como **Delinquent (**$\hat{y} = 1$**)**.
- CostoPrimerExperimento: Costo de aplicar el primer experimento (definido más adelante en la sección *PRIMER EXPERIMENTO*)
- learning_cost: Es el costo total de aprendizaje que contempla las observaciones con incertidumbre (definido más adelante en la sección *SEGUNDO EXPERIMENTO*)
- CostoSegundoExperimento: Es el costo de aplicar el segundo experimento (definido más adelante en la sección *SEGUNDO EXPERIMENTO*).
- %_Inversión: Valor porcentual que mide el porcentaje de inversión de aplicar el segundo experimento por sobre el primer experimento (definido más adelante en la sección *SEGUNDO EXPERIMENTO*).

Entonces, una vez efectuada la corrida de cómputos para alpha y threshold llegamos a los siguientes resultados:

| alpha | threshold | count_zero | count_one | CostoPrimer Experimento | learning_ cost | CostoSegundo Experimento | %_Inversion |
|---|---|---|---|---|---|---|---|
| 200 | 0.4 | 5,000 | 7,000 | 200,000 | 10,000 | 210,000 | 5 |
| 220 | 0.3 | 3,500 | 8,500 | 220,000 | 5,000 | 225,000 | 2.27 |
| 250 | 0.4 | 5,000 | 7,000 | 210,000 | 20,000 | 230,000 | 9.52 |
| 300 | 0.35 | 4,200 | 7,800 | 200,000 | 15,000 | 215,000 | 7.5 |

***Tabla 1.*** *Conjunto de datos de ejemplo que representa los cómputos realizados por el algoritmo NGBoost para ciertos valores de alpha y threshold.*

Resultado obtenido en base al primer criterio:

| alpha | threshold | count_zero | count_one | CostoPrimer Experimento | learning_ cost | CostoSegundo Experimento | %_Inversion |
|---|---|---|---|---|---|---|---|
| 200 | 0.4 | 5,000 | 70,00 | 200,000 | 10,000 | 210,000 | 5 % |

***Tabla 2.*** *Resultado del primer criterio.*

Resultado obtenido en base al segundo criterio:

| alpha | threshold | count_zero | count_one | CostoPrimer Experimento | learning _cost | CostoSegundo Experimento | %_Inversion |
|---|---|---|---|---|---|---|---|
| 220 | 0.3 | 3,500 | 8,500 | 220,000 | 5,000 | 225,000 | 2.27 |

***Tabla 3.*** *Resultado del segundo criterio.*

Resultado obtenido en base al tercer criterio:

| alpha | threshold | count_zero | count_one | CostoPrimer Experimento | learning _cost | CostoSegundo Experimento | %_Inversion |
|---|---|---|---|---|---|---|---|
| 200 | 0.4 | 5,000 | 7,000 | 200,000 | 10,000 | 210,000 | 5 |

***Tabla 4.*** *Resultado del tercer criterio.*

Resultado obtenido en base al cuarto criterio:

| alpha | threshold | count_zero | count_one | CostoPrimer Experimento | learning_ cost | CostoSegundo Experimento | %_Inversion |
|---|---|---|---|---|---|---|---|
| 220 | 0.3 | 3,500 | 8,500 | 220,000 | 5,000 | 225,000 | 2.27 |

***Tabla 5.*** *Resultado del cuarto criterio.*

En base a estos resultados el tomador de decisión deberá optar por uno de estos en base a satisfacer la política impuesta por el banco y al criterio seleccionado.

Entre los beneficios esperados respecto a la versión de tener una matriz de costos respecto a los errores de clasificación versus este mismo enfoque pero con la particularidad de que se considerarán aquellas observaciones con incertidumbre, es de esperar que nuestro algoritmo predictivo contemple muchas más observaciones con el segundo enfoque que con el primero.

# 2. DATOS

La siguiente información demográfica y financiera de 150,000 prestatarios está disponible de manera pública en un conjunto de datos usado para una competencia de Kaggle **GiveMeSomeCredit**[10], realizada en el año 2011.
Los features de cada uno están representados por once variables como se puede observar en la **Tabla 6**. El objetivo de esta competencia era predecir si un cliente defaulteará en los próximos dos años o no, indicado por el feature **SeriousDlqin2yrs**.

## 2.1. DICCIONARIO DE DATOS

El conjunto de datos contiene 150,000 instancias.

| **Nombre de variable** | **Descripción** | **Chequeo de nulos** | **Tipo de dato** |
|---|---|---|---|
| SeriousDlqin2yrs | Persona experimentada con 90 días de morosidad o más. | 150,000 no nulos | 0 (Responsible) / 1 (Delinquent) |
| RevolvingUtilizationOfUnsecuredLines | Saldo total de tarjetas de crédito y líneas de crédito personales, excepto bienes inmuebles y ninguna deuda a plazos, como préstamos para automóviles, dividido por la suma de los límites de crédito. | 150,000 no nulos | porcentaje |
| age | Edad del prestatario en años. | 150,000 no nulos | entero |

[10] https://www.kaggle.com/c/GiveMeSomeCredit/data

| | | | |
|---|---|---|---|
| NumberOfTime30-59DaysPastDueNotWorse | Número de veces que el prestatario ha estado atrasado entre 30 y 59 días, pero no ha empeorado en los últimos 2 años. | 150,000 no nulos | entero |
| DebtRatio | Pagos mensuales de la deuda, pensión alimenticia, costos de vida divididos por el ingreso bruto mensual. | 150,000 no nulos | porcentaje |
| MonthlyIncome | Ingreso Mensual. | 120,269 no nulos | real |
| NumberOfOpenCreditLinesAndLoans | Número de préstamos abiertos (cuotas como préstamos para automóviles o hipotecas) y líneas de crédito (por ejemplo, tarjetas de crédito). | 150,000 no nulos | entero |
| NumberOfTimes90DaysLate | Número de veces que el prestatario ha estado atrasado 90 días o más. | 150,000 no nulos | entero |
| NumberRealEstateLoansOrLines | Número de préstamos hipotecarios e inmobiliarios, incluidas líneas de crédito sobre el valor neto de la vivienda. | 150,000 no nulos | entero |
| NumberOfTime60-89DaysPastDueNotWorse | Número de veces que el prestatario ha vencido 60-89 días, pero no ha empeorado en los últimos 2 años. | 150,000 no nulos | entero |
| NumberOfDependents | Número de dependientes en su familia excluyéndose a sí mismos (cónyuge, hijos, etc.). | 146,076 no nulos | entero |

***Tabla 6.*** *Variables del conjunto de datos a utilizar en el análisis.*

## 2.2 DESCRIPCIÓN DE LOS DATOS[11]

### 2.2.1. SeriousDlqin2yrs

Indica si la persona que toma el crédito experimentó alguna cuota pasada hasta 90 días en los 2 años anteriores. Es dicotómica y en nuestro caso es además, la variable target, por lo que el algoritmo a diseñar predecirá este valor.
Si el valor de este feature es True (o 1), la persona que tomó el préstamo tuvo cuotas con demoras de más de 90 días en los 2 años anteriores, es decir el prestatario no pagó el EMI[12] 90 días después de la fecha de vencimiento del EMI. Si este feature toma el valor False, es el caso contrario.

### 2.2.2 RevolvingUtilizationOfUnsecuredLines

Indica los límites de la tarjeta de crédito del prestatario después de excluir cualquier deuda de préstamo actual y bienes inmuebles.
Por ejemplo, supongamos que tenemos una tarjeta de crédito y su límite de crédito es de $ 1,000. En la cuenta bancaria personal, tenemos $ 1,000. El saldo de nuestra tarjeta de crédito es de $ 500 de $ 1,000. El saldo máximo total será:

$$límite\ tarjeta\ de\ crédito\ +\ saldo\ cuenta\ bancaria\ personal\ =\ \$1{,}000\ +\ \$1{,}000\ =\ \$2{,}000.$$

Al usar $500 del límite de la tarjeta de crédito, el saldo total actual es de:

$$saldo\ \ tarjeta\ de\ crédito\ +\ saldo\ cuenta\ bancaria\ personal\ =\ \$500\ +\ \$1{,}000\ =\ \$1{,}500.$$

Si el titular de la cuenta toma un préstamo y paga el EMI por ese préstamo, entonces no consideramos el valor EMI para el préstamo, por lo que sólo consideraremos el saldo de la tarjeta de crédito del titular de la cuenta y el saldo de la cuenta personal, quedando entonces lo siguiente:

$$RevolvingUtilizationOfUnsecuredLines = \frac{saldo\ total\ actual}{saldo\ máximo\ total} = \frac{\$1{,}500}{\$2{,}000} = 0.75.$$

[11] *J. Thanaki. Machine Learning Solutions: Expert techniques to tackle complex machine learning problems using Python (2018).*
[12] *Equated Monthly Installment (EMI) es un monto de pago fijo realizado por un prestatario a un prestamista en una fecha específica cada mes calendario.*

### 2.2.3. Age

Edad del prestatario.

### 2.2.4. NumberOfTime30-59DaysPastDueNotWorse

Indica el número de veces que los prestatarios han pagado sus EMI con retraso pero los han pagado entre los 30 días y los 59 días después de la fecha de vencimiento.

### 2.2.5. DebtRatio

Métrica que incluye pagos mensuales de la deuda, pensión alimenticia, costos de vida divididos por el ingreso bruto mensual. Está dado por la siguiente fórmula:

$$DebtRatio = \frac{deuda\ mensual + otros\ gastos}{ingreso\ mensual} = \frac{700}{1{,}000} = 0.7$$

### 2.2.6. MonthlyIncome

Es el valor que corresponde a los ingresos mensuales de los prestatarios.

### 2.2.7. NumberOfOpenCreditLinesAndLoans

Indica la cantidad de préstamos abiertos más la cantidad de tarjetas de crédito que tiene el prestatario.

### 2.2.8. NumberOfTimes90DaysLate

Indica cuántas veces un prestatario ha pagado sus cuotas 90 días después de la fecha de vencimiento de sus EMI.

### 2.2.9 NumberRealEstateLoansOrLines

Indica la cantidad de préstamos que el prestatario tiene para sus bienes inmuebles o la cantidad de préstamos para la vivienda que tiene un prestatario.

### 2.2.10. NumberOfTime60-89DaysPastDueNotWorse

Indica cuántas veces los prestatarios han pagado sus EMI con retraso, pero les han pagado entre los 60 días y 89 días después de su fecha de vencimiento.

### 2.2.11. NumberOfDependents

Indica el número de familiares dependientes que tienen los prestatarios, excluyendo a quien toma el préstamo.

# 3. MARCO TEÓRICO

Definiremos la teoría que hay por detrás de las técnicas y algoritmos a implementar en nuestros experimentos, tomando como referencia los papers y bibliografía sobre los cuales se sustentan y explicando brevemente su aplicación en la práctica del presente trabajo.

## 3.1 PRINCIPAL COMPONENTS ANALYSIS (PCA)

Cuando nos enfrentamos a un gran número de variables correlacionadas, los componentes principales nos permiten resumir variables en un número más pequeño de variables representativas que colectivamente explican la mayoría de la variabilidad del conjunto de datos original.

**Principal Component Analysis (PCA)** se refiere al proceso de reducción de dimensionalidad mediante el cual se computan los componentes principales, los cuales permiten comprender los datos. PCA es un enfoque no supervisado, ya que involucra sólo un conjunto de features $X_1, X_2, .., X_N$ y no involucra una respuesta asociada Y. Además de producir variables derivadas para su uso en problemas de aprendizaje no supervisado, PCA también sirve como una herramienta de visualización de datos.[13]

Para nuestro experimento, lo usaremos bajo este propósito exploratorio, es decir, mediante una visualización. Para lo cual utilizaremos la clase **PCA** del módulo **decomposition** del paquete python **sklearn**, sobre la cual sólo utilizaremos un parámetro que es **n_componentes**, que según la documentación oficial corresponde al número de componentes a mantener.[14]

## 3.2. LOCAL OUTLIER FACTOR[15]

El algoritmo de **Local Outlier Factor** es un método de detección de anomalías no supervisado que calcula la desviación de la densidad local en un punto de datos dado con respecto a sus vecinos. Considera como valores atípicos las muestras que tienen una densidad sustancialmente menor que sus vecinas.

Es importante definir antes, la métrica *reachability distance* definida en el paper de Ng y Sander, que se define matemáticamente de la siguiente manera:

---

[13] *G. James, D. Witten, T. Hastie, R. Tibshirani, An Introduction to Statistical Learning with Applications in R (2013), capítulo 10.*

[14] *Documentación oficial de sklearn que hace referencia a la clase PCA https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html*

[15] *Raymond T. Ng & Joerg Sander. LOF: Identifying Density-Based Local Outliers.*

$$Reachability\ distance\ (A,B)_k\ =\ max\ \{Distance(A,B), Distance_k(B)\}$$

Es decir, dado dos puntos A y B, *reachability distance* se define como el máximo de la k-ésima distancia de B y la distancia entre A y B. Por lo que, si el punto A está dentro de los k vecinos del punto B, *reachability distance* será la k-ésima distancia de B. De lo contrario, será la distancia real entre A y B.

En este algoritmo se emplea la métrica anteriormente definida, la cual es tomada para calcular otra métrica llamada *local reachability density*, definida del siguiente modo, dado un objeto A y otro objeto B:

$$lrd_k(A) := \frac{1}{\left(\frac{\Sigma_{B\varepsilon N_k(A)}\ reachability\ distance_k\ (A,B)}{|N_k(A)|}\right)}$$

Que se define como el inverso de la media de *reachability density* del objeto A de sus vecinos. k se define como el número de vecinos más cercanos y $N_k(A)$ es el conjunto de los k vecinos. Por tanto, luego es posible calcular la métrica **LOF** comparando las densidades de accesibilidad local para ese dicho objeto entre los distintos vecinos, utilizando lo siguiente:

$$LOF_{MinPts}\ (A)\ =\ \frac{\Sigma_{B\varepsilon N_k(A)}\ \frac{lrd_k(B)}{lrd_k(A)}}{|N_k(A)|}$$

Que se define como el promedio de *local reachability density* dividido por el *local reachability density* del objeto A. Finalmente, teniendo este valor, la regla es la siguiente:

- Si LOF(k) es próximo a 1, la densidad es similar a la de los vecinos.
- Si LOF(k) es menor a 1, la densidad es mayor que la de los vecinos (inlier).
- Si LOF(k) es mayor que 1, la densidad es menor que la de los vecinos (outlier).

Para el proceso de limpieza a implementar en los experimentos, utilizaremos la clase de Python **sklearn.neighbors.LocalOutlierFactor** del paquete **sklearn**, para la cual, utilizaremos el siguiente parámetro determinante en este proceso:[16]

- Contamination: Proporción de valores atípicos en el conjunto de datos utilizado como umbral en los puntajes de las muestras. Si toma el valor 'auto', el umbral será el valor

[16] *Documentación oficial del paquete scikit-learn https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.LocalOutlierFactor.html*

definido en el paper original de Ng y Sander, sino, puede tomar un valor decimal comprendido en [0, 0.5].

Implementaremos **Local Outlier Factor** debido a que, a diferencia de otros algoritmos, contempla todos los features a la hora de borrar datos "contaminados".

## 3.3. PROBLEMA DEL DESBALANCE DE CLASES[17]

El **problema del desbalance de clases** está presente en muchos conjuntos de datos de clasificación del mundo real (algunos ejemplos son fraude, detección de enfermedades, delincuencia en scoring crediticio, entre otros[18]) y consiste en una desproporción del número de ejemplos de las diferentes clases en el problema. Este problema dificulta el rendimiento de los modelos de machine learning de clasificación, debido a su diseño orientado a la precisión, lo que generalmente hace que se pase por alto la clase minoritaria.

Existen varias técnicas para contrarrestar este problema, entre las cuales se clasifican en 4 grupos principales, según el enfoque de cómo abordan el problema:

1. Los enfoques basados en el nivel del algoritmo (o internos) intentan adaptar los clasificadores para sesgar el aprendizaje hacia la clase minoritaria, en vez de centrarse en modificar el conjunto de datos de entrenamiento para combatir el sesgo de clase. Para realizar la adaptación, se requiere un conocimiento tanto sobre el clasificador como del dominio de aplicación para comprender por qué el clasificador falla cuando la distribución de la clase es desigual.
2. Los enfoques de nivel de datos (o externos) tienen como objetivo re-equilibrar la distribución de la clase mediante el remuestreo del espacio de datos. De esta forma, se evita la modificación del algoritmo de machine learning, ya que el efecto causado por el desequilibrio disminuye con un paso de preprocesamiento. Estos métodos pueden ser de undersampling, oversampling o híbridos. El que utilizaremos en el presente trabajo se denomina **SMOTE**, que detallaremos en la siguiente sección.
3. El enfoque de aprendizaje sensible al costo es un intermedio entre los enfoques de nivel de datos y algoritmos. Se incorporan tanto las transformaciones de nivel de datos (agregando costos a las instancias) como las modificaciones de nivel de algoritmo (modificando el proceso de aprendizaje para aceptar costos). El clasificador se inclina

[17] *A. Fernández, S.I Galar, R. Prati, B. Krawczyk, F. Herrera, Learning from Imbalanced Data Sets (2018). Springer.*
[18] *L. Zhang, H. Ray, J. Priestley & S. Tan, Journal of Applied Statistics, A descriptive study of variable discretization and cost-sensitive logistic regression on imbalanced credit data (23 de Julio del 2019)*

hacia la clase minoritaria asumiendo costos de clasificación erróneos más altos para esta clase y buscando minimizar los errores de costo total de ambas clases. Este enfoque será detallado más adelante.

4. Los métodos basados en ensambles generalmente consisten en una combinación entre un algoritmo de aprendizaje por ensambles (como bagging, boosting y otros enfoques híbridos) y una de las técnicas anteriores, específicamente, nivel de datos o sensible al costo. Se sabe que los clasificadores de tipo ensamble, es decir, la combinación de varios clasificadores en uno solo, mejoran la precisión en comparación con el uso de un sólo clasificador, aunque, estas técnicas no resuelven el problema de desequilibrio de clases. Al agregar un enfoque de nivel de datos al algoritmo de ensamble, el nuevo método híbrido generalmente procesa previamente los datos antes de entrenar a cada clasificador, mientras que los conjuntos sensibles al costo, en lugar de modificar el clasificador base para aceptar costos en el proceso de aprendizaje, guían la minimización de costos a través del algoritmo de ensambles.

## 3.4. SYNTHETIC MINORITY OVER-SAMPLING TECHNIQUE (SMOTE)[19]

Como mencionamos anteriormente, en caso de que el conjunto de datos esté desbalanceado, lo cual impacta gravemente en la capacidad predictiva de nuestro modelo, se implementarán ciertas técnicas de balanceo de datos, entre las cuales implementaremos **SMOTE**.

Es una técnica estadística basada en vecinos más cercanos juzgados con distancia euclídea que sirve para aumentar el número de casos en un conjunto de datos de manera equilibrada. El modelo funciona generando nuevas instancias a partir de los casos existentes de la clase minoritaria. Los ejemplos sintéticos hacen que el clasificador forme regiones de decisión más grandes y menos específicas, en lugar de regiones más pequeñas y más específicas. Ahora se aprenden regiones más generales para las muestras de clase minoritaria en lugar de las que están incluidas en las muestras de la clase mayoritaria a su alrededor. El efecto es que los árboles de decisión generalizan mejor.

Es muy común su aplicación en problemas de fraude y churn, cuando los conjuntos de datos están desbalanceados.

[19] *N. V. Chawla, K. W. Bowyer, L. O.Hall, W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," Journal of artificial intelligence research, 321-357, 2002.*

Para su aplicación usaremos la clase **imblearn.over_sampling.SMOTE** del paquete Python **imblearn**, que es específico para el tratamiento de desbalance de clases. A fines de replicar los mismos resultados, sólo le definiremos una semilla.[20]

## 3.5. BOOSTING

Es un enfoque general que puede ser aplicado a muchos métodos de aprendizaje estadístico para problemas de regresión o clasificación.
Se refiere a cualquier método de ensamble que puede combinar varios weak learners en un strong learner. La idea general de la mayoría de los métodos de boosting es entrenar predictores secuencialmente, cada uno tratando de corregir a su predecesor. Algunos ejemplos de métodos de Boosting son: AdaBoost, Gradient Boosting, CatBoost, XGBoost, LightGBM, NGBoost, etc.

## 3.6. GRADIENT BOOSTING

Gradient Boosting, como bien se explicó en el anterior apartado, es un método que funciona agregando secuencialmente modelos a un conjunto, cada uno corrigiendo a su predecesor. Sin embargo, en lugar de ajustar los pesos de las instancias para cada iteración, este método intenta ajustar el nuevo predictor a los errores residuales cometidos por el predictor anterior[21].

## 3.7. NATURAL GRADIENT BOOSTING

El gradiente natural es definido matemáticamente como:[22]

$$\tilde{\nabla}_\theta \mathcal{L}(\theta) = \mathrm{F}^{-1} \nabla_\theta \mathcal{L}(\theta)$$

Donde $\mathcal{L}(\theta)$ es la función de pérdida y F es la Matriz de Información de Fisher.

Como corolario, tenemos el siguiente algoritmo:

1. Repetir:
    1.1. Hacer un forward pass en nuestro modelo y computar la función de pérdida $\mathcal{L}(\theta)$

[20] *Documentación oficial del paquete imblearn* https://imbalanced-learn.readthedocs.io/en/stable/generated/imblearn.over_sampling.SMOTE.html
[21] *Hands-On Machine Learning with Scikit-Learn, Keras and Tensorflow, 2nd Edition, Aurelien Geron, pp 182.*
[22] *Natural Gradient Descent (2018), blog post de Agustinus Kristiadi,* https://wiseodd.github.io/techblog/2018/03/14/natural-gradient/

1.2. Procesar el gradiente $\nabla_\theta \mathcal{L}(\theta)$

1.3. Calcular la Matriz de Información de Fisher F, o su versión empírica (con respecto a nuestros datos de entrenamiento).

1.4. Calcular gradiente natural $\tilde{\nabla}_\theta \mathcal{L}(\theta) = \mathrm{F}^{-1} \nabla_\theta \mathcal{L}(\theta)$

1.5. Actualizar el parámetro $\theta = \theta - \alpha \tilde{\nabla}_\theta \mathcal{L}(\theta)$, donde $\alpha$ es la tasa de aprendizaje.

2. Hasta converger

En el mundo de Natural Gradient, no restringimos el movimiento de los parámetros en el espacio de soluciones. Lo que se hace es detener el movimiento de la distribución de probabilidad de salida en cada paso. ¿Cómo medimos la distribución de probabilidad? Usando Log Likelihood. Además, la matriz de Fisher ofrece la curvatura de Log Likelihood.

El gradiente ordinario no conoce la curvatura de la función de pérdida porque es un método de optimización de primer orden. Pero cuando incluimos la matriz de Fisher en el gradiente, lo que estamos haciendo es escalar las actualizaciones de los parámetros con la curvatura de la función de probabilidad logarítmica. Por lo tanto, en los lugares del espacio de distribución donde la Log Likelihood cambia rápidamente con un parámetro, la actualización del parámetro sería menor que un plano en el espacio de distribución.

Además, Natural Gradient nos permite controlar directamente el movimiento del modelo en el espacio predicho. En el gradiente ordinario, su movimiento movimiento está estrictamente en el espacio de soluciones y se restringe el movimiento en ese espacio con una tasa de aprendizaje con la esperanza de que el movimiento en el espacio de predicción también esté restringido. En cambio, en la actualización de Natural Gradient, se restringe directamente el movimiento en el espacio de predicción al estipular que el modelo sólo se mueve en una distancia fija en términos de divergencia KL.[23]

El uso de gradiente natural en lugar del gradiente ordinario es crucial. El descenso de gradiente ordinario no es apropiado cuando se trata con parámetros de distribuciones de probabilidad.

¿Qué tan importante es usar gradiente natural? Según el paper de **NGBoost**, los autores compararon los resultados de la regresión unidimensional con la figura aplicando el gradiente ordinario (a) y con los gradientes naturales de la figura (b).

[23] *Natural Gradient (2020), blog post escrito por Manu Joseph,* https://towardsdatascience.com/natural-gradient-ce454b3dcdfa

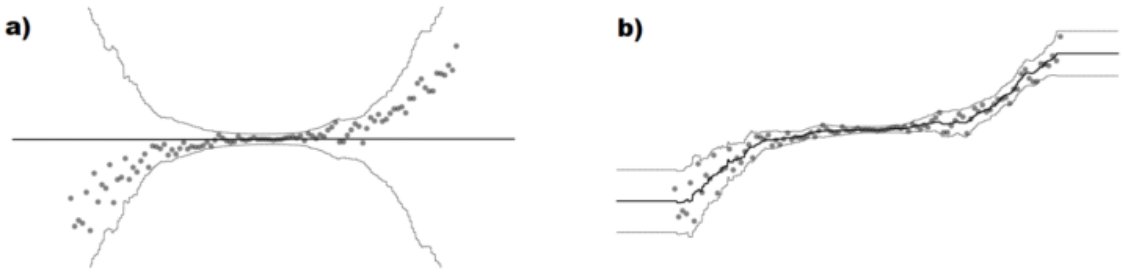


***Figura 1**. Gradiente ordinario (a) y gradiente natural (b) usados en el entrenamiento.* [24]

Los intervalos de confianza del 95% en los extremos se comportan de manera extraña para **NGBoost** con gradientes ordinarios. Esta es una fuerte evidencia de que las dinámicas de aprendizaje son mucho mejores cuando se utilizan gradientes naturales.

## 3.8. ALGORITMO NGBOOST

Los científicos de datos que compiten en las competencias de Kaggle a menudo generan soluciones que utilizan ensambles de algoritmos avanzados de machine learning. Un modelo particular que es parte de estos ensambles es Gradient Boosting Machines (GBM). Teniendo en cuenta el gráfico anterior, **NGBoost** ha demostrado tener buenos resultados de performance, así como los otros modelos de ensamble.

**Natural Gradient Boosting** es un algoritmo que tiene la capacidad de generar predicciones probabilísticas utilizando gradient descent de manera genérica. La predicción probabilística de la incertidumbre es crucial en muchas aplicaciones tales como la asistencia sanitaria o el pronóstico del tiempo. La predicción probabilística es el enfoque en el que el modelo genera una distribución de probabilidad completa en todo el espacio de resultados, siendo una forma natural de cuantificar esas incertidumbres.

**NGBoost** es un enfoque basado en gradient boosting que utiliza el gradiente natural para abordar desafíos técnicos que dificultan la predicción probabilística genérica con gradient boosting. Este algoritmo posee 3 componentes modulares abstractos que se eligen como configuración:

[24] *https://stanfordmlgroup.github.io/projects/ngboost/ - Página oficial del paper "NGBoost: Natural Gradient Boosting for Probabilistic Prediction (Febrero 2020), T. Duan*, A. Avati*, D. Yi Ding, S. Basu, A. Ng, A. Schuler"*

1. Base Learner ($f$): La opción más común es decision tree, que tienden a funcionar bien en inputs estructurados.
2. Distribución de Probabilidad Paramétrica ($P_\theta$ ): La distribución suele ser compatible con el tipo de salida. Por ejemplo, la Distribución Normal para salidas con valor real, Distribución Bernoulli para salidas binarias.
3. Regla de Puntuación ($S_{MLE}$): La estimación de máxima verosimilitud es una opción obvia. También son adecuadas reglas más robustas, como el puntaje de probabilidad de clasificación continua.[25]

Estas opciones se pueden mezclar y combinar para personalizarlas para el problema de predicción específico en cuestión.

En sí, la idea del algoritmo es sencilla:

*Entrenamos al base learner para que produzca para cada muestra de training una distribución de probabilidad que minimice un score adecuado.* ***NGBoosting*** *se utiliza como algoritmo de optimización y el base learner es un conjunto de weak learners entrenados con un enfoque de boosting.*

### 3.8.1. PSEUDOCÓDIGO

1, Tenemos M etapas de boosting. Los parámetros iniciales θ se establecen con valores correspondientes a la distribución marginal ajustada en los valores del target de training $\{y_i\}_{i=1}^{n}$ .

2. En cada etapa $m = \{1, \ldots, M\}$ se evalúa la regla de puntuación adecuada para cada muestra de training $S_{MLE}^{(m)} = -log\ P_{\theta_i^{(m)}}(y_i)$.

3. Los gradientes naturales $\boldsymbol{\nabla}_i^{(m)}$ se calculan como gradientes normales escalados $\mathcal{L}_S^{-1}\boldsymbol{\nabla} S_{MLE}^{(m)}$.

4. El base learner (weak) $f^{(m)}$se calcula para cada muestra de training i mediante una pequeña actualización en la dirección del gradiente $\theta_i^{(m)} = \theta_i^{(m-1)} - \eta\rho^{(m)} f^{(m)}(x_i)$ y comienza la siguiente etapa del algoritmo de boosting.[26]

---

[25] *"NGBoost: Natural Gradient Boosting for Probabilistic Prediction.". T. Duan, A. Avati, D. Y. Ding, S. Basu, Andrew Y. Ng, and A. Schuler (2019).*

[26] *https://dkopczyk.quantee.co.uk/ngboost-explained/ - Blog post sobre NGBoost explicado por Dawid Kopczyk.*

A fines prácticos, para entrenar nuestros modelos de machine learning, utilizaremos el paquete de Python **ngboost** desarrollado por el *Machine Learning Group* de la universidad de Stanford. Principalmente para nuestro problema de clasificación utilizaremos la clase **NGBClassifier**.[27]

A continuación, detallaremos los hiper parámetros del modelo que exploraremos en los experimentos del presente trabajo, los cuales, acorde a su definición en el paquete python de **ngboost**, son los siguientes:

- Base: Base Learner utilizado en el algoritmo de boosting. Consiste en algún modelo regresor del paquete python **sklearn**, por ejemplo, **DecisionTreeRegressor**. Sólo exploraremos un hiper parámetro de este modelo base, acorde a lo que dice la documentación del paquete python **sklearn**[28], se define de la siguiente manera:
    - max_depth: Es la profundidad máxima del árbol. Si este valor es None, entonces los nodos se expanden hasta que todas las hojas sean puras o hasta que todas las hojas contengan menos de min_samples_split (otro parámetro del modelo, que por default toma un valor de 2) muestras.

  El resto de los hiper parámetros serán aquellos que, según la documentación, están definidos por defecto.
- Estimators: Cantidad de iteraciones de booting a entrenar
- Learning rate: Tasa de aprendizaje, parámetro de ajuste en un algoritmo de optimización que determina el tamaño del paso en cada iteración mientras se desplaza hacía un mínimo de una función de pérdida.

## 3.9. COST-SENSITIVE LEARNING

El aprendizaje sensible al costo es un tipo de aprendizaje que toma en cuenta los costos de clasificación errónea (y posiblemente otros costos) al entrenar un modelo de Machine Learning. El objetivo de este tipo de aprendizaje es minimizar el costo total. La principal diferencia con el aprendizaje no sensible al costo es que trata las clasificaciones erróneas de manera diferente. Es decir, el costo de etiquetar un ejemplo positivo como negativo puede ser diferente del costo e etiquetar un ejemplo negativo como positivo. Sin embargo, los métodos de clasificación estándar (o no sensibles al costo) no tienen en cuenta estos costos y suponen un costo constante de errores de clasificación errónea.

[27] *Guía de uso del paquete ngboost https://stanfordmlgroup.github.io/ngboost/intro.html*
[28] *Documentación sobre DecisionTreeRegressor https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html*

Los problemas de clasificación binaria desbalanceados suelen tener una interpretación diferente para cada uno de los errores de clasificación que se pueden realizar. Esto tiene sentido si consideramos que el objetivo del clasificador para este caso binario y desbalanceado es detectar los casos positivos correctamente y los casos positivos representan un evento excepcional en el que estamos más interesados.
A modo de ejemplo podemos citar los siguientes casos:

1. Problema de préstamo bancario (Caso de este trabajo): Considere un problema en el que un banco quiere determinar la asignación de un crédito o no. Negar un préstamo a un buen cliente no es tan malo como otorgarle un préstamo a un cliente malo que nunca lo pagará.
2. Problema de diagnóstico de cáncer: Considere un problema en el que un médico quiera determinar si un paciente tiene cáncer o no. Es mejor diagnosticar a un paciente sano con cáncer y hacer un seguimiento con más pruebas médicas que dar de alta a un paciente que tiene cáncer.[29]
3. Problema de detección de fraude: Considere el problema de una compañía de seguros que quiere determinar si un reclamo es fraudulento. Identificar buenos como fraudulentos y hacer un seguimiento con el cliente es mejor que favorecer los reclamos de seguro fraudulentos.

## 3.10. INTERPRETABLE MACHINE LEARNING

A pesar de su amplia adopción, la mayoría de los modelos de Machine Learning son cajas negras. Sin embargo, comprender las razones detrás de las predicciones es bastante importante para evaluar la confianza, lo cual es fundamental a la hora de tomar decisiones basadas en una predicción, o al evaluar la implantación de un nuevo modelo. Dicha exploración, además, proporciona información sobre el modelo que se puede utilizar para transformar un modelo o predicción no confiable en una más confiable. Es válido recalcar, además, que estas técnicas permiten brindar explicaciones más amigables para gente no técnica que no está muy familiarizada con la implementación de los algoritmos.

Es importante diferenciar entre dos definiciones diferentes (pero relacionadas) de confianza:

1. Confiar en una predicción, es decir, si un usuario confía en una predicción individual lo suficiente como para tomar alguna acción basada en ella.

[29] *Claude Sammut, Geoffrey Webb, Encyclopedia of Machine Learning (2010).*

2. Confiar en un modelo, es decir, si el usuario confía en que un modelo se comportará de manera razonable si se implementa.

Ambos se ven directamente afectados por lo mucho que el humano entiende el comportamiento de un modelo, en vez de verlo como una caja negra.

El **aprendizaje automático interpretable**, entonces, se refiere a los métodos y modelos que hacen que el comportamiento y las predicciones de los sistemas de Machine Learning sean comprensibles para los humanos.[30]

## 3.11. SHAPLEY ADDITIVE EXPLANATIONS (SHAP)

SHAP es un enfoque desde la teoría de juegos cooperativa que permite explicar el resultado de cualquier modelo de machine learning, fue publicado por primera vez en 2017 por Lundberg y Lee[31]. La analogía con teoría de juegos es la siguiente:

*Una predicción puede explicarse asumiendo que cada valor del feature de una observación es un "jugador" en un juego donde la predicción es el pago. Los valores Shapley nos dicen entonces cómo distribuir equitativamente el "pago" entre los features, es decir, cuantificar la contribución que cada feature aporta a la predicción realizada por el modelo.*[32]

Supongamos un ejemplo donde entrenamos un modelo para predecir precios de viviendas en Delhi, tomaremos como referencia la moneda india INR dado el caso. Para una determinada vivienda, nuestro modelo predice INR 51,00,000 y se debe explicar esta predicción. La vivienda tiene un tamaño de 50 yardas (*size-50*), tiene una piscina privada (*has_pool*) y tiene garage (*has_garage*). Además sabemos que la predicción promedio para todos los apartamentos es de INR 50,00,000. Nuestro objetivo entonces es explicar cómo cada uno de estos valores de features contribuyó a la predicción en comparación con la predicción promedio.

[30] *Christoph Molnar, Interpretable Machine Learning, a Guide for Making Black Box Models Explainable (2020), sección 1. Fuente: https://christophm.github.io/interpretable-ml-book/*
[31] *S. Lundberg, S. Lee, A Unified Approach to Interpreting Model Predictions (22 de mayo del 2017) https://arxiv.org/abs/1705.07874*
[32] *Christoph Molnar, Interpretable Machine Learning, a Guide for Making Black Box Models Explainable (2020), sección 5.9. Fuente: https://christophm.github.io/interpretable-ml-book/*

Entonces, en términos de teoría de juego, entendemos por "juego" a la predicción para una sola observación del conjunto de datos. Los "jugadores" son los valores de los features de la instancia que colaboran para participar en el juego (predecir un valor).
Para nuestro ejemplo, los valores de los features size-50, has_pool y has_garage contribuyeron juntos para lograr la predicción de INR 51,00,000. Nuestro objetivo es explicar la diferencia entre la predicción real (INR 51,00,000) y la predicción promedio (INR 50,00,000), es decir, la diferencia de INR 1,00,000. Una posible explicación podría ser que has_pool contribuyó INR 30,000, has_garage contribuyó INR 50,000 y size-50 aportó INR 20,000. Las contribuciones suman INR 1,00,000, es decir, la predicción final menos el precio promedio de la vivienda (variable de target) en el dataset.

Para resumir, el valor de Shapley para cada feature (pago) básicamente está tratando de encontrar el peso correcto de modo que la suma de todos los valores Shapley sea la diferencia entre las predicciones y el valor promedio del modelo. En otras palabras, los valores de Shapley corresponden a la contribución de cada feature para alejar la predicción del valor esperado.[33]

Los valores SHAP proporcionan 2 grandes ventajas:[34]

- Interpretabilidad global: Los valores SHAP pueden mostrar cuánto contribuye cada predictor, ya sea positiva o negativamente, a la variable objetivo. Esto es como el gráfico de feature importance, pero puede mostrar la relación positiva o negativa para cada feature con el target.
- Interpretabilidad local: Cada observación obtiene su propio conjunto de valores SHAP, lo que favorece a la transparencia del modelo, con lo cual es posible explicar por qué un caso recibe su predicción y las contribuciones de los predictores. Los algoritmos tradicionales de feature importance sólo muestran los resultados en toda la población, pero no en cada caso individual.

Entre los casos de aplicación de esta técnica podemos mencionar:

- Un modelo afirma que un banco no debería prestarle dinero a alguien, y el banco está legalmente obligado a explicar la base de cada rechazo de préstamo.

[33] *Explicación y ejemplo práctico de las viviendas sacado de "A Unique Method for Machine Learning Interpretability: Game Theory & Shapley Values!", blog post por A. Choudhary (Noviembre 2019)* *https://www.analyticsvidhya.com/blog/2019/11/shapley-value-machine-learning-interpretability-game-theory/*
[34] *Artículo con caso práctico y explicaciones sobre interpretabilidad global y local usando SHAP* *https://towardsdatascience.com/explain-your-model-with-the-shap-values-bc36aac4de3d*

- Un proveedor de atención médica desea identificar qué factores están impulsando el riesgo de alguna enfermedad de cada paciente para poder abordar directamente dichos factores de riesgo con intervenciones de salud específicas.[35]

En la práctica, utilizaremos el paquete python **shap**[36], del cual utilizaremos la implementación **TreeExplainer** para explicar la salida de nuestro mejor modelo **NGBoost** para nuestro primer experimento, que, como bien dijimos anteriormente, es un método de ensamble basado en árboles. Del mismo podemos obtener entonces nuestros valores shap (**shap_values**, acorde a lo que dice la documentación oficial[37]) y luego realizar sobre los mismos el análisis de interpretación global sobre los features, mediante los métodos **summary_plot** y **dependence_plot** que ofrece el módulo shap.

## 3.12. LOCAL INTERPRETABLE MODEL-AGNOSTIC EXPLANATIONS (LIME)[38]

**LIME** es una solución que provee explicaciones para las predicciones individuales, atacando al problema de "confiar en una predicción" y seleccionar múltiples predicciones (y explicaciones) en contraste al problema de "confiar en el modelo".

A modo de ejemplo se ilustra el caso de un modelo que predice si cierto paciente tiene gripe. Luego, la predicción se explica mediante un "explainer" que resalta los síntomas que son más importantes para el modelo. Con esta información sobre las justificaciones del modelo, el médico ahora tiene la opción de poder confiar o no en el modelo.

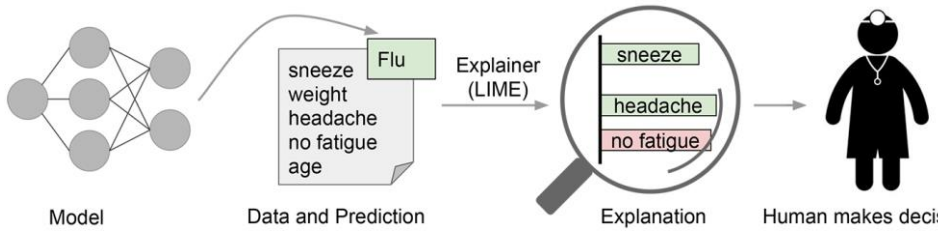


***Figura 2.*** *Ilustración gráfica del proceso de Explanation sobre predicciones individuales a un decision maker.*

[35] *Breve explicación sobre SHAP e ilustración de caso práctico* https://www.kaggle.com/dansbecker/shap-values

[36] *Repositorio que contiene código fuente y referencias importantes acerca de la implementación de SHAP* https://github.com/slundberg/shap

[37] *Documentación oficial del paquete python SHAP* https://shap.readthedocs.io/en/latest/

[38] *Marco Tulio Ribeiro, Sameer Singh, Carlos Guestrin. Why should I trust you? Explaining the predictions of any classifier (2016).*

A fines prácticos, se utilizará el paquete python **lime**, utilizando el módulo **lime_tabular** para generar un objeto **LimeTabularExplainer** que explicará aquellas observaciones que se le indiquen. Además utilizaremos el módulo **submodular_pick** para extraer una muestra representativa de explicaciones.[39]

## 3.13. PERMUTATION FEATURE IMPORTANCE[40]

**Permutation Feature Importance** mide el aumento en el error de predicción del modelo después de permutar los valores de la característica, lo que rompe la relación entre el feature y el output real.
El concepto es relativamente sencillo: debe medirse la importancia de un feature calculando el aumento en el error de predicción del modelo después de permutar dicho feature.
Un feature es importante en términos de que si mezclamos sus valores, aumenta el error del modelo, porque en este caso el modelo se basó en la característica para la predicción.
Un feature no es importante si, al mezclar sus valores, el error del modelo se mantiene invariable, por lo que, en este caso, el modelo ignoró el feature para la predicción.

