Universidad de San Andrés

Departamento de Matemática y Ciencias

Maestría en Ciencia de Datos

Optimización de Decisiones Crediticias: Un Enfoque de Modelado con Random Forest

Autora: Melina Heliszkowski

Directora: Marcela Svarc

2023

Maestría en Ciencia de Datos

Departamento de Matemática y Ciencias

Optimización de Decisiones Crediticias: Un Enfoque de Modelado con *Random Forest*

Melina Heliszkowski **2023**

Directora: Marcela Svarc

Universidad de SanAndrés

Universidad de
SanAndrés

Resumen

En el ámbito financiero contemporáneo, la toma de decisiones en la concesión de créditos es crucial para garantizar un equilibrio entre el riesgo y la rentabilidad. La creciente disponibilidad de información y las nuevas tecnologías han revolucionado este proceso, permitiendo un análisis más preciso y efectivo.

Este trabajo se enfoca en desarrollar un modelo para determinar el incumplimiento en la concesión de créditos a corto plazo. La metodología se basa en la utilización del algoritmo construido a partir de un modelo de Random Forest, una técnica de aprendizaje automático que ha demostrado ser eficaz en la predicción de resultados crediticios.

Agradecimientos

A mi familia, por su apoyo siempre. Y a Marcela, por su ayuda, dedicación, motivación y apoyo.

Índice general

Resumen **i**

Agradecimientos **ii**

Índice general **iii**

1 Introducción **1**

2 Presentación y tratamiento de los datos **3**
2.1. Descripción general de los datos 3
2.2. Estadística descriptiva 4
2.3. Tratamiento de valores nulos 6
2.4. División de la base de datos 8

3 Metodología estadística implementada **9**
3.1. Aprendizaje Supervisado 9
3.2. Árbol de Decisión 9
3.3. Random Forest 11
3.4. Modelo seleccionado 12

4 Resultados **17**
4.1. Métricas de rendimiento 17
4.2. Resultados monetarios 19
4.3. Resultados por rangos 21
4.4. Importancia de las variables 22

5 Conclusión **25**

Bibliografía **26**

A Detalle de las variables utilizadas **27**

B Código de Python **40**

Universidad de
SanAndrés

CAPÍTULO 1

Introducción

En el mundo financiero contemporáneo, la toma de decisiones en el otorgamiento de créditos reviste una importancia vital para las instituciones y empresas que operan en este ámbito. La evaluación precisa y efectiva de la solvencia crediticia de los solicitantes es esencial para mantener un equilibrio entre el riesgo y la rentabilidad. Sin embargo, en un entorno donde los datos son vastos y complejos, la toma de decisiones basada únicamente en métodos tradicionales se ha vuelto cada vez más insuficiente y susceptible a la subjetividad.

La creciente disponibilidad de datos y los avances en el aprendizaje automático (*machine learning*) han redefinido la forma en que se abordan los desafíos en la toma de decisiones crediticias. Esta nueva era tecnológica ha abierto la puerta al desarrollo de modelos predictivos que aprovechan algoritmos avanzados para evaluar y predecir la capacidad crediticia de los solicitantes. La adopción de estos modelos brinda la posibilidad de mejorar la eficacia de las decisiones, optimizar la asignación de recursos y, en última instancia, impulsar los resultados financieros de las instituciones crediticias.

Tal como menciona Schmitt [6], durante la última década, el mundo ha sido testigo de diversos avances tecnológicos que han tenido un impacto tremendo en las empresas de servicios financieros. La toma de decisiones basada en datos impulsada por la inteligencia artificial y el aprendizaje automático se ha vuelto indispensable en la economía global y ultra competitiva de hoy. Agrega que estos desarrollos y otros similares han generado un cambio en la estructura del mercado para las empresas de préstamos lo que ha llevado a una mayor importancia en la precisión de predicción para las evaluaciones de crédito en lugar de depender únicamente de la protección a posteriori en forma de reducción de pérdida en caso de incumplimiento a través de garantías.

Chopra y Bhilare [2], establecen que la mayoría de los modelos de calificación crediticia deben actualizarse constantemente con nuevas variables y técnicas estadísticas para lograr una mayor precisión. Agregan, que los modelos de calificación crediticia han utilizado la regresión logística y el análisis discriminante lineal para la identificación de posibles incumplidores. Y establecen que las técnicas más nuevas y contemporáneas de aprendizaje automático tienen la capacidad de superar las técnicas clásicas. Estas nuevas técnicas incluyen *Bagging*, *Boosting* y *Random Forest*.

Tripathi et al. [7], afirman que los modelos de calificación crediticia se desarrollan para fortalecer el proceso de toma de decisiones en instituciones financieras, con el objetivo de lidiar con el riesgo asociado a un candidato a

crédito al solicitar un nuevo producto crediticio. Establecen que, el *ensemble learning* es un enfoque sólido para acercarse a un clasificador ideal, fortaleciendo los clasificadores con la agregación de varios modelos para obtener un resultado mejor que el modelo individual. Agregan,que diversos estudios han demostrado que los modelos de *ensemble learning* han logrado un rendimiento de clasificación superior en comparación con los modelos de aprendizaje automático existentes y que la mejora en el rendimiento predictivo resultará en grandes ahorros de ingresos para la institución financiera. Además, con el fin de brindar mayor estabilidad y precisión, el *ensemble learning* produce resultados destacados debido a sus propiedades inherentes para mejorar la eficacia del modelo de calificación crediticia.

En este contexto, la presente tesis se propone abordar el desafío de implementar un modelo de *machine learning* para la toma de decisiones en el otorgamiento de créditos a corto plazo para una entidad financiera que otorga créditos en Brasil, con el objetivo primordial de mejorar los resultados financieros de dicha entidad crediticia. Para lograrlo, se explorarán en detalle diferentes estrategias y enfoques que permitan maximizar tanto la precisión de los resultados como la rentabilidad del proceso crediticio.

En resumen, esta tesis busca