El algoritmo de **Permutation Feature Importance** detallado por Fisher, Rudin y Dominici (2018)[41] en base al paper de Random Forest (2001) de L. Breiman, se detalla de la siguiente manera:

Input: Modelo entrenado f, matriz de features X, vector objetivo y medida de error L (y, f).

Algoritmo:

1. Estimar el error original del modelo $e^{orig} = L(y, f(x))$ (por ejemplo, mean squared error).
2. Para cada feature j=1,...,p, hacer:
    - Generar la matriz de features $X^{perm}$ permutando el feature j en los datos X. Esto rompe la asociación entre el feature j y el verdadero output y.
    - Estimar el error $e^{perm}$ basado en las predicciones de los datos permutados.

[39] *Documentación oficial del paquete lime https://lime-ml.readthedocs.io/en/latest/*
[40] *Christoph Molnar, Interpretable Machine Learning, a Guide for Making Black Box Models Explainable (2020).*
[41] *Aaron Fisher, Cynthia Rudin, Francesca Dominici, All Models are Wrong, but Many are Useful: Learning a Variable's Importance by Studying an Entire Class of Prediction Models Simultaneously (2018)*

- Calcular el Permutation Feature Importance $FI^j = e^{perm}/e^{orig}$. Alternativamente, se puede usar la diferencia: $FI^j = e^{perm} - e^{orig}$

3. Ordenar los features FI de manera descendiente.

A modo de ejemplo, tenemos la siguiente tabla:[42]

| Height at age 20 (cm) | Height at age 10 (cm) | ... | Socks owned at age 10 |
|---|---|---|---|
| 182 | 155 | ... | 20 |
| 175 | 147 | ... | 10 |
| ... | ... | ... | ... |
| 156 | 142 | ... | 8 |
| 153 | 130 | ... | 24 |

***Tabla 7.*** *Ejemplo de conjunto de datos para explicar Permutation Feature Importance.*

La idea sería predecir la altura de una persona cuando cumpla 20 años de edad, utilizando los datos disponibles en el momento en el que tiene 10 años.
La importancia de la permutación se calcula después de que se haya ajustado un modelo. Por lo tanto, no se cambia el modelo ni las predicciones obtenidas a partir del mismo, que fue entrenado bajo un cierto conjunto de datos.
En este caso, nos preguntaremos: Si barajo aleatoriamente una sola columna de los datos de validación, dejando el target y todas las demás columnas en su lugar. ¿Cómo afectaría eso a la precisión de las predicciones en esos datos (ahora barajados)?

| Height at age 20 (cm) | Height at age 10 (cm) | ... | Socks owned at age 10 |
|---|---|---|---|
| 182 | 155 | ... | 20 |
| 175 | 147 | ... | 10 |
| ... | ... | ... | ... |
| 156 | 142 | ... | 8 |
| 153 | 130 | ... | 24 |

***Tabla 8.*** *Efecto de Permutation Feature Importance sobre el conjunto de datos propuesto como ejemplo.*

[42] *Ejemplo sobre Permutation Importance* https://www.kaggle.com/dansbecker/permutation-importance

Reordenar aleatoriamente una sola columna debería generar predicciones menos precisas, ya que los datos resultantes no corresponden a nada de lo observado para el modelo. La precisión del modelo se ve especialmente afectada si barajamos una columna en la que el modelo dependía en gran medida para las predicciones. En este caso, como bien dijimos anteriormente, alterar un feature importante como la altura a los 10 años causaría grandes cambios en las predicciones y alterar features menos significativos no cambiaría mucho la performance predictiva.

A fines prácticos utilizaremos 2 paquetes de Python:

- sklearn[43]: Provee también un método que permite obtener los promedios de las importancias de permutación, mediante el método **permutation_importance** del módulo **sklearn.inspection.**
- ELI5[44]: Ayuda a depurar clasificadores de machine learning y explica sus predicciones. Tiene gran soporte para diferentes implementaciones. También implementa varios algoritmos para inspeccionar modelos de caja negra, entre ellos, la clase **PermutationImportance** del módulo **eli5.sklearn**, el cual luego nos permitirá observar los pesos de importancia asignados para cada feature, mediante el método **show_weigths**.

[43] *Documentación oficial sobre el método de permutation importance del package sklearn* https://scikit-learn.org/stable/modules/generated/sklearn.inspection.permutation_importance.html
[44] *Documentación oficial del paquete ELI5* https://eli5.readthedocs.io/en/latest/index.html

# 4. PROPUESTA DE TRABAJO Y METODOLOGÍA

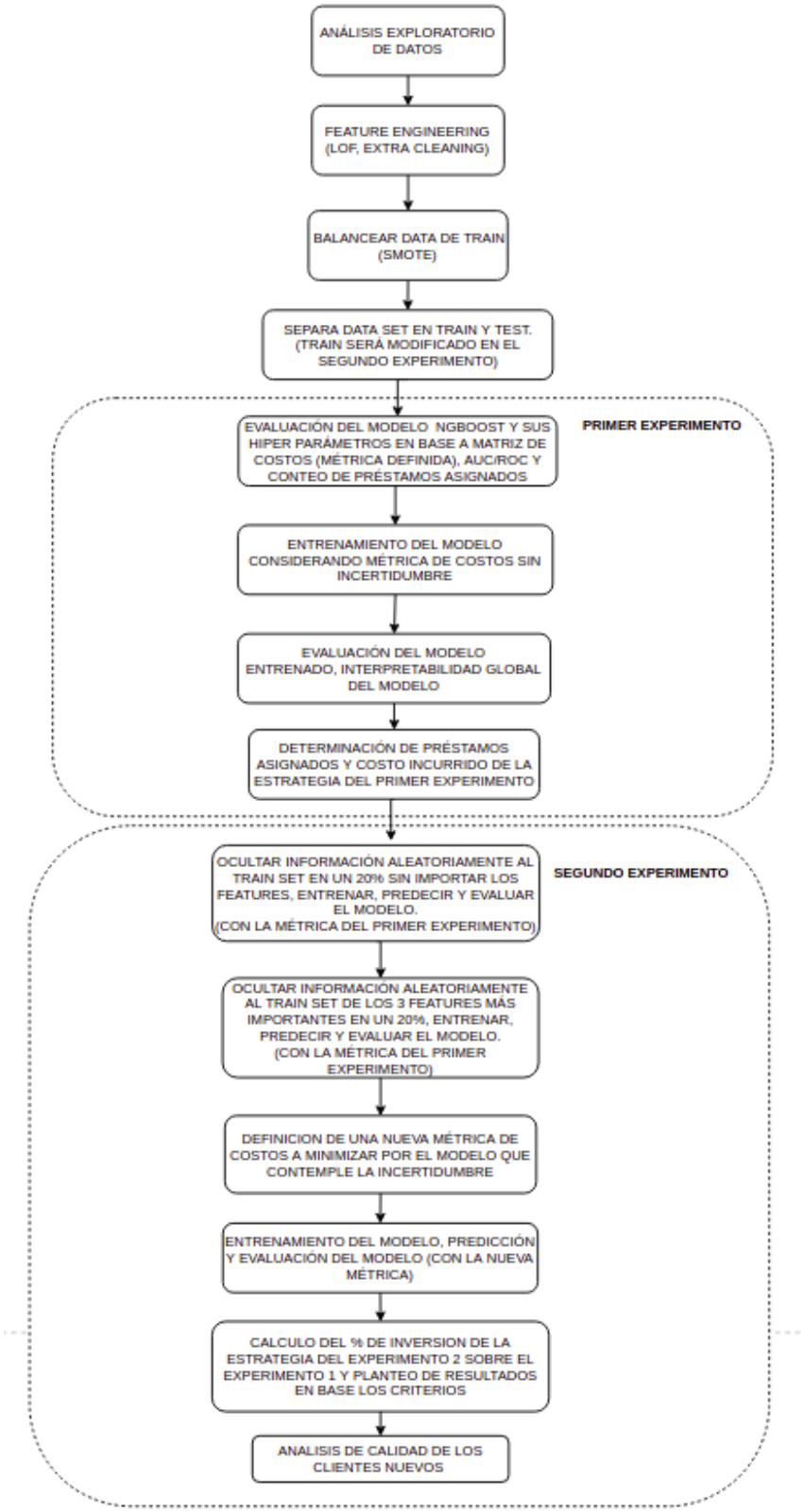


***Figura 3.*** *Flujograma para explicar la metodología de trabajo propuesta.*

## 4.1. ANÁLISIS EXPLORATORIO DE LOS DATOS

Se realizará un análisis exploratorio de los datos a fines de comprender la naturaleza del conjunto de datos de estudio.

A continuación detallaremos estadísticas descriptivas del data set:

| Métricas/ Features | SeriousDlqin2yrs | RevolvingUtilizationOfUnsecuredLines | age | NumberOfTime30-59DaysPastDueNotWorse | DebtRatio | MonthlyIncome |
|---|---|---|---|---|---|---|
| **count** | 150,000 | 150,000 | 150,000 | 150,000 | 150,000 | 120,269 |
| **mean** | 0.07 | 6.05 | 52.30 | 0.42 | 353.01 | 6,670.22 |
| **std** | 0.25 | 249.76 | 14.77 | 4.19 | 2,037.82 | 14,384.67 |
| **min** | 0 | 0 | 0 | 0 | 0 | 0 |
| **25%** | 0 | 0.03 | 41 | 0 | 0.18 | 3,400 |
| **50%** | 0 | 0.15 | 52 | 0 | 0.37 | 5,400 |
| **75%** | 0 | 0.56 | 63 | 0 | 0.87 | 8,249 |
| **max** | 1 | 50,708 | 109 | 98 | 329,664 | 3,008,750 |

***Tabla 9.*** *Estadísticas descriptivas del conjunto de datos tomado de la competencia GiveMeSomeCredit de Kaggle (primera parte).*

| Métricas/ Features | NumberOfOpenCreditLinesAndLoans | NumberOfTimes90DaysLate | NumberRealEstateLoansOrLines | NumberOfTime60-89DaysPastDueNotWorse | NumberOfDependents |
|---|---|---|---|---|---|
| **count** | 150,000 | 150,000 | 150,000 | 150,000 | 146,076 |
| **mean** | 8.45 | 0.27 | 1.02 | 0.24 | 0.76 |
| **std** | 5.15 | 4.17 | 1.13 | 4.16 | 1.12 |
| **min** | 0 | 0 | 0 | 0 | 0 |
| **25%** | 5 | 0 | 0 | 0 | 0 |
| **50%** | 8 | 0 | 1 | 0 | 0 |

| | | | | | |
|---|---|---|---|---|---|
| **75%** | 11 | 0 | 2 | 0 | 1 |
| **max** | 58 | 98 | 54 | 98 | 20 |

***Tabla 10.*** *Estadísticas descriptivas del conjunto de datos tomado de la competencia GiveMeSomeCredit de Kaggle (segunda parte).*

Podemos hacer las siguientes observaciones:

- Para la variable target, **SeriousDlqIn2yrs**, que toma valores 0 y 1, podemos observar el desbalance que hay, ya que la media es de 0.07, con lo cual, esto será un tema a tratar posteriormente.
- Existen ciertos valores que, por su definición, son porcentajes, tales como **DebtRatio** y **RevolvingOfUtilizationOfUnsecuredLines** y poseen valores mayores a 1, es decir, presenta valores que son outliers, que deberán ser tratados.
- Existen valores outliers en el campo **age** que también deberán ser tratados.

Además, podemos encontrar que en el conjunto de datos hay missings con lo cual podríamos calcular lo siguiente:

| Features | Cantidad de nulos |
|---|---|
| SeriousDlqin2yrs | 0 |
| RevolvingUtilizationOfUnsecuredLines | 0 |
| age | 0 |
| NumberOfTime30-59DaysPastDueNotWorse | 0 |
| DebtRatio | 0 |
| MonthlyIncome | 29,731 |
| NumberOfOpenCreditLinesAndLoans | 0 |
| NumberOfTimes90DaysLate | 0 |
| NumberRealEstateLoansOrLines | 0 |

| | |
|---|---|
| NumberOfTime60-89DaysPastDueNotWorse | 0 |
| NumberOfDependents | 3,924 |

***Tabla 11.*** *Tabla que ilustra las cantidades de valores nulos por feature.*

Dicha tabla afirma lo siguiente:

- **MonthlyIncome** tiene el 19.82% valores nulos sobre el total de filas del conjunto de datos.
- **NumberOfDependents** tiene el 2.62% de valores nulos sobre el total de filas del conjunto de datos.

Esto, a fines de nuestro análisis, será útil para definir la estrategia de feature engineering.

### 4.1.1. PROPORCIÓN DEL TARGET

Con respecto al target, se puede observar que hay 139974 (93.32%) observaciones que están etiquetados como **Responsible (target = 0)** y 10026 (6.68%) observaciones que están identificadas como **Delinquent (target = 1)**. Con lo cual observamos un desbalance de clases en el conjunto de datos, esto lo podemos observar en la siguiente Figura:


***Figura 4.*** *Gráfico de barras que ilustra la proporción del target que presenta el conjunto de datos a analizar, sobre el cual podemos observar un claro desbalance.*

Teniendo en cuenta esto, aplicaremos la técnica **SMOTE** para atacar el problema del desbalance presentado.

### 4.1.2. VARIANCE INFLATION FACTOR (VIF)

A fin de analizar la colinealidad de los features, implementaremos VIF para calcular los factores para cada feature. Este análisis permite, en forma simplificada, interpretar las correlaciones entre las diferentes variables con respecto a una, para cada uno de los features del conjunto de datos. Si este valor es más grande, entonces es probable que exista multicolinealidad y debería tratarse dicho feature.[45]

| Features | VIF Factor |
|---|---|
| Intercept | 17 |
| RevolvingUtilizationOfUnsecuredLines | 1 |
| age | 1.09 |
| NumberOfTime30-59DaysPastDueNotWorse | 26.77 |
| DebtRatio | 1 |
| MonthlyIncome | 1.02 |
| NumberOfOpenCreditLinesAndLoans | 1.28 |
| NumberOfTimes90DaysLate | 49.21 |
| NumberRealEstateLoansOrLines | 1.25 |
| NumberOfTime60-89DaysPastDueNotWorse | 60.64 |
| NumberOfDependents | 1.07 |

***Tabla 12.*** *Tabla que ilustra los factores VIF para cada feature y el intercepto.*

Respecto a esta tabla, podemos afirmar que existe posibilidad de que exista multicolinealidad entre **NumberOfTime30-59DaysPastDueNotWorse**, **NumberOfTimes90DaysLate** y **NumberOfTime60-89DaysPastDueNotWorse**. Estos features hacen referencia a la cantidad de veces que el prestatario estuvo moroso para cierta cantidad de días.

### 4.1.3. VISUALIZACIONES DE DATOS CRUDOS (CON OUTLIERS)

Se realizaron visualizaciones del conjunto de datos, calculando el promedio de los features (excluyendo al target) con respecto a la edad (campo **Age**), separando aquellos que fueron identificados con **Responsible (target=0)** y **Delinquent (target=1)** para poder apreciar diferencias en la interpretación de ambos casos.

[45] Documentación oficial del paquete python statsmodels que explica el método de variance inflation factor implementado en el presente trabajo.
https://www.statsmodels.org/stable/generated/statsmodels.stats.outliers_influence.variance_inflation_factor.html

La razón del por qué se tomó como eje x al campo **Age** es a fines de poder detectar datos anómalos para observaciones con edad menor a 18 años y mayor a 60 años (valores outliers). Entonces, tomar como referencia la edad de unas personas nos puede permitir, por ejemplo, medir los niveles de ingresos mensuales a medida que la persona posee mayor edad o por ejemplo la posibilidad de que a mayor edad, entonces mayor (hasta cierto punto) en promedio deberían ser la cantidad de personas dependientes a dicho prestatario.
Es válido recalcar que estas visualizaciones poseen datos con outliers, es decir, no se realizó ningún proceso de limpieza sobre estos. En las próximas secciones se podrán apreciar otras visualizaciones con los datos sin outliers.

#### 4.1.3.1. RevolvingUtilizationOfUnsecuredLines

En este gráfico podemos observar la presencia de muchos outliers a lo largo de la curva, por lo que deberán ser atendidos en el proceso de limpieza de los datos.

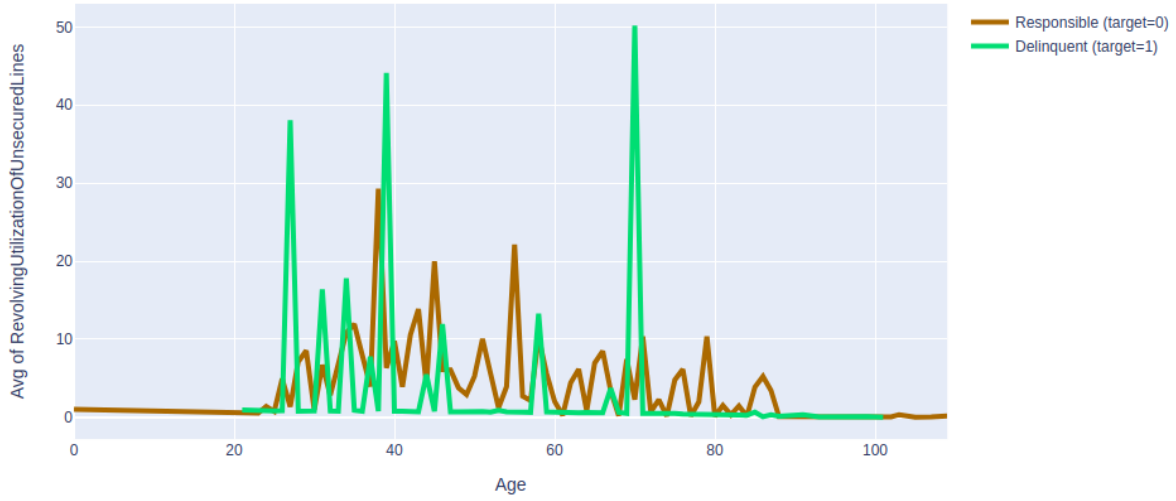


***Figura 5.*** *Relación entre RevolvingUtilizationOfUnsecuredLines (promedio) y Age.*

#### 4.1.3.2. NumberOfTime30-59DaysPastDueNotWorse

En este gráfico podemos apreciar que quienes están etiquetados como **Responsible (target = 0)** suelen tener niveles más bajo de este feature que los que están etiquetados como **Delinquent (target = 1)**, es decir, quienes tienden a ser responsables se endeudan, en promedio, menos veces que los que delinquen.

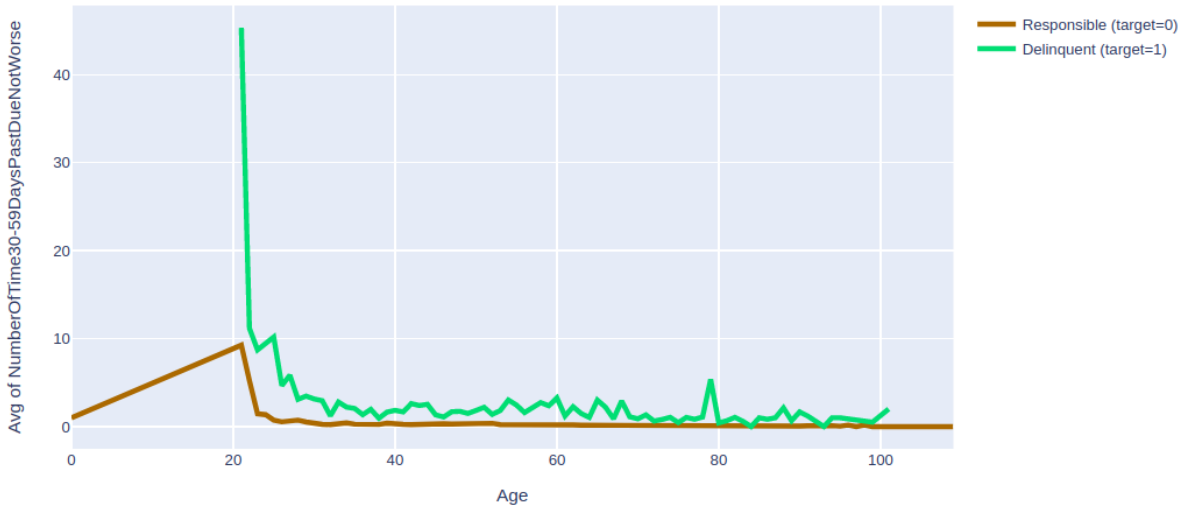


***Figura 6.*** *Relación entre NumberOfTime30-59DaysPastDueNotWorse (promedio) y Age.*

#### 4.1.3.3. DebtRatio

En cuanto al promedio del feature **DebtRatio** con respecto a la edad, podemos observar presencia de outliers en el rango de edad entre 90 y 100. Durante el resto del gráfico parece respetar cierta escala en forma consistente, para lo cual, dado este problema de escala, se tratará de hacer respetar esta consistencia en el proceso de limpieza de outliers.

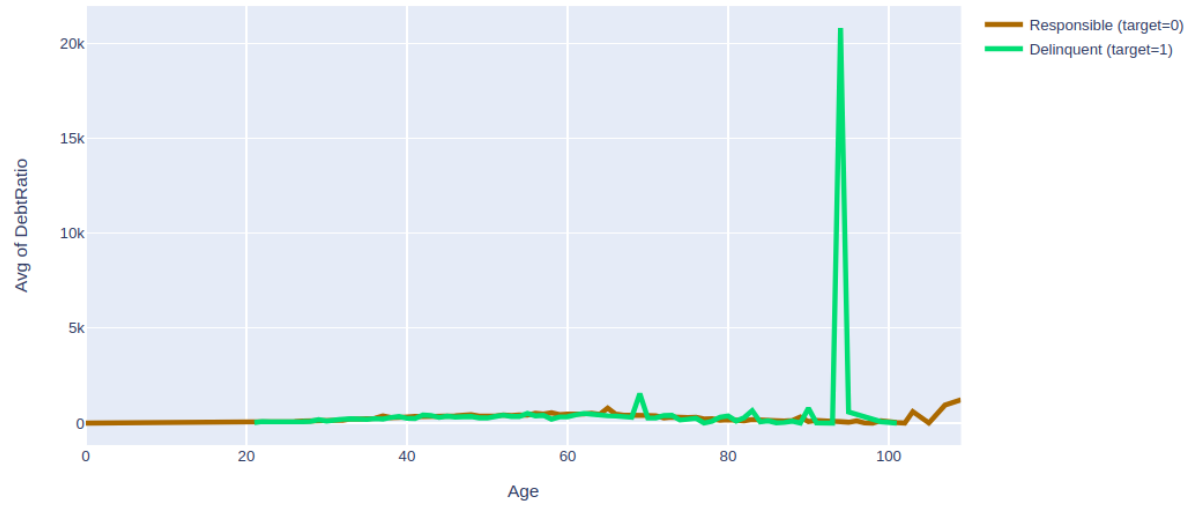


***Figura 7.*** *Relación entre DebtRatio (promedio) y Age.*

#### 4.1.3.4. MonthlyIncome

Del siguiente gráfico podemos apreciar que los prestatarios etiquetados como **Responsible (target = 0)** entre los 30 y 70 años tienden a tener, en promedio, un mayor ingreso mensual que los que están etiquetados como **Delinquent (target = 1)**.

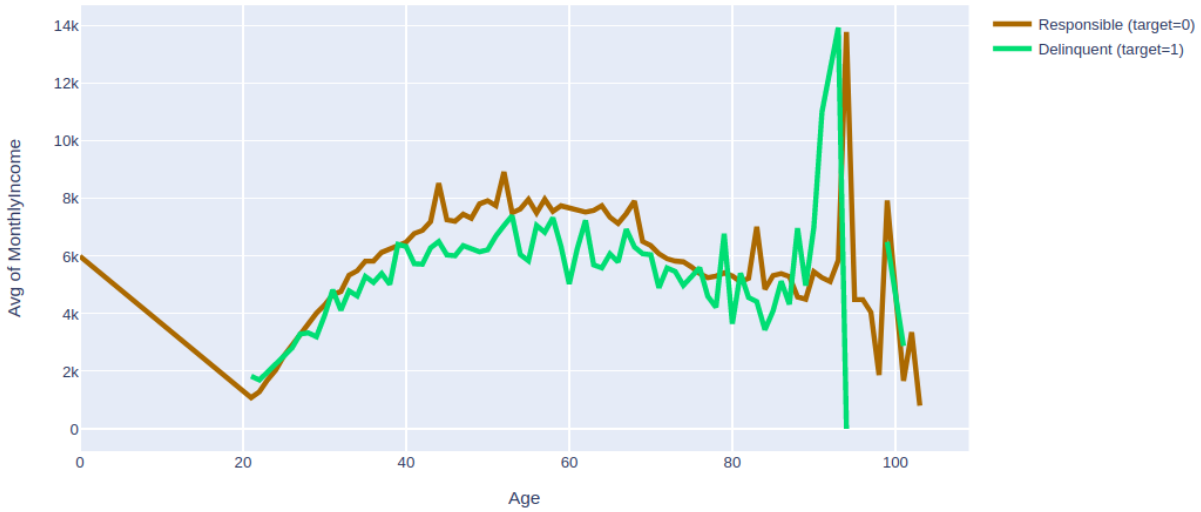


***Figura 8.*** *Relación entre MonthlyIncome (promedio) y Age.*

#### 4.1.3.5. NumberOfOpenCreditLinesAndLoans

En el siguiente gráfico podemos apreciar una tendencia ascendente para **Delinquent (target = 1)** y descendente para **Responsible (target = 0)** a partir de los 60 años, es decir, la tendencia para este feature cambia a partir de esa edad. Es decir, a partir de los 60 años, quienes delinquen tienen tendencia a tener más créditos abiertos que quienes son responsables.


***Figura 9.*** *Relación entre NumberOfOpenCreditLinesAndLoans (promedio) y Age.*

#### 4.1.3.6. NumberOfTimes90DaysLate

En este gráfico podemos apreciar que quienes están etiquetados como **Responsible (target = 0)** suelen tener niveles más bajo de este feature que los que están etiquetados como

**Delinquent (target = 1)**, es decir, quienes tienden a ser responsables se endeudan, en promedio, menos veces que los que delinquen.

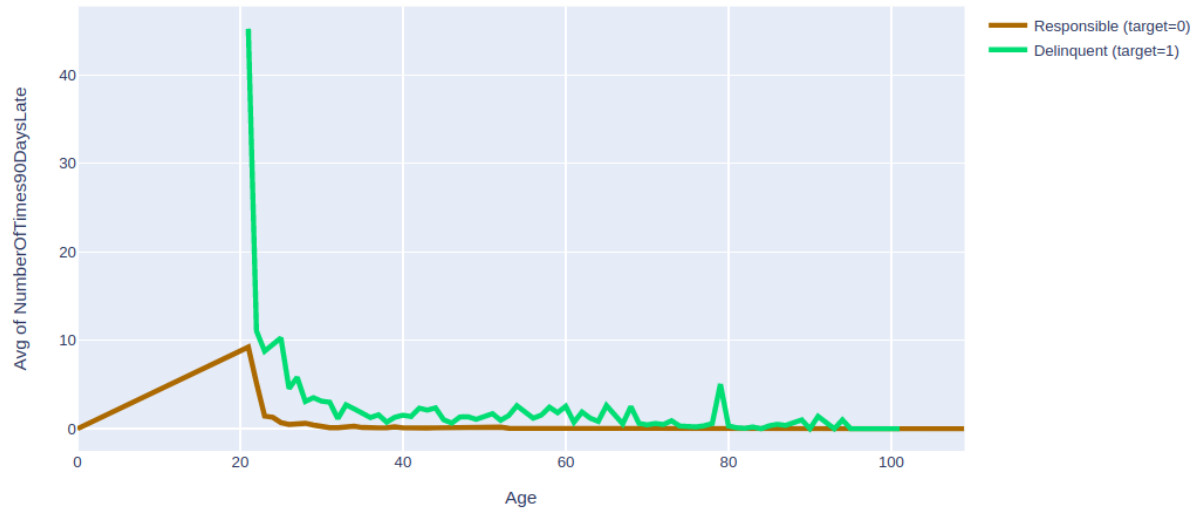


***Figura 10.*** *Relación entre NumberOfTimes90DaysLate (promedio) y Age.*

#### 4.1.3.7. NumberRealEstateLoansOrLines

Podemos observar en el siguiente gráfico que, a partir de los 20 años, la tendencia para el promedio de **NumberRealEstateLoansOrLines** se respeta hasta los 65 años aproximadamente, luego en gran parte, la gráfica para **Delinquent (target = 1)** parece crecer con respecto a **Responsible (target=0)**, salvo la parte final, a partir de los 100 años.


***Figura 11.*** *Relación entre NumberRealEstateLoansOrLines (promedio) y Age.*

#### 4.1.3.8. NumberOfTime60-89DaysPastDueNotWorse

En este gráfico podemos apreciar que quienes están etiquetados como **Responsible (target = 0)** suelen tener niveles más bajo de este feature que los que están etiquetados como

**Delinquent (target = 1)**, es decir, quienes tienden a ser responsables se endeudan, en promedio, menos veces que los que delinquen.

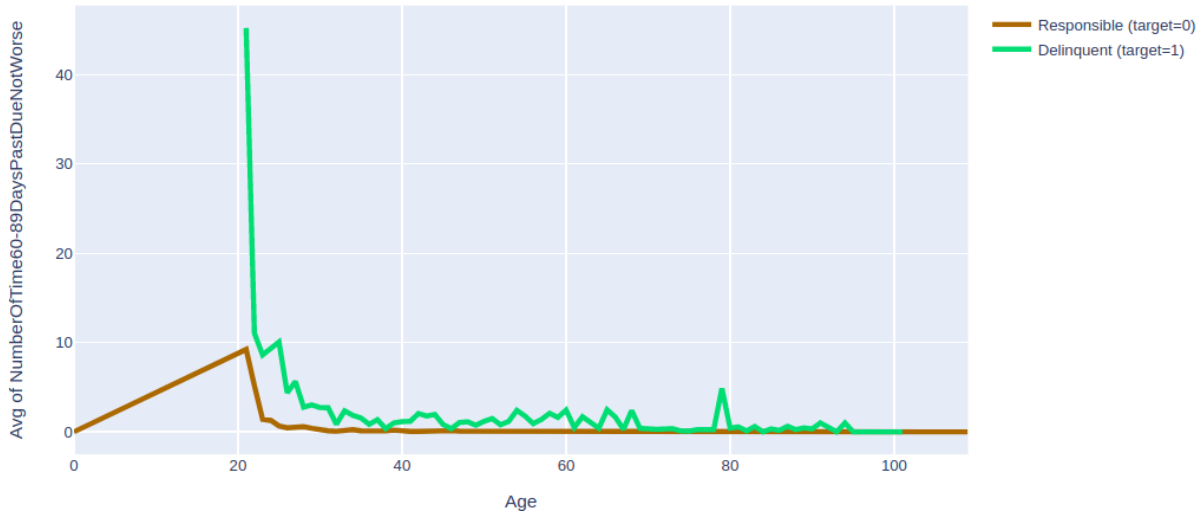


***Figura 12.*** *Relación entre NumberOfTime60-89DaysPastDueNotWorse (promedio) y Age.*

#### 4.1.3.9. NumberOfDependents

Podemos observar presencia de outliers en la primera parte del gráfico. Además, se ve una tendencia similar para ambos casos a lo largo del resto del gráfico.


***Figura 13.*** *Relación entre NumberOfDependents (promedio) y Age.*

### 4.1.4. ANÁLISIS DE COMPONENTES PRINCIPALES

A continuación, a modo exploratorio, trataremos de reducir los features del conjunto de datos en 3 componentes aplicando PCA dado que es la máxima cantidad de dimensiones graficable y que además, explican el 57% de la varianza. Luego, nos traeremos del primer conjunto de

datos (previo al PCA) la información del feature **SeriousDlqin2yrs** para poder hacer una distinción entre estos datos, como podemos observar:

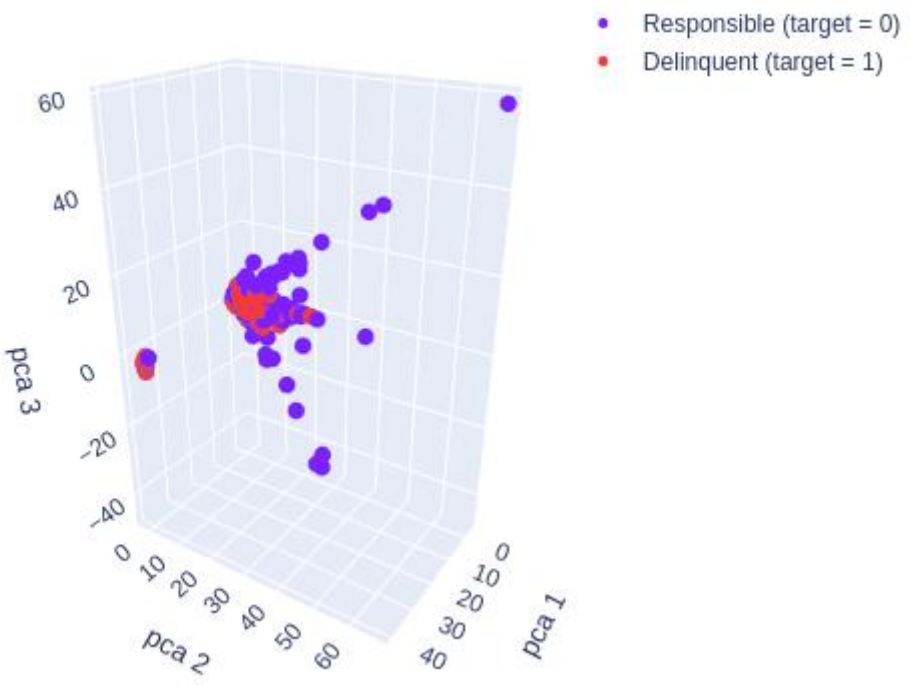


***Figura 14.*** *Scatter plot en 3D de los componentes principales, distinguidos por el valor que toma la variable a predecir, es decir, SeriousDlqin2yrs.*

Respecto al gráfico, podemos validar que en la gran mayoría de los puntos de datos para ambos grupos se cruzan, es decir, “se parecen” entre sí, por lo que podríamos intuir la presencia de ruido o incertidumbre en los datos, lo cual enfatiza la dificultad este ejercicio sobre el ámbito de credit scoring, que requiere de algoritmos complejos y de analítica avanzada para realizar un buen trabajo. Esto hace notar que asignar créditos en este tipo de problemas no es trivial.

### 4.1.5. BOXPLOT - VISUALIZACIONES SOBRE NUMBER OF TIMES

Procederemos a analizar los features **NumberOfTime30-59DaysPastDueNotWorse**, **NumberOfTimes90DaysLate** y **NumberOfTime60-89DaysPastDueNotWorse,** sobre los cuales se puede apreciar una notable presencia de valores outliers, con lo que se puede intuir que estos features podrían contener información útil, ya que, parece ser que la gente que ya tuvo comportamientos morosos en el pasado, vuelve a tenerlos en el futuro.

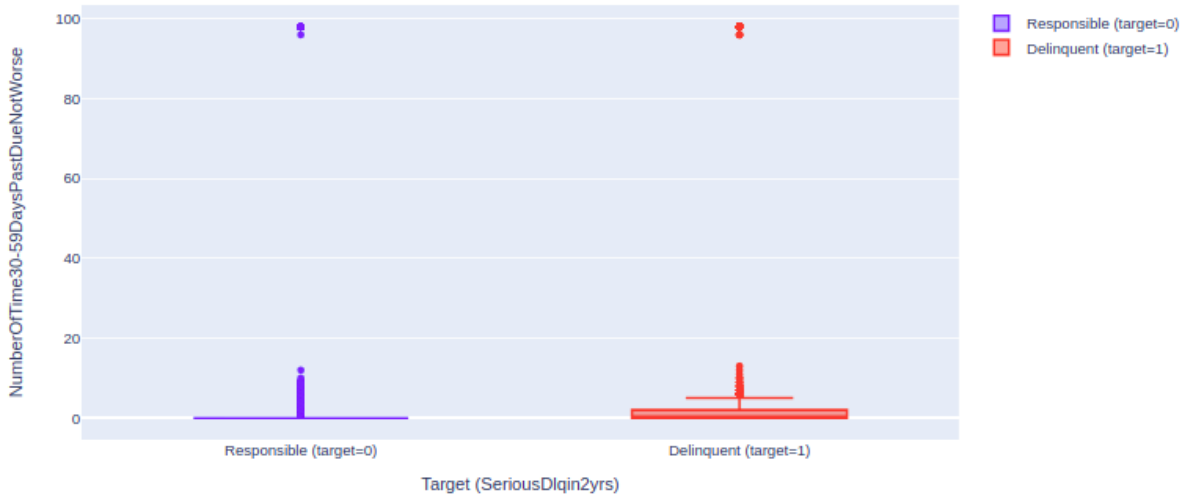


***Figura 15.*** *Boxplot sobre NumberOfTime30-59DaysPastDueNotWorse.*

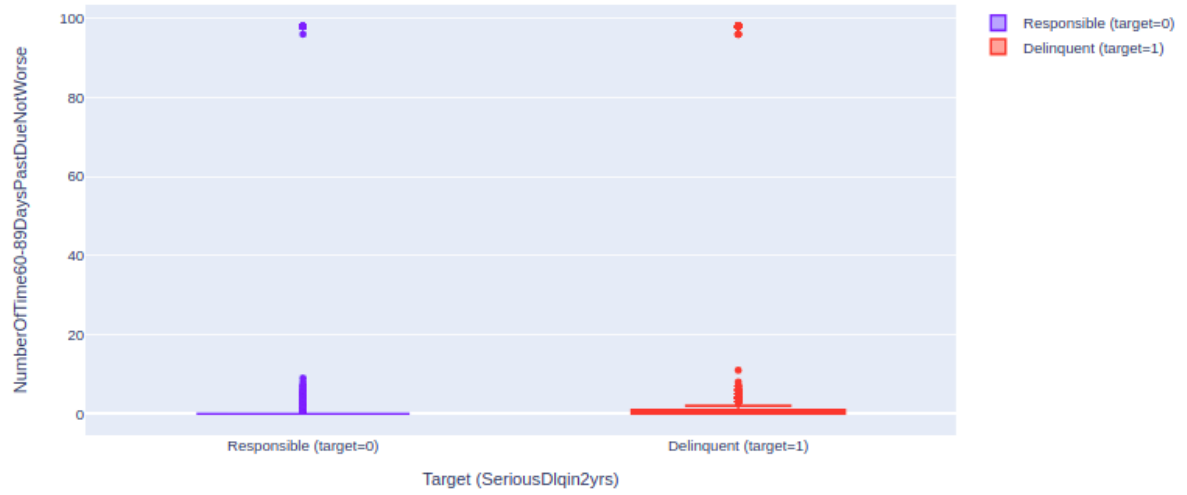


***Figura 16.*** *Boxplot sobre NumberOfTime60-89DaysPastDueNotWorse.*

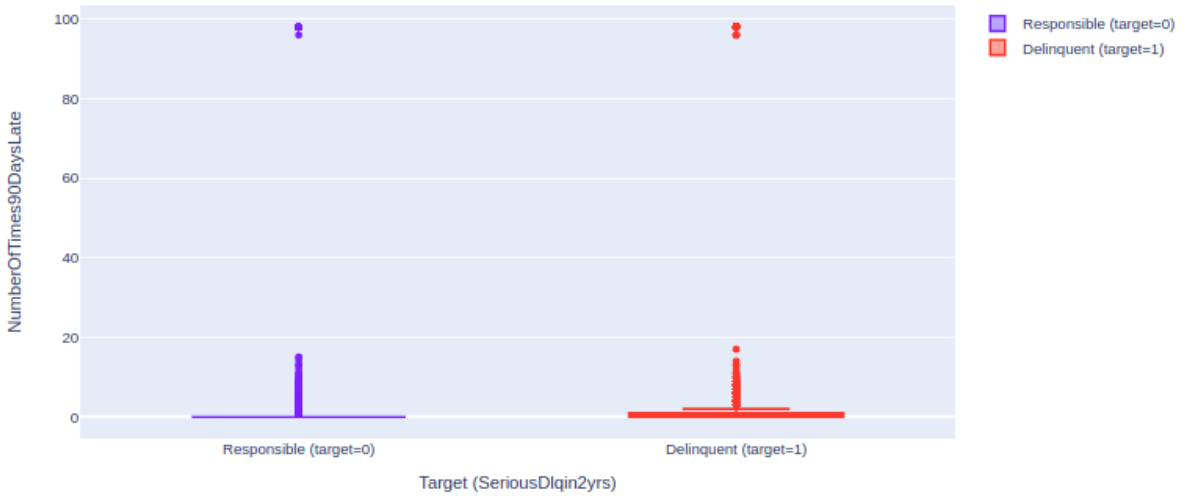


***Figura 17.*** *Boxplot sobre NumberOfTimes90DaysLate.*

#### 4.1.5.2. Age

En el siguiente gráfico podemos observar un KDE, sobre el cual podemos deducir por ejemplo que la gente que no paga suele ser más joven. O por ejemplo, en caso de que estén formando una familia, entonces sería lógico que tengan más problemas para pagar préstamos bancarios.

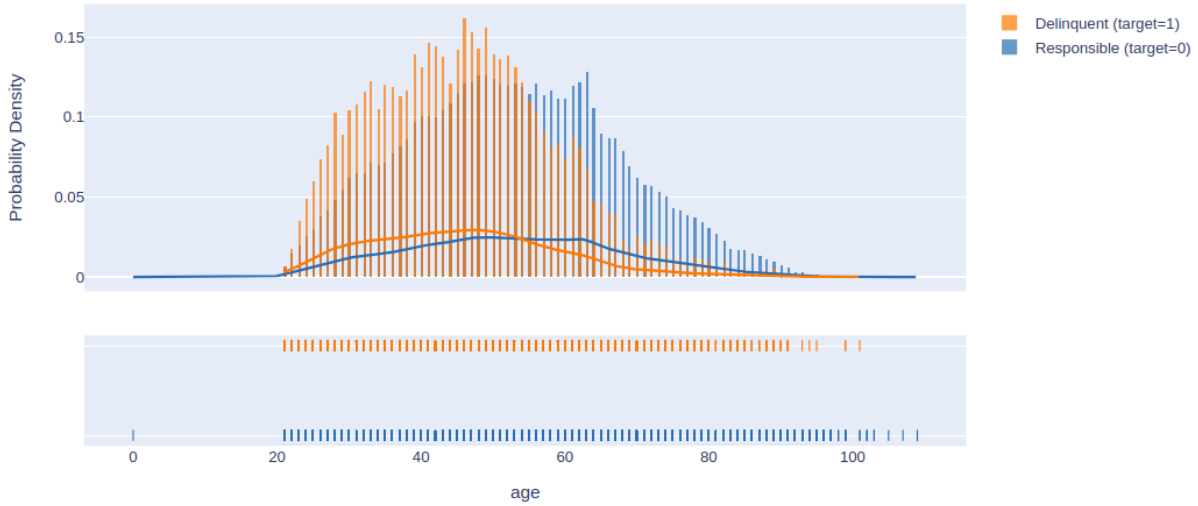


***Figura 18**. KDE plot de Age acorde al target.*

#### 4.1.5.3. NumberOfDependents

En la siguiente figura podemos observar que si el prestatario no tiene personas a cargo, es más probable que cumpla con los plazos del préstamo, a medida que este número incrementa es más probable que no pueda pagar a tiempo. Los últimos números sobre el eje x podrían indicar que se tratan de valores outliers. Es válido aclarar que la periodicidad vista en el siguiente gráfico se debe a que los valores que asume este feature son discretos, teniendo en cuenta que los valores están normalizados.

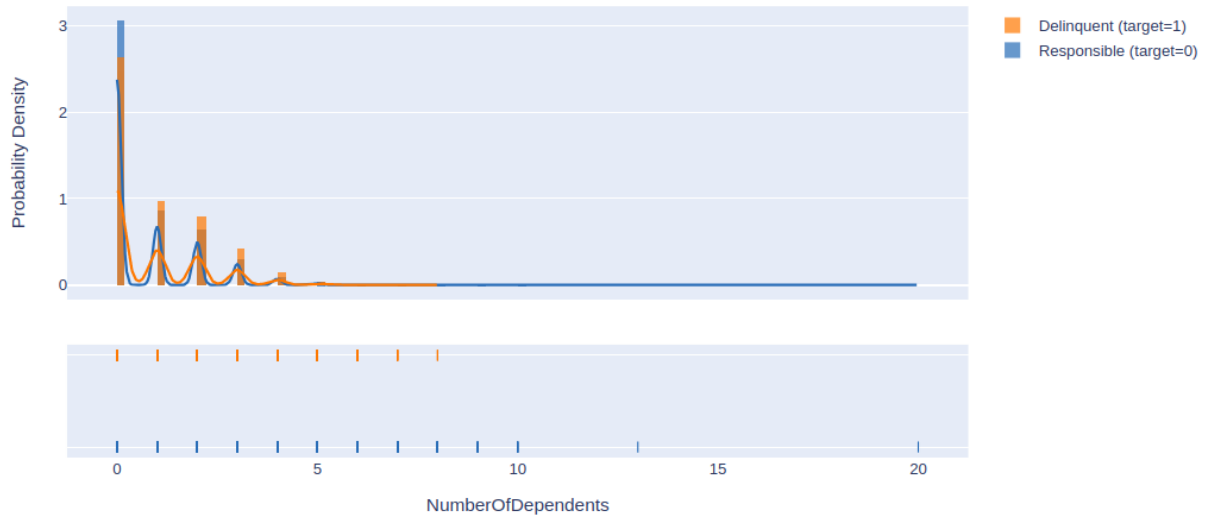


***Figura 19**. KDE plot de NumberOfDependents acorde al target.*

### 4.1.6. RELACIÓN ENTRE DebtRatio y MonthlyIncome

Previo al armado de esta visualización se computó logaritmo base 10 para achicar la magnitud para ambos features. Teniendo en cuenta la fórmula de **DebtRatio** anteriormente detallada, podemos observar en este gráfico la relación inversa entre ambos atributos. Es decir, por ejemplo, si la persona no posee ingresos mensuales es mucho más propenso a tener un **DebtRatio** muy alto y a medida que **MonthlyIncome** es mayor, dicho ratio va disminuyendo.

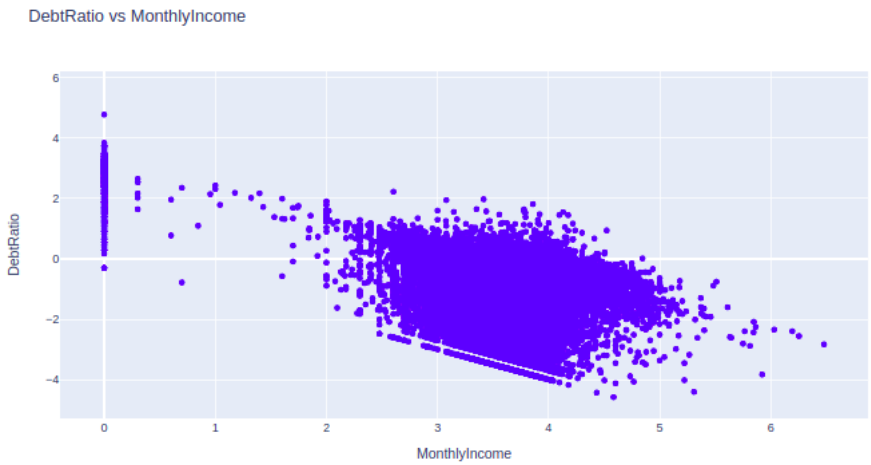


***Figura 20.** Scatter plot de DebtRatio vs MonthlyIncome.*

Para cuantificar la relación entre ambas variables se calculó el coeficiente de correlación de Spearman, que contempla relaciones no lineales entre las variables, el cual nos dió un resultado de -0.493.

## 4.2. FEATURE ENGINEERING

### 4.2.1. VALORES MISSING

Se encontraron valores missings para los siguientes features **MonthlyIncome** y **NumberOfDependents**. Para ambos casos se amputó un valor de cero para contar con un dato numérico en reemplazo a ese valor faltante. Además, implementamos una lógica para que nuestro modelo pueda identificar este reemplazo. A modo de ejemplo mostraremos un pequeño caso y la aplicación de la lógica anteriormente nombrada.

Teniendo como input lo siguiente:

| age | MonthlyIncome |
|---|---|
| 24 | 100 |
| 34 | ? |

***Tabla 13.*** *Mini ejemplo práctico para ilustrar cómo se reemplazan los valores missings.*

Definimos nuestra lógica, la cual fue escrita en código Python de la siguiente manera:

```
def register_imputation(df):
    """
    Register imputations of certain df
    Args:
        - df (DataFrame): Dataframe to be computed
    Return df with filled values and booleans that indicate if each row was changed
    """
    for c in cols[1:]:
        df[f"{c}_imputed"] = df[f"{c}"].fillna(0)
        df[f"dummy_{c}"] = (df[f"{c}"] != df[f"{c}_imputed"]).astype(int)
        del df[f"{c}"]
        df.columns = df.columns.str.replace(f"{c}_imputed", f"{c}")
    return df
```

***Figura 21.*** *Código Python que ilustra la definición de la función que rellena los missings con cero y registra dicha ocurrencia de valores faltantes en una nueva columna.*

Con esta función, rellenamos los valores missings con cero y registramos para cada observación la ocurrencia de missings en cada columna del conjunto de datos, ignorando **SeriousDlqin2yrs**, que es la variable a predecir. Entonces, obtenemos el siguiente output:

| age | MonthlyIncome | dummy_age | dummy_MonthlyIncome |
|---|---|---|---|
| 24 | 100 | 0 | 0 |
| 34 | 0 | 0 | 1 |

***Tabla 14.*** *Resultado final del proceso de identificación de presencia de valores missings en cada observación y para cada columna del conjunto de datos.*

### 4.2.2. LOCAL OUTLIER FACTOR (LOF)

Se aplicó este algoritmo con el fin de limpiar valores outliers, tomando como parámetro un nivel de contaminación del 10%.

En cuanto a resultados obtenidos en los datos aplicando este algoritmo, podemos observar, en la siguiente sección, visualizaciones de algunos features del conjunto de datos sin outliers, con lo que se puede concluir que, al considerar todas las dimensiones del conjunto de datos con **LOF**, obtendremos datos un poco más consistentes y por consiguiente, el modelo podría ser más robusto.

A fines prácticos, es válido recalcar que la aplicación de este proceso de limpieza de outliers mejoró notablemente la performance del modelo.

Debido a que, a pesar de haber aplicado **LOF**, aún persiste la existencia de outliers, aplicamos una limpieza extra sobre los features que representan porcentajes, es decir, **DebtRatio** y **RevolvingUtilizationOfUnsecuredLines**. Esta limpieza extra consiste en rellenar con 0 aquellos valores para **DebtRatio** que son mayores a 10 y rellenar con 0 aquellos valores para **RevolvingUtilizationOfUnsecuredLines** que son mayores a 1.

Para dar visibilidad a esto, representaremos los datos mediante un box plot de los siguientes estados:

1. Con los datos crudos, es decir, sin limpiar.

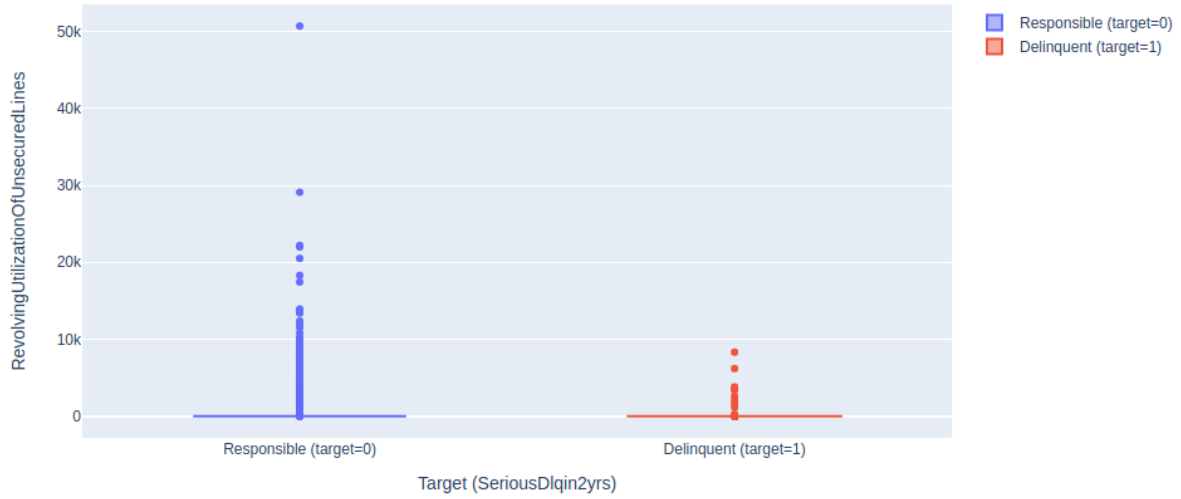


***Figura 22.*** *Box plot con datos crudos para analizar RevolvingUtilizationOfUnsecuredLines separando por el valor del target.*

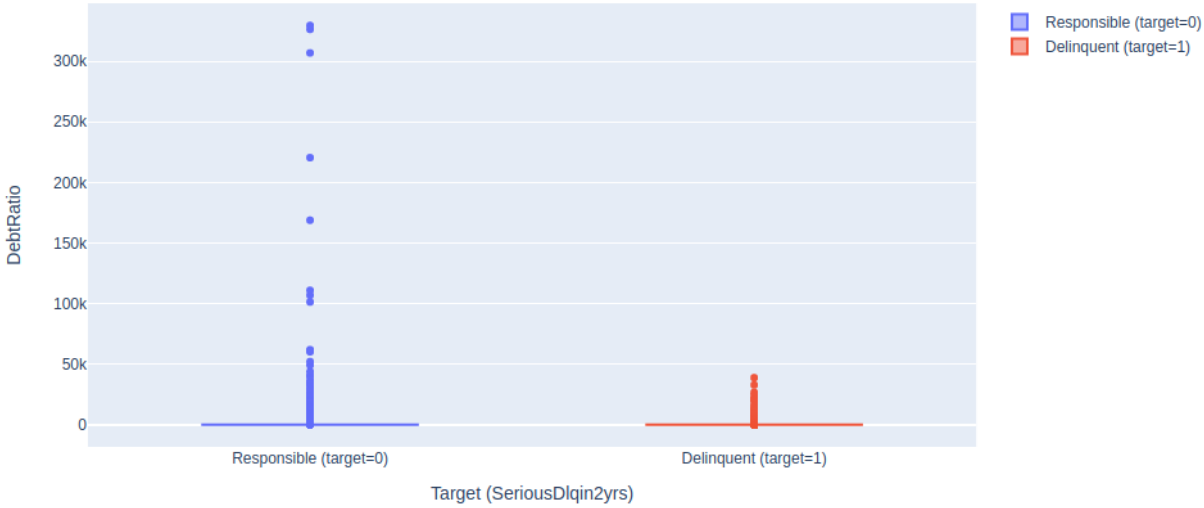


***Figura 23.*** *Box plot con datos crudos para analizar DebtRatio separando por el valor del target.*

2. Con los datos limpiados por **LOF**.

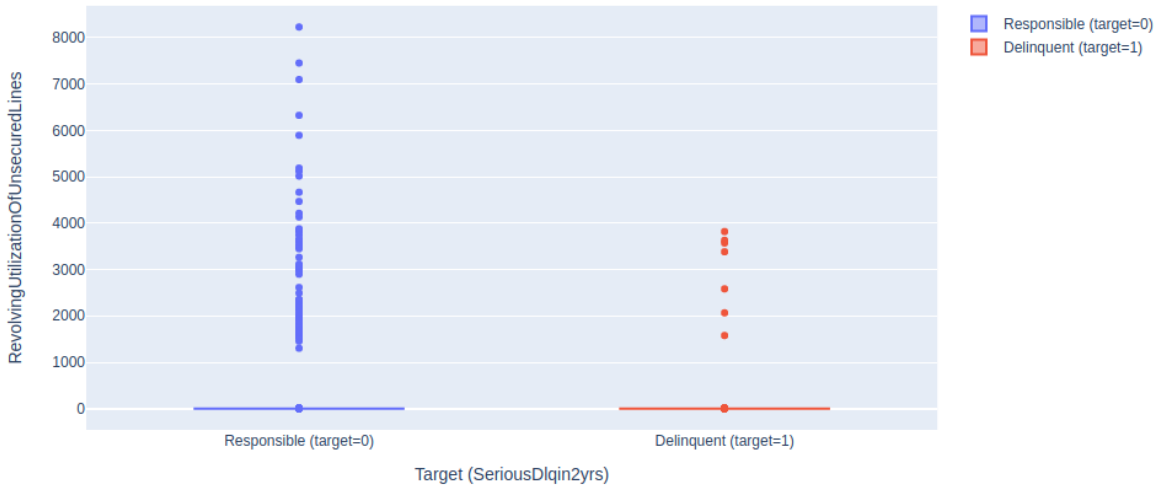


***Figura 24.*** *Box plot con datos procesados con LOF para analizar RevolvingUtilizationOfUnsecuredLines separando por el valor del target.*

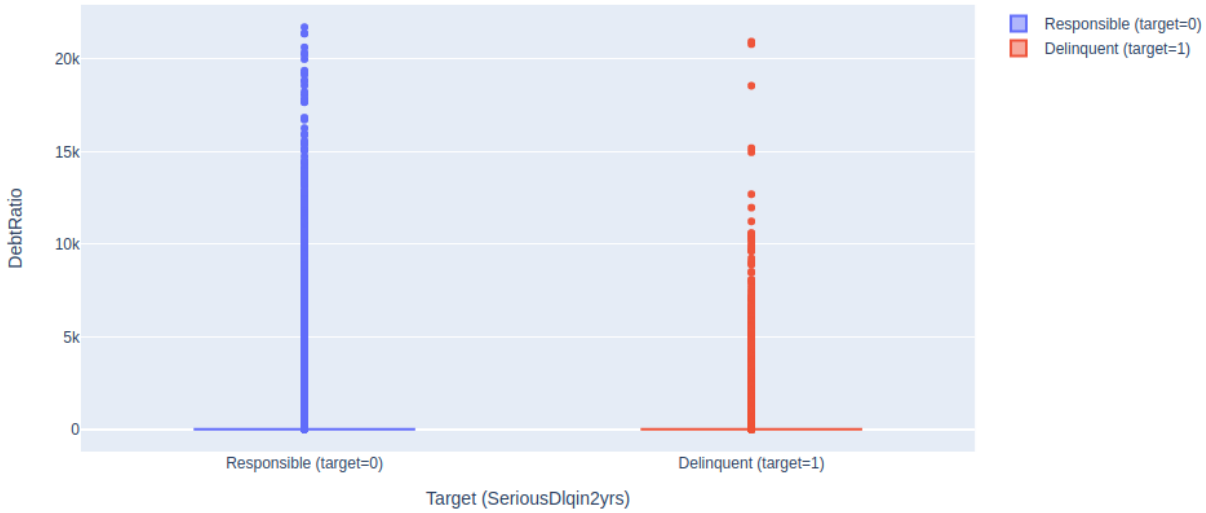

***Figura 25.*** *Box plot con datos procesados con LOF para analizar Debt Ratio separando por el valor del target.*

3. Con los datos limpiados por **LOF** y además con la limpieza extra.

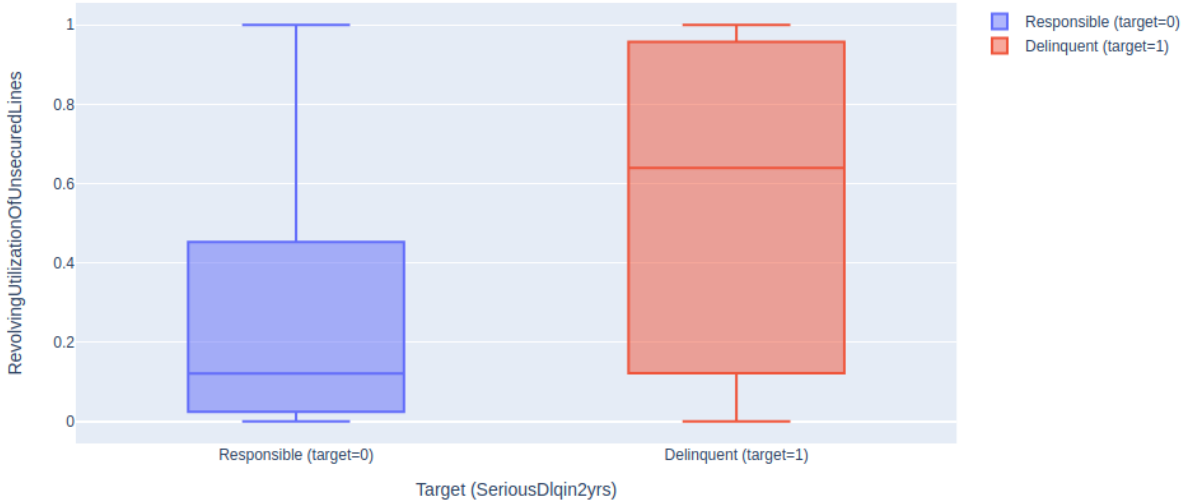


***Figura 26.*** *Box plot con datos procesados con LOF y una limpieza extra para analizar RevolvingUtilizationOfUnsecuredLines separando por el valor del target.*

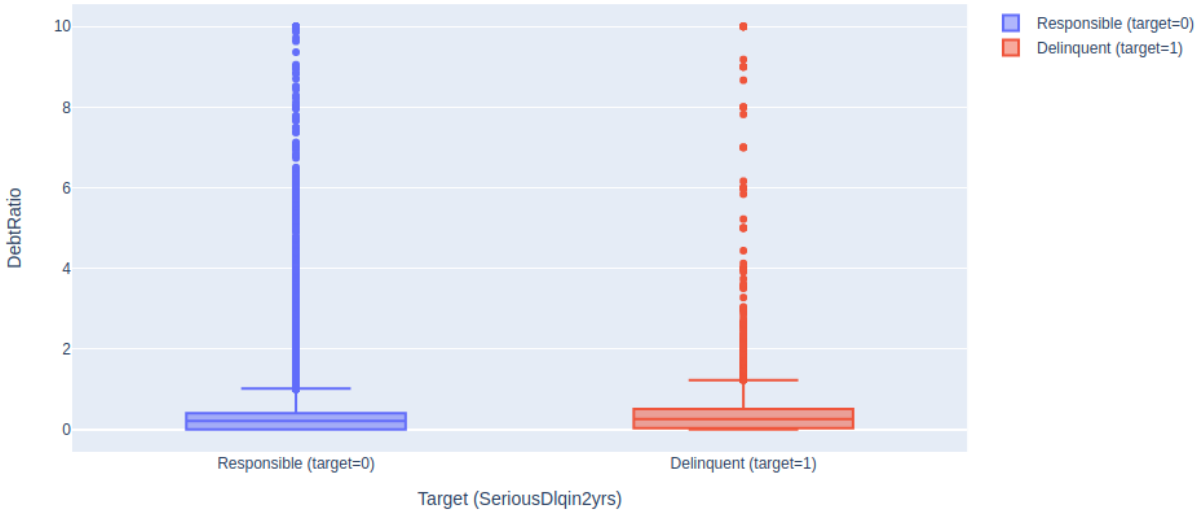


***Figura 27.*** *Box plot con datos procesados con LOF para analizar DebtRatio separando por el valor del target.*

### *4.2.2.1. VISUALIZACIONES CON DATOS SIN OUTLIERS*

Al igual que cuando contábamos con los datos sin limpiar, representaremos nuevamente algunas visualizaciones para hacer notar los resultados del proceso de limpieza de outliers con **LOF**.

Podemos observar en gran parte del siguiente gráfico que quienes asumen mayores deudas posiblemente tiendan a delinquir.

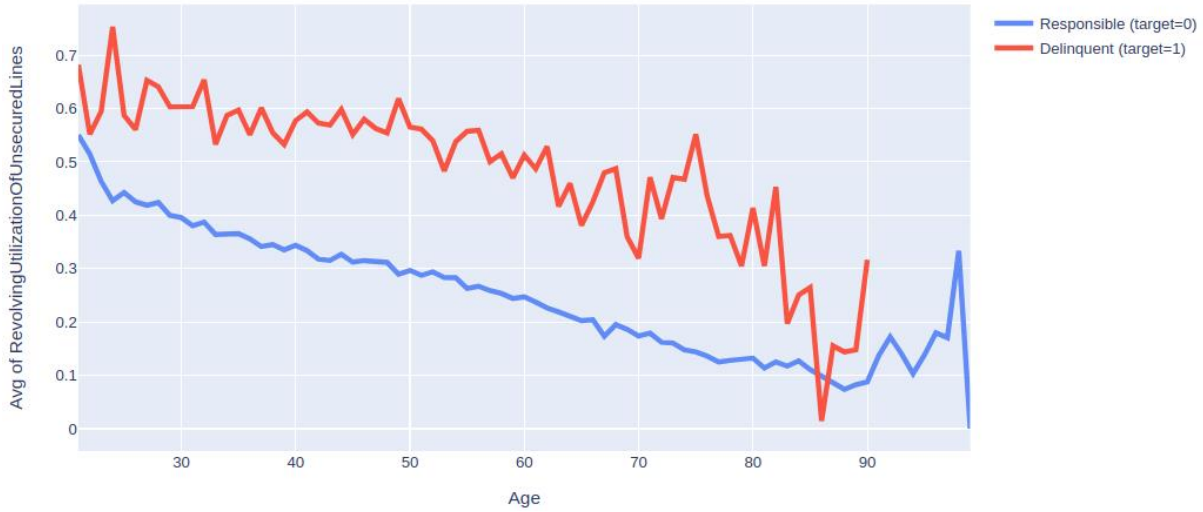


***Figura 28.*** *Scatter line por target del promedio de RevolvingUtilizationOfUnsecuredLines en función de la edad (datos sin outliers).*

Del siguiente gráfico podemos hacer la misma conclusión del gráfico anterior, se puede apreciar mucho mejor este gráfico respecto al anterior que poseía un pico alto de outlier en un punto, lo cual dificulta el análisis y la identificación de algún patrón.


***Figura 29.*** *Scatter line por target del promedio de DebtRatio en función a la edad (datos sin outliers).*

A continuación, para el siguiente gráfico podemos observar que en algún punto, a medida que la edad crece, el promedio de **MonthlyIncome** también crece. Esto además, hace notar que quienes tienden a tener mayor ingreso mensual, es posible que sean más responsables a la hora de pagar el crédito. A su vez, se pueden observar después de la mitad de la curva, varios casos de delincuencia a medida que decrece el promedio del ingreso mensual conforme al incremento de la edad.

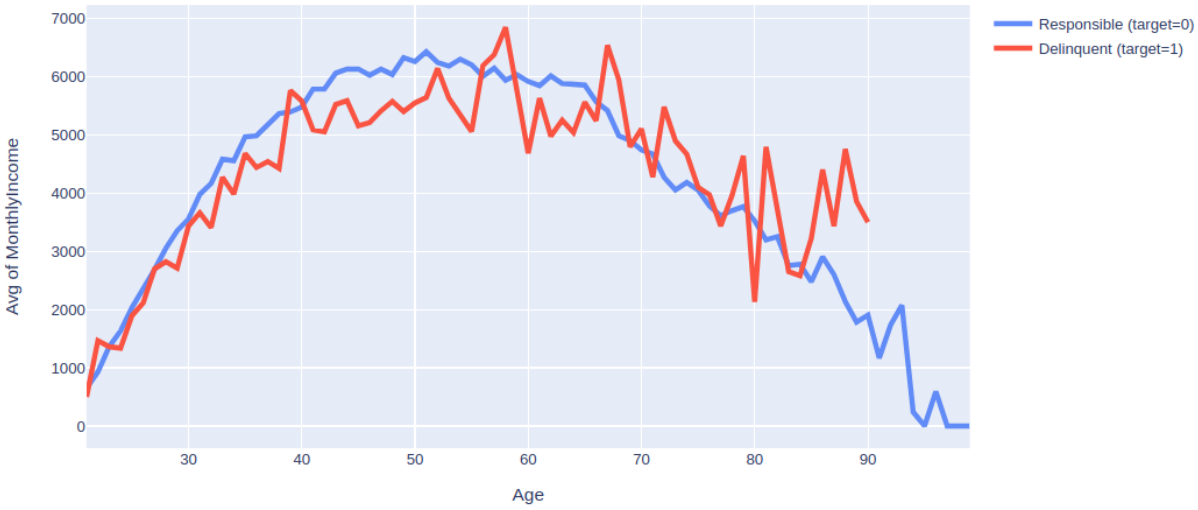


***Figura 30.*** *Scatter line por target del promedio de MonthlyIncome en función a la edad (datos sin outliers).*

Para el siguiente gráfico podemos concluir que posiblemente quienes tengan una edad mayor a 60 años tienen, en promedio, una cantidad mayor de líneas de créditos y préstamos abiertos y que además posiblemente delinquen, a diferencia de quienes tienen, en promedio, menores cantidades para esas edades, que sí cumplen responsablemente con el pago del crédito.

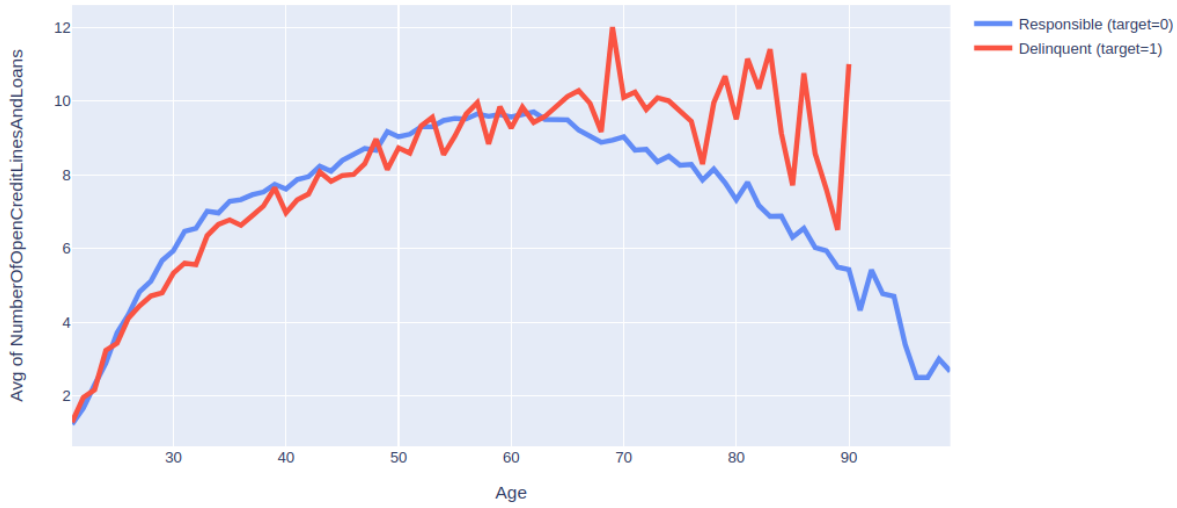


***Figura 31.*** *Scatter line por target del promedio de NumberOfOpenCreditLinesAndLoans en función de la edad (datos sin outliers).*

Este último gráfico indica que quienes, en promedio, tienen más personas a cargo y menor edad, es posible que delinquen. En gran parte del gráfico hay casos donde a mayor cantidad de personas a cargo (promedio), entonces posiblemente sean malos pagadores.

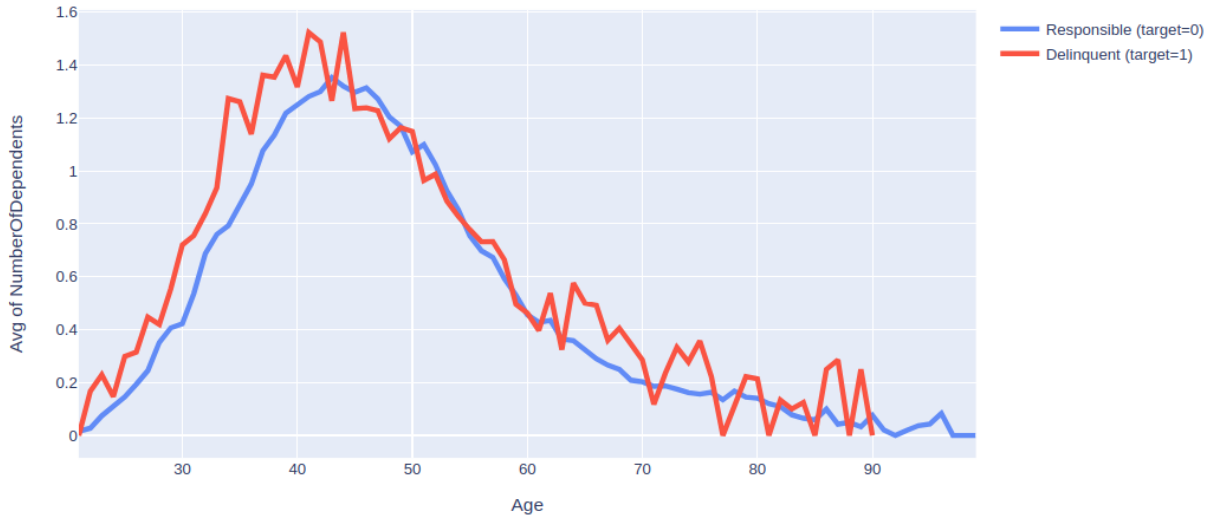


***Figura 32.*** *Scatter line por target del promedio de NumberOfDependents en función de la edad (datos sin outliers).*

### 4.2.3. PREPARACIÓN DE DATOS (PREVIO AL ENTRENAMIENTO DEL MODELO)

Se decidió aplicar el split de train y test dejando un 20% para test.

Como se pudo observar anteriormente, el conjunto de datos presenta un claro desbalance, con lo cual, previo al entrenamiento del modelo, aplicamos una técnica de oversampling llamada Synthetic Minority Over-sampling Technique (**SMOTE**) sobre los datos de train, con lo cual obtuvimos un total de 106,781 observaciones para **Responsible (target=0)** y 106,781 observaciones para **Delinquent (target=1)**.

Otro factor determinante es la obtención de los pesos (**sample_weights**) asignados a cada target a la hora de entrenar, para mejorar la clasificación del modelo en este problema con datos desbalanceados, los cuales están determinados de la siguiente manera:

| Weight para Responsible (target=0) | Weight para Delinquent (target=1) |
|---|---|
| 0.53380283 | 7.89583045 |

***Tabla 15.*** *Tabla que ilustra los pesos asignados para cada target, teniendo en cuenta su desbalance.*

Una aclaración al respecto es que se tomó estos valores como fijo (lo que es una limitación del presente trabajo) ya que explorar dicho hiper parámetro implica una complejidad algorítmica mucho mayor para el entrenamiento del modelo **NGBoost** en este trabajo. De igual modo, podría ser interesante explorar este hiper parámetro en el futuro.

## 4.3. CONSIDERACIONES ANTES DE LOS EXPERIMENTOS

El ejercicio consiste en analizar la performance de la regla de asignación de crédito en el caso en que existan potenciales clientes donde hay mucha incertidumbre sobre su capacidad de pago, pero también interés en aprender sobre ellos.

La idea central de esta aplicación consiste en esconder información al clasificador. Por ejemplo, dejando missing ciertos predictores con una probabilidad que depende de qué tan nuevo es el usuario del sistema financiero que está en ese registro. A modo de ejemplo, un caso de esto, sería que la probabilidad de que **MonthlyIncome** esté missing baje con la edad de la persona. Teniendo en cuenta esto, los features posibles a modificar podrían ser el ingreso mensual o algun feature con capacidad predictiva importante.

Una vez creado el dataset con missings, se le pide a un modelo de boosting que separe en buenos y malos deudores. Se calcula el profit de lo anterior para la empresa. Para esto, necesitamos cuantificar los errores: dar crédito cuando no conviene darlo (generando una pérdida del default) y no dar cuando debería haberlo dado (generando una pérdida de los fees). Además de calcular en el segundo experimento, también se hará este cálculo para el primer experimento, a fin de poder comparar ambos.

Luego, se aplica el modelo de **NGBoost** para calcular incertidumbre en cada pronóstico y se intenta optimizar la clasificación tomando como función objetivo clasificar bien, más explorar dónde hay incertidumbre. Se calcula el profit de esta estrategia.

## 4.4. PRIMER EXPERIMENTO

Comenzaremos nuestro primer experimento detallando una métrica de negocio que se armó a fin de vincularla con el modelo que entrenaremos, optaremos por aquellos hiper parámetros del modelo entrenado tales que minimicen dicha métrica. Sumado a esto, hay que tener en cuenta que este problema de clasificación a tratar es sensible al costo, es decir, además de explorar los diferentes hiper parámetros, buscaremos un threshold que también minimice dicha métrica

Teniendo en cuenta de que la competencia fue creada en 2011 y que no sabemos el país de donde provienen los datos, nos basaremos en una tasa de interés aplicada sobre una moneda

estable como el euro, por lo que adoptaremos una tasa de interés del 1%[46]. Otro factor a asumir es el valor del crédito a otorgar, el cual será 2 veces el ingreso mensual de esa persona.

Por lo tanto, en base a nuestros supuestos procederemos a formular nuestra matriz de costos, considerando una tasa de interés (r) y un monto de crédito asignado (LP).

Existen 4 casos ante nuestro caso de uso:

1. Caso 1: El prestatario no es deudor **(Responsible - y_real = 0)** y decidimos otorgarle el crédito **(No default - y_pred = 0)**, en este caso, ganamos el interés sobre el monto prestado, es decir $r\ x\ LP$.
2. Caso 2: El prestatario no es deudor **(Responsible - y_real = 0)** y decidimos no otorgarle el crédito **(Default - y_pred = 1)**, con lo cual, perdemos $r\ x\ LP$.
3. Caso 3: El prestatario es deudor **(Delinquent - y_real = 1)** y decidimos otorgarle el crédito **(No default - y_pred = 0)**, por lo tanto, perdemos lo que otorgamos, que equivale a $LPx\ (1 + r)$.
4. Caso 4: El prestatario es deudor **(Delinquent - y_real = 1)** y decidimos no otorgarle el crédito **(Default - y_pred = 1)**, con lo cual, ni ganamos, ni perdemos.

Dado estos 4 casos podemos formular la siguiente matriz de costos:

| Target Value/Decision | No default (y_pred = 0) | Default (y_pred = 1) |
|---|---|---|
| **Responsible (y_real = 0)** | r * LP | - r * LP |
| **Delinquent (y_real = 1)** | - LP x (1 + r) | 0 |

***Tabla 16.*** *Tabla que ilustra la matriz de costos que el algoritmo tomará para minimizar la función de costos.*

A modo de ejemplo, suponiendo que tomamos como muestra a una persona con un ingreso mensual de 4,166 euros, que solicita un monto como crédito (dado nuestro supuesto) de 8,322 euros, y además considerando nuestra tasa de interés sobre esta moneda para el año 2011 (que nombramos anteriormente) de 1%, nuestra matriz para este caso sería la siguiente:

| Target Value/Decision | No default (y_pred = 0) | Default (y_pred = 1) |
|---|---|---|

[46] https://www.euribor-rates.eu/en/ecb-refinancing-rate/

| | | |
|---|---|---|
| **Responsible (y_real = 0)** | 0.01 * 8322 = 83.22 | - 0.01 * 8322 = - 83.22 |
| **Delinquent (y_real = 1)** | - 8,322 x (1 + 0.01) = - 8,405.22 | 0 |

***Tabla 17.*** *Tabla con valores definidos a modo de ejemplo para la matriz de costos anteriormente presentada.*

Definiremos matemáticamente el costo unitario, para cada observación $i$ de nuestro conjunto de datos:

$$CostoUnitario_i = \begin{cases} -LP_i \times r & si \quad y_{pred_i} = 0 \vee y_{real_i} = 0 \\ LP_i \times r & si \quad y_{pred_i} = 0 \vee y_{real_i} = 1 \\ LP_i \times (1+r) & si \quad y_{pred_i} = 1 \vee y_{real_i} = 0 \\ 0 & si \quad y_{pred_i} = 1 \vee y_{real_i} = 1 \end{cases}$$

Con esto, podemos definir entonces nuestra función de costos, considerando $i$ a cada observación hasta $n$ que está dado por el total de observaciones del conjunto de datos. Por lo tanto, tenemos lo siguiente:

$$CostoPrimerExperimento = \sum_{i=1}^{N} CostoUnitario_i$$

Para esto, implementamos una función de score de costo “personalizada” que contempla costos unitarios, basado en la matriz de costos anteriormente explicada. Para mayor entendimiento, compartiremos una pequeña porción del código:

```
def process_unit_cost(x, rate):
    """
    Process the unit cost for each observation, according to our cost matrix.

    Args:
        - x (pd.Series): data to be identified with name column (real, predicted or
LoanPrincipal)
        - rate (float): interest rate

    Returns for each case his cost value.
    """
    cost = (
        x["predicted"] * (1 - x["real"]) * x["LoanPrincipal"] * rate
        + (1 - x["predicted"]) * x["real"] * x["LoanPrincipal"] * (1 + rate)
        - x["predicted"] * (x["real"]) * x["LoanPrincipal"] * rate
        + (1 - x["predicted"]) * (1 - x["real"]) * 0
    )
    return cost
```

***Figura 33.*** *Función para procesar costo unitario*

```
def cost_score_first_experiment(loan, y_pred, y_true):
    """
    From input data, generates auxiliar dataframe in order to apply process_unit_cost for each
row and then summarize that.
```

```
    Args:
        - loan (pd.Series): data about the requested amount of money
        - y_pred (pd.Series): list of predictions
        - y_true (pd.Series): list of true values

    Returns sum of unit costs
    """
    aux_df = pd.DataFrame(
        data={"LoanPrincipal": loan, "predicted": y_pred, "real": y_true}
    )
    return sum(aux_df.apply(lambda x: process_unit_cost(x, 0.01), axis=1))
```

***Figura 34.*** *Función de Score de Costo a minimizar*

Por tanto, seguiremos la estrategia de esta función de costo. Para ello, entrenaremos más de 100 modelos **NGBoost** con diferentes hiper parámetros buscando la configuración de hiperparametros que minimicen nuestra métrica de **CostoPrimerExperimento**.

Los hiper parámetros que se optimizarán son **Base**, **n_estimators** y **learning_rate**. Para lo cual realizaremos una exploración de los valores de estos hiper parámetros de forma tal que encontremos un modelo con el menor costo incurrido en sus predicciones, para ello recorreremos distintas posibilidades de valores para los hiper parámetros anteriormente nombrados, entrenando el modelo en cada una de ellas y ajustando un valor de **threshold** (que refiere a $P(y = 1 \mid X)$) que también minimice nuestro costo, moviéndolo desde 0.1 a 0.9, para alterar las predicciones de nuestro modelo (dado que por defecto el considerado es 0.5) y recalcular en cada una de estas iteraciones, el correspondiente valor del costo asociado a las predicciones modificadas.
Esta optimización del threshold se implementa con el siguiente snippet código para mayor entendimiento:

```
def generate_y_pred_with_custom_threshold(model, x_data, threshold):
    """
    Generates new y_predictions according to a threshold.

    Args:
        - model: NGBoost model that was trained.
        - x_data: Data on which we predict probabilities
        - threshold: Float value to determine 1 or 0 for new predictions

    Returns updated y_predictions
    """
    y_predictions = model.predict_proba(x_data)
    y_pred = []
    for i in range(len(list(y_predictions))):
        if y_predictions[i][1] > threshold:
            y_pred.append(1)
```

```
        else:
            y_pred.append(0)
    return y_pred
```

***Figura 35.*** *Función para modificar los y_predictions (en una variable y_pred nueva) acorde a un threshold dado, que en nuestro caso, iterará desde 0.1 a 0.9, sumando 0.05 en cada paso.*

Al generar estas nuevas predicciones (variable **y_pred** de la **Figura 25**), como bien dijimos, calculamos nuevos costos. Todos estos cálculos son almacenados en una dataframe nueva que almacena datos correspondientes a:

- model: Instancia de la clase **NGBoost.Classifier** con los hiper parámetros sobre los cuales entrenó.
- cost: Costo que incurre dicho modelo (dado unos hiper parámetros establecidos) para un determinado threshold.
- threshold: Valor que determina las nuevas predicciones teniendo en cuenta las predicciones cuando P (y = 1 | X), para ponderar el error en la clasificación para dicho valor del target.

### 4.4.1 ANÁLISIS DE LA PERFORMANCE VS HIPER PARÁMETROS DE NGBOOST

Realizaremos el análisis de la performance del modelo **NGBoost**, para lo cual, de los 3 hiper parámetros que exploramos, que son **estimators**, **learning_rate** y **max_depth** del **Base Learner**, tomaremos uno de cada uno, manteniendo constantes los demás hiper parámetros definidos por default en la implementación del paquete python de **ngboost**. Algo a considerar es que el threshold considerado para nuestro análisis es igual a 0.20 para P (y = 1 | X).

#### 4.4.1.1. Evolución de la performance respecto a la cantidad de estimadores

Incrementando la cantidad de estimadores desde 50 a 700, podemos observar que el valor que asume la métrica de costos anteriormente definida a lo largo del gráfico no varía. Este costo es de € 2,817,421.83.

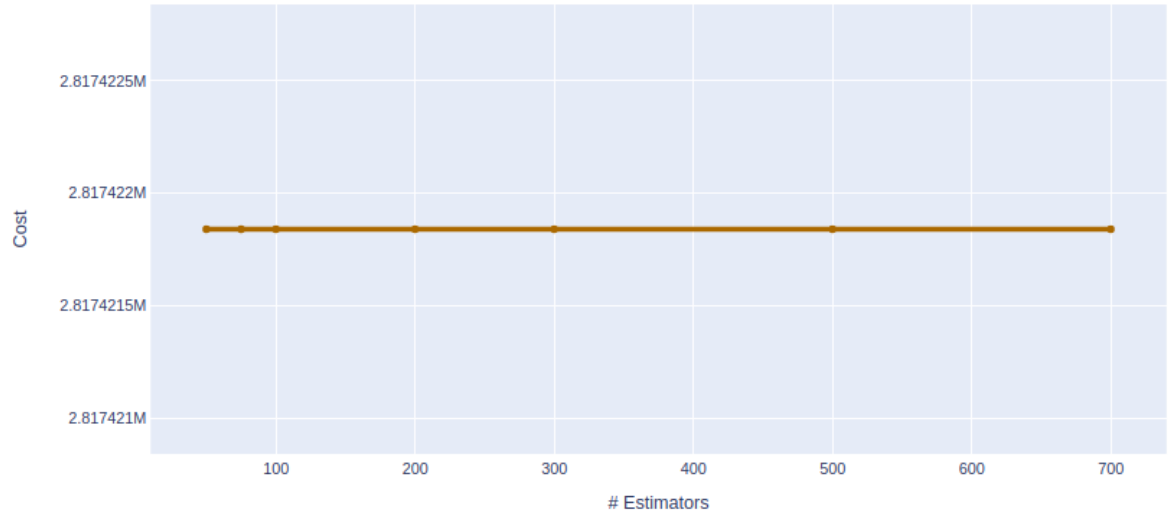


***Figura 36.*** *Gráfico que ilustra la evolución del costo mediante el aumento de la cantidad de estimadores.*

Incrementando la cantidad de estimadores desde 50 a 700, podemos observar que el valor de ambas métricas AUC de train y test incrementan, reduciendo su distancia entre ellas. Luego, a partir de los 300 estimadores aproximadamente podemos observar que el AUC para ambos tiende a ser constante sin mayores cambios.

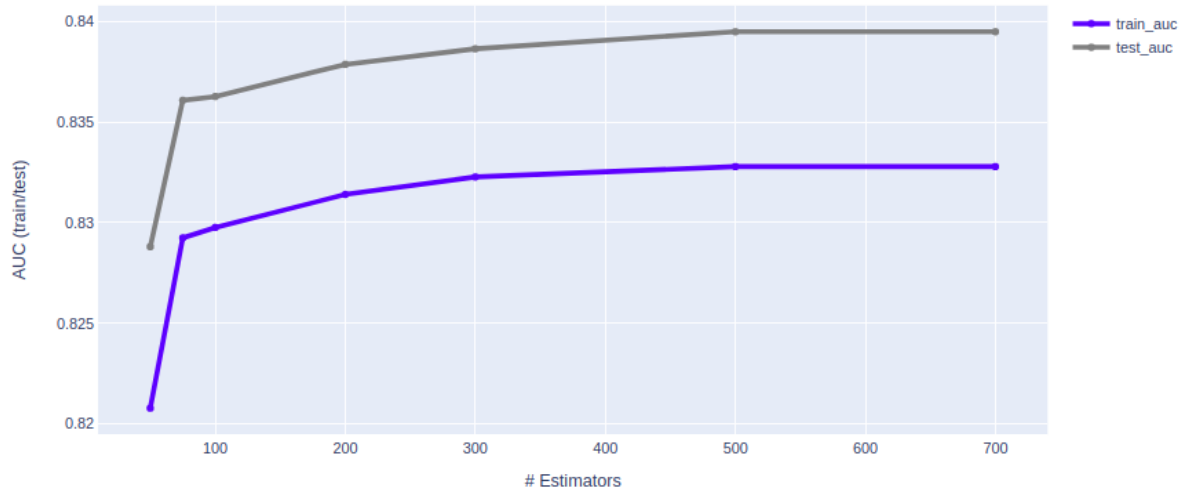


***Figura 37.*** *Gráfico que ilustra la evolución del AUC de train y test mediante el aumento de la cantidad de estimadores.*

#### 4.4.1.2. Evolución de la performance respecto a learning_rate

Incrementando learning_rate desde 0.0001 a 0.1, podemos observar que el valor que asume la métrica de costos anteriormente definida a lo largo del gráfico no varía. Este costo, al igual que en el caso anterior, es de € 2,817,421.83.

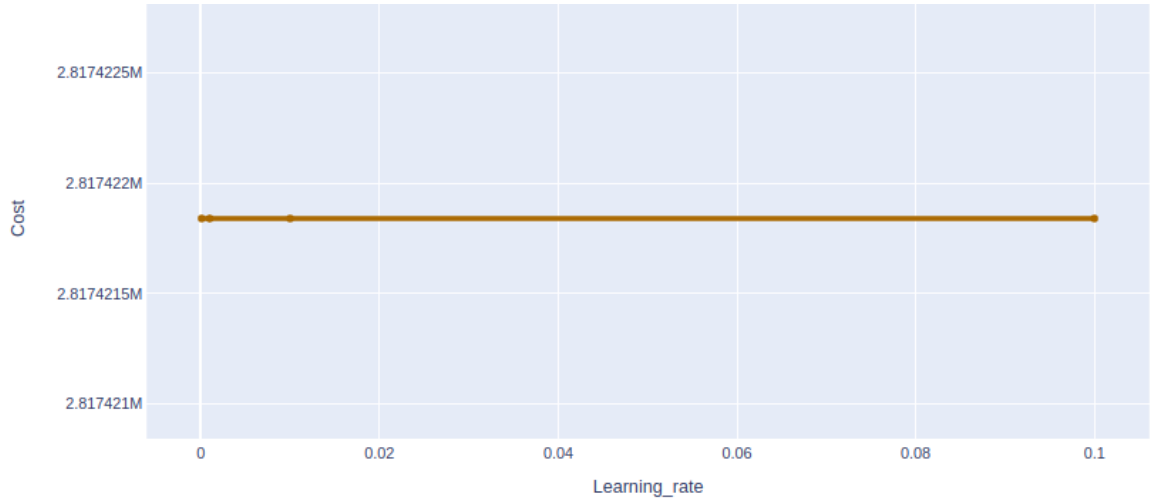


***Figura 38.*** *Gráfico que ilustra la evolución del costo mediante el aumento del hiper parámetro learning_rate.*

Incrementando learning_rate desde 0.0001 a 0.1, podemos observar que las métricas AUC de train y test incrementan notoriamente hasta 0.01 y luego se mantiene constante hasta 0.1.

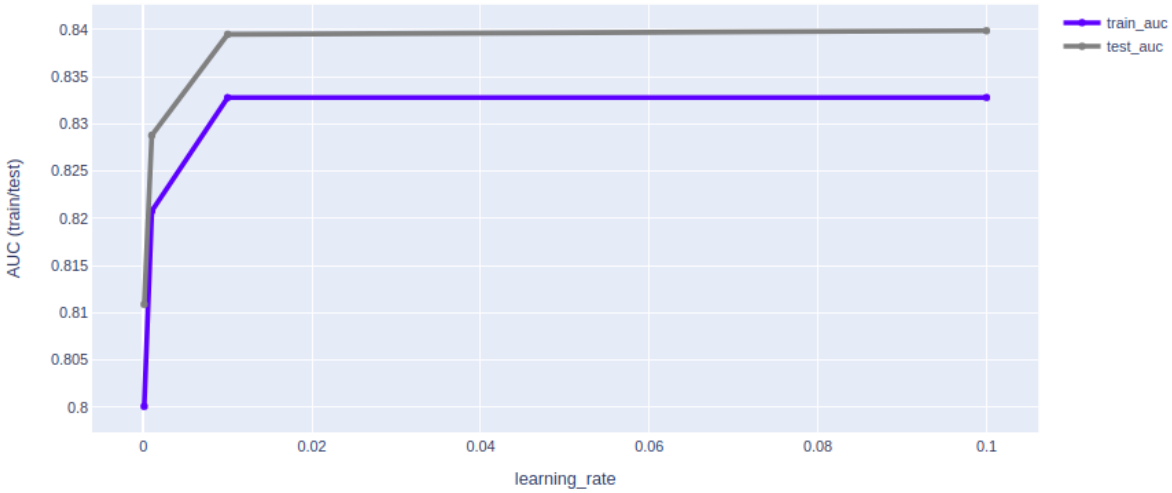


***Figura 39.*** *Gráfico que ilustra la evolución del AUC de train y test mediante el aumento del hiper parámetro learning_rate.*

#### 4.4.1.3. Evolución de la performance respecto a max_depth del Base Learner

Incrementando el hiper parámetro max_depth del Base Learner desde 4 a 16, podemos observar que el valor que asume la métrica de costos (anteriormente definida) al comienzo disminuye hasta max_depth = 8 y luego incrementa notoriamente durante el resto del gráfico.

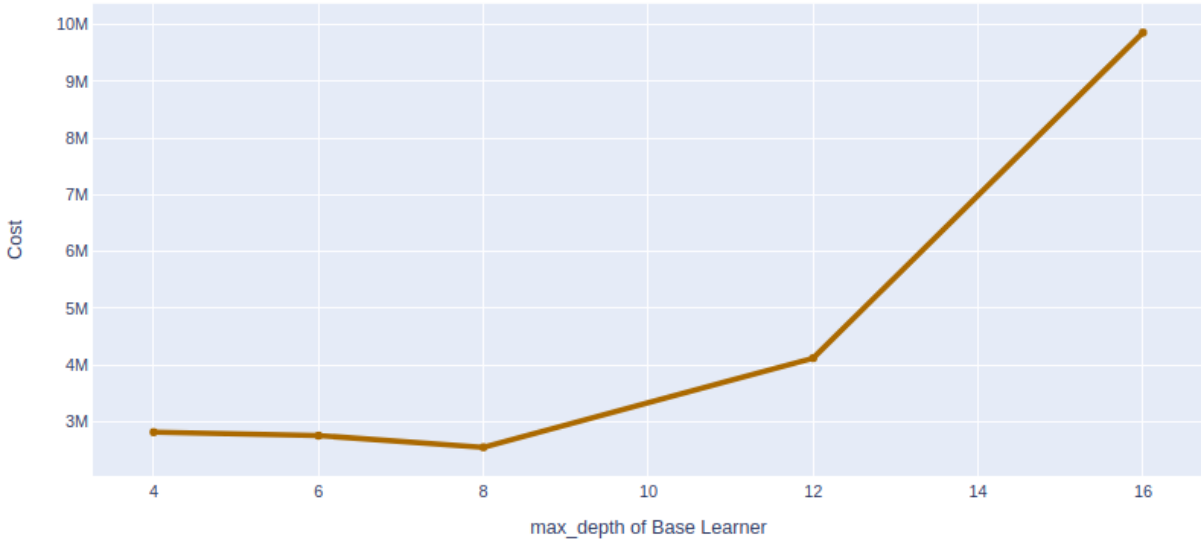


***Figura 40.*** *Gráfico que ilustra la evolución del costo mediante el aumento del hiper parámetro max_depth del Base Learner.*

Incrementando el hiper parámetro max_depth del Base Learner desde 4 a 16, podemos observar que el valor de la métrica AUC de train aumenta notoriamente y el valor de la métrica AUC de test disminuye notoriamente desde que max_depth es igual a 8. Esto podría dar indicio de overfitting, lo cual afecta a nuestro modelo a la hora de hacer predicciones, por lo que hay que considerar la correcta configuración de este hiper parámetro a la hora de entrenar el modelo. Al comienzo del gráfico podemos observar que la performance predictiva en test es mejor que la de train hasta max_depth = 7 apróximadamente.

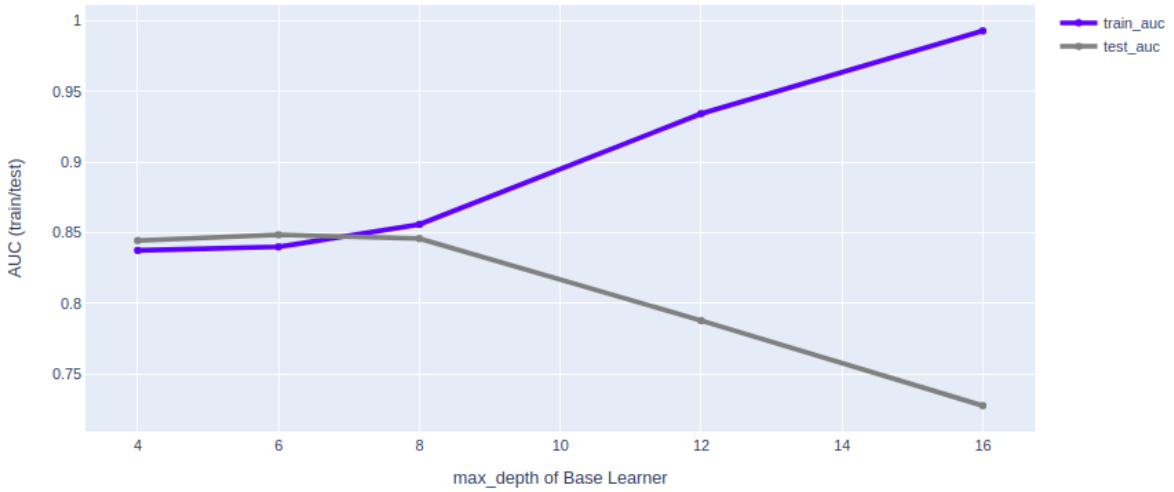


***Figura 41.*** *Gráfico que ilustra la evolución del AUC de train y cost mediante el aumento del hiper parámetro max_depth del Base Learner.*

### 4.4.2. ANÁLISIS DE LOS PRÉSTAMOS ASIGNADOS VS HIPER PARÁMETROS DE NGBOOST

A continuación realizamos el mismo análisis que en el apartado anterior, pero evaluando la cantidad de préstamos asignados (y no asignados). Para lo cual, como comentamos antes, procesaremos la evolución de cada hiper parámetro del modelo en forma individual, manteniendo los demás en su definición por default dada su implementación en Python. Además, haremos la distinción para ambos casos del target, es decir, Responsible **(target = 0)** y **Delinquent (target = 1)**.

Para ver los valores que asumen cada punto en las visualizaciones de las siguientes subsecciones, recurrir al *ANEXO 1 - RESULTADOS DEL PRIMER EXPERIMENTO ITERANDO INDIVIDUALMENTE CADA HIPER PARÁMETRO.*

#### 4.4.2.1. Evolución de préstamos asignados y no asignados respecto a la cantidad de estimadores

Podemos observar que, para ambos casos, estimators no influye en la asignación de estos créditos asignados, manteniéndose esta cantidad constante a lo largo de la iteración. En todo el gráfico, observamos la cantidad de observaciones etiquetadas como **Responsible (target = 0)** por el algoritmo entrenado es nula y para el caso de **Delinquent (target = 1)** aplica a todas las observaciones del conjunto de datos.

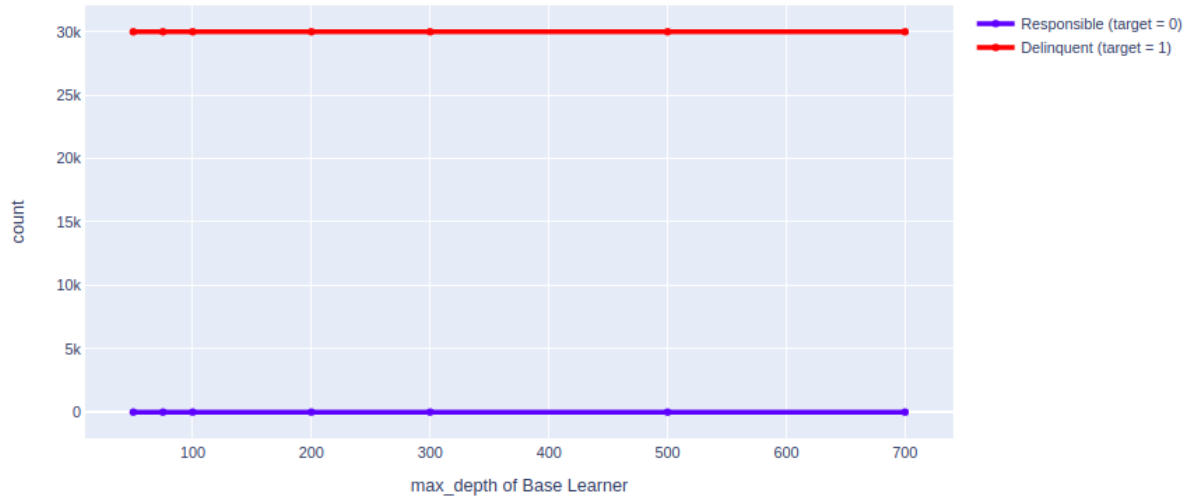


***Figura 42.*** *Gráfico que ilustra la evolución de los créditos asignados para* $\hat{y} = 0$ *y para* $\hat{y} = 1$ *conforme aumenta el valor de la cantidad de estimadores.*

#### 4.4.2.2. Evolución de préstamos asignados y no asignados respecto a learning_rate

Podemos observar que, para ambos casos, learning_rate no influye en la asignación de estos créditos asignados, manteniéndose esta cantidad constante a lo largo de la iteración. En todo el gráfico, observamos la cantidad de observaciones etiquetadas como **Responsible (target = 0)** por el algoritmo entrenado es nula y para el caso de **Delinquent (target = 1)** aplica a todas las observaciones del conjunto de datos.

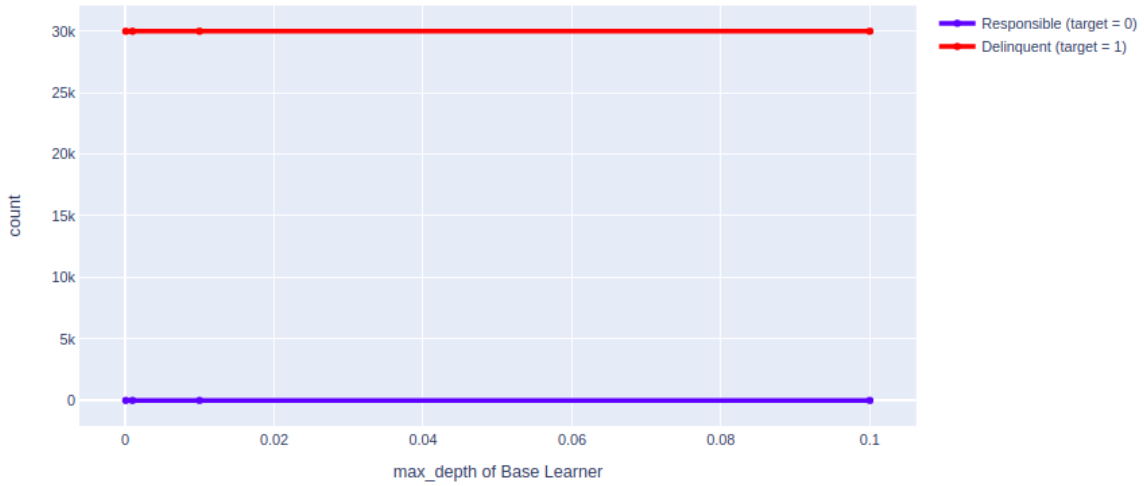


***Figura 43.*** *Gráfico que ilustra la evolución de los créditos asignados para* $\hat{y} = 0$ *y para* $\hat{y} = 1$ *conforme aumenta el valor de learning_rate.*

#### 4.4.2.3. Evolución de préstamos asignados y no asignados respecto a max_depth

A diferencia de los dos análisis anteriores, en este caso si se produce una variación de los créditos asignados, siendo decreciente para **Delinquent (target = 1)** y creciente para **Responsible (target = 0)**. Podemos visualizar además una intersección entre ambas rectas en un valor entre 14,000 y 15,000 para un max_depth entre 12 y 13 (es válido mencionar que en la práctica esto último no es posible de aplicar).

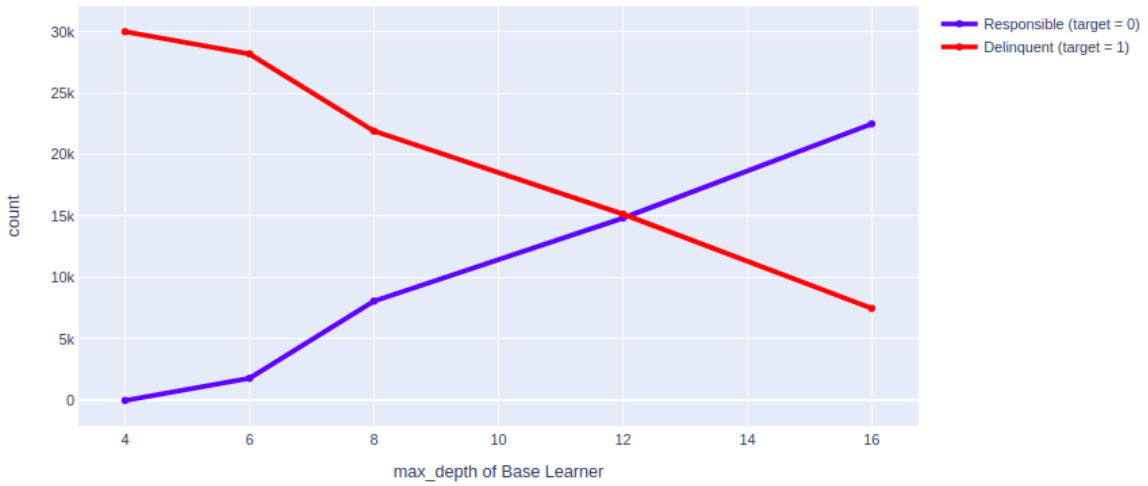


***Figura 44.*** *Gráfico que ilustra la evolución de los créditos asignados para* $\hat{y} = 0$ *y para* $\hat{y} = 1$ *conforme aumenta el valor de max_depth del Base Learner.*

### 4.4.3. MUESTRA DE DATOS PREVIO AL ENTRENAMIENTO

El fin de esta sección es poder ilustrar una muestra para explicar cómo es la estructura de los datos con la cual nuestros modelos **NGBoost** entrenarán, dado a nuestro proceso de Feature Engineering sus valores se vieron normalizados y su definición de esquema también, dado que añadimos columnas, por lo tanto, es necesario recordar que este esquema de datos es distinto al ilustrado en la **Tabla 6**, pero tomó su base a partir de ese esquema.

Dicho esquema nuevo es el siguiente:

1. RevolvingUtilizationOfUnsecuredLines
2. RevolvingUtilizationOfUnsecuredLines_bool_missing
3. age
4. age_bool_missing
5. NumberOfTime30-59DaysPastDueNotWorse
6. NumberOfTime30-59DaysPastDueNotWorse_bool_missing
7. DebtRatio
8. DebtRatio_bool_missing
9. MonthlyIncome
10. MonthlyIncome_bool_missing
11. NumberOfOpenCreditLinesAndLoans
12. NumberOfOpenCreditLinesAndLoans_bool_missing
13. NumberOfTimes90DaysLate
14. NumberOfTimes90DaysLate_bool_missing
15. NumberRealEstateLoansOrLines

16. NumberRealEstateLoansOrLines_bool_missing
17. NumberOfTime60-89DaysPastDueNotWorse
18. NumberOfTime60-89DaysPastDueNotWorse_bool_missing
19. NumberOfDependents
20. NumberOfDependents_bool_missing
21. SeriousDlqin2yrs

Y algunos datos que están en estas columnas son los siguientes:

| | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.07 | 0 | 52 | 0 | 0 | 0 | 0.02 | 0 | 69,000 | 0 | 10 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0.68 | 0 | 29 | 0 | 0 | 0 | 0.48 | 0 | 3,545 | 0 | 7 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 2 | 0 | 0 |
| 0.11 | 0 | 59 | 0 | 0 | 0 | 0.39 | 0 | 43,00 | 0 | 12 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 2 | 0 | 0 |
| 0.94 | 0 | 55 | 0 | 0 | 0 | 0.36 | 0 | 1,800 | 0 | 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0.95 | 0 | 47 | 0 | 0 | 0 | 0.13 | 0 | 4,283 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

***Tabla 18.*** *Muestra de datos con los que los modelos NGBoost entrenarán.*

### 4.4.4. MODELO DEL PRIMER EXPERIMENTO

A continuación, detallaremos los valores de los hiper parámetros obtenidos para nuestro mejor modelo, es decir, el que nos brinda el menor costo para nuestra métrica definida, subrayando el nombre de aquellos que fueron explorados:

| Hiper parámetro | Valores |
|---|---|
| Base | DecisionTreeRegressor(ccp_alpha=0.0, criterion='friedman_mse', max_depth=8, max_features=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, presort='deprecated', random_state=2020, splitter='best') |
| Dist | Bernoulli |
| Score | LogScore (ngboost score module) |
| learning_rate | 0.01 |
| minibatch_frac | 1.0 |

| | |
|---|---|
| n_estimators | 250 |
| natural_gradient | True |
| tol | 0.0001 |

***Tabla 19.*** *Tabla con los hiper parámetros asignados al modelo con el cual entrenamos para realizar nuestras predicciones. Los features que fueron explorados están subrayados.*

### 4.4.5. PERFORMANCE DEL MODELO - AUC Y ROC

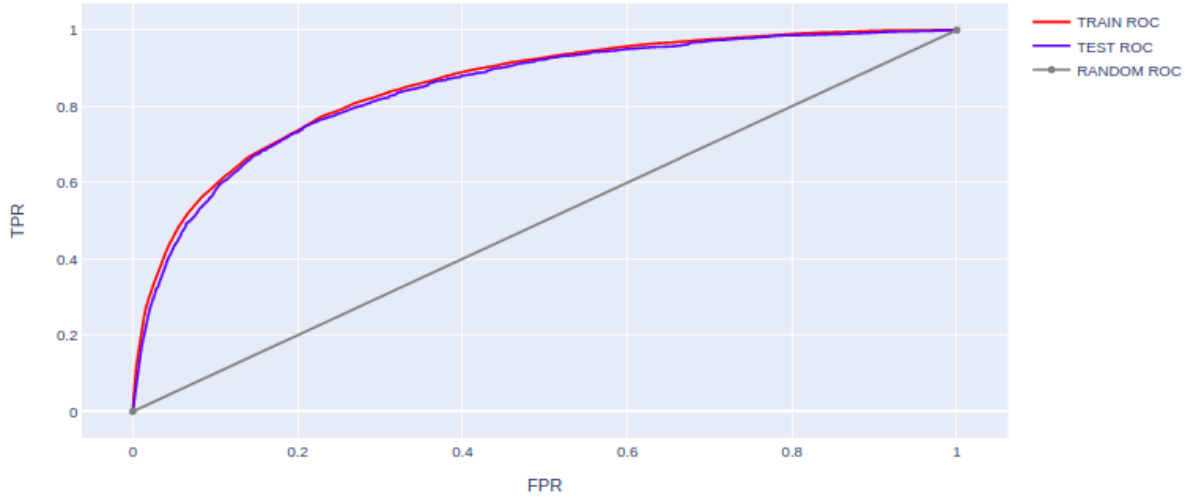


***Figura 45.*** *Curva de ROC para el modelo entrenado con los parámetros de la Tabla 10.*

En cuanto a ROC curve, podemos notar un buen rendimiento, aunque no sea de nuestro interés de estudio, ya que mediremos en base a la métrica de costos anteriormente definida. La métrica AUC para train dió un resultado de 0.852 y para test dió un resultado de 0.844. Algo destacable de estos resultados es que la brecha es considerablemente buena a pesar de haber sido optimizada bajo una métrica basada en una matriz de costos. Aún así, esta métrica AUC no considera ponderaciones, por lo tanto no es lo mismo dar crédito y generar default que no darlo y perder un fee de transacción.

### 4.4.6. PERFORMANCE DEL MODELO - MÉTRICA DE COSTOS

Entrenando el modelo **NGBoost** presentado en la Tabla 10, considerando nuestra métrica de costos anteriormente definida y un threshold de 0.20 para $P(y = 1 \mid X)$ obtuvimos, entre todos los modelos entrenados en la búsqueda de hiper parámetros óptimos, un costo mínimo de € **2,443,875.00**, para el cual, acorde a lo que nuestro algoritmo nos propone, deberíamos otorgarle créditos a 6,553 clientes y negar créditos a 23,447 clientes.
Podemos notar que es un enfoque demasiado conservador, esto se explica dada nuestra matriz de costos que definimos y el threshold obtenido en este primer experimento.

Con el modelo seleccionado, procederemos a realizar un ejercicio de feature importances a fines de determinar qué features ocultaremos en el segundo experimento.

### 4.4.7. KDE DE LAS PROBABILIDADES PREDICHAS POR NGBOOST

El siguiente gráfico representa la distribución de las probabilidades predichas para $y = 1$ por nuestro algoritmo **NGBoost** del primer experimento. Es útil tener esto en cuenta para compararlo con los modelos del segundo experimento a fines de poder observar gráficamente cómo varía esta densidad en las regiones con mayor incertidumbre, es decir, cercanos a 0.5. Una aclaración al respecto es que no damos a visualizar el threshold dado que visualizamos la distribución empírica de la probabilidad estimada de que $y = 1$ .

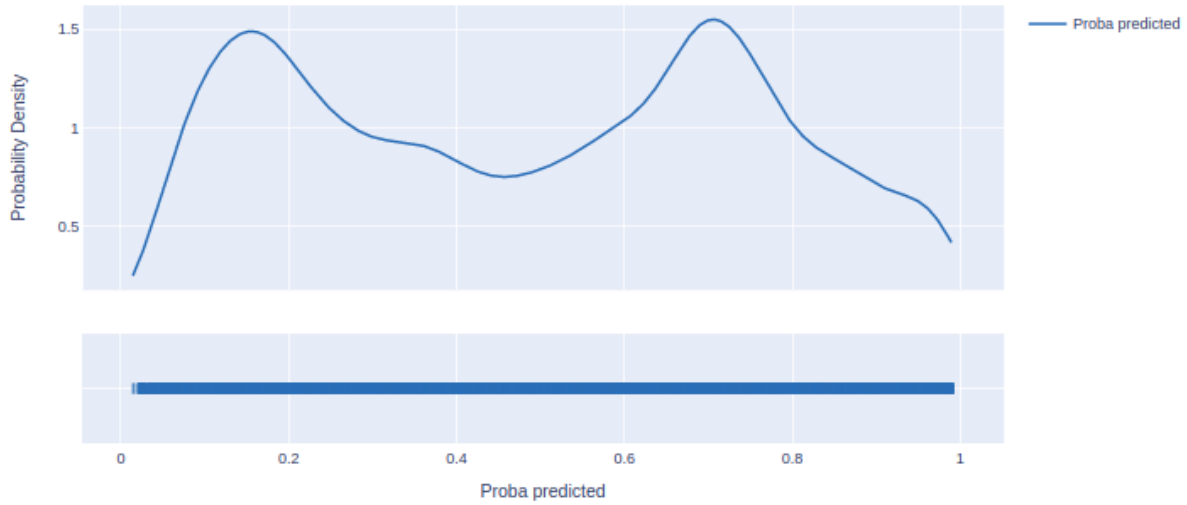


***Figura 46.*** *Gráfico KDE de las probabilidades predichas para $y = 1$ por nuestro algoritmo NGBoost .*

### 4.4.8. INTERPRETABILIDAD GLOBAL DEL MODELO

Dado que necesitamos insertar valores missing a features importantes, para tratar de poner a prueba al modelo en términos de incertidumbre, es necesario realizar una interpretabilidad global del modelo para conocer aquellos features importantes sobre los cuales se realizará la modificación de los datos para efectuar el segundo experimento.

Además, es importante la interpretabilidad a fines de poder entender si nuestro modelo es robusto, dada también la interacción entre features y el impacto de estos sobre la salida del modelo.

#### 4.4.8.1. FEATURE IMPORTANCE

Necesitamos saber cuáles son los features importantes de nuestro modelo **NGBoost,** por lo que ilustraremos lo siguiente:

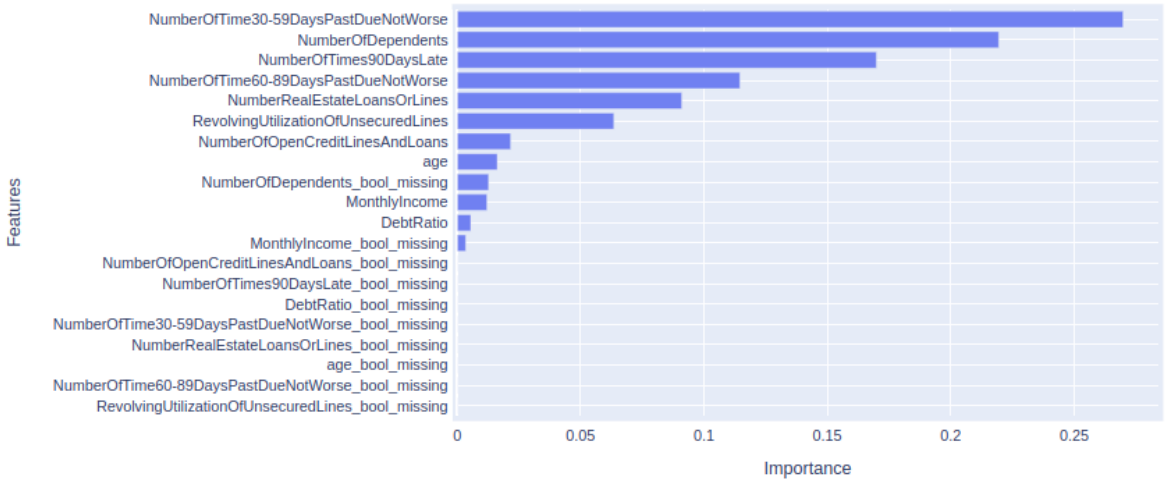


***Figura 47.*** *Plot Feature Importance para el modelo entrenado con los parámetros de la Tabla 10.*

En base a lo presentado podemos intuir lo siguiente:

- Respecto a **NumberOfTime30-59DaysPastDueNotWorse** se puede hacer la intuición que es posible que quienes tiendan a demorarse entre 30 días y 60 días, suelan delinquir.
- **NumberOfDependents** podría indicar que si el prestatario tiene mayor número de personas dependientes, entonces es mucho más probable que delinque.
- Se podría intuir respecto a **NumberOfTimes90DaysLate** que es posible que quienes tiendan a endeudarse por más de 90 días, suelan delinquir.
- De **NumberOfTime60-89DaysPastDueNotWorse** se puede hacer la intuición que es posible que quienes tiendan a demorarse entre 60 días y 90 días, suelan delinquir.

Es importante destacar que la distribución de los features es muy buena, es decir, no existen pocos features (por ejemplo, uno o dos) que consuman toda la importancia para el modelo, esto quiere decir por ejemplo, para este caso, que presenta cierta fiabilidad o robustez, en el sentido de que pequeños cambios no conducen a grandes cambios en la predicción.

#### 4.4.8.2. PERMUTATION FEATURE IMPORTANCE

Mediante el paquete de sklearn sobre permutation features importance, calculamos los mismos y observamos, en formato de gráfico de barras, lo siguiente:


***Figura 48.*** *Plot Permutation Feature Importance para el modelo entrenado con los parámetros de la Tabla 10.*

Este gráfico intenta explicar qué features son más y menos importantes, entendiéndose los primeros por aquellos con magnitud positiva y los segundos por aquellos con magnitud negativa. Podemos observar que este método del paquete sklearn llega más o menos a los mismos resultados que lo que veremos a continuación con el paquete de Python **ELI5**.

Podemos hacer también la misma observación que en el apartado anterior, donde podemos observar que están distribuidos de forma tal que no hay pocos features que consumen toda la importancia.

#### 4.4.8.3. ENTENDIENDO LA INTERPRETABILIDAD GLOBAL DEL MODELO CON ELI5

A partir de la siguiente figura que nos devolvió ELI5 (paquete de Python para analizar permutation importances), se puede observar que los features más influyentes son **MonthlyIncome**, **NumberRealEstateLoansOrLines**, **NumberOfOpenCreditLinesAndLoans** y **MonthlyIncome_bool_missing** . Aquellos features que son menos importantes son **NumberOfTime30-59DaysPastDueNotWorse, NumberOfTimes90DaysLate**, **RevolvingUtilizationOfUnsecuredLines** y **NumberOfTime60-89DaysPastDueNotWorse**.

Algo importante a destacar es que nuestro modelo pudo identificar a **MonthlyIncome_bool_missing** como un feature importante, lo cual es positivo, ya que la transformación de los missings favoreció a que el modelo pudiera identificarla, siendo esto muy importante para nuestro próximo experimento.

| Weight | Feature |
|---|---|
| 0.0171 ± 0.0024 | MonthlyIncome |
| 0.0142 ± 0.0010 | NumberOfOpenCreditLinesAndLoans |
| 0.0138 ± 0.0017 | NumberRealEstateLoansOrLines |
| 0.0109 ± 0.0006 | MonthlyIncome_bool_missing |
| 0.0021 ± 0.0019 | NumberOfDependents |
| 0.0011 ± 0.0009 | age |
| 0 ± 0.0000 | age_bool_missing |
| 0 ± 0.0000 | NumberOfTime60-89DaysPastDueNotWorse_bool_missing |
| 0 ± 0.0000 | RevolvingUtilizationOfUnsecuredLines_bool_missing |
| 0 ± 0.0000 | NumberRealEstateLoansOrLines_bool_missing |
| 0 ± 0.0000 | NumberOfTimes90DaysLate_bool_missing |
| 0 ± 0.0000 | NumberOfOpenCreditLinesAndLoans_bool_missing |
| 0 ± 0.0000 | DebtRatio_bool_missing |
| 0 ± 0.0000 | NumberOfTime30-59DaysPastDueNotWorse_bool_missing |
| -0.0001 ± 0.0008 | DebtRatio |
| -0.0024 ± 0.0002 | NumberOfDependents_bool_missing |
| -0.0249 ± 0.0010 | NumberOfTime60-89DaysPastDueNotWorse |
| -0.0403 ± 0.0016 | RevolvingUtilizationOfUnsecuredLines |
| -0.0432 ± 0.0021 | NumberOfTimes90DaysLate |
| -0.0476 ± 0.0024 | NumberOfTime30-59DaysPastDueNotWorse |

***Tabla 20.*** *Tabla que indica en forma descendente la importancia de permutación de los features.*

El primer número en cada fila indica la reducción en el performance del modelo por la permutación producida para ese feature.
El segundo número es una medida de la aleatoriedad de la reducción de la performance para diferentes permutaciones del feature.

#### 4.4.8.4. ENTENDIENDO LA INTERACCIÓN ENTRE VARIABLES CON SHAP

Los siguientes gráficos corresponden a los summary plot provistos por shap que permiten entender la interpretabilidad global del modelo utilizando un ploteo de feature importance respecto a los valores Shapley.
El primer gráfico es una versión más compleja en donde se representan todas las observaciones. Es válido aclarar que la ubicación horizontal (izquierda o derecha) indica si el efecto de ese valor está asociado con una predicción menor o mayor. El color muestra si ese feature es muy importante (rojo) o poco importante (azul) para esa observación, esto está reflejado en la barra horizontal del lado derecho del gráfico.
Podemos observar que el siguiente gráfico condice con el apartado anterior en el análisis de permutation feature importance realizado por ELI5, observando un gran volumen (acumulación de puntos) de valores Shapley bajos del lado izquierdo, es decir, de impacto negativo respecto a la salida del modelo, lo que condice con lo visto en el ranking provisto por ELI5 ya que estos features figuran como menos importantes en la Tabla 20. Es necesario recordar que los valores que arroja SHAP no necesariamente deben ser los mismos de los otros análisis ya que son análisis que contemplan distinto tipo de métrica, aunque pueden llegar a tener alguna vinculación como en este caso.

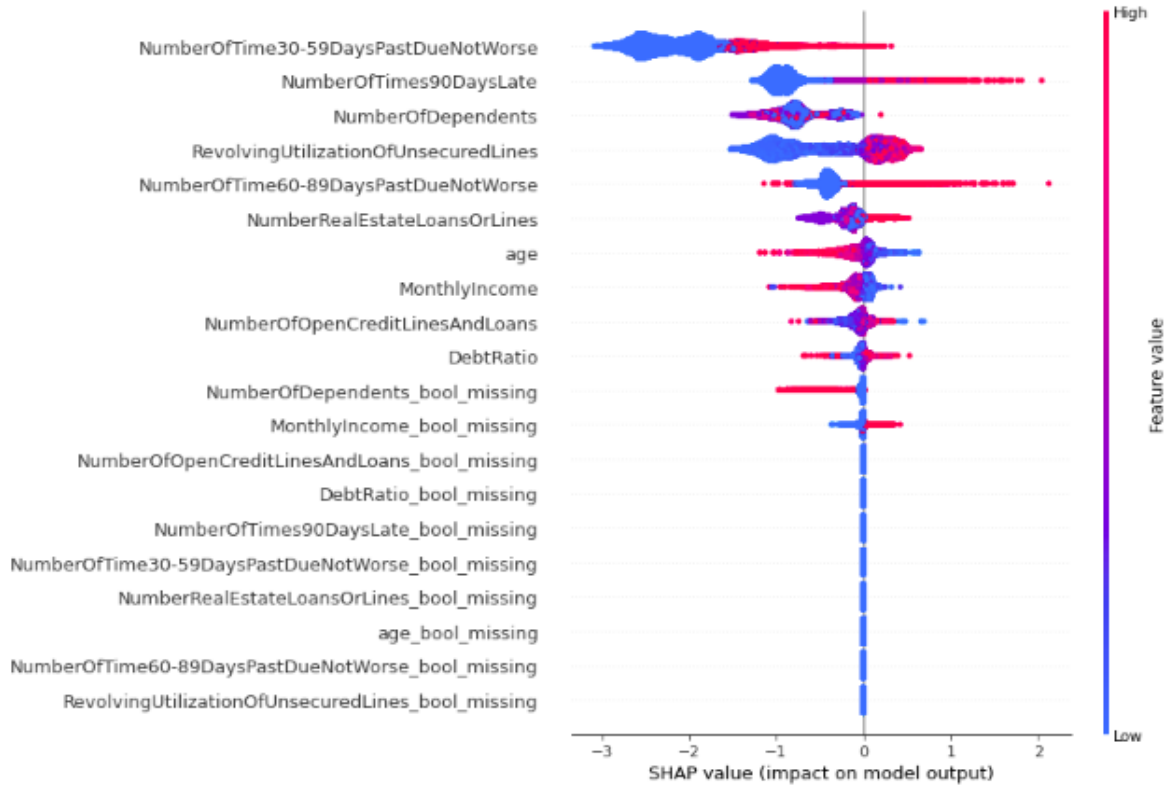


***Figura 49.*** *Importancia de los features (versión compleja) considerando valores Shapley.*

#### 4.4.8.5. ENTENDIENDO LAS PREDICCIONES PARTICULARES DEL MODELO CON LIME

Trataremos de explicar las predicciones de nuestro algoritmo **NGBoost,** para lo cual utilizaremos LIME. Como bien dijimos antes, LIME nos permite obtener casos y, mediante el explainer podemos obtener el valor de las probabilidades de predicción para cada target, cuánta influencia tiene cada feature en la decisión tomada por el algoritmo y una tabla con los valores correspondiente a ese caso seleccionado. Esta extracción, como expresamos en el **Marco Teórico**, es realizada por el módulo **submodular_pickle** que toma una muestra representativa de explicaciones.

Estos features, cuyos valores están normalizados, estarán visualizados en la tabla que figura a la izquierda de cada reporte del explainer.

##### *4.4.8.5.1. Caso 1 - Delinquent (target = 0)*

Para este caso podemos observar en la siguiente Figura un caso donde el algoritmo predijo para esta observación que es **Responsible (target = 0)**. Para ello, tuvo en cuenta el valor de **NumberOfDependents** y de **MonthlyIncome**, los cuales probablemente el modelo considere que son valores apropiados para su predicción. Luego, se puede observar que por ejemplo los valores de **NumberOfTimes30-59DaysPastDueNotWorse**, **NumberRealEstateLoans**,

**NumberOfTimes90DaysLate** y **RevolvingUtilizationOfUnsecuredLines** empujaron a que el algoritmo incline por su decisión dado su bajísimo valor, lo cual es razonable.

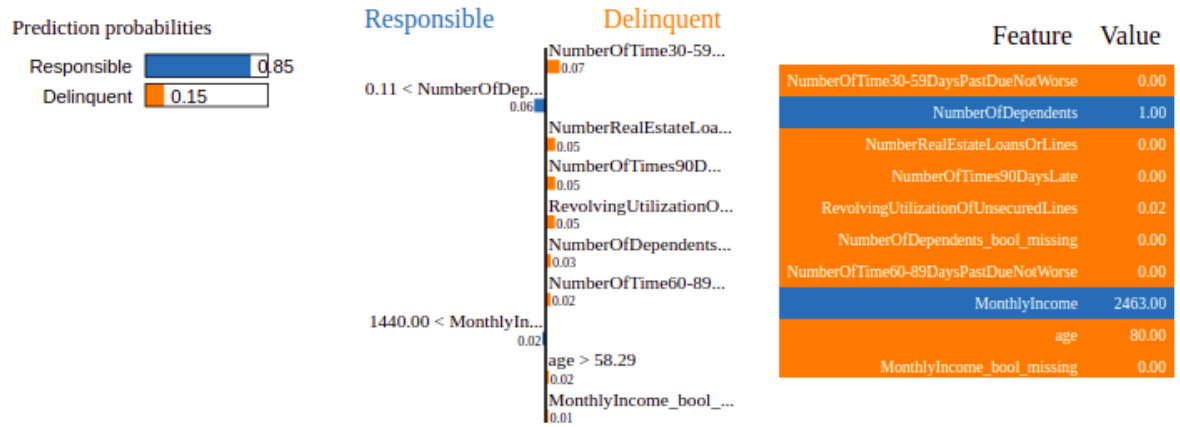


***Figura 50.*** *Reporte del Explainer del paquete LIME respecto a una observación sobre la cual el algoritmo predijo que es Responsible (target=0).*

### *4.4.8.5.2. Caso 2 - Delinquent (target = 1)*

Podemos observar en la siguiente Figura un caso tomado al azar donde el algoritmo predice con mucha certeza que es **Delinquent (target = 1)**, dado los valores que se observan en la tabla de la derecha. Podemos observar también la influencia de los features **NumberOfTime30-59DaysPastDueNotWorse**, **NumberOfDependents, RevolvingUtilizationOfUnsecuredLines** y **age** para que el algoritmo decida que es **Delinquent**, dado que estos valores son relativamente altos (teniendo en cuenta que fueron normalizados).

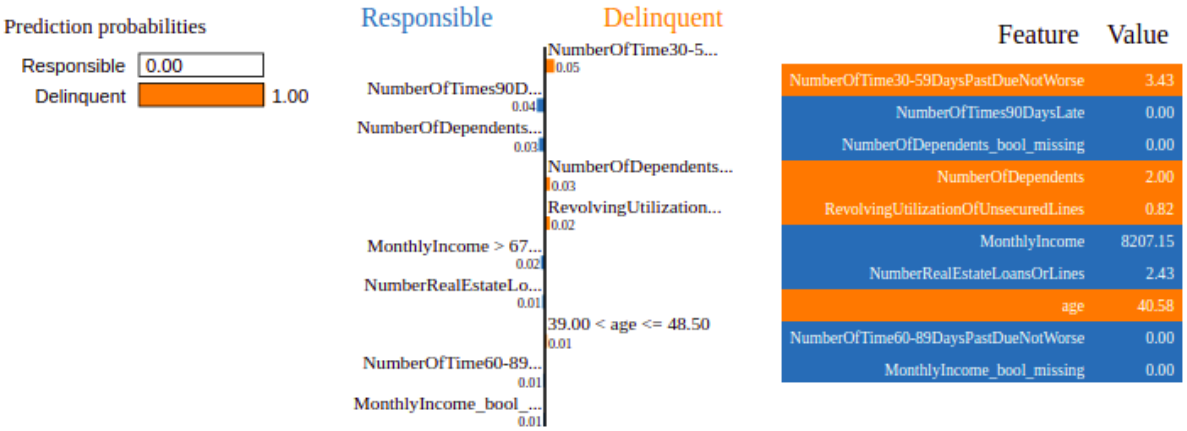


***Figura 51.*** *Reporte del Explainer del paquete LIME respecto a una observación sobre la cual el algoritmo predijo que es Delinquent (target=1).*

## 4.4.9. ANÁLISIS DE LOS HIPER PARÁMETROS DE NUESTRO MODELO

### 4.4.9.1. Evolución del costo con respecto a los diferentes hiper parámetros

Ahora bien, analizando todos los modelos entrenados, que son más de 100, podemos también visualizar la evolución del costo conforme a las diferentes combinaciones de los hiper parámetros a fines de poder sacar intuiciones respecto a qué hiper parámetros son determinantes con respecto a la métrica a observar. Al igual que en el anterior análisis, sólo tendremos en cuenta los valores para un threshold igual a 0.2.

Con respecto a la métrica de costo, podemos mostrar lo siguiente:

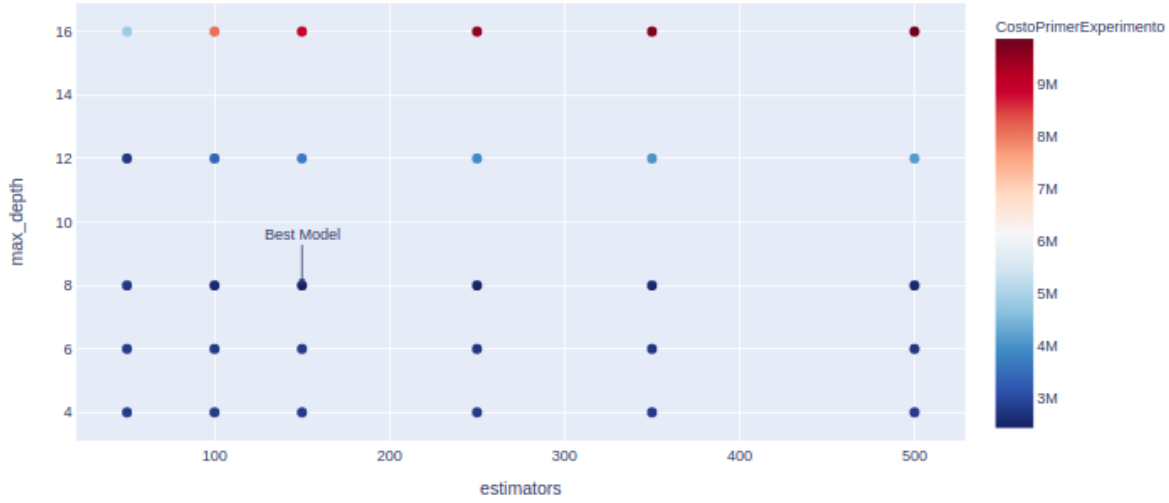


***Figura 52.*** *Scatter plot en el que cada punto es el valor de CostoPrimerExperimento que asume la corrida del entrenamiento del modelo NGBoost para esa configuración de hiper parámetros y el color indica la magnitud del valor obtenido para la métrica de costos, teniendo en cuenta que el threshold es igual a 0.2 y que learning_rate es igual a 0.01. Otra consideración es que cuanto más azul oscuro sea el punto, más chico es el valor de la métrica de costos (mejor) y que cuanto más rojo oscuro sea el punto, más grande es el valor de la métrica de costos (peor).*

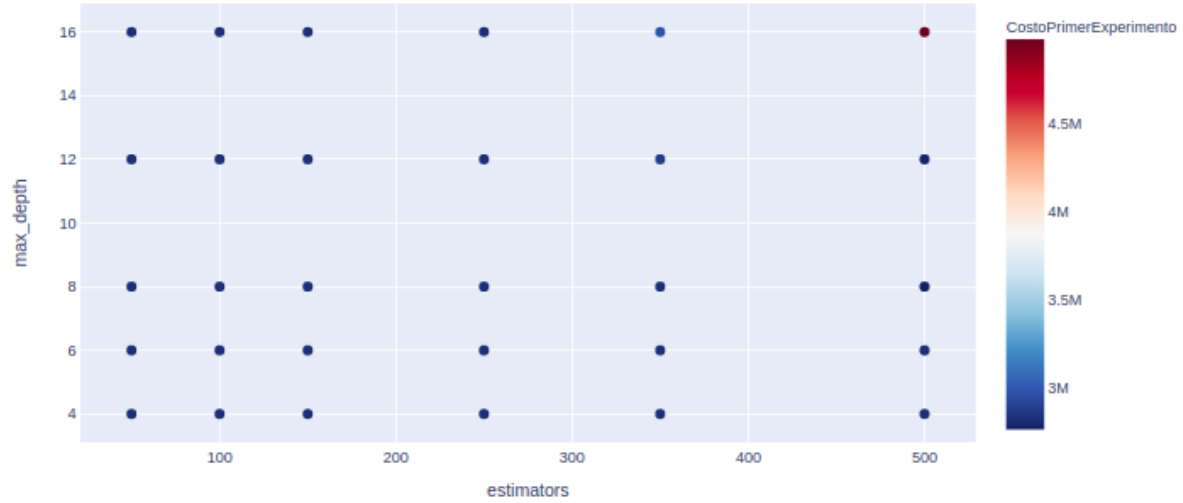


***Figura 53.*** *Scatter plot en el que cada punto es el valor de CostoPrimerExperimento que asume la corrida del entrenamiento del modelo NGBoost para esa configuración de hiper parámetros y el color indica la magnitud del valor obtenido para la métrica de costos, teniendo en cuenta que el threshold es igual a 0.2 y que learning_rate es igual a 0.001. Otra consideración es que cuanto más azul oscuro sea el punto, más chico es el valor de la métrica de costos (mejor) y que cuanto más rojo oscuro sea el punto, más grande es el valor de la métrica de costos (peor).*

Los valores volcados en estas representaciones pueden ser apreciados en el **ANEXO 1**.
Respecto a los gráficos podemos afirmar que, para un **learning_rate** de 0.01, cuanto más **estimadores** y con un **max_depth** apropiado (no tan alto, en este caso), entonces mejor será nuestro modelo y que cuando tenga mayor cantidad de **estimadores** y **max_depth** será peor.

#### 4.4.9.2. Evolución de la cantidad de préstamos asignados y no asignados con respecto al threshold

Realizando iteraciones para el threshold, desde 0.05 a 0.95, podemos obtener el siguiente gráfico de evolución corresponde a la cantidad de préstamos asignados y no asignados que predijo nuestro algoritmo **NGBoost**:

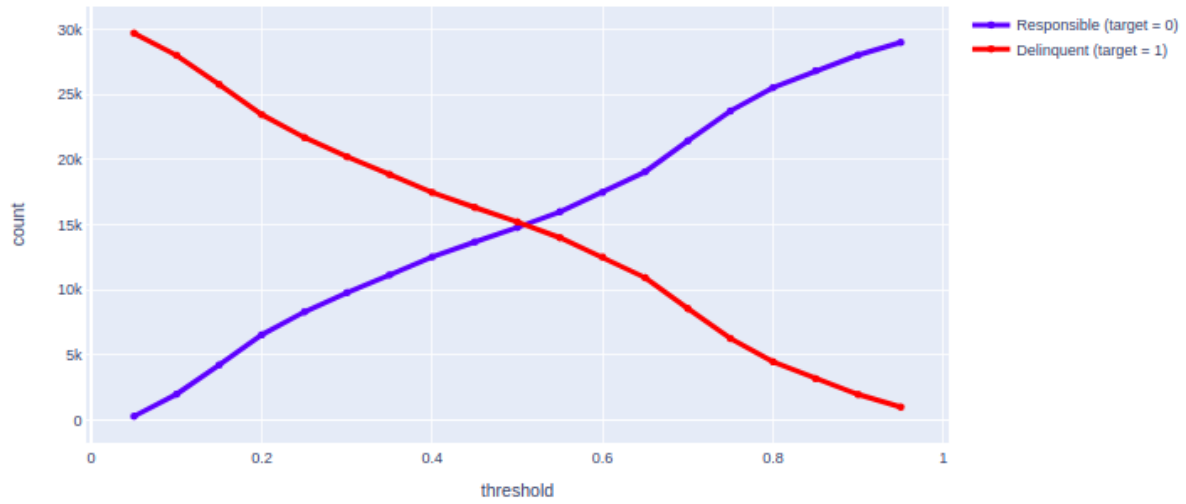


***Figura 54.*** *Evolución en formato gráfico de la cantidad de préstamos asignados y no asignados acorde a la predicción de nuestro algoritmo obtenido en el primer experimento. La línea azul representa las observaciones predichas con* $\hat{y} = 0$ *y la línea roja representa las observaciones predichas con* $\hat{y} = 1$*. Las unidades*

Podemos apreciar los valores del anterior gráfico en formato tabular:

| threshold | count_zero | count_one |
|---|---|---|
| 0.05 | 303 | 29,697 |
| 0.1 | 1,998 | 28,002 |
| 0.15 | 4,241 | 25,759 |
| 0.2 | 6,553 | 23,447 |
| 0.25 | 8,308 | 21,692 |
| 0.3 | 9,786 | 20,214 |
| 0.35 | 11,146 | 18,854 |
| 0.4 | 12,528 | 17,472 |
| 0.45 | 13,678 | 16,322 |

| | | |
|---|---|---|
| 0.5 | 14,785 | 15,215 |
| 0.55 | 15,998 | 14,002 |
| 0.6 | 17,479 | 12,521 |
| 0.65 | 19,062 | 10,938 |
| 0.7 | 21,428 | 8,572 |
| 0.75 | 23,730 | 6,270 |
| 0.8 | 25,513 | 4,487 |
| 0.85 | 26,809 | 3,191 |
| 0.9 | 28,027 | 1,973 |
| 0.95 | 28,993 | 1,007 |

***Tabla 21.*** *Evolución en formato tabular de la cantidad de préstamos asignados y no asignados acorde a la predicción de nuestro algoritmo obtenido en el primer experimento.*

#### 4.4.9.3. Evolución del costo con respecto al threshold

Realizando iteraciones para el threshold, desde 0.05 a 0.95, podemos obtener el siguiente gráfico de evolución de nuestro costo definido conforme el threshold va aumentando y, por ende, cambiando nuestra cantidad de créditos que son asignados y los que no son asignados.

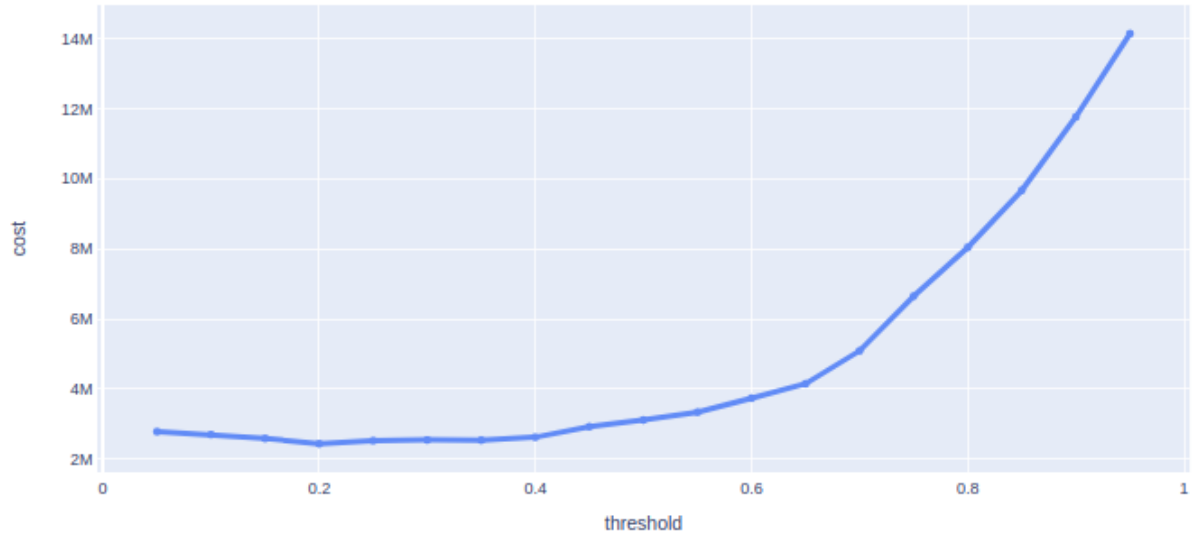

***Figura 55.*** *Evolución en formato gráfico del costo definido para nuestro primer experimento conforme al aumento del threshold.*

Podemos apreciar esta evolución en forma tabular también:

| threshold | cost (CostoPrimer Experimento) |
|---|---|
| 0.05 | 2,788,670.28 |
| 0.1 | 2,705,948.90 |
| 0.15 | 2,594,765.28 |
| 0.2 | 2,443,875.00 |
| 0.25 | 2,531,405.86 |
| 0.3 | 2,552,349.22 |
| 0.35 | 2,546,171.48 |
| 0.4 | 2,629,692.00 |
| 0.45 | 2,931,691.76 |
| 0.5 | 3,121,077.04 |
| 0.55 | 3,346,601.16 |
| 0.6 | 3,744,903.36 |
| 0.65 | 4,155,563.24 |
| 0.7 | 5,094,596.96 |
| 0.75 | 6,655,837.96 |
| 0.8 | 8,045,415.76 |

| | |
|---|---|
| 0.85 | 9,672,013.98 |
| 0.9 | 11,767,218.56 |
| 0.95 | 14,141,030.68 |

***Tabla 22.*** *Evolución en formato tabular del costo definido para nuestro primer experimento conforme al aumento del threshold.*

### 4.4.10. CONCLUSIONES DEL PRIMER EXPERIMENTO

En este experimento llegamos entonces, a las siguientes conclusiones:

- Un buen **learning_rate** para nuestro modelo es 0.01. Se concluye esto, dado que con valores más pequeños, el modelo tendía a estar más “confuso”, con lo cual, predecía todos 1 (o 0) con demasiada incertidumbre, es decir, con probabilidades muy cercanas al threshold.
- Un buen **threshold** para que el modelo prediga si la observación es **Deliquent (y_pred = 1)** es 0.20, es decir, nuestro modelo está ponderando más el error de clasificación para este valor del target.
- El parámetro **max_depth** de nuestro hiper parámetro **Base** (que es un DecisionTreeRegressor) de nuestro modelo **NGBoost** afecta mucho también las predicciones, en el sentido de que tiende a sobreajustar si el mismo es muy grande, por lo que los mejores modelos suelen dar con números más chicos de este hiper parámetro de nuestro modelo base.

## 4.5. SEGUNDO EXPERIMENTO

Procederemos entonces a realizar nuestro segundo experimento, en donde a diferencia del primer experimento donde buscamos un modelo, entrenaremos 3 modelos nuevos:

1. Modelo entrenado con datos missings insertados en un 20% aplicados de forma aleatoria sin feature importance. Se utiliza para evaluar y predecir la métrica de costos definida en el primer experimento.
2. Modelo entrenado con datos missings insertados en un 20% aplicados a los features más importantes según el reporte de ELI5. Se utiliza para evaluar y predecir la métrica de costos definida en el primer experimento.
3. Modelo entrenado con datos missings insertados en un 20% aplicados a los features más importantes según el reporte de ELI5. Se utiliza para evaluar y predecir una nueva métrica de costos (a definir más adelante) que contemple la incertidumbre en las predicciones.

### 4.5.1. APLICANDO MISSINGS EN FORMA ALEATORIA SIN FEATURE IMPORTANCES

Antes de insertar valores missings sobre los features más importantes representados en la **Tabla 24**, procederemos a aplicarlos en forma aleatoria sin considerar los features sobre los cuales aplicamos esta alteración de los datos.
Entonces, entrenamos nuevamente nuevo modelo **NGBoost** tal que minimice la función de costos definida en el primer experimento.
Obtenemos el siguiente modelo y la siguiente performance predictiva:

| Hiper parámetro | Valores |
|---|---|
| Base | DecisionTreeRegressor(ccp_alpha=0.0, criterion='friedman_mse', max_depth=8, max_features=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, presort='deprecated', random_state=2020, splitter='best') |
| Dist | Bernoulli |
| Score | LogScore (ngboost score module) |

| | |
|---|---|
| learning_rate | 0.01 |
| minibatch_frac | 1.0 |
| n_estimators | 500 |
| natural_gradient | True |
| tol | 0.0001 |

***Tabla 23.*** *Tabla con los hiper parámetros asignados al modelo con el cual entrenamos para realizar nuestras predicciones. Los features que fueron explorados están subrayados.*

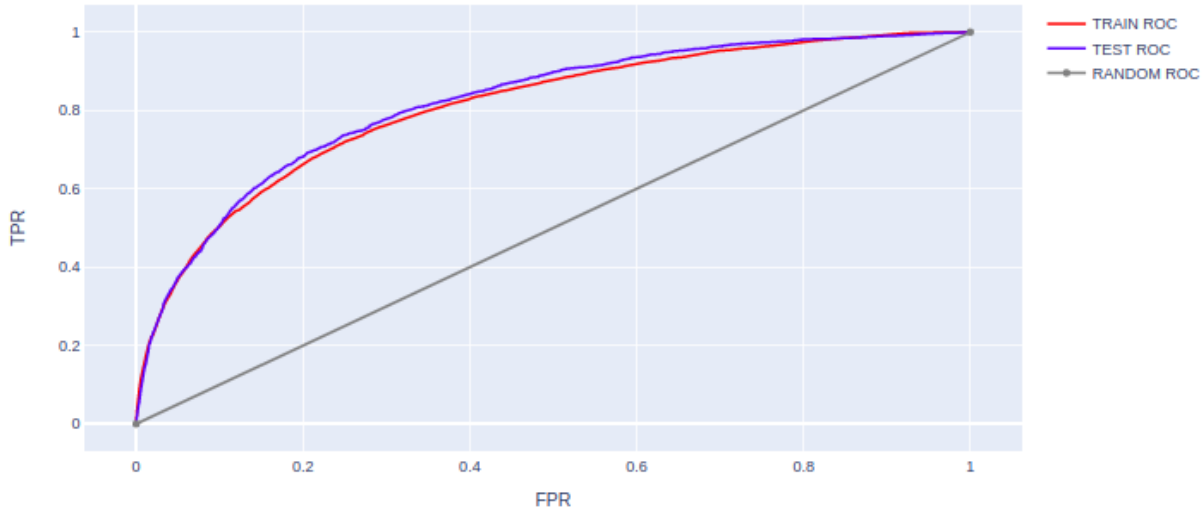


***Figura 56.*** *Curva de ROC del modelo NGBoost que contempla la métrica de costos del primer experimento y entrenado con datos con el 20% con missings asignados de manera aleatoria sobre todo el conjunto de datos.*

En cuanto a las métricas de performance, el modelo tuvo una peor performance con respecto al modelo del primer experimento, obteniendo un auc para train de 0.808 y para test de 0.818. Además, el modelo, para un threshold de 0.25, recomienda asignar 4,559 créditos y no asignar 25,441 créditos. La función de costos del primer experimento, en base a este modelo, resultó en **2,679,354.87 euros**. Esto significa igualmente que es mucho peor que nuestro modelo del primer experimento, dado que perdemos mucho más con una asignación mucho menor.

Esto puede justificarse mediante el KDE de las probabilidades predichas para $y = 1$, observando que la mayoría de la concentración tendió a irse hacia 1.

Una aclaración al respecto es que no damos a visualizar el threshold dado que visualizamos la distribución empírica de la probabilidad estimada de que $y = 1$ .

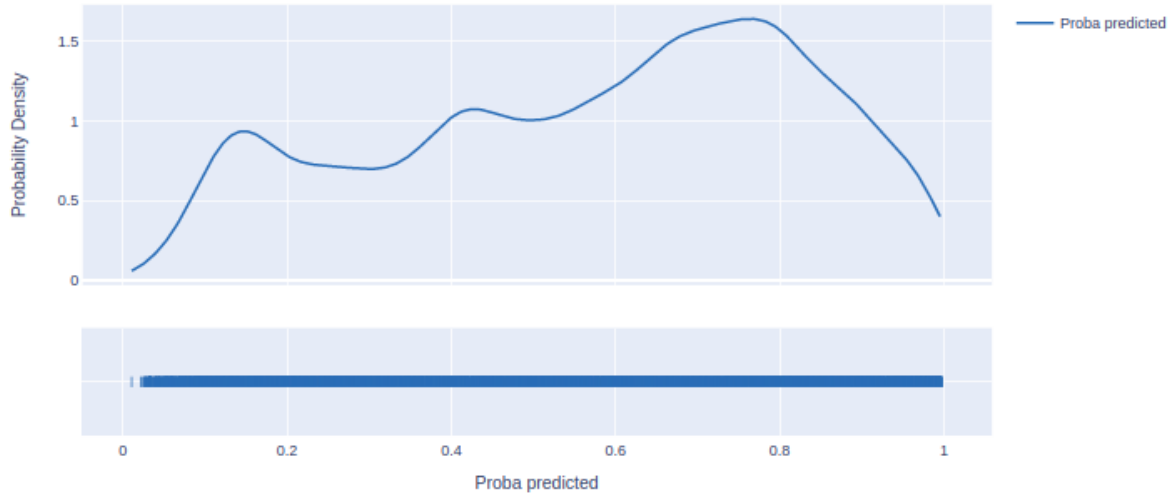


***Figura 57.*** *KDE de probabilidades predichas para $y = 1$ para el nuevo modelo NGBoost entrenado con datos missings asignados de manera aleatoria.*

### 4.5.2. APLICANDO MISSINGS SOBRE FEATURES IMPORTANTES

Recordamos entonces que uno de nuestros objetivos para el presente trabajo es poder asignar crédito a usuarios dónde hay mucha incertidumbre en las predicciones, de manera tal que se pueda aprender sobre ellos, por lo que procederemos a realizar nuestro segundo experimento, tomando como referencia lo experimentado anteriormente.

Por lo tanto, a partir de lo que observamos en la **Tabla 20**, es decir, nuestra tabla de Permutation Feature Importances según el paquete Python **ELI5**, tomaremos los primeros 3 features importantes:

| Feature | Importance |
|---|---|
| MonthlyIncome | 0.0171 |
| NumberRealEstateLoansOrLines | 0.0142 |
| NumberOfOpenCreditLinesAndLoans | 0.0138 |

***Tabla 24.*** *Tabla que indica los 3 features más importantes acorde a lo indicado en la Figura 33.*

Aplicaremos los mismos procesos de feature engineering y limpieza de outliers aplicados en el experimento anterior, con la particularidad de que, previo a aplicar **SMOTE** y la identificación de valores missings, ocultaremos estos datos con un porcentaje del 20% sobre estos 3 features.

Exploraremos nuevamente y buscaremos un nuevo modelo que satisfaga la minimización de la métrica **CostoPrimerExperimento**, pero con la salvedad de que esta vez los datos de entrenamiento son totalmente distintos a los del primer experimento.
Obtuvimos entonces el siguiente modelo y la siguiente performance predictiva:

| Hiper parámetro | Valores |
|---|---|
| Base | DecisionTreeRegressor(ccp_alpha=0.0, criterion='friedman_mse', max_depth=8, max_features=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, presort='deprecated', random_state=2020, splitter='best') |
| Dist | Bernoulli |
| Score | LogScore (ngboost score module) |
| learning_rate | 0.01 |
| minibatch_frac | 1.0 |
| n_estimators | 100 |
| natural_gradient | True |
| tol | 0.0001 |

***Tabla 25.*** *Tabla con los hiper parámetros asignados al modelo con el cual entrenamos para realizar nuestras predicciones. Los features que fueron explorados están subrayados.*

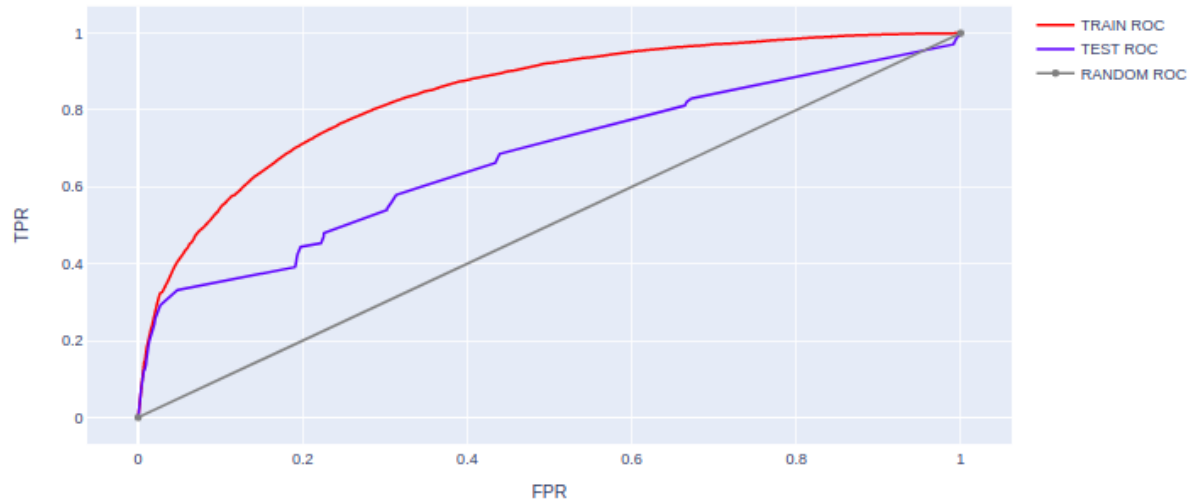

***Figura 58.*** *Curva de ROC del modelo NGBoost que contempla la métrica de costos del primer experimento y entrenado con datos con el 20% con missings sobre los features más importantes indicados también en el primer experimento.*

Desde el enfoque tradicional de machine learning podemos observar un AUC para train de 0.840 y un AUC para test de 0.673.

En términos de costos obtuvimos un valor menor, resultando un costo de **€ 2,817,421.84**, esto se debe a que el modelo predijo menos observaciones como **Responsible (target = 0)** para un mismo threshold, es decir, 0.20 considerando $P(y = 1 \mid X)$. Este modelo predijo asignar 1 crédito y negar 29,999 créditos. Esto ya indica claramente que la performance es pésima.

Esto también podemos representarlo mediante un KDE y podemos observar claramente esta bajísima performance:

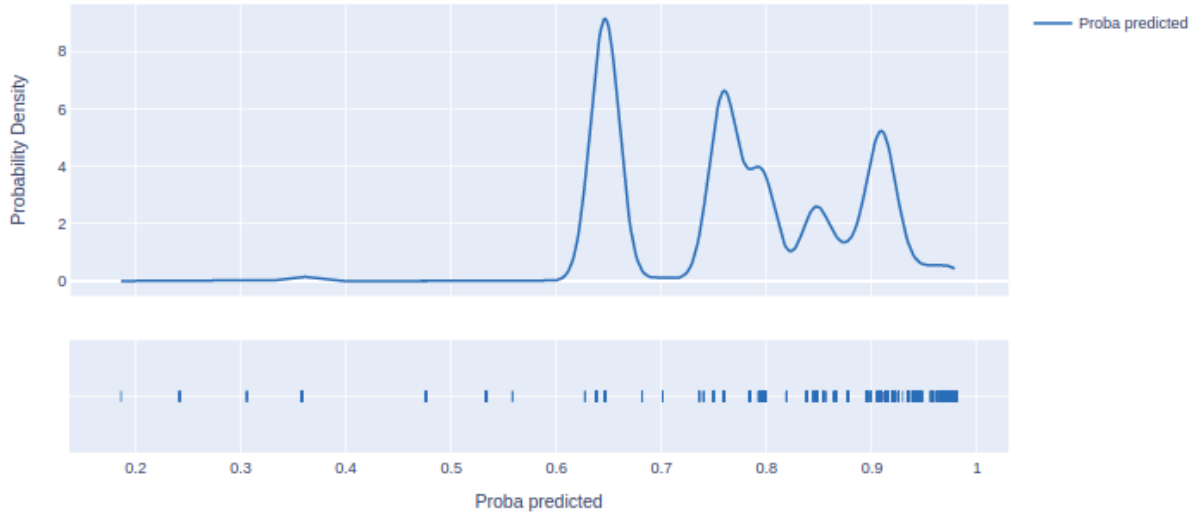


***Figura 59.*** *Gráfico KDE de las probabilidades predichas para $y = 1$ para el modelo NGBoost entrenado con los datos con missings sobre los features más importantes indicados en el primer experimento.*

Una aclaración al respecto es que no damos a visualizar el threshold dado que visualizamos la distribución empírica de la probabilidad estimada de que $y = 1$ .

Como es importante poder asignar más en un ámbito de mayor incertidumbre, ya que necesitamos tener un enfoque menos conservador (en el sentido de la cantidad de asignaciones) y más exploratorio (en el sentido de enfocarse en observaciones sobre las cuales el modelo tiene incertidumbre), procederemos entonces a desarrollar una nueva métrica que contemple dicha incertidumbre en las predicciones para poder asignar una mayor

cantidad de créditos y desarrollar además una evaluación respecto al enfoque anterior del primer experimento.

### 4.5.3. DEFINIENDO UNA NUEVA FUNCIÓN OBJETIVO

Procederemos a definir una nueva métrica que llamaremos **Costo de Aprender**, cuyo objetivo es ponderar aquellas observaciones sobre las cuales el modelo tiene incertidumbre. Dicho costo está formado por un parámetro $\alpha$ que es lo suficientemente grande como para ponderar aquellas $i$ observaciones. Siendo N el tamaño del conjunto de datos, $\hat{y}$ la predicción de nuestro modelo **NGBoost** y $\hat{p}$ la probabilidad predicha para dicha predicción (considerando cuando $\hat{y}$=1), entonces, matemáticamente definimos lo siguiente:

$$CostodeAprender = \alpha \times \sum_{i=1}^{N} (1 - \hat{y}) \times \hat{p}_i \times (1 - \hat{p}_i)$$

Una intuición respecto a esta fórmula es que si $\hat{y} = 1$ entonces el **Costo de Aprender** se anula, ya que no estaría asignando el crédito, por lo que no tendría importancia dicho costo. Otra aclaración, es que la multiplicación $\widehat{p_i} \times (1 - \widehat{p_i})$ para $p_i$ en un intervalo de $[0; 1]$ es máxima cuando $p_i = 0.5$, lo que muestra que el **CostodeAprender** es más grande cuando más incertidumbre hay.

Definiremos entonces, una nueva función objetivo tal que contemple el costo de aprender, recientemente definido. Para ello, consideraremos la métrica de **CostoPrimerExperimento** definida en el primer experimento y le sumaremos el valor del costo de aprendizaje, con lo cual, tenemos lo siguiente:

$$CostoSegundoExperimento = CostoPrimerExperimento + \alpha \times \sum_{i=1}^{N} (1 - \hat{y}) \times \hat{p}_i \times (1 - \hat{p}_i)$$

Como bien dijimos anteriormente acerca del segundo sumando, podemos aclarar que es menor a **CostoPrimerExperimento**, ya que nuestro objetivo, en comparación al primer experimento, es poder asignar más créditos en aquellas situaciones donde la incertidumbre es mayor.
Por tanto, esta nueva métrica, entonces, nos permitirá poder asignar más créditos ya que estamos haciendo énfasis en aquellas observaciones en las que hay mucha incertidumbre ($p_i \simeq 0.5$).

Con lo cual, finalmente tenemos lo siguiente:

$$CostoSegundoExperimento = CostoPrimerExperimento + CostodeAprender$$

Tal como hicimos con el experimento anterior, procederemos entonces a ilustrar las implementaciones de estas métricas en código Python:

```python
def process_learning_unit_cost(x, alpha):
    """
    Process the learning unit cost for each observation.

    Args:
        - x (pd.Series): data to be identified with name column (predicted, proba_predicted)
        - alpha (float): hyperparam to set a weight of learning

    Returns for each case his cost value.
    """
    return alpha * (1- x["predicted"]) *  x["proba_predicted"] * (1-x["proba_predicted"])
```

***Figura 60.*** *Función que procesa el costo unitario de aprendizaje para cada observación.*

```python
def calculate_learning_cost(y_pred, proba_pred, alpha):
    """
    From input data, generates auxiliar dataframe in order to apply process_learning_unit.
    Args:
        - y_pred (pd.Series): list of predictions
        - y_proba (pd.Series): list of probabilities for each prediction
        - alpha (int): hyperparam to set a weight of learning

    Returns sum of learning unit costs
    """
    aux_df = pd.DataFrame(
        data={"predicted": y_pred, "proba_predicted": proba_pred}
    )
    return sum(aux_df.apply(lambda x: process_learning_unit_cost(x, alpha), axis=1))
```

***Figura 61.*** *Función que suma los costos unitarios de aprendizaje de todo el conjunto de datos.*

```python
def calculate_cost_score_with_learning(cost_score_without_learning, learning_cost):
    """
    Process cost score with learning (second experiment).
    """
    return cost_score_without_learning + learning_cost
```

***Figura 62.*** *Función que recibe como argumentos CostoPrimerExperimento (costo sin aprendizaje) y el costo sumarizado de aprendizaje anteriormente ilustrado, para luego efectuar la suma, tal como indica la definición de la nueva función objetivo.*

### 4.5.4. DEFINIENDO PORCENTAJE DE INVERSIÓN DEL SEGUNDO EXPERIMENTO

A fines de poder expresar la aplicación de este segundo experimento en términos porcentuales, definiremos lo siguiente:

$$\% \, Inversión \; = \; - \frac{CostoPrimerExperimento \; - \; CostoSegundoExperimento}{CostoSegundoExperimento}$$

Esta métrica es muy distinta a las anteriormente definidas como funciones objetivos, ya que dichas métricas optimizan errores. **% Inversion** posee el costo del primer experimento (que se definiría como un baseline, en este caso), es decir, el costo de no aplicar la consideración sobre aquellas observaciones sobre las cuales tengo mayor incertidumbre y posee además el costo del modelo actual, el cual recordamos que es la función objetivo de nuestro segundo experimento, que sí contempla aquellas observaciones con mayor incertidumbre a la hora de asignar créditos. Entonces, esta métrica permite cuantificar el impacto de aplicar este segundo experimento a medida que alpha cambia y, por tanto, considerar observaciones sobre las cuales el algoritmo predice con mayor incertidumbre.
Una aclaración al respecto es que la métrica **CostoPrimerExperimento** fue procesada con datos del segundo experimento, es decir, con la aplicación del 20% de missings sobre los features más importantes expresados en el primer experimento.

#### 4.5.4.1 ANÁLISIS DE LA PERFORMANCE DEL SEGUNDO MODELO NGBOOST

Analizaremos el caso para cuando el threshold = 0.4, analizando los 100 mejores resultados con respecto a **CostoSegundoExperimento.** Sabemos entonces que tomamos fijo **learning_rate** como 0.01, por lo que analizaremos los valores para estimators, max_depth del Base Learner y alpha.
Por cuestiones de simplicidad, desglosaremos los gráficos en 3, uno por cada valor explorado para max_depth, los cuales son 8, 10 y 12 respectivamente.

##### *4.5.4.1.1. ANÁLISIS DE COSTO DEL SEGUNDO MODELO NGBOOST*

Podemos apreciar en los siguientes gráficos que a mayor max_depth, a mayor alpha y a menor número de estimadores, entonces mayor será el costo.
Podemos observar que los colores cambian notablemente con los niveles de alpha y que a medida que crece la cantidad de estimadores, hay mucho más volumen de valores altos para la métrica de costos.

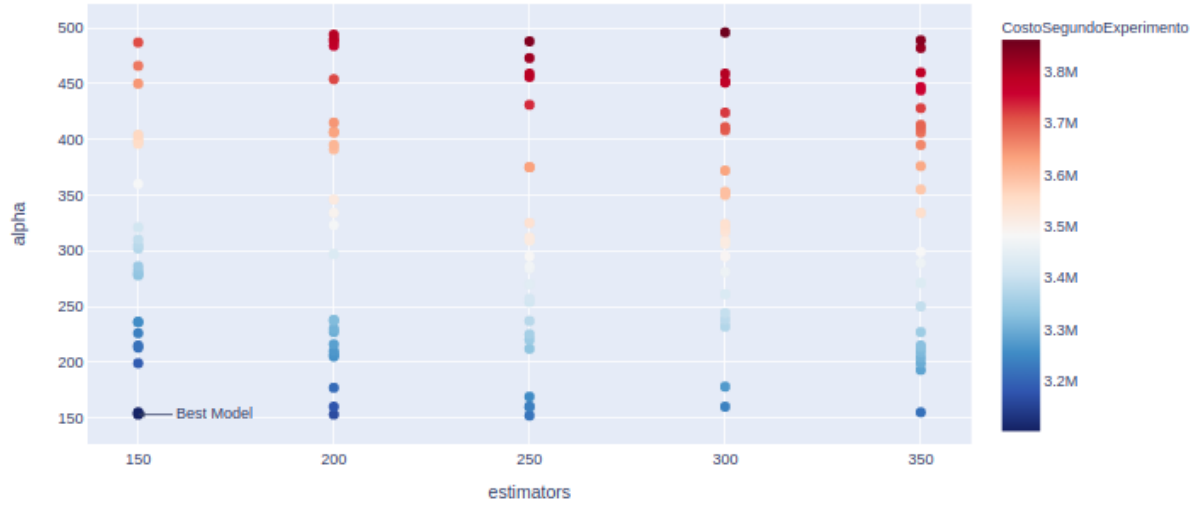


***Figura 63.*** *Scatter plot que ilustra mediante los colores la magnitud de la métrica de costo de nuestro segundo experimento, considerando mayores aquellos puntos que tienden a tener un color más rojo oscuro y menores aquellos puntos con un color más azul oscuro. Este gráfico corresponde a corridas con max_depth igual a 8.*

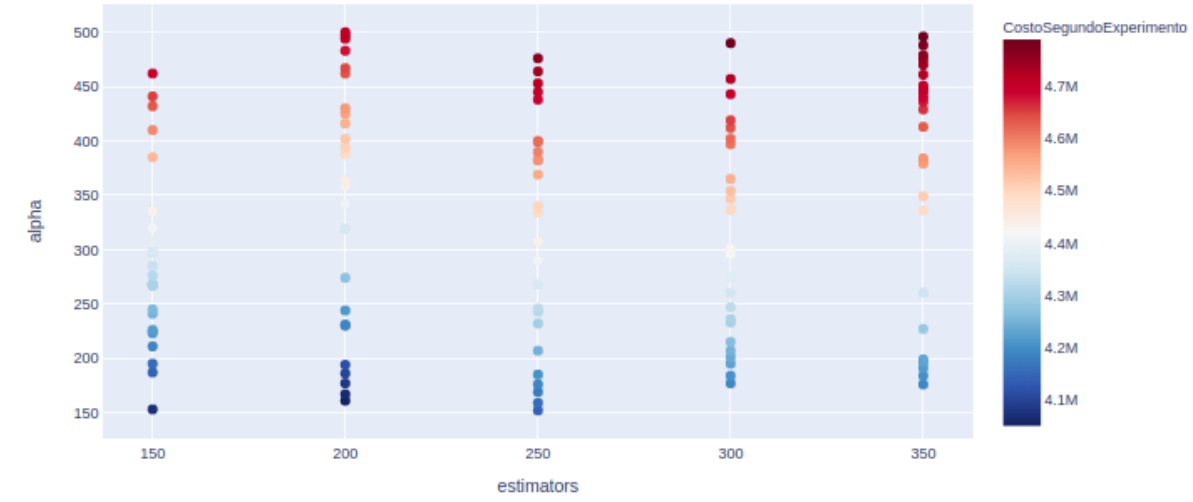


***Figura 64.*** *Scatter plot que ilustra mediante los colores la magnitud de la métrica de costo de nuestro segundo experimento, considerando mayores aquellos puntos que tienden a tener un color más rojo oscuro y menores aquellos puntos con un color más azul oscuro. Este gráfico corresponde a corridas con max_depth igual a 10.*

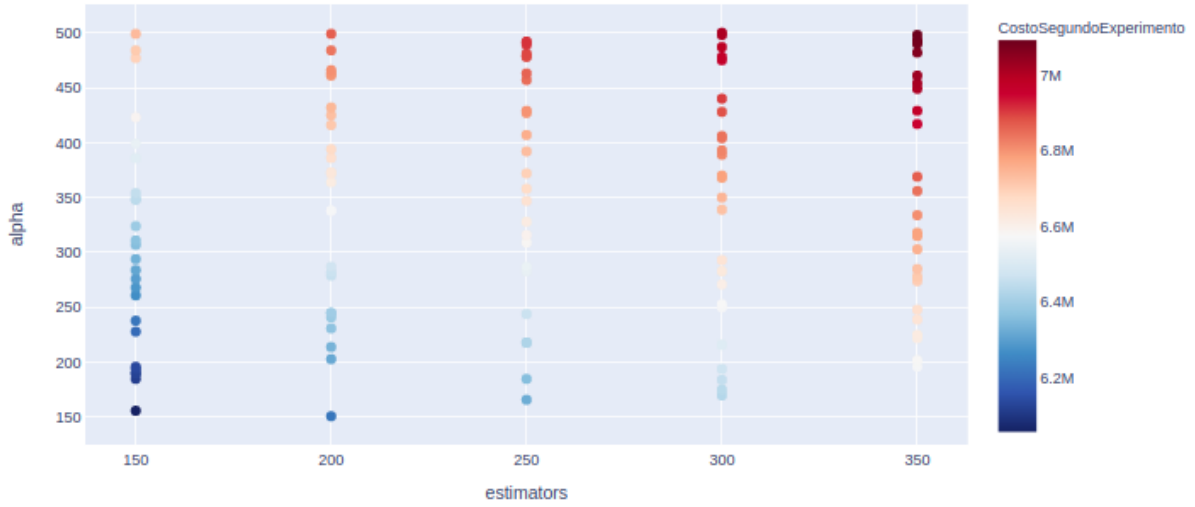


***Figura 65.*** *Scatter plot que ilustra mediante los colores la magnitud de la métrica de costo de nuestro segundo experimento, considerando mayores aquellos puntos que tienden a tener un color más rojo oscuro y menores aquellos puntos con un color más azul oscuro. Este gráfico corresponde a corridas con max_depth igual a 12.*

#### *4.5.4.1.2 ANÁLISIS DE AUC DEL SEGUNDO MODELO NGBOOST*

Considerando la diferencia entre el AUC de train y el AUC de test, podemos apreciar los siguientes gráficos, en los cuales se puede ver que a mayor max_depth, entonces mayor es la magnitud de esta diferencia.

A diferencia de la anterior sección podemos observar que la escala de colores cambia con respecto al eje x, es decir, con estimators.

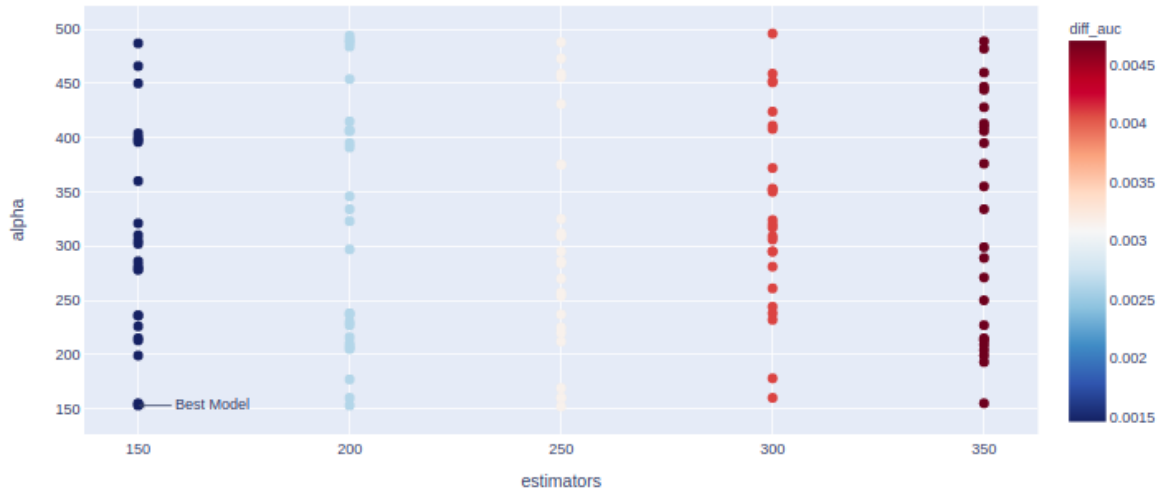


***Figura 66.*** *Scatter plot que ilustra mediante los colores la magnitud de la diferencia del AUC de train y el AUC de test del modelo entrenado para nuestro segundo experimento, considerando mayores aquellos puntos que tienden a tener un color más rojo oscuro y menores aquellos puntos con un color más azul oscuro. Este gráfico corresponde a corridas con max_depth igual a 8.*

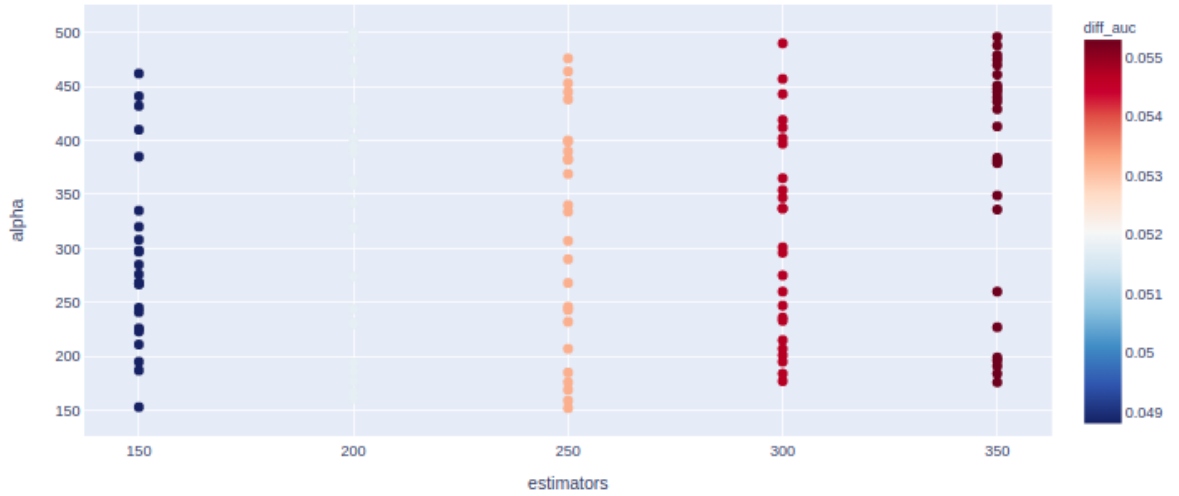


***Figura 67.*** *Scatter plot que ilustra mediante los colores la magnitud de la diferencia del AUC de train y el AUC de test del modelo entrenado para nuestro segundo experimento, considerando mayores aquellos puntos que tienden a tener un color más rojo oscuro y menores aquellos puntos con un color más azul oscuro. Este gráfico corresponde a corridas con max_depth igual a 10.*

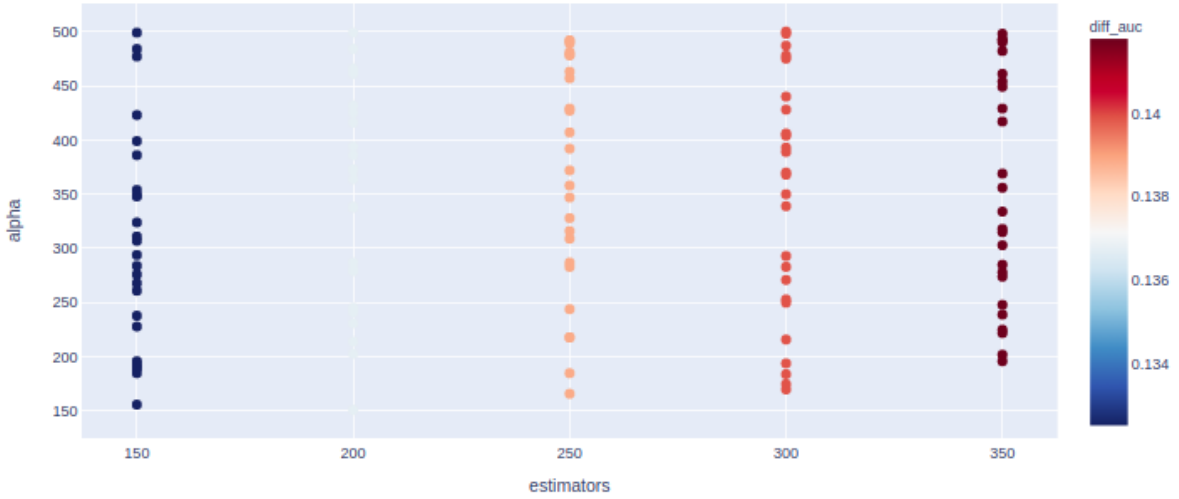


***Figura 68.*** *Scatter plot que ilustra mediante los colores la magnitud de la diferencia del AUC de train y el AUC de test del modelo entrenado para nuestro segundo experimento, considerando mayores aquellos puntos que tienden a tener un color más rojo oscuro y menores aquellos puntos con un color más azul oscuro. Este gráfico corresponde a corridas con max_depth igual a 12*

### 4.5.5. SEGUNDO MODELO NGBOOST

Entrenando bajo esta nueva modalidad y con una nueva función objetivo, entrenaremos nuevamente nuestros modelos para determinar entonces los mejores hiper parámetros tales que optimicen los mismos del primer experimento (es decir, estimators, learning_rate, max_depth y threshold), pero con la particularidad de que contaremos nuevamente con un

hiper parámetro alpha que pondera la asignación de crédito sobre aquellas observaciones con mayor incertidumbre.

Podemos observar la tabla del **Anexo 8**, que representa el top 50 de todos los resultados obtenidos de la exploración de hiper parámetros para encontrar el nuestro segundo modelo **NGBoost** más óptimo, para el cual, se entrenaron más de 20 modelos **NGBoost**, donde se mantuvo fijo **learning_rate** en 0.01, dado los buenos resultados del primer experimento. Sólamente se exploraron valores sobre los hiper parámetros **n_estimators** y **Base** (más específicamente **max_depth**) del modelo. Otros valores que se exploraron (externos a la implementación del modelo) son los siguientes:

- alpha: Se hicieron entre más de 3000 corridas (debido a que se fue refinando en cada iteración para poder encontrar un rango apropiado del mismo y además, algunos de estos resultaban en costos finales negativos, lo cual no tiene sentido) tomando valores aleatorios entre 100 y 10,000 para una primera etapa. Finalmente, se pudo concluir que un rango de valores acorde para alpha serían aproximadamente entre 100 a 700, dando así resultados más coherentes en términos de costo.
- threshold: Recordamos que es para $P(y = 1 \mid X)$. Se exploró para valores desde 0.25 a 0.45, recordando que hacemos esto ya que consideramos asignar más créditos sobre observaciones con mayor incertidumbre.

Algo a destacar de estos experimentos para obtener nuestro segundo mejor modelo, es que tanto para threshold que asumen valores de 0.35, 0.4 y 0.45, el mejor modelo para cada uno de estos mantiene la misma configuración en los 3 casos, como se puede observar en los Anexos 5, 6 y 7.

Acorde a la tabla del **Anexo 8**, que representa el top 50 de todos los resultados obtenidos de la exploración de hiper parámetros para encontrar el nuestro segundo modelo **NGBoost** más óptimo, seleccionaremos entonces el mejor modelo resultante representado en la siguiente Tabla:

| Hiper parámetro | Valores |
|---|---|
| Base | DecisionTreeRegressor(ccp_alpha=0.0, criterion='friedman_mse', max_depth=8, max_features=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, presort='deprecated', random_state=2020, splitter='best' |
| Dist | Bernoulli |

| Score | LogScore (ngboost score module) |
|---|---|
| learning_rate | 0.01 |
| minibatch_frac | 1.0 |
| n_estimators | 150 |
| natural_gradient | True |
| tol | 0.0001 |

***Tabla 26.*** *Tabla con los hiper parámetros asignados al modelo con el cual entrenamos para realizar nuestras predicciones. Los features que fueron explorados están subrayados.*

### 4.5.5.1. RESULTADOS DEL SEGUNDO MODELO NGBOOST

#### *4.5.5.1.1. PERFORMANCE DEL MODELO - AUC Y ROC*

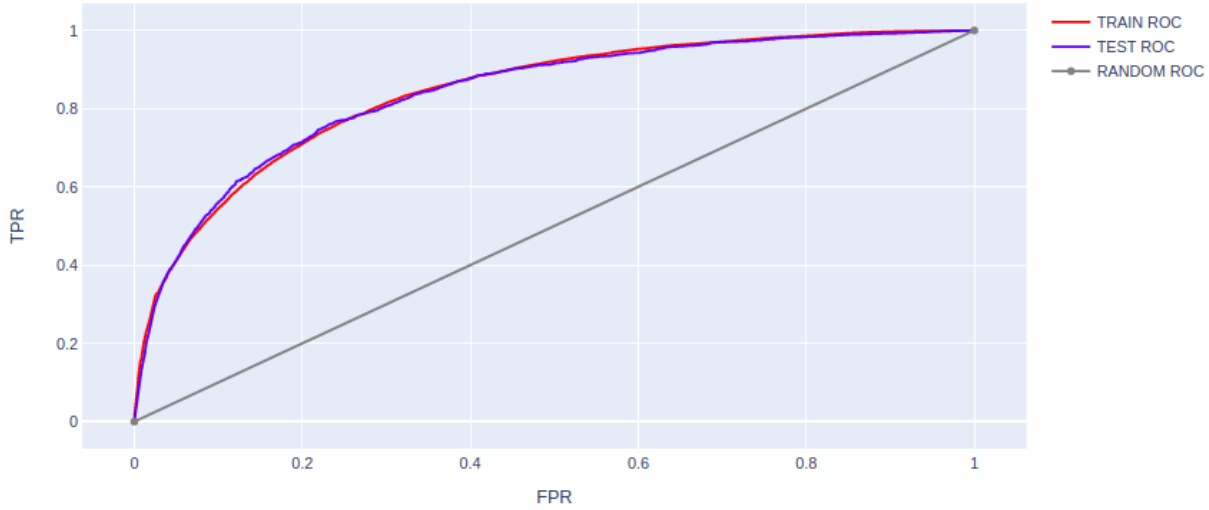


***Figura 69.*** *Curva de ROC para nuestro modelo del segundo experimento con los parámetros de la Tabla 26.*

En cuanto a ROC curve, podemos notar un buen rendimiento. La métrica AUC para train dió un resultado de 0.841 y para test dió un resultado de 0.839. Recordemos como bien dijimos en el anterior experimento, esta métrica AUC no considera ponderaciones, por lo tanto no es lo mismo dar crédito y generar default que no darlo y perder un fee de transacción. Entonces, para nuestro análisis no es necesario que estos resultados sean buenos, dado que nuestro foco no son las técnicas de machine learning tradicional, sino más bien basado en un enfoque cost-sensitive.

Sin embargo, es válido mencionar que entre los resultados obtenidos por el algoritmo, esta diferencia entre el AUC de train y el AUC de test varía entre 0.001 y 0.14, con lo cual, como bien observamos en nuestros resultados, estamos en el extremo inferior de este rango.

#### *4.5.5.1.2. KDE DE LAS PROBABILIDADES PREDICHAS DEL SEGUNDO MODELO NGBOOST*

A fines de poder comparar con lo representado en el primer experimento en la Figura 46, podemos observar que sobre la región de incertidumbre, es decir, cercano a 0.5, la curva se torna constante y es más alta en comparación a la observada en el primer experimento.
Una aclaración al respecto es que no damos a visualizar el threshold dado que visualizamos la distribución empírica de la probabilidad estimada de que $y = 1$ .

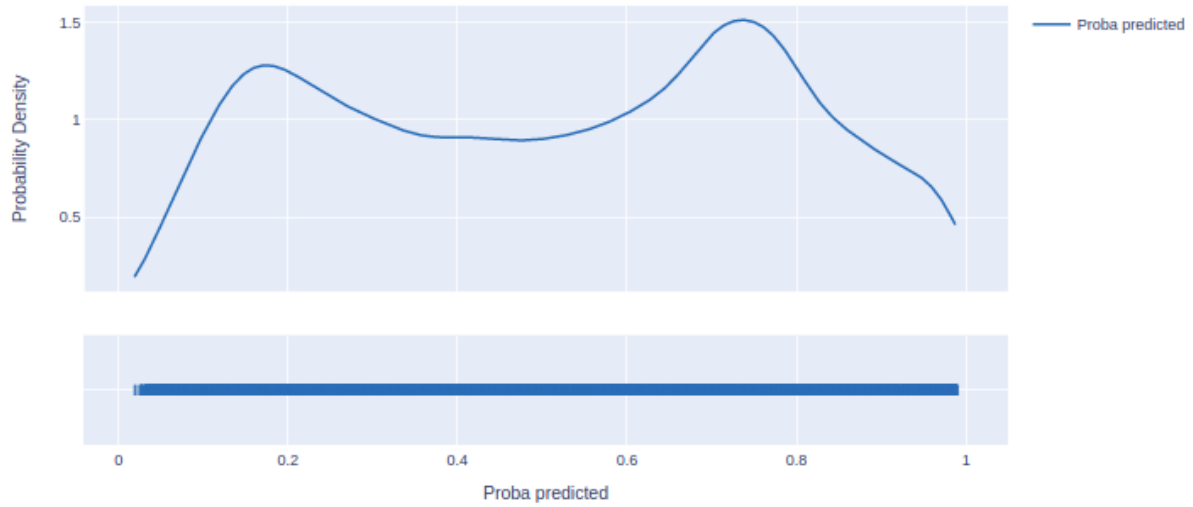


***Figura 70.*** *KDE de las probabilidades predichas para $y = 1$ para el modelo NGBoost del segundo experimento que contempla la incertidumbre y fue entrenado con 20% de missings sobre los features importantes expresados en el primer experimento.*

#### *4.5.5.1.3. PERFORMANCE DEL MODELO - COSTO*

Para un $\alpha = 153$ y un $threshold = 0.4$ nuestro modelo, en términos de costo, da los siguientes resultados:

- CostoPrimerExperimento (sin considerar incertidumbre): € 2,825,111.32
- Costo de Aprender: € 278,510.00
- CostoSegundoExperimento (considerando incertidumbre): € 3,103,621.32

Teniendo en cuenta que, según el algoritmo, serán otorgados 11,207 créditos y no se otorgarán 18,793 créditos.

A diferencia de nuestro primer experimento, donde obtuvimos 6,553 clientes a los cuales se les otorgará el crédito y 23,447 clientes a los que no se le otorgará, esta notable diferencia

es lógica ya que acorde a que esperábamos poder otorgar más créditos sobre observaciones con más incertidumbre.

## 4.6 EVALUACIÓN DE LA INVERSIÓN

### 4.6.1. CASO PRÁCTICO

Considerando que el banco restringe la inversión de aplicar el experimento 2 por sobre el experimento 1 en un porcentaje de máximo el 12%, procederemos a analizar la métrica **%_Inversión** a fines de armar un conjunto de reglas tales que el empleado del banco encargado de tomar la decisión respecto a la asignación de créditos pueda tener opciones, teniendo en cuenta distintos criterios, los cuales serán:

- Primer criterio: Menor costo a la hora de asignar créditos considerando la incertidumbre en las predicciones.
- Segundo criterio: Mayor **%_Inversión** (lo más cercano posible al 12% impuesto).
- Tercer criterio: Mayor cantidad de préstamos asignados
- Cuarto criterio: Menor costo de aprender

Entonces, seleccionando la configuración de nuestro mejor modelo del segundo experimento como referencia (definido en la **Tabla 26**), aplicaremos cálculos explorando los distintos valores de alpha y threshold para calcular la métrica definida para el primer experimento y la métrica definida para nuestro segundo experimento. Para cada iteración, obtendremos la métrica **%_Inversión** y definiremos threshold y alpha para dicho modelo (teniendo en cuenta la política impuesta y los criterios anteriormente mencionados) que el encargado de asignación de créditos debería contemplar para tomar su decisión.

Procesando 100 alphas aleatorios para cada threshold (desde 0.35 a 0.45 incluido), obtuvimos los siguientes resultados:

| alpha | threshold | count_zero | count_one | CostoPrimerExperimento | learning_cost | CostoSegundo Experimento | %_Inversión |
|---|---|---|---|---|---|---|---|
| 261 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 392,132.72 | 3,014,905.78 | 13.01 |
| 261 | 0.40 | 11,207 | 18,793 | 2,825,111.32 | 475,105.30 | 3,300,216.62 | 14.40 |
| 261 | 0.45 | 12,602 | 17,398 | 2,908,261.74 | 563,991.22 | 3,472,252.96 | 16.24 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 334 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 501,809.69 | 3,124,582.75 | 16.06 |
| 334 | 0.40 | 11,207 | 18,793 | 2,825,111.32 | 607,989.16 | 3,433,100.48 | 17.71 |
| 334 | 0.45 | 12,602 | 17,398 | 2,908,261.74 | 721,735.89 | 3,629,997.63 | 19.88 |
| 280 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 420,678.78 | 3,043,451.84 | 13.82 |
| 280 | 0.40 | 11,207 | 18,793 | 2,825,111.32 | 509,691.51 | 3,334,802.83 | 15.28 |
| 280 | 0.45 | 12,602 | 17,398 | 2,908,261.74 | 605,048.05 | 3,513,309.79 | 17.22 |
| 294 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 441,712.72 | 3,064,485.78 | 14.41 |
| 294 | 0.40 | 11,207 | 18,793 | 2,825,111.32 | 535,176.08 | 3,360,287.40 | 15.93 |
| 294 | 0.45 | 12,602 | 17,398 | 2,908,261.74 | 635,300.45 | 3,543,562.19 | 17.93 |
| 192 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 288,465.45 | 2,911,238.51 | 9.91 |
| 192 | 0.40 | 11,207 | 18,793 | 2,825,111.32 | 349,502.75 | 3,174,614.07 | 11.01 |
| 192 | 0.45 | 12,602 | 17,398 | 2,908,261.74 | 414,890.09 | 3,323,151.83 | 12.48 |
| 261 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 392,132.72 | 3,014,905.78 | 13.01 |
| 261 | 0.40 | 11,207 | 18,793 | 2,825,111.32 | 475,105.30 | 3,300,216.62 | 14.40 |
| 261 | 0.45 | 12,602 | 17,398 | 2,908,261.74 | 563,991.22 | 3,472,252.96 | 16.24 |
| 152 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 228,368.48 | 2,851,141.54 | 8.01 |
| 152 | 0.40 | 11,207 | 18,793 | 2,825,111.32 | 276,689.68 | 3,101,801.00 | 8.92 |

***Tabla 27.*** *Muestra de 20 cómputos para una corrida de 100 alphas para cada valor de threshold explorado.*

Podemos observar que hay cómputos para los cuales **%_Inversión** es mayor al requerimiento solicitado, es decir, 12%, entonces procederemos a filtrarlos y a analizar los criterios anteriormente detallados.

#### 4.6.1.1.Primer criterio

Si tomamos como criterio el menor **CostoSegundoExperimento,** obtenemos el siguiente listado de resultados:

| alpha | threshold | count_zero | count_one | CostoPrimer Experimento | learning_cost | CostoSegundo Experimento | %_Inversión |
|---|---|---|---|---|---|---|---|
| 152 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 228,368.48 | 2,851,141.54 | 8.01 |
| 154 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 231,373.33 | 2,854,146.39 | 8.11 |
| 155 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 232,875.75 | 2,855,648.81 | 8.15 |
| 157 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 235,880.60 | 2,858,653.66 | 8.25 |
| 163 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 244,895.15 | 2,867,668.21 | 8.54 |
| 164 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 246,397.57 | 2,869,170.63 | 8.59 |
| 171 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 256,914.54 | 2,879,687.60 | 8.92 |
| 173 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 259,919.39 | 2,882,692.45 | 9.02 |
| 180 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 270,436.36 | 2,893,209.42 | 9.35 |
| 187 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 280,953.33 | 2,903,726.39 | 9.68 |
| 191 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 286,963.03 | 2,909,736.09 | 9.86 |
| 192 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 288,465.45 | 2,911,238.51 | 9.91 |
| 193 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 289,967.87 | 2,912,740.93 | 9.96 |
| 194 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 291,470.30 | 2,914,243.36 | 10.00 |
| 197 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 295,977.57 | 2,918,750.63 | 10.14 |
| 198 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 297,480.00 | 2,920,253.06 | 10.19 |
| 199 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 298,982.42 | 2,921,755.48 | 10.23 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 152 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 228,368.48 | 2,851,141.54 | 8.01 |
| 154 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 231,373.33 | 2,854,146.39 | 8.11 |
| 155 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 232,875.75 | 2,855,648.81 | 8.15 |

***Tabla 28.*** *Mejores 20 cómputos para una corrida de 100 alphas para cada valor de threshold explorado, filtrando por la condición solicitada y ordenando por CostoSegundoExperimento en forma ascendente.*

Entonces, para este criterio, nuestro alpha será de 152 para un threshold de 0.35, otorgando 9,849 créditos y negando 20,151 créditos. El costo incurrido en aplicar el segundo experimento será de 2,851,141.54 euros, resultando en un porcentaje de inversión del 8,01%.

#### 4.6.1.2. Segundo criterio

Si consideramos ahora menor **%_Inversion** tenemos lo siguiente:

| alpha | threshold | count_zero | count_one | CostoPrimer Experimento | learning_cost | CostoSegundo Experimento | %_Inversión |
|---|---|---|---|---|---|---|---|
| 210 | 0.4 | 11,207 | 18,793 | 2,825,111.32 | 382,268.63 | 3,207,379.95 | 11.92 |
| 234 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 351,567.27 | 2,974,340.33 | 11.82 |
| 180 | 0.45 | 12,602 | 17,398 | 2,908,261.74 | 388,959.46 | 3,297,221.20 | 11.80 |
| 233 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 350,064.84 | 2,972,837.90 | 11.78 |
| 207 | 0.4 | 11,207 | 18,793 | 2,825,111.32 | 376,807.65 | 3,201,918.97 | 11.77 |
| 206 | 0.4 | 11,207 | 18,793 | 2,825,111.32 | 374,987.32 | 3,200,098.64 | 11.72 |
| 231 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 347,059.99 | 2,969,833.05 | 11.69 |
| 230 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 345,557.57 | 2,968,330.63 | 11.64 |
| 229 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 344,055.15 | 2,966,828.21 | 11.60 |
| 226 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 339,547.87 | 2,962,320.93 | 11.46 |
| 225 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 338,045.45 | 2,960,818.51 | 11.42 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 173 | 0.45 | 12,602 | 17,398 | 2,908,261.74 | 373,833.26 | 3,282,095.00 | 11.39 |
| 224 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 336,543.02 | 2,959,316.08 | 11.37 |
| 199 | 0.4 | 11,207 | 18,793 | 2,825,111.32 | 362,245.04 | 3,187,356.36 | 11.37 |
| 198 | 0.4 | 11,207 | 18,793 | 2,825,111.32 | 360,424.71 | 3,185,536.03 | 11.31 |
| 171 | 0.45 | 12,602 | 17,398 | 2,908,261.74 | 369,511.49 | 3,277,773.23 | 11.27 |
| 197 | 0.4 | 11,207 | 18,793 | 2,825,111.32 | 358,604.38 | 3,183,715.70 | 11.26 |
| 221 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 332,035.75 | 2,954,808.81 | 11.24 |
| 220 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 330,533.33 | 2,953,306.39 | 11.19 |
| 194 | 0.4 | 11,207 | 18,793 | 2,825,111.32 | 353,143.40 | 3,178,254.72 | 11.11 |

***Tabla 29.*** *Mejores 20 cómputos para una corrida de 100 alphas para cada valor de threshold explorado, filtrando por la condición solicitada y ordenando por %_Inversión en forma descendente.*

Para este criterio, nuestro alpha será de 210 para un threshold de 0.35, otorgando 11,207 créditos y negando 18,793 créditos. El costo incurrido en aplicar el segundo experimento será de 3,207,379.95 euros, resultando en un porcentaje de inversión del 11,92%.

#### 4.6.1.3. Tercer criterio

Ahora consideraremos el tercer criterio para el cual nos enfocaremos en considerar la mayor cantidad de préstamos asignados (count_zero).

| alpha | threshold | count_zero | count_one | CostoPrimer Experimento | learning_cost | CostoSegundo Experimento | %_Inversión |
|---|---|---|---|---|---|---|---|
| 163 | 0.45 | 12,602 | 17,398 | 2,908,261.74 | 352,224.40 | 3,260,486.14 | 10.80 |
| 173 | 0.45 | 12,602 | 17,398 | 2,908,261.74 | 373,833.26 | 3,282,095.00 | 11.39 |
| 157 | 0.45 | 12,602 | 17,398 | 2,908,261.74 | 339,259.08 | 3,247,520.82 | 10.45 |
| 154 | 0.45 | 12,602 | 17,398 | 2,908,261.74 | 332,776.43 | 3,241,038.17 | 10.27 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 180 | 0.45 | 12,602 | 17,398 | 2,908,261.74 | 388,959.46 | 3,297,221.20 | 11.80 |
| 152 | 0.45 | 12,602 | 17,398 | 2,908,261.74 | 328,454.66 | 3,236,716.40 | 10.15 |
| 155 | 0.45 | 12,602 | 17,398 | 2,908,261.74 | 334,937.31 | 3,243,199.05 | 10.33 |
| 164 | 0.45 | 12,602 | 17,398 | 2,908,261.74 | 354,385.29 | 3,262,647.03 | 10.86 |
| 171 | 0.45 | 12,602 | 17,398 | 2,908,261.74 | 369,511.49 | 3,277,773.23 | 11.27 |
| 164 | 0.4 | 11,207 | 18,793 | 2,825,111.32 | 298,533.60 | 3,123,644.92 | 9.56 |
| 155 | 0.4 | 11,207 | 18,793 | 2,825,111.32 | 282,150.66 | 3,107,261.98 | 9.08 |
| 173 | 0.4 | 11,207 | 18,793 | 2,825,111.32 | 314,916.54 | 3,140,027.86 | 10.03 |
| 194 | 0.4 | 11,207 | 18,793 | 2,825,111.32 | 353,143.40 | 3,178,254.72 | 11.11 |
| 152 | 0.4 | 11,207 | 18,793 | 2,825,111.32 | 276,689.68 | 3,101,801.00 | 8.92 |
| 210 | 0.4 | 11,207 | 18,793 | 2,825,111.32 | 382,268.63 | 3,207,379.95 | 11.92 |
| 157 | 0.4 | 11,207 | 18,793 | 2,825,111.32 | 285,791.31 | 3,110,902.63 | 9.19 |
| 171 | 0.4 | 11,207 | 18,793 | 2,825,111.32 | 311,275.89 | 3,136,387.21 | 9.92 |
| 197 | 0.4 | 11,207 | 18,793 | 2,825,111.32 | 358,604.38 | 3,183,715.70 | 11.26 |
| 193 | 0.4 | 11,207 | 18,793 | 2,825,111.32 | 351,323.08 | 3,176,434.40 | 11.06 |
| 192 | 0.4 | 11,207 | 18,793 | 2,825,111.32 | 349,502.75 | 3,174,614.07 | 11.01 |

***Tabla 30.*** *Mejores 20 cómputos para una corrida de 100 alphas para cada valor de threshold explorado, filtrando por la condición solicitada y ordenando por count_zero en forma descendente.*

Acorde a este criterio, nuestro alpha será de 163 para un threshold de 0.45, otorgando 12,602 créditos y negando 17,398 créditos. El costo incurrido en aplicar el segundo experimento será de 3,260,486.14 euros, resultando en un porcentaje de inversión del 10.80%.

#### 4.6.4.4. Cuarto criterio

Finalmente consideraremos el cuarto criterio, tomando el menor costo de aprendizaje como consideración para los siguientes cómputos:

| alpha | threshold | count_zero | count_one | CostoPrimer Experimento | learning_cost | CostoSegundo Experimento | %_Inversion |
|---|---|---|---|---|---|---|---|
| 152 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 228,368.48 | 2,851,141.54 | 8.01 |
| 154 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 231,373.33 | 2,854,146.39 | 8.11 |
| 155 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 232,875.75 | 2,855,648.81 | 8.15 |
| 157 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 235,880.60 | 2,858,653.66 | 8.25 |
| 163 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 244,895.15 | 2,867,668.21 | 8.54 |
| 164 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 246,397.57 | 2,869,170.63 | 8.59 |
| 171 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 256,914.54 | 2,879,687.60 | 8.92 |
| 173 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 259,919.39 | 2,882,692.45 | 9.02 |
| 180 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 270,436.36 | 2,893,209.42 | 9.35 |
| 152 | 0.4 | 11,207 | 18,793 | 2,825,111.32 | 276,689.68 | 3,101,801.00 | 8.92 |
| 154 | 0.4 | 11,207 | 18,793 | 2,825,111.32 | 280,330.33 | 3,105,441.65 | 9.03 |
| 187 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 280,953.33 | 2,903,726.39 | 9.68 |
| 155 | 0.4 | 11,207 | 18,793 | 2,825,111.32 | 282,150.66 | 3,107,261.98 | 9.08 |
| 157 | 0.4 | 11,207 | 18,793 | 2,825,111.32 | 285,791.31 | 3,110,902.63 | 9.19 |
| 191 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 286,963.03 | 2,909,736.09 | 9.86 |
| 192 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 288,465.45 | 2,911,238.51 | 9.91 |
| 193 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 289,967.87 | 2,912,740.93 | 9.96 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 194 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 291,470.30 | 2,914,243.36 | 10.00 |
| 197 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 295,977.57 | 2,918,750.63 | 10.14 |
| 163 | 0.4 | 11,207 | 18,793 | 2,825,111.32 | 296,713.27 | 3,121,824.59 | 9.50 |

***Tabla 31.*** *Mejores 20 cómputos para una corrida de 100 alphas para cada valor de threshold explorado, filtrando por la condición solicitada y ordenando por learning_cost en forma ascendente.*

Acorde a este criterio, nuestro alpha será de 152 para un threshold de 0.35, otorgando 9,849 créditos y negando 20,151 créditos. El costo incurrido en aplicar el segundo experimento será de 2,851,141.54 euros, resultando en un porcentaje de inversión del 8,01%.

### 4.6.2. MIDIENDO LA CALIDAD DE LOS CLIENTES NUEVOS

Para un rango de threshold dado, podemos medir la calidad de los clientes que reciben préstamos mediante el cálculo de sus lifetime value (CLV). Basándonos en un estudio sobre problemas de clasificación cost-sensitive[47], utilizaremos la siguiente fórmula de CLV:

$$CLV_{i,1} = \frac{\mu * s_{i,1}}{r - g}$$

En la que $\mu$ es el beneficio marginal promedio por unidad de uso del producto, $s_{i,1}$ se refiere al consumo del cliente $i$ durante el primer período, r es la tasa de descuento y g es la tasa de crecimiento en el consumo del cliente. $CLV_{i,1}$ entonces es el Customer Lifetime Value del cliente $i$ durante el primer período.

Este ejercicio requeriría conocer todos estos parámetros por lo que vamos a trabajar con los CLV relativos, en relación al CLV de cliente de mejor valor que en versión propuesta de la métrica se reduce al de mayor $s_{i,1}$. El CLV relativo entonces se simplifica como:

$$CLVRelativo_i = \frac{CLV_{i,1}}{CLV_{j-máx,1}} * 100 = \frac{s_{i,1}}{s_{j-máx,1}} * 100$$

[47] A. Correa Bahnsen, Example-Dependent Cost-Sensitive Classification, Applications in Financial Risk Modelling and Marketing Analytics (2015), Página 49 https://github.com/albahnsen/phd-thesis/blob/master/Thesis_ExampleDependentCostSensitiveClassification.pdf

Donde el numerador es el valor del consumo del cliente $i$ durante el primer período y el denominador es el máximo valor de consumo de los clientes que ingresan.

Entonces, realizando este enfoque definimos nuestra métrica $s_{i1}$ como:

```python
def process_comsumption_customer_first_period(x, rate):
    """
    Compute metric s_i_1

    Args:
        - x (pd.Series): row that contains necessary data to compute the metric.
        - rate (float): disccount rate

    Returns computed value for metric s_i_1.
    """
    return x["LoanPrincipal"] * rate * (1-x["y_true"]) - x["LoanPrincipal"] *  x["y_true"]
```

***Figura 71.*** *Definición de la función que calcula $s_{i1}$ para un cliente $i$*

Los CLV en términos relativos, con respecto al del cliente de mayor valor, se visualizan en los siguientes gráficos. Las líneas verticales representan los threshold optimizados para cada modelo mientras que el eje x tiene las probabilidades de default en base al modelo para cada observación.

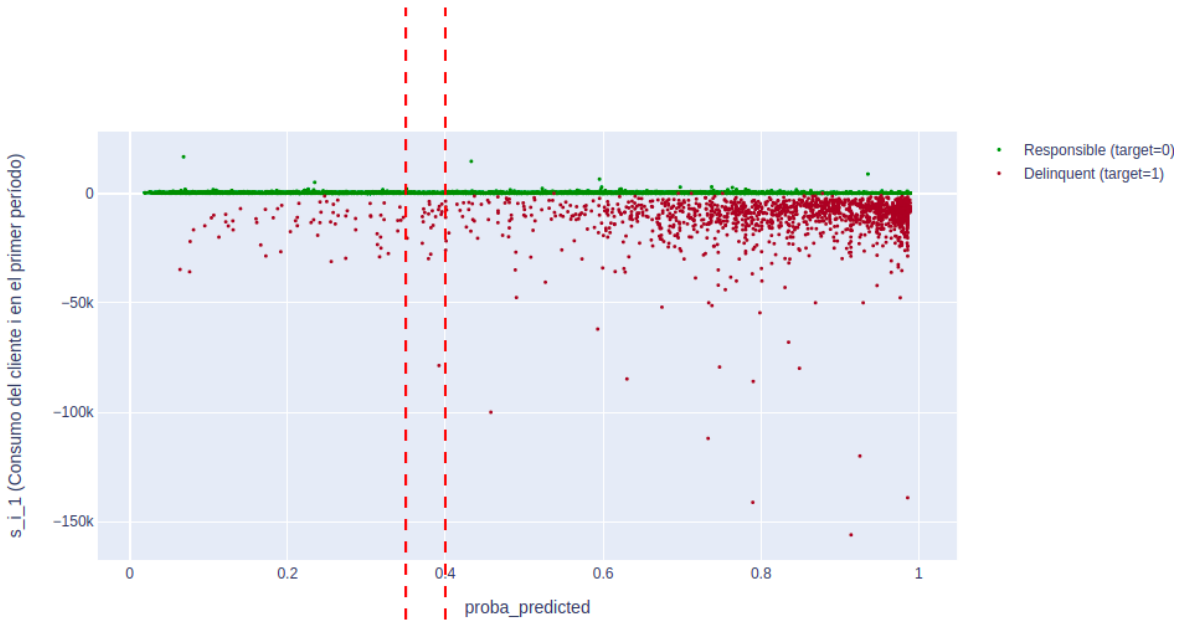


***Figura 72.*** *Scatter plot que representa los $s_{i1}$ de todos los clientes con su respectiva probabilidad predicha por nuestro modelo NGBoost definido en la Tabla 26. El color verde indica el valor que asume la variable target en el conjunto de test (y_test) cuando es igual a 0 y un color rojo cuando es igual a 1. Las franjas verticales cortadas de color rojo indican el intervalo del threshold entre 0.35 a 0.4.*

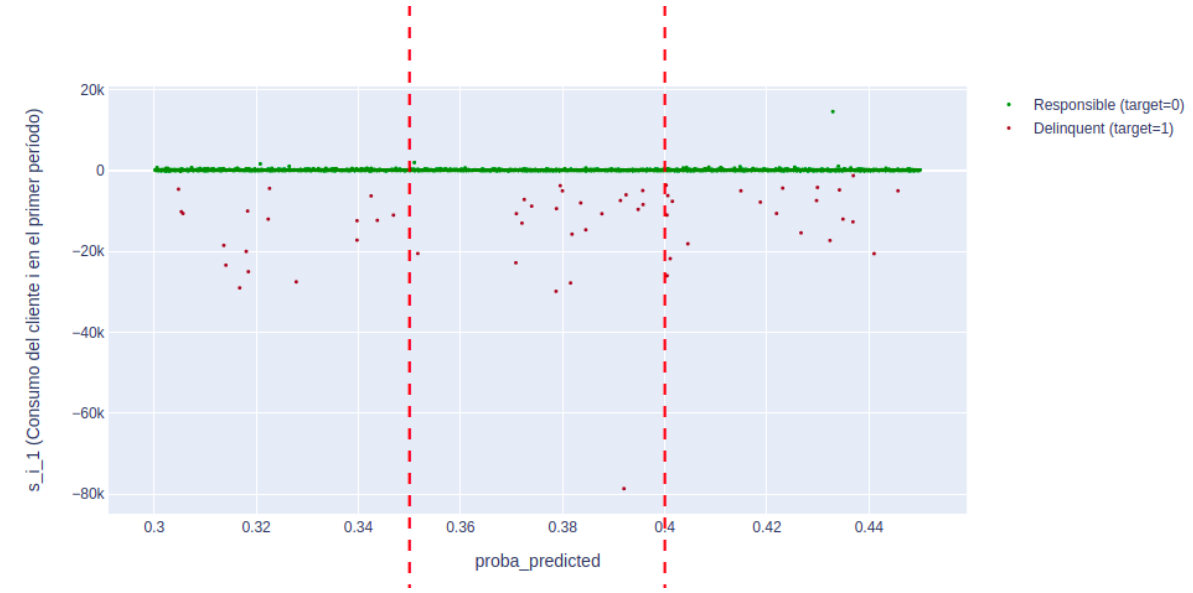


***Figura 73.*** *Scatter plot (acotado para probabilidades predichas desde 0.3 a 0.45) que representa los $s_{i1}$ de todos los clientes con su respectiva probabilidad predicha por nuestro modelo NGBoost definido en la Tabla 26. El color verde indica el valor que asume la variable target en el conjunto de test (y_test) cuando es igual a 0 y un color rojo cuando es igual a 1. Las franjas verticales cortadas de color rojo indican el intervalo del threshold entre 0.35 a 0.4.*

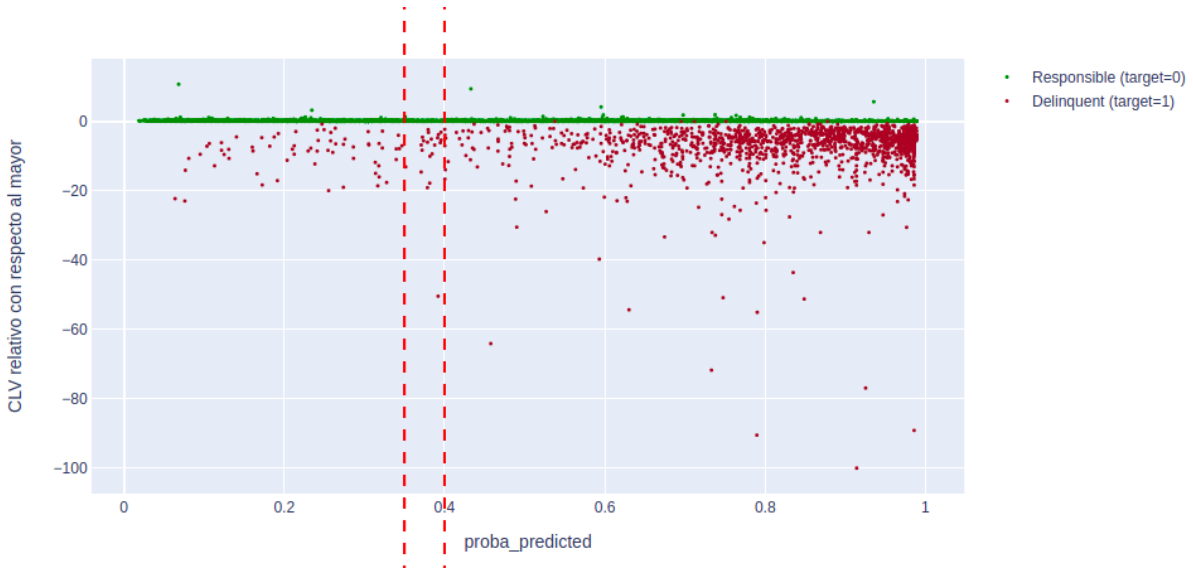


***Figura 74.*** *Scatter plot que representa los $CLV\ Relativo_{i1}$ de todos los clientes (con respecto al mayor valor) con su respectiva probabilidad predicha por nuestro modelo NGBoost definido en la Tabla 26. El color verde indica el valor que asume la variable target en el conjunto de test (y_test) cuando es igual a 0 y un color rojo cuando es igual a 1. Las franjas verticales cortadas de color rojo indican el intervalo del threshold entre 0.35 a 0.4.*

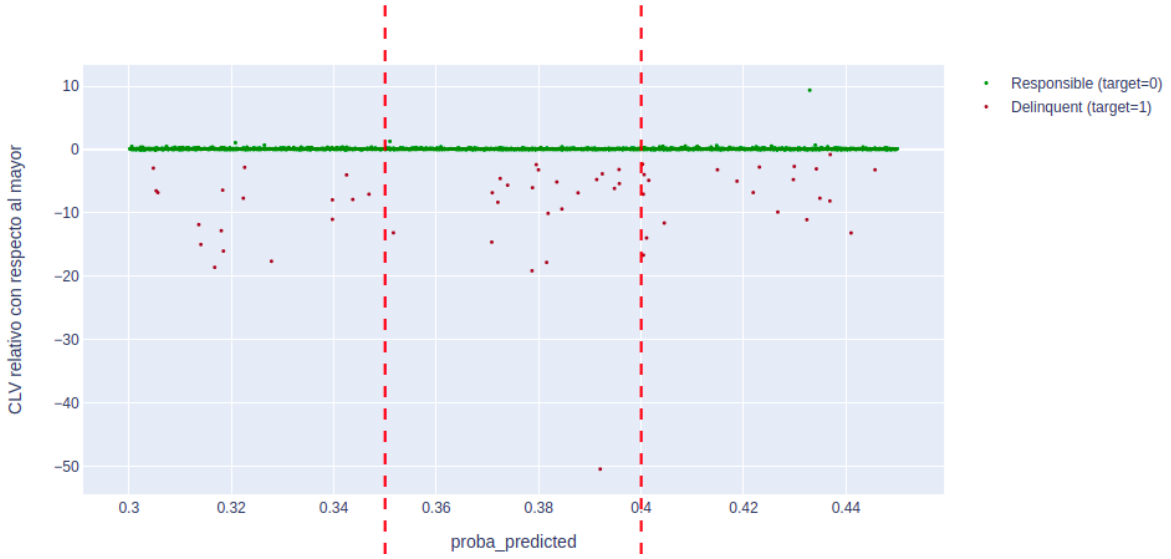


***Figura 74.*** *Scatter plot (acotado para probabilidades predichas entre 0.3 y 0.45) que representa los* $CLV\ Relativo_{i1}$ *de todos los clientes (con respecto al mayor valor) con su respectiva probabilidad predicha por nuestro modelo NGBoost definido en la Tabla 26. El color verde indica el valor que asume la variable target en el conjunto de test (y_test) cuando es igual a 0 y un color rojo cuando es igual a 1. Las franjas verticales cortadas de color rojo indican el intervalo del threshold entre 0.35 a 0.4.*

Podemos apreciar dentro del intervalo de thresholds (las líneas rojas verticales) de los gráficos que la gran concentración de préstamos se da para los casos donde **y_test = 0 (Responsible)** siendo estos considerablemente pequeños. Por otro lado, a comparación del caso anterior, para **y_test = 1 (Delinquent)** hay muchísima menos concentración de puntos a lo largo de los gráficos.

Mirando los gráficos a lo largo de las probabilidades predichas, se puede observar también una gran concentración de préstamos que fueron etiquetados como **y_test = 1 (Delinquent)** a medida que su valor asociado de probabilidad predicha se acerca a 1. Para **y_test = 0 (Responsible)** notamos la misma observación del párrafo anterior, su concentración aparece principalmente para valores bajos a lo largo de los gráficos ilustrados.

# 5. ESCALABILIDAD

En esta sección tiene como objetivo brindar una breve descripción sobre distintas extensiones posibles del presente trabajo.

## 5.1. Datos

El dataset trabajado no incluyó información generada por las interacciones de usuario en el contexto de las aplicaciones móviles. Algunos estudios recientes indican que dicha información es particularmente útil para predecir el comportamiento financiero en personas jóvenes y de bajos recursos, quienes también tienen más probabilidades de incumplir con los préstamos otorgados. Un ejemplo de este tipo de información serían transacciones en algunos *marketplaces* sobre los cuales el usuario opera, como por ejemplo las aplicaciones de delivery.

Otra información relevante podría ser aquella referida al movimiento de las personas en un área geográfica, como patrones de movilidad o el método de transporte más utilizado. Estos datos son generados por los usuarios de sistemas como Didi, Uber, Lime, Mobike, entre otros.[48]

Existen otros estudios que también hacen referencia al potencial de este tipo de información para mejorar la inclusión financiera en los países en desarrollo al facilitar el acceso al crédito para los prestatarios con poco o ningún historial crediticio. Los beneficios de esta aplicación son sustanciales ya que pueden ayudar a aumentar el bienestar financiero de numerosas personas en todo el mundo. Sin embargo, su éxito depende en gran medida del uso adecuado de features con poder predictivo en estos modelos de clasificación crediticia.[49]

Un ejemplo de un conjunto de datos sobre el cual sería interesante aplicar la metodología presentada en este trabajo es el provisto por FreddieMac[50], que provee una amplia variedad de información, tanto en cuanto a los features como a lo largo del tiempo, ya que provee información anual y cuatrimestral. Esta información disponible pertenece a un periodo desde

[48] *L. Roa, A. Correa-Bahnsen, G. Suárez, F. Cortés-Tejada, M. Luque, C. Bravo, Super-App Behavioral Patterns in Credit Risk Models: Financial, Statistical and Regulatory Implications (9 de Mayo del 2020)*

[49] *M. Óskarsdóttir, C. Bravo , C. Sarraute , B. Baesens, J. Vanthienen, Credit Scoring for Good: Enhancing Financial Inclusion with Smartphone-Based Microlending (Enero 2020)* https://arxiv.org/pdf/2001.10994.pdf

[50] *Single Family Loan-Level Dataset* http://www.freddiemac.com/research/datasets/sf_loanlevel_dataset.page

el 1 de enero de 1999 y el 31 de diciembre de 2018, cuyo conjunto de datos cubre aproximadamente 27.8 millones de hipotecas de tasa fija.

## 5.2. Técnicas de Machine Learning

En el presente trabajo no se implementaron técnicas tales como Regresión Logística, Decision Trees, Random Forest o las variaciones de boosting como es el caso de XGBoost, el cual suele ser un modelo que performa bastante bien en las competencias de Kaggle de este estilo. Es importante aclarar que la implementación de CatBoost[51] [52] en Python[53], de acuerdo con la documentación oficial, posee soporte para implementar métricas de costo tipo cost-sensitive.[54] [55] Además, no se consideraron técnicas sofisticadas de Deep Learning.

## 5.3. Métricas

### 5.3.1. H-Measure

En cuanto a las métricas, este análisis se podría extender considerando **H-measure**[56]. Esta métrica fue propuesta por D.J. Hand[57], en respuesta a las desventajas de la métrica del área bajo la curva. Permite fijar la distribución de las severidades relacionadas a la configuración independiente del clasificador en un problema dado. **H-measure** suele ajustarse muy bien en problemas donde los conjuntos de datos son desbalanceados.[58]

La misma posee una implementación en Python desarrollada por Christoforos Anagnostopoulos, que ha escrito papers sobre esta métrica. Este paquete implementa un

[51] *A. V. Dorogush, V. Ershov, A. Gulin, CatBoost: gradient boosting with categorical features support (2018), http://learningsys.org/nips17/assets/papers/paper_11.pdf*

[52] *L Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, A. Gulin, CatBoost: unbiased boosting with categorical features (2017), https://arxiv.org/abs/1706.09516*

[53] *Implementación de CatBoost en Python, https://github.com/catboost/catboost*

[54] *Variables used in formulas, sección de la documentación oficial de CatBoost, https://catboost.ai/docs/concepts/loss-functions-variables-used.html*

[55] *Classification: objectives and metrics, sección de la documentación oficial de CatBoost, https://catboost.ai/docs/concepts/loss-functions-classification.html*

[56] *https://www.hmeasure.net/ - Página web que contiene información relevante acerca de la métrica.*

[57] *D. Hand, Measuring classifier performance: a coherent alternative to the area under the ROC curve (2009).*

[58] *D. Hand, C. Anagnostopoulos, A better Beta for the H measure of classification performance (2013)*

gran número de métricas de performance de clasificación, entre ellas, AUC, Error Rate, Sensitivity y Specificity, además de H-measure. [59] También posee su implementación en R.[60]

### 5.3.2. Expected Maximum Profit Measure

Otra alternativa es la métrica **Expected Maximum Profit (EMP)**. Es una métrica de desempeño alineada a la maximización de beneficios. En general aplica a problemas de churn, aunque existen estudios realizados sobre problemas de riesgo crediticio. EMP es, entonces, una alternativa al AUC que representa los beneficios generados por los préstamos que fueron cumplidos y los costos causados por incumplimientos de préstamos. Como resultado, esta métrica permite la selección de modelos basados en las ganancias, es decir, permite a los expertos elegir el modelo de calificación crediticia que aumenta más la rentabilidad y, además, proporciona el valor de corte óptimo que se requiere en un contexto donde el puntaje de un modelo de puntaje crediticio debe transformarse en una decisión binaria. Esto es una característica que otras métricas de rendimiento no proporcionan y es una ventaja de la métrica EMP.[61] [62]

Posee implementación en R, creada por Cristian Bravo, Seppe vanden Broucke y Thomas Verbraken, con implementaciones en churn y credit scoring.[63] También posee su implementación en Python, pero principalmente enfocada en casos de churn.[64]

### 5.3.3. Customer Lifetime Value (CLV)

Es posible extender el presente trabajo realizando el cálculo de la métrica CLV, realizando otro enfoque de medición de la calidad de los clientes que ingresan cuando decidimos pasar de un threshold dado (por ejemplo, 0.35) a uno mayor (por ejemplo, 0.4).

Entre las extensiones con mayor complejidad podemos mencionar un paper donde proponen una metodología para poder calcular el mismo.[65]

---

[59] https://github.com/canagnos/hmeasure-python - *Implementación de H-Measure en Python, por C. Anagnostopoulos.*

[60] https://cran.r-project.org/web/packages/hmeasure/index.html - *Implementación de H-Measure en R por D. Hand, C. Anagnostopoulos.*

[61] *T. Verbraken, Business-Oriented Data Analytics: Theory and Case Studies (2013)*

[62] *T. Verbraken, W. Verbeke, B. Baesens, A Novel Profit Maximizing Metric for Measuring Classification Performance of Customer Churn Prediction Models (2012).*

[63] https://cran.r-project.org/web/packages/EMP/EMP.pdf - *Implementación de EMP Measure en R por C. Bravo, S. v. Broucke, T. Verbraken.*

[64] https://github.com/estripling/proflogit-python/ - *Implementación de EMP Measure en Python particularmente aplicado para problemas de churn.*

[65] *H. Aeron, T. Bhaskar, R. Sundararajan, A. Kumar, A metric for customer lifetime value of credit card customers in Journal of Database Marketing & Customer Strategy Management (Septiembre 2008)*

### 5.3.4. Implementación de una métrica de costos en NGBoost

Destacamos que los resultados presentados en este trabajo son realmente muy buenos, a pesar de no haber podido implementar las funciones de costos desarrolladas en este trabajo en la implementación del algoritmo **NGBoost**, por lo que implementar otro tipo de enfoques, tomando como base este trabajo, podría dar mejores resultados que los presentados.

Acorde a algunos estudios[66] mencionan que el Log Likelihood ponderado negativo es un límite superior ajustado y convexo de la pérdida empírica. Además, este estudio sostiene que LogLoss es analíticamente más manejable, siendo incluso ampliamente adoptada en las implementaciones de algoritmos de machine learning analizadas previo a este trabajo.

Existen formas de poder adaptar el código fuente de la implementación de **NGBoost** que no son triviales y superan el alcance del presente trabajo, ya que implican modificar en bajo nivel los algoritmos que hoy no soportan entrenar con métricas de error personalizadas y, por consiguiente, validar su correcto funcionamiento a nivel de la convergencia de la solución.[67] Para ampliar esto para quien considere extender el presente trabajo, podría implementarse una nueva scoring rule (teniendo en cuenta una distribución Bernoulli) basada en una adaptación de Cross-Entropy Loss.[68]

Una posibilidad de extensión de este trabajo, como mencionamos antes (ver *Sección 5.2 - Técnicas de Machine Learning*), es aplicar esta metodología implementando este tipo de métricas entrenando modelos CatBoost.

Otra posibilidad, un poco más compleja pero con posibilidad de extensiones, es adaptar el código fuente de la implementación del paquete Python costcla[69], aplicando la misma idea empleada en los otros modelos sobre los cuales tiene soporte, según su documentación.[70] Existe una implementación respecto a esto último que podría tomarse como referencia, pero

---

[66] *J. P. Dmochowski, P. Sajda, L. C. Parra, Maximum Likelihood in Cost-Sensitive Learning: Model Specification, Approximations, and Upper Bounds (2010)*
[67] *Código fuente del módulo Scores de NGBoost* https://github.com/stanfordmlgroup/ngboost/blob/master/ngboost/scores.py
[68] *A. Herpburn, R. McConville, R. Santos-Rodriguez, J. Cid-Sueiro, D García-García, Proper Losses for Learning with Example-Dependent Costs Proper Losses for Learning with Example-Dependent Costs (2018)*
[69] *Implementación de CostSensitiveClassification por PhD Alejandro Correa Bahnsen, como parte de su investigación de tesis para su doctorado,* https://github.com/albahnsen/CostSensitiveClassification
[70] *Modelos soportados por CostCla según su documentación oficial,* http://albahnsen.github.io/CostSensitiveClassification/Models.html

que, como se observa, requiere la aplicación de ciertos métodos para su adaptación.[71] Esto podría facilitar la extensión de este trabajo analizando otras métricas de interés[72] y los otros modelos ya soportados por costcla que mencionamos anteriormente.

[71] *Ejemplo de implementación en python de un DecisionTreeClassifier de scikit-learn en costcla,* https://github.com/albahnsen/CostSensitiveClassification/blob/master/costcla/models/cost_tree.py
[72] *Métricas, sección de la documentación oficial de costcla,* http://albahnsen.github.io/CostSensitiveClassification/Metrics.html

# 6. CONCLUSIÓN

En base a lo observado en ambos experimentos podemos ver que hay una buena performance predictiva desde el machine learning tradicional junto a una métrica de error basada en la matriz de costos del problema.

La ventaja de este enfoque es que podemos reflejar adecuadamente el costo asimétrico de los distintos tipos de errores. Por lo tanto, es posible plantear alternativas de decisión alineadas al objetivo de negocio, cuantificando el impacto de un approach (segundo experimento) sobre el otro (primer experimento). Esto posibilita que el encargado de riesgo crediticio cuente con varias opciones basadas en distintos criterios de inversión asociados a determinadas configuraciones de hiper parámetros como alpha y el threshold del modelo.

En el primer experimento el análisis de *Interpretabilidad Global* del modelo indica que no hay predominancia en importancia de unos pocos features mientras que el feature **MonthlyIncome_bool_missing** resulta importante. Recordamos que este feature marca registros con missings para **MonthlyIncome**. Esto es muy importante ya que este análisis podría extenderse a casos donde existen muchos registros con missings.

Otro punto a notar es que al modificar el conjunto de entrenamiento con un porcentaje de missings afectamos la performance del modelo, algo que era de esperar, pero nuestro nuevo modelo que evalúa bajo la métrica que considera la incertidumbre pudo resolver esta situación sin mayores problemas, notándose sólo un leve crecimiento en la región de incertidumbre que ilustra la **Figura 70**.

La metodología propuesta en este trabajo nos permite realizar evaluaciones de una estrategia base respecto de otra estrategia que contemple una nueva función objetivo basada en la misma matriz de costos pero con el objetivo adicional de aprender sobre los clientes. A su vez, podemos cuantificar la calidad de los clientes ingresantes cuando movemos el threshold de asignación del crédito lo que resulta muy práctico para entender en forma cuantitativa la performance de la regla de asignación del crédito.

# 7. BIBLIOGRAFÍA Y REFERENCIAS


Kaggle, plataforma de competencias de Data Science.

Anderson, R. (2007). The Credit Scoring Toolkit : Theory and Practice for Retail Credit Risk Management and Decision Automation. Oxford University Press.

W. Verbeke, B. Baesens, C. Bravo, Profit-Driven Business Analytics (2018)

S. Barocas, M. Hardt, A. Narayanan, Fairness and machine learning, Limitations and Opportunities. https://fairmlbook.org/

J. Thanaki. Machine Learning Solutions: Expert techniques to tackle complex machine learning problems using Python (2018).

Marco Tulio Ribeiro, Sameer Singh, Carlos Guestrin. Why should I trust you? Explaining the predictions of any classifier (2016).

Raymond T. Ng & Joerg Sander. LOF: Identifying Density-Based Local Outliers. (2000) [Paper Link]

A. Fernández, S.I Galar, R. Prati, B. Krawczyk, F. Herrera, Learning from Imbalanced Data Sets (2018). Springer.

L. Zhang, H. Ray, J. Priestley & S. Tan, Journal of Applied Statistics, A descriptive study of variable discretization and cost-sensitive logistic regression on imbalanced credit data (23 de Julio del 2019)

N. V. Chawla, K. W. Bowyer, L. O.Hall, W. P. Kegelmeyer, “SMOTE: synthetic minority over-sampling technique,” Journal of artificial intelligence research, 321-357, 2002. [Paper Link]

Hands-On Machine Learning with Scikit-Learn, Keras and Tensorflow (2019), 2nd Edition, Aurelien Geron.

T. Duan, A. Avati, D. Y. Ding, S. Basu, Andrew Y. Ng, and A. Schuler (2019). "NGBoost: Natural Gradient Boosting for Probabilistic Prediction." [Paper Link] [Project Page Link]

https://dkopczyk.quantee.co.uk/ngboost-explained/ - Blog post sobre NGBoost explicado por Dawid Kopczyk.

A. Burkov. The Hundred-Page Machine Learning Book (2019).

Claude Sammut, Geoffrey Webb, Encyclopedia of Machine Learning (2010).

S. Lundberg, S. Lee, A Unified Approach to Interpreting Model Predictions (22 de mayo del 2017) https://arxiv.org/abs/1705.07874

Christoph Molnar, Interpretable Machine Learning, a Guide for Making Black Box Models Explainable (2020). Fuente: https://christophm.github.io/interpretable-ml-book/

Marco Tulio Ribeiro, Sameer Singh, Carlos Guestrin. Why should I trust you? Explaining the predictions of any classifier (2016)

Christoph Molnar, Interpretable Machine Learning, a Guide for Making Black Box Models Explainable (2020).

Aaron Fisher, Cynthia Rudin, Francesca Dominici, All Models are Wrong, but Many are Useful: Learning a Variable's Importance by Studying an Entire Class of Prediction Models Simultaneously (2018).

Lessmann, S., Baesens, B., Seow, H.V. and Thomas, L.C.,(2015). Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. European Journal of Operational Research,247(1), pp.124-136

Quantifying uncertainty in Machine Learning predictions. María Navarro. PyData London 2019 (Slides).
Quantifying uncertainty in Machine Learning predictions. María Navarro. PyData London 2019 (Video).

Liliang Ren, Gen Sun and Jiaman Wu (2019).RoNGBa: A Robustly Optimized Natural Gradient Boosting Training Approach with Leaf Number Clipping [Paper Link]

Vineeth Balasubramanian, Shen-Shyang Ho, Vladimir Vovk (2014). Conformal Prediction for Reliable Machine Learning: Theory, Adaptations and Applications 1º Ed.

S. Amary, S.C. Douglas (1998). Why Natural Gradient? [Paper Link]

What is the natural gradient, and how does it work? Kevin Frans [Article Link]

L. Roa, A. Correa-Bahnsen, G. Suárez, F. Cortés-Tejada, M. Luque, C. Bravo, Super-App Behavioral Patterns in Credit Risk Models: Financial, Statistical and Regulatory Implications (9 de Mayo del 2020)

M. Óskarsdóttir, C. Bravo , C. Sarraute , B. Baesens, J. Vanthienen, Credit Scoring for Good: Enhancing Financial Inclusion with Smartphone-Based Microlending (Enero 2020) https://arxiv.org/pdf/2001.10994.pdf

D. Hand, Measuring classifier performance: a coherent alternative to the area under the ROC curve (2009).

D. Hand, C. Anagnostopoulos, A better Beta for the H measure of classification performance (2013)

T. Verbraken, Business-Oriented Data Analytics: Theory and Case Studies (2013)

T. Verbraken, W. Verbeke, B. Baesens, A Novel Profit Maximizing Metric for Measuring Classification Performance of Customer Churn Prediction Models (2012).

H. Aeron, T. Bhaskar, R. Sundararajan, A. Kumar, A metric for customer lifetime value of credit card customers in Journal of Database Marketing & Customer Strategy Management (Septiembre 2008)

L Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, A. Gulin, CatBoost: unbiased boosting with categorical features (2017), https://arxiv.org/abs/1706.09516

A. V. Dorogush, V. Ershov, A. Gulin, CatBoost: gradient boosting with categorical features support (2018), http://learningsys.org/nips17/assets/papers/paper_11.pdf

J. P. Dmochowski, P. Sajda, L. C. Parra, Maximum Likelihood in Cost-Sensitive Learning: Model Specification, Approximations, and Upper Bounds (2010)

A. Herpburn, R. McConville, R. Santos-Rodriguez, J. Cid-Sueiro, D García-García, Proper Losses for Learning with Example-Dependent Costs Proper Losses for Learning with Example-Dependent Costs (2018)

Documentación oficial del paquete python statsmodels https://www.statsmodels.org/stable

Documentación oficial del paquete python sklearn https://scikit-learn.org/stable/

Documentación oficial del paquete python imblearn https://imbalanced-learn.readthedocs.io/en/stable/

Guía de usuario del paquete python ngboost https://stanfordmlgroup.github.io/ngboost/intro.html

Documentación oficial del paquete python lime https://lime-ml.readthedocs.io/en/latest/

Documentación oficial del paquete python ELI5 https://eli5.readthedocs.io/en/latest/index.html

Documentación oficial del paquete python shap https://shap.readthedocs.io/en/latest/

Documentación oficial del paquete LIME https://lime.readthedocs.io/en/latest/

# 8. ANEXOS

## 8.1. ANEXO 1 - RESULTADOS DEL PRIMER EXPERIMENTO ITERANDO INDIVIDUALMENTE CADA HIPER PARÁMETRO

Las siguientes tablas representan las corridas de cada hiper parámetro (estimators, learning_rate y max_depth) en forma individual, manteniendo los demás hiper parámetros del modelo **NGBoost** por default, tal como está definido en la implementación de la librería.

### 8.1.1. Estimators

| estimators | CostoPrimerExperimento | count_zero | count_one | train_auc | test_auc |
|---|---|---|---|---|---|
| 50 | 2,817,421.83 | 0 | 30,000 | 0.82 | 0.83 |
| 75 | 2,817,421.83 | 0 | 30,000 | 0.83 | 0.84 |
| 100 | 2,817,421.83 | 0 | 30,000 | 0.83 | 0.84 |
| 200 | 2,817,421.83 | 0 | 30,000 | 0.83 | 0.84 |
| 300 | 2,817,421.83 | 0 | 30,000 | 0.83 | 0.84 |
| 500 | 2,817,421.83 | 0 | 30,000 | 0.83 | 0.84 |
| 700 | 2,817,421.83 | 0 | 30,000 | 0.83 | 0.84 |

### 8.1.2. Learning_rate

| learning_rate | CostoPrimerExperimento | count_zero | count_one | train_auc | test_auc |
|---|---|---|---|---|---|
| 0.1 | 2,817,421.83 | 0 | 30,000 | 0.83 | 0.84 |
| 0.01 | 2,817,421.83 | 0 | 30,000 | 0.83 | 0.84 |
| 0.001 | 2,817,421.83 | 0 | 30,000 | 0.82 | 0.83 |

| | | | | | |
|---|---|---|---|---|---|
| 0.0001 | 2,817,421.83 | 0 | 30,000 | 0.80 | 0.81 |

### 8.1.3. max_depth

| max_depth | CostoPrimer Experimento | count_zero | count_one | train_auc | test_auc |
|---|---|---|---|---|---|
| 4 | 2,817,421.84 | 0 | 30,000 | 0.84 | 0.84 |
| 6 | 2,756,356.94 | 1,806 | 28,194 | 0.84 | 0.85 |
| 8 | 2,550,849.74 | 8,084 | 21,916 | 0.86 | 0.85 |
| 12 | 4,121,976.58 | 14,838 | 15,162 | 0.93 | 0.79 |
| 16 | 9,856,730.12 | 22,499 | 7,501 | 0.99 | 0.73 |

## 8.2. ANEXO 2 - RESULTADOS DE CORRIDAS DEL PRIMER EXPERIMENTO

El siguiente anexo corresponde al top 50 de resultados del primer experimento, considerando el valor de **CostoPrimerExperimento**, que es la métrica cuyo valor quisimos minimizar en los entrenamientos de los modelos.

| estimators | learning_rate | max_depth | threshold | CostoPrimer Experimento | count_zero | count_one | train_auc | test_auc |
|---|---|---|---|---|---|---|---|---|
| 150 | 0.01 | 8 | 0.2 | 2,443,875.00 | 6,553 | 23,447 | 0.85 | 0.84 |
| 250 | 0.01 | 8 | 0.2 | 2,489,196.36 | 7,619 | 22,381 | 0.85 | 0.85 |
| 100 | 0.01 | 8 | 0.25 | 2,524,173.38 | 7,441 | 22,559 | 0.85 | 0.84 |
| 150 | 0.01 | 8 | 0.25 | 2,531,405.86 | 8,308 | 21,692 | 0.85 | 0.84 |
| 350 | 0.01 | 8 | 0.2 | 2,538,652.10 | 7,859 | 22,141 | 0.86 | 0.85 |
| 500 | 0.01 | 8 | 0.2 | 2,550,849.74 | 8,084 | 21,916 | 0.86 | 0.85 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 150 | 0.01 | 8 | 0.3 | 2,552,349.22 | 9,786 | 20,214 | 0.85 | 0.84 |
| 100 | 0.01 | 8 | 0.3 | 2,554,968.32 | 9,116 | 20,884 | 0.85 | 0.84 |
| 100 | 0.01 | 8 | 0.2 | 2,564,375.24 | 5,234 | 24,766 | 0.85 | 0.84 |
| 50 | 0.01 | 8 | 0.3 | 2,583,292.32 | 6,571 | 23,429 | 0.85 | 0.84 |
| 250 | 0.01 | 8 | 0.25 | 2,587,614.14 | 9,157 | 20,843 | 0.85 | 0.85 |
| 500 | 0.001 | 8 | 0.3 | 2,605,409.26 | 6,554 | 23,446 | 0.85 | 0.84 |
| 500 | 0.01 | 6 | 0.3 | 2,607,434.86 | 5,841 | 24,159 | 0.84 | 0.85 |
| 350 | 0.01 | 6 | 0.3 | 2,611,003.76 | 5,784 | 24,216 | 0.84 | 0.85 |
| 250 | 0.01 | 8 | 0.3 | 2,620,899.80 | 10,619 | 19,381 | 0.85 | 0.85 |
| 250 | 0.01 | 6 | 0.3 | 2,629,245.98 | 5,642 | 24,358 | 0.84 | 0.85 |
| 500 | 0.01 | 8 | 0.25 | 2,642,065.08 | 9,690 | 20,310 | 0.86 | 0.85 |
| 350 | 0.01 | 8 | 0.3 | 2,650,479.14 | 10,938 | 19,062 | 0.86 | 0.85 |
| 150 | 0.01 | 6 | 0.3 | 2,650,535.24 | 5,366 | 24,634 | 0.84 | 0.85 |
| 350 | 0.01 | 8 | 0.25 | 2,658,913.58 | 9,454 | 20,546 | 0.86 | 0.85 |
| 350 | 0.001 | 8 | 0.3 | 2,669,149.88 | 3,444 | 26,556 | 0.84 | 0.84 |
| 500 | 0.01 | 8 | 0.3 | 2,698,618.76 | 11,188 | 18,812 | 0.86 | 0.85 |
| 100 | 0.01 | 6 | 0.3 | 2,702,228.18 | 4,988 | 25,012 | 0.84 | 0.85 |
| 500 | 0.01 | 6 | 0.25 | 2,717,801.84 | 4,002 | 25,998 | 0.84 | 0.85 |
| 100 | 0.01 | 6 | 0.25 | 2,718,706.18 | 2,937 | 27,063 | 0.84 | 0.85 |
| 350 | 0.01 | 6 | 0.25 | 2,722,260.36 | 3,954 | 26,046 | 0.84 | 0.85 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 250 | 0.01 | 6 | 0.25 | 2,726,481.68 | 3,885 | 26,115 | 0.84 | 0.85 |
| 500 | 0.001 | 8 | 0.25 | 2,728,722.12 | 3,359 | 26,641 | 0.85 | 0.84 |
| 150 | 0.01 | 6 | 0.25 | 2,729,621.18 | 3,453 | 26,547 | 0.84 | 0.85 |
| 50 | 0.01 | 8 | 0.25 | 2,733,827.18 | 3,222 | 26,778 | 0.85 | 0.84 |
| 50 | 0.01 | 6 | 0.3 | 2,739,897.88 | 2,259 | 27,741 | 0.84 | 0.85 |
| 50 | 0.01 | 8 | 0.2 | 2,755,867.56 | 966 | 29,034 | 0.85 | 0.84 |
| 500 | 0.001 | 6 | 0.3 | 2,756,295.28 | 1,876 | 28,124 | 0.84 | 0.85 |
| 500 | 0.01 | 6 | 0.2 | 2,756,356.94 | 1,806 | 28,194 | 0.84 | 0.85 |
| 350 | 0.01 | 6 | 0.2 | 2,759,325.28 | 1,772 | 28,228 | 0.84 | 0.85 |
| 250 | 0.01 | 6 | 0.2 | 2,760,032.94 | 1,756 | 28,244 | 0.84 | 0.85 |
| 350 | 0.001 | 12 | 0.25 | 2,760,660.32 | 6,341 | 23,659 | 0.90 | 0.79 |
| 50 | 0.01 | 12 | 0.2 | 2,763,658.62 | 7,212 | 22,788 | 0.91 | 0.79 |
| 500 | 0.001 | 8 | 0.2 | 2,763,842.08 | 950 | 29,050 | 0.85 | 0.84 |
| 500 | 0.001 | 12 | 0.2 | 2,778,157.84 | 7,071 | 22,929 | 0.90 | 0.79 |
| 100 | 0.01 | 6 | 0.2 | 2,781,904.32 | 959 | 29,041 | 0.84 | 0.85 |
| 150 | 0.01 | 6 | 0.2 | 2,786,986.58 | 1,444 | 28,556 | 0.84 | 0.85 |
| 50 | 0.01 | 6 | 0.25 | 2,788,337.10 | 318 | 29,682 | 0.84 | 0.85 |
| 500 | 0.001 | 6 | 0.25 | 2,792,592.96 | 183 | 29,817 | 0.84 | 0.85 |
| 250 | 0.001 | 8 | 0.3 | 2,793,355.54 | 510 | 29,490 | 0.84 | 0.84 |
| 350 | 0.001 | 8 | 0.25 | 2,793,988.54 | 699 | 29,301 | 0.84 | 0.84 |
| 350 | 0.001 | 8 | 0.2 | 2,814,000.38 | 30 | 29,970 | 0.84 | 0.84 |

## 8.3. ANEXO 3 - RESULTADOS DEL PRIMER EXPERIMENTO CON THRESHOLD = 0.2

La siguiente representación tabular es la filtración de los resultados del primer experimento cuyo threshold es igual a 0.2. Se pueden observar algunos valores inconsistentes sobre el final de la misma dado a que dicha combinación de hiper parámetros es pésima dado este threshold. En general son resultados con un **learning_rate** muy bajo.

| estimators | learning_rate | max_depth | threshold | CostoPrimer Experimento | count_zero | count_one | train_auc | test_auc |
|---|---|---|---|---|---|---|---|---|
| 150 | 0.01 | 8 | 0.2 | 2,443,875.00 | 6,553 | 23,447 | 0.85 | 0.84 |
| 250 | 0.01 | 8 | 0.2 | 2,489,196.36 | 7,619 | 22,381 | 0.85 | 0.85 |
| 350 | 0.01 | 8 | 0.2 | 2,538,652.10 | 7,859 | 22,141 | 0.86 | 0.85 |
| 500 | 0.01 | 8 | 0.2 | 2,550,849.74 | 8,084 | 21,916 | 0.86 | 0.85 |
| 100 | 0.01 | 8 | 0.2 | 2,564,375.24 | 5,234 | 24,766 | 0.85 | 0.84 |
| 50 | 0.01 | 8 | 0.2 | 2,755,867.56 | 966 | 29,034 | 0.85 | 0.84 |
| 500 | 0.01 | 6 | 0.2 | 2,756,356.94 | 1,806 | 28,194 | 0.84 | 0.85 |
| 350 | 0.01 | 6 | 0.2 | 2,759,325.28 | 1,772 | 28,228 | 0.84 | 0.85 |
| 250 | 0.01 | 6 | 0.2 | 2,760,032.94 | 1,756 | 28,244 | 0.84 | 0.85 |
| 50 | 0.01 | 12 | 0.2 | 2,763,658.62 | 7,212 | 22,788 | 0.91 | 0.79 |
| 500 | 0.001 | 8 | 0.2 | 2,763,842.08 | 950 | 29,050 | 0.85 | 0.84 |
| 500 | 0.001 | 12 | 0.2 | 2,778,157.84 | 7,071 | 22,929 | 0.90 | 0.79 |
| 100 | 0.01 | 6 | 0.2 | 2,781,904.32 | 959 | 29,041 | 0.84 | 0.85 |
| 150 | 0.01 | 6 | 0.2 | 2,786,986.58 | 1,444 | 28,556 | 0.84 | 0.85 |
| 350 | 0.001 | 8 | 0.2 | 2,814,000.38 | 30 | 29,970 | 0.84 | 0.84 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 250 | 0.001 | 4 | 0.2 | 2,817,421.84 | 0 | 30,000 | 0.82 | 0.83 |
| 50 | 0.01 | 4 | 0.2 | 2,817,421.84 | 0 | 30,000 | 0.83 | 0.84 |
| 250 | 0.01 | 4 | 0.2 | 2,817,421.84 | 0 | 30,000 | 0.84 | 0.84 |
| 150 | 0.001 | 16 | 0.2 | 2,817,421.84 | 0 | 30,000 | 0.95 | 0.71 |
| 250 | 0.001 | 6 | 0.2 | 2,817,421.84 | 0 | 30,000 | 0.83 | 0.85 |
| 350 | 0.001 | 4 | 0.2 | 2,817,421.84 | 0 | 30,000 | 0.83 | 0.83 |
| 500 | 0.001 | 6 | 0.2 | 2,817,421.84 | 0 | 30,000 | 0.84 | 0.85 |
| 500 | 0.001 | 4 | 0.2 | 2,817,421.84 | 0 | 30,000 | 0.83 | 0.84 |
| 500 | 0.01 | 4 | 0.2 | 2,817,421.84 | 0 | 30,000 | 0.84 | 0.84 |
| 350 | 0.001 | 6 | 0.2 | 2,817,421.84 | 0 | 30,000 | 0.83 | 0.85 |
| 350 | 0.01 | 4 | 0.2 | 2,817,421.84 | 0 | 30,000 | 0.84 | 0.84 |
| 250 | 0.001 | 16 | 0.2 | 2,817,421.84 | 0 | 30,000 | 0.96 | 0.71 |
| 250 | 0.001 | 12 | 0.2 | 2,817,421.84 | 1 | 29,999 | 0.89 | 0.78 |
| 250 | 0.001 | 8 | 0.2 | 2,817,421.84 | 0 | 30,000 | 0.84 | 0.84 |
| 150 | 0.001 | 12 | 0.2 | 2,817,421.84 | 0 | 30,000 | 0.88 | 0.77 |
| 100 | 0.001 | 8 | 0.2 | 2,817,421.84 | 0 | 30,000 | 0.83 | 0.82 |
| 100 | 0.001 | 6 | 0.2 | 2,817,421.84 | 0 | 30,000 | 0.83 | 0.84 |
| 100 | 0.001 | 4 | 0.2 | 2,817,421.84 | 0 | 30,000 | 0.82 | 0.83 |
| 50 | 0.001 | 4 | 0.2 | 2,817,421.84 | 0 | 30,000 | 0.81 | 0.82 |
| 100 | 0.01 | 4 | 0.2 | 2,817,421.84 | 0 | 30,000 | 0.83 | 0.84 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 50 | 0.001 | 16 | 0.2 | 2,817,421.84 | 0 | 30,000 | 0.93 | 0.70 |

## 8.4.A. ANEXO 4A - RESULTADOS DE CORRIDAS DEL PRIMER MODELO CON DATOS MISSINGS (RANDOM)

| estimators | learning_rate | max_depth | threshold | count_zero | count_one | CostoPrimer Experimento | train_auc | test_auc |
|---|---|---|---|---|---|---|---|---|
| 250 | 0.01 | 8 | 0.3 | 4,478 | 22,522 | 1,837,304.94 | 0.81 | 0.77 |
| 300 | 0.01 | 8 | 0.3 | 4,561 | 22,439 | 1,844,118.94 | 0.81 | 0.77 |
| 350 | 0.01 | 8 | 0.3 | 4,620 | 22,380 | 1,850,426.90 | 0.81 | 0.77 |
| 200 | 0.01 | 8 | 0.3 | 4,335 | 22,665 | 1,874,877.92 | 0.81 | 0.77 |
| 350 | 0.01 | 8 | 0.25 | 3,553 | 23,447 | 1,885,547.20 | 0.81 | 0.77 |
| 300 | 0.01 | 8 | 0.25 | 3,483 | 23,517 | 1,895,388.70 | 0.81 | 0.77 |
| 150 | 0.01 | 8 | 0.3 | 3,914 | 23,086 | 1,903,119.80 | 0.81 | 0.77 |
| 250 | 0.01 | 8 | 0.25 | 3,395 | 23,605 | 1,905,395.66 | 0.81 | 0.77 |
| 200 | 0.01 | 8 | 0.25 | 3,225 | 23,775 | 1,923,622.52 | 0.81 | 0.77 |
| 150 | 0.01 | 8 | 0.25 | 2,782 | 24,218 | 1,930,222.62 | 0.81 | 0.77 |
| 350 | 0.01 | 8 | 0.2 | 2,426 | 24,574 | 1,963,834.54 | 0.81 | 0.77 |
| 200 | 0.01 | 8 | 0.2 | 2,192 | 24,808 | 1,965,930.24 | 0.81 | 0.77 |
| 300 | 0.01 | 8 | 0.2 | 2,387 | 24,613 | 1,967,930.46 | 0.81 | 0.77 |
| 150 | 0.01 | 8 | 0.2 | 1,883 | 25,117 | 1,969,287.96 | 0.81 | 0.77 |
| 250 | 0.01 | 8 | 0.2 | 2,322 | 24,678 | 1,975,760.66 | 0.81 | 0.77 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 250 | 0.01 | 4 | 0.3 | 0 | 27,000 | 2,010,144.70 | 0.79 | 0.78 |
| 300 | 0.01 | 4 | 0.2 | 0 | 27,000 | 2,010,144.70 | 0.79 | 0.78 |
| 300 | 0.01 | 4 | 0.25 | 0 | 27,000 | 2,010,144.70 | 0.79 | 0.78 |
| 250 | 0.01 | 4 | 0.25 | 0 | 27,000 | 2,010,144.70 | 0.79 | 0.78 |
| 350 | 0.01 | 4 | 0.2 | 0 | 27,000 | 2,010,144.70 | 0.79 | 0.78 |
| 350 | 0.01 | 4 | 0.25 | 0 | 27,000 | 2,010,144.70 | 0.79 | 0.78 |
| 350 | 0.01 | 4 | 0.3 | 0 | 27,000 | 2,010,144.70 | 0.79 | 0.78 |
| 300 | 0.01 | 4 | 0.3 | 0 | 27,000 | 2,010,144.70 | 0.79 | 0.78 |
| 250 | 0.01 | 4 | 0.2 | 0 | 27,000 | 2,010,144.70 | 0.79 | 0.78 |
| 150 | 0.01 | 4 | 0.2 | 0 | 27,000 | 2,010,144.70 | 0.79 | 0.78 |
| 200 | 0.01 | 4 | 0.3 | 0 | 27,000 | 2,010,144.70 | 0.79 | 0.78 |
| 150 | 0.01 | 4 | 0.3 | 0 | 27,000 | 2,010,144.70 | 0.79 | 0.78 |
| 150 | 0.01 | 4 | 0.25 | 0 | 27,000 | 2,010,144.70 | 0.79 | 0.78 |
| 200 | 0.01 | 4 | 0.2 | 0 | 27,000 | 2,010,144.70 | 0.79 | 0.78 |
| 200 | 0.01 | 4 | 0.25 | 0 | 27,000 | 2,010,144.70 | 0.79 | 0.78 |
| 150 | 0.01 | 10 | 0.2 | 4,742 | 22,258 | 2,187,946.46 | 0.84 | 0.75 |
| 200 | 0.01 | 10 | 0.2 | 5,198 | 21,802 | 2,211,882.62 | 0.85 | 0.74 |
| 250 | 0.01 | 10 | 0.2 | 5,444 | 21,556 | 2,223,788.54 | 0.85 | 0.74 |
| 150 | 0.01 | 10 | 0.25 | 6,019 | 20,981 | 2,359,154.14 | 0.84 | 0.75 |
| 300 | 0.01 | 10 | 0.2 | 5,640 | 21,360 | 2,361,963.42 | 0.85 | 0.74 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 350 | 0.01 | 10 | 0.2 | 5,806 | 21,194 | 2,369,783.48 | 0.85 | 0.74 |
| 150 | 0.01 | 10 | 0.3 | 7,465 | 19,535 | 2,477,358.94 | 0.84 | 0.75 |
| 200 | 0.01 | 10 | 0.25 | 6,499 | 20,501 | 2,515,717.32 | 0.85 | 0.74 |
| 350 | 0.01 | 10 | 0.25 | 7,099 | 19,901 | 2,530,862.30 | 0.85 | 0.74 |
| 250 | 0.01 | 10 | 0.25 | 6,749 | 20,251 | 2,531,146.46 | 0.85 | 0.74 |
| 200 | 0.01 | 10 | 0.3 | 7,966 | 19,034 | 2,535,020.28 | 0.85 | 0.74 |
| 300 | 0.01 | 10 | 0.25 | 6,953 | 20,047 | 2,540,740.26 | 0.85 | 0.74 |
| 300 | 0.01 | 10 | 0.3 | 8,354 | 18,646 | 2,561,207.20 | 0.85 | 0.74 |
| 250 | 0.01 | 10 | 0.3 | 8,180 | 18,820 | 2,566,162.96 | 0.85 | 0.74 |
| 350 | 0.01 | 10 | 0.3 | 8,488 | 18,512 | 2,613,595.74 | 0.85 | 0.74 |

## 8.4.B. ANEXO 4B - RESULTADOS DE CORRIDAS DEL PRIMER MODELO CON DATOS MISSINGS (FEATURE IMPORTANCE)

La siguiente tabla resulta del entrenamiento del primer modelo pero con datos afectados con un 20% de missings en los 3 features más importantes según ELI5 expresados en la Tabla 24.

| estimators | learning_rate | max_depth | threshold | CostoPrimerExperimento | count_zero | count_one | train_auc | test_auc |
|---|---|---|---|---|---|---|---|---|
| 100 | 0.01 | 8 | 0.2 | 2,817,421.84 | 1 | 29,999 | 0.84 | 0.67 |
| 250 | 0.01 | 8 | 0.25 | 2,827,097.16 | 7 | 29,993 | 0.84 | 0.67 |
| 350 | 0.01 | 8 | 0.2 | 2,827,097.16 | 7 | 29,993 | 0.84 | 0.69 |
| 350 | 0.01 | 8 | 0.25 | 2,827,097.16 | 7 | 29,993 | 0.84 | 0.69 |
| 150 | 0.01 | 8 | 0.3 | 2,827,097.16 | 7 | 29,993 | 0.84 | 0.68 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 150 | 0.01 | 8 | 0.25 | 2,827,097.16 | 7 | 29,993 | 0.84 | 0.68 |
| 150 | 0.01 | 8 | 0.2 | 2,827,097.16 | 7 | 29,993 | 0.84 | 0.68 |
| 500 | 0.01 | 8 | 0.2 | 2,827,097.16 | 7 | 29,993 | 0.84 | 0.69 |
| 250 | 0.01 | 8 | 0.2 | 2,827,097.16 | 7 | 29,993 | 0.84 | 0.67 |
| 100 | 0.01 | 8 | 0.3 | 2,827,097.16 | 7 | 29,993 | 0.84 | 0.67 |
| 100 | 0.01 | 8 | 0.25 | 2,827,097.16 | 7 | 29,993 | 0.84 | 0.67 |
| 100 | 0.01 | 8 | 0.35 | 2,832,703.32 | 11 | 29,989 | 0.84 | 0.67 |
| 250 | 0.01 | 8 | 0.3 | 2,832,703.32 | 11 | 29,989 | 0.84 | 0.67 |
| 150 | 0.01 | 8 | 0.35 | 2,883,987.34 | 158 | 29,842 | 0.84 | 0.68 |
| 500 | 0.01 | 8 | 0.25 | 2,883,987.34 | 158 | 29,842 | 0.84 | 0.69 |
| 350 | 0.01 | 8 | 0.3 | 2,889,593.50 | 162 | 29,838 | 0.84 | 0.69 |
| 500 | 0.01 | 8 | 0.3 | 2,889,593.50 | 162 | 29,838 | 0.84 | 0.69 |
| 500 | 0.01 | 8 | 0.35 | 2,889,593.50 | 162 | 29,838 | 0.84 | 0.69 |
| 250 | 0.01 | 8 | 0.35 | 2,889,593.50 | 162 | 29,838 | 0.84 | 0.67 |
| 350 | 0.01 | 8 | 0.35 | 2,889,593.50 | 162 | 29,838 | 0.84 | 0.69 |
| 50 | 0.01 | 12 | 0.25 | 3,307,035.34 | 161 | 29,839 | 0.89 | 0.62 |
| 50 | 0.01 | 12 | 0.2 | 3,309,453.18 | 120 | 29,880 | 0.89 | 0.62 |
| 50 | 0.01 | 12 | 0.3 | 4,158,691.16 | 856 | 29,144 | 0.89 | 0.62 |
| 50 | 0.01 | 12 | 0.35 | 4,947,283.66 | 3,873 | 26,127 | 0.89 | 0.62 |
| 100 | 0.01 | 12 | 0.2 | 5,250,468.04 | 4,088 | 25,912 | 0.91 | 0.57 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 100 | 0.01 | 12 | 0.25 | 5,627,938.24 | 4,328 | 25,672 | 0.91 | 0.57 |
| 100 | 0.01 | 12 | 0.3 | 6,910,150.20 | 12,692 | 17,308 | 0.91 | 0.57 |
| 150 | 0.01 | 12 | 0.2 | 7,032,482.98 | 13,070 | 16,930 | 0.92 | 0.59 |
| 150 | 0.01 | 12 | 0.25 | 8,127,291.38 | 14,184 | 15,816 | 0.92 | 0.59 |
| 100 | 0.01 | 12 | 0.35 | 8,163,008.58 | 13,839 | 16,161 | 0.91 | 0.57 |
| 250 | 0.01 | 12 | 0.2 | 8,442,822.04 | 14,431 | 15,569 | 0.92 | 0.59 |
| 150 | 0.01 | 12 | 0.3 | 8,729,958.22 | 14,412 | 15,588 | 0.92 | 0.59 |
| 150 | 0.01 | 12 | 0.35 | 8,782,538.00 | 14,443 | 15,557 | 0.92 | 0.59 |
| 350 | 0.01 | 12 | 0.2 | 9,494,481.14 | 15,869 | 14,131 | 0.92 | 0.59 |
| 250 | 0.01 | 12 | 0.25 | 9,502,193.24 | 15,873 | 14,127 | 0.92 | 0.59 |
| 250 | 0.01 | 12 | 0.3 | 10,026,080.36 | 16,129 | 13,871 | 0.92 | 0.59 |
| 350 | 0.01 | 12 | 0.25 | 10,231,795.16 | 16,266 | 13,734 | 0.92 | 0.59 |
| 500 | 0.01 | 12 | 0.2 | 10,244,746.92 | 16,133 | 13,867 | 0.93 | 0.59 |
| 350 | 0.01 | 12 | 0.3 | 10,437,244.22 | 16,309 | 13,691 | 0.92 | 0.59 |
| 250 | 0.01 | 12 | 0.35 | 10,499,145.22 | 16,324 | 13,676 | 0.92 | 0.59 |
| 500 | 0.01 | 12 | 0.25 | 10,652,372.92 | 16,350 | 13,650 | 0.93 | 0.59 |
| 350 | 0.01 | 12 | 0.35 | 10,681,552.08 | 16,351 | 13,649 | 0.92 | 0.59 |
| 500 | 0.01 | 12 | 0.3 | 10,711,252.82 | 16,364 | 13,636 | 0.93 | 0.59 |
| 500 | 0.01 | 12 | 0.35 | 10,757,699.48 | 16,377 | 13,623 | 0.93 | 0.59 |
| 100 | 0.01 | 8 | 0.2 | 2,817,421.84 | 1 | 29,999 | 0.84 | 0.67 |

## 8.5. ANEXO 5 - RESULTADOS DE CORRIDAS DEL SEGUNDO EXPERIMENTO PARA THRESHOLD = 0.35

La siguiente tabla resulta de filtrar las corridas para encontrar el mejor modelo **NGBoost** para el segundo experimento para un threshold de 0.35, ordenandola además por **CostoSegundoExperimento**.

| alpha | estimators | learning_rate | max_depth | count_zero | count_one | CostoPrimerExperimento | learning_cost | train_auc | test_auc | CostoSegundoExperimento | %_Inversion |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 156 | 150 | 0.01 | 8 | 9,849 | 20,151 | 2,622,773.06 | 234,378.18 | 0.84 | 0.84 | 2,857,151.24 | 8.94 |
| 159 | 150 | 0.01 | 8 | 9,849 | 20,151 | 2,622,773.06 | 238,885.45 | 0.84 | 0.84 | 2,861,658.51 | 9.11 |
| 173 | 150 | 0.01 | 8 | 9,849 | 20,151 | 2,622,773.06 | 259,919.39 | 0.84 | 0.84 | 2,882,692.45 | 9.91 |
| 174 | 150 | 0.01 | 8 | 9,849 | 20,151 | 2,622,773.06 | 261,421.81 | 0.84 | 0.84 | 2,884,194.87 | 9.97 |
| 181 | 150 | 0.01 | 8 | 9,849 | 20,151 | 2,622,773.06 | 271,938.78 | 0.84 | 0.84 | 2,894,711.84 | 10.37 |
| 184 | 150 | 0.01 | 8 | 9,849 | 20,151 | 2,622,773.06 | 276,446.06 | 0.84 | 0.84 | 2,899,219.12 | 10.54 |
| 190 | 150 | 0.01 | 8 | 9,849 | 20,151 | 2,622,773.06 | 285,460.60 | 0.84 | 0.84 | 2,908,233.66 | 10.88 |
| 194 | 150 | 0.01 | 8 | 9,849 | 20,151 | 2,622,773.06 | 291,470.30 | 0.84 | 0.84 | 2,914,243.36 | 11.11 |
| 153 | 200 | 0.01 | 8 | 10,508 | 19,492 | 2,691,299.64 | 232,512.21 | 0.84 | 0.84 | 2,923,811.85 | 8.64 |
| 156 | 200 | 0.01 | 8 | 10,508 | 19,492 | 2,691,299.64 | 237,071.28 | 0.84 | 0.84 | 2,928,370.92 | 8.81 |
| 212 | 150 | 0.01 | 8 | 9,849 | 20,151 | 2,622,773.06 | 318,513.93 | 0.84 | 0.84 | 2,941,286.99 | 12.14 |
| 217 | 150 | 0.01 | 8 | 9,849 | 20,151 | 2,622,773.06 | 326,026.06 | 0.84 | 0.84 | 2,948,799.12 | 12.43 |
| 158 | 350 | 0.01 | 8 | 11,222 | 18,778 | 2,708,409.14 | 244,892.70 | 0.84 | 0.84 | 2,953,301.84 | 9.04 |
| 173 | 350 | 0.01 | 8 | 11,222 | 18,778 | 2,708,409.14 | 268,142.01 | 0.84 | 0.84 | 2,976,551.15 | 9.90 |
| 165 | 300 | 0.01 | 8 | 11,112 | 18,888 | 2,722,371.86 | 255,956.89 | 0.84 | 0.84 | 2,978,328.75 | 9.40 |
| 238 | 150 | 0.01 | 8 | 9,849 | 20,151 | 2,622,773.06 | 357,576.96 | 0.84 | 0.84 | 2,980,350.02 | 13.63 |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 167 | 300 | 0.01 | 8 | 11,112 | 18,888 | 2,722,371.86 | 259,059.39 | 0.84 | 0.84 | 2,981,431.25 | 9.52 |
| 173 | 300 | 0.01 | 8 | 11,112 | 18,888 | 2,722,371.86 | 268,366.92 | 0.84 | 0.84 | 2,990,738.78 | 9.86 |
| 177 | 300 | 0.01 | 8 | 11,112 | 18,888 | 2,722,371.86 | 274,571.93 | 0.84 | 0.84 | 2,996,943.79 | 10.09 |
| 202 | 200 | 0.01 | 8 | 10,508 | 19,492 | 2,691,299.64 | 306,976.91 | 0.84 | 0.84 | 2,998,276.55 | 11.41 |

## 8.6. ANEXO 6 - RESULTADOS DE CORRIDAS DEL SEGUNDO EXPERIMENTO PARA THRESHOLD = 0.4

La siguiente tabla resulta de filtrar las corridas para encontrar el mejor modelo **NGBoost** para el segundo experimento para un threshold de 0.4, ordenandola además por **CostoSegundoExperimento**.

| alpha | estimators | learning_rate | max_depth | count_zero | count_one | CostoPrimerExperimento | learning_cost | train_auc | test_auc | CostoSegundoExperimento | %_Inversion |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 153 | 150 | 0.01 | 8 | 11,207 | 18,793 | 2,825,111.32 | 278,510.00 | 0.84 | 0.84 | 3,103,621.32 | 9.86 |
| 154 | 150 | 0.01 | 8 | 11,207 | 18,793 | 2,825,111.32 | 280,330.33 | 0.84 | 0.84 | 3,105,441.65 | 9.92 |
| 155 | 150 | 0.01 | 8 | 11,207 | 18,793 | 2,825,111.32 | 282,150.66 | 0.84 | 0.84 | 3,107,261.98 | 9.99 |
| 153 | 200 | 0.01 | 8 | 11,848 | 18,152 | 2,883,965.38 | 280,505.24 | 0.84 | 0.84 | 3,164,470.62 | 9.73 |
| 160 | 200 | 0.01 | 8 | 11,848 | 18,152 | 2,883,965.38 | 293,338.81 | 0.84 | 0.84 | 3,177,304.19 | 10.17 |
| 199 | 150 | 0.01 | 8 | 11,207 | 18,793 | 2,825,111.32 | 362,245.04 | 0.84 | 0.84 | 3,187,356.36 | 12.82 |
| 177 | 200 | 0.01 | 8 | 11,848 | 18,152 | 2,883,965.38 | 324,506.06 | 0.84 | 0.84 | 3,208,471.44 | 11.25 |
| 213 | 150 | 0.01 | 8 | 11,207 | 18,793 | 2,825,111.32 | 387,729.61 | 0.84 | 0.84 | 3,212,840.93 | 13.72 |
| 155 | 350 | 0.01 | 8 | 12,501 | 17,499 | 2,928,995.46 | 286,659.26 | 0.84 | 0.84 | 3,215,654.72 | 9.79 |
| 215 | 150 | 0.01 | 8 | 11,207 | 18,793 | 2,825,111.32 | 391,370.27 | 0.84 | 0.84 | 3,216,481.59 | 13.85 |
| 152 | 250 | 0.01 | 8 | 12,218 | 17,782 | 2,941,618.36 | 280,278.56 | 0.84 | 0.84 | 3,221,896.92 | 9.53 |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 226 | 150 | 0.01 | 8 | 11,207 | 18,793 | 2,825,111.32 | 411,393.86 | 0.84 | 0.84 | 3,236,505.18 | 14.56 |
| 160 | 250 | 0.01 | 8 | 12,218 | 17,782 | 2,941,618.36 | 295,030.06 | 0.84 | 0.84 | 3,236,648.42 | 10.03 |
| 160 | 250 | 0.01 | 8 | 12,218 | 17,782 | 2,941,618.36 | 295,030.06 | 0.84 | 0.84 | 3,236,648.42 | 10.03 |
| 160 | 300 | 0.01 | 8 | 12,378 | 17,622 | 2,943,629.58 | 295,613.30 | 0.84 | 0.84 | 3,239,242.88 | 10.04 |
| 169 | 250 | 0.01 | 8 | 12,218 | 17,782 | 2,941,618.36 | 311,625.50 | 0.84 | 0.84 | 3,253,243.86 | 10.59 |
| 236 | 150 | 0.01 | 8 | 11,207 | 18,793 | 2,825,111.32 | 429,597.13 | 0.84 | 0.84 | 3,254,708.45 | 15.21 |
| 236 | 150 | 0.01 | 8 | 11,207 | 18,793 | 2,825,111.32 | 429,597.13 | 0.84 | 0.84 | 3,254,708.45 | 15.21 |
| 205 | 200 | 0.01 | 8 | 11,848 | 18,152 | 2,883,965.38 | 375,840.35 | 0.84 | 0.84 | 3,259,805.73 | 13.03 |
| 207 | 200 | 0.01 | 8 | 11,848 | 18,152 | 2,883,965.38 | 379,507.09 | 0.84 | 0.84 | 3,263,472.47 | 13.16 |

## 8.7. ANEXO 7 - RESULTADOS DE CORRIDAS DEL SEGUNDO EXPERIMENTO PARA THRESHOLD = 0.45

La siguiente tabla resulta de filtrar las corridas para encontrar el mejor modelo **NGBoost** para el segundo experimento para un threshold de 0.45, ordenandola además por **CostoSegundoExperimento**.

| alpha | estimators | learning_rate | max_depth | count_zero | count_one | CostoPrimerExperimento | learning_cost | train_auc | test_auc | CostoSegundoExperimento | %_Inversion |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 150 | 150 | 0.01 | 8 | 12,602 | 17,398 | 2,908,261.74 | 324,132.88 | 0.84 | 0.84 | 3,232,394.62 | 11.15 |
| 168 | 150 | 0.01 | 8 | 12,602 | 17,398 | 2,908,261.74 | 363,028.83 | 0.84 | 0.84 | 3,271,290.57 | 12.48 |
| 150 | 200 | 0.01 | 8 | 13,199 | 16,801 | 2,966,242.78 | 324,474.77 | 0.84 | 0.84 | 3,290,717.55 | 10.94 |
| 153 | 200 | 0.01 | 8 | 13,199 | 16,801 | 2,966,242.78 | 330,964.27 | 0.84 | 0.84 | 3,297,207.05 | 11.16 |
| 168 | 200 | 0.01 | 8 | 13,199 | 16,801 | 2,966,242.78 | 363,411.74 | 0.84 | 0.84 | 3,329,654.52 | 12.25 |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 169 | 200 | 0.01 | 8 | 13,199 | 16,801 | 2,966,242.78 | 365,574.91 | 0.84 | 0.84 | 3,331,817.69 | 12.32 |
| 156 | 300 | 0.01 | 8 | 13,686 | 16,314 | 3,004,756.48 | 338,025.09 | 0.84 | 0.84 | 3,342,781.57 | 11.25 |
| 150 | 350 | 0.01 | 8 | 13,802 | 16,198 | 3,018,070.36 | 325,050.91 | 0.84 | 0.84 | 3,343,121.27 | 10.77 |
| 163 | 300 | 0.01 | 8 | 13,686 | 16,314 | 3,004,756.48 | 353,192.88 | 0.84 | 0.84 | 3,357,949.36 | 11.75 |
| 210 | 150 | 0.01 | 8 | 12,602 | 17,398 | 2,908,261.74 | 453,786.04 | 0.84 | 0.84 | 3,362,047.78 | 15.60 |
| 172 | 300 | 0.01 | 8 | 13,686 | 16,314 | 3,004,756.48 | 372,694.33 | 0.84 | 0.84 | 3,377,450.81 | 12.40 |
| 168 | 250 | 0.01 | 8 | 13,534 | 16,466 | 3,016,174.94 | 363,738.75 | 0.84 | 0.84 | 3,379,913.69 | 12.06 |
| 192 | 200 | 0.01 | 8 | 13,199 | 16,801 | 2,966,242.78 | 415,327.71 | 0.84 | 0.84 | 3,381,570.49 | 14.00 |
| 178 | 300 | 0.01 | 8 | 13,686 | 16,314 | 3,004,756.48 | 385,695.30 | 0.84 | 0.84 | 3,390,451.78 | 12.84 |
| 175 | 350 | 0.01 | 8 | 13,802 | 16,198 | 3,018,070.36 | 379,226.06 | 0.84 | 0.84 | 3,397,296.42 | 12.57 |
| 228 | 150 | 0.01 | 8 | 12,602 | 17,398 | 2,908,261.74 | 492,681.98 | 0.84 | 0.84 | 3,400,943.72 | 16.94 |
| 201 | 200 | 0.01 | 8 | 13,199 | 16,801 | 2,966,242.78 | 434,796.19 | 0.84 | 0.84 | 3,401,038.97 | 14.66 |
| 187 | 300 | 0.01 | 8 | 13,686 | 16,314 | 3,004,756.48 | 405,196.74 | 0.84 | 0.84 | 3,409,953.22 | 13.49 |
| 237 | 150 | 0.01 | 8 | 12,602 | 17,398 | 2,908,261.74 | 512,129.96 | 0.84 | 0.84 | 3,420,391.70 | 17.61 |
| 243 | 150 | 0.01 | 8 | 12,602 | 17,398 | 2,908,261.74 | 525,095.27 | 0.84 | 0.84 | 3,433,357.01 | 18.06 |

## 8.8. ANEXO 8 - RESULTADOS DE CORRIDAS DEL SEGUNDO EXPERIMENTO

Debido a la complejidad de exploración del hiper parámetro alpha y de la intención de ilustrar los valores de los hiperparametros de los diferentes modelos, thresholds, préstamos asignados, costos y porcentajes de inversión. Hemos decidido poder representar el top 50 de 2,000 resultados satisfactorios, es decir, resultados que no dieron como resultados **CostoSegundoExperimento** negativos, thresholds y cantidad de préstamos asignados inconsistentes (que no condicen con lo propuesto o que son irreales, dada la exploración de

hiper parámetros), etc. Todos estos resultados no deseados se deben a la exploración que hacemos sobre los datos, a favor de poder encontrar aquellos que satisfagan el objetivo del presente trabajo. Es válido recordar que todas estas corridas se hicieron con un learning_rate fijo de 0.01.

| alpha | estimators | max_depth | threshold | count_zero | count_one | CostoPrimerExperimento | learning_cost | CostoSegundoExperimento | diff_auc | %_Inversion |
|---|---|---|---|---|---|---|---|---|---|---|
| 156 | 150 | 8 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 234,378.18 | 2,857,151.24 | 0.001458 | 8.94 |
| 159 | 150 | 8 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 238,885.45 | 2,861,658.51 | 0.001458 | 9.11 |
| 173 | 150 | 8 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 259,919.39 | 2,882,692.45 | 0.001458 | 9.91 |
| 174 | 150 | 8 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 261,421.81 | 2,884,194.87 | 0.001458 | 9.97 |
| 181 | 150 | 8 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 271,938.78 | 2,894,711.84 | 0.001458 | 10.37 |
| 184 | 150 | 8 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 276,446.06 | 2,899,219.12 | 0.001458 | 10.54 |
| 190 | 150 | 8 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 285,460.60 | 2,908,233.66 | 0.001458 | 10.88 |
| 194 | 150 | 8 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 291,470.30 | 2,914,243.36 | 0.001458 | 11.11 |
| 153 | 200 | 8 | 0.35 | 10,508 | 19,492 | 2,691,299.64 | 232,512.21 | 2,923,811.85 | 0.002616 | 8.64 |
| 156 | 200 | 8 | 0.35 | 10,508 | 19,492 | 2,691,299.64 | 237,071.28 | 2,928,370.92 | 0.002616 | 8.81 |
| 212 | 150 | 8 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 318,513.93 | 2,941,286.99 | 0.001458 | 12.14 |
| 217 | 150 | 8 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 326,026.06 | 2,948,799.12 | 0.001458 | 12.43 |
| 158 | 350 | 8 | 0.35 | 11,222 | 18,778 | 2,708,409.14 | 244,892.70 | 2,953,301.84 | 0.004705 | 9.04 |
| 173 | 350 | 8 | 0.35 | 11,222 | 18,778 | 2,708,409.14 | 268,142.01 | 2,976,551.15 | 0.004705 | 9.90 |
| 165 | 300 | 8 | 0.35 | 11,112 | 18,888 | 2,722,371.86 | 255,956.89 | 2,978,328.75 | 0.004100 | 9.40 |
| 238 | 150 | 8 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 357,576.96 | 2,980,350.02 | 0.001458 | 13.63 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 167 | 300 | 8 | 0.35 | 11,112 | 18,888 | 2,722,371.86 | 259,059.39 | 2,981,431.25 | 0.004100 | 9.52 |
| 173 | 300 | 8 | 0.35 | 11,112 | 18,888 | 2,722,371.86 | 268,366.92 | 2,990,738.78 | 0.004100 | 9.86 |
| 177 | 300 | 8 | 0.35 | 11,112 | 18,888 | 2,722,371.86 | 274,571.93 | 2,996,943.79 | 0.004100 | 10.09 |
| 202 | 200 | 8 | 0.35 | 10,508 | 19,492 | 2,691,299.64 | 306,976.91 | 2,998,276.55 | 0.002616 | 11.41 |
| 179 | 300 | 8 | 0.35 | 11,112 | 18,888 | 2,722,371.86 | 277,674.44 | 3,000,046.30 | 0.004100 | 10.20 |
| 179 | 300 | 8 | 0.35 | 11,112 | 18,888 | 2,722,371.86 | 277,674.44 | 3,000,046.30 | 0.004100 | 10.20 |
| 208 | 200 | 8 | 0.35 | 10,508 | 19,492 | 2,691,299.64 | 316,095.04 | 3,007,394.68 | 0.002616 | 11.75 |
| 186 | 300 | 8 | 0.35 | 11,112 | 18,888 | 2,722,371.86 | 288,533.22 | 3,010,905.08 | 0.004100 | 10.60 |
| 218 | 350 | 8 | 0.35 | 11,222 | 18,778 | 2,708,409.14 | 337,889.93 | 3,046,299.07 | 0.004705 | 12.48 |
| 282 | 150 | 8 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 423,683.63 | 3,046,456.69 | 0.001458 | 16.15 |
| 188 | 250 | 8 | 0.35 | 10,903 | 19,097 | 2,760,031.78 | 288,806.06 | 3,048,837.84 | 0.003157 | 10.46 |
| 211 | 300 | 8 | 0.35 | 11,112 | 18,888 | 2,722,371.86 | 327,314.56 | 3,049,686.42 | 0.004100 | 12.02 |
| 190 | 250 | 8 | 0.35 | 10,903 | 19,097 | 2,760,031.78 | 291,878.47 | 3,051,910.25 | 0.003157 | 10.58 |
| 288 | 150 | 8 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 432,698.17 | 3,055,471.23 | 0.001458 | 16.50 |
| 217 | 300 | 8 | 0.35 | 11,112 | 18,888 | 2,722,371.86 | 336,622.09 | 3,058,993.95 | 0.004100 | 12.37 |
| 245 | 200 | 8 | 0.35 | 10,508 | 19,492 | 2,691,299.64 | 372,323.48 | 3,063,623.12 | 0.002616 | 13.83 |
| 249 | 200 | 8 | 0.35 | 10,508 | 19,492 | 2,691,299.64 | 378,402.23 | 3,069,701.87 | 0.002616 | 14.06 |
| 252 | 200 | 8 | 0.35 | 10,508 | 19,492 | 2,691,299.64 | 382,961.29 | 3,074,260.93 | 0.002616 | 14.23 |
| 229 | 300 | 8 | 0.35 | 11,112 | 18,888 | 2,722,371.86 | 355,237.13 | 3,077,608.99 | 0.004100 | 13.05 |
| 208 | 250 | 8 | 0.35 | 10,903 | 19,097 | 2,760,031.78 | 319,530.11 | 3,079,561.89 | 0.003157 | 11.58 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 256 | 200 | 8 | 0.35 | 10,508 | 19,492 | 2,691,299.64 | 389,040.04 | 3,080,339.68 | 0.002616 | 14.46 |
| 260 | 200 | 8 | 0.35 | 10,508 | 19,492 | 2,691,299.64 | 395,118.80 | 3,086,418.44 | 0.002616 | 14.68 |
| 315 | 150 | 8 | 0.35 | 9,849 | 20,151 | 2,622,773.06 | 473,263.63 | 3,096,036.69 | 0.001458 | 18.04 |
| 153 | 150 | 8 | 0.4 | 11,207 | 18,793 | 2,825,111.32 | 278,510.00 | 3,103,621.32 | 0.001458 | 9.86 |
| 154 | 150 | 8 | 0.4 | 11,207 | 18,793 | 2,825,111.32 | 280,330.33 | 3,105,441.65 | 0.001458 | 9.92 |
| 155 | 150 | 8 | 0.4 | 11,207 | 18,793 | 2,825,111.32 | 282,150.66 | 3,107,261.98 | 0.001458 | 9.99 |
| 255 | 300 | 8 | 0.35 | 11,112 | 18,888 | 2,722,371.86 | 395,569.73 | 3,117,941.59 | 0.004100 | 14.53 |
| 266 | 350 | 8 | 0.35 | 11,222 | 18,778 | 2,708,409.14 | 412,287.71 | 3,120,696.85 | 0.004705 | 15.22 |
| 247 | 250 | 8 | 0.35 | 10,903 | 19,097 | 2,760,031.78 | 379,442.01 | 3,139,473.79 | 0.003157 | 13.75 |
| 250 | 250 | 8 | 0.35 | 10,903 | 19,097 | 2,760,031.78 | 384,050.61 | 3,144,082.39 | 0.003157 | 13.91 |
| 275 | 300 | 8 | 0.35 | 11,112 | 18,888 | 2,722,371.86 | 426,594.81 | 3,148,966.67 | 0.004100 | 15.67 |
| 277 | 300 | 8 | 0.35 | 11,112 | 18,888 | 2,722,371.86 | 429,697.32 | 3,152,069.18 | 0.004100 | 15.78 |
| 259 | 250 | 8 | 0.35 | 10,903 | 19,097 | 2,760,031.78 | 397,876.44 | 3,157,908.22 | 0.003157 | 14.42 |
| 293 | 350 | 8 | 0.35 | 11,222 | 18,778 | 2,708,409.14 | 454,136.47 | 3,162,545.61 | 0.004705 | 16.77 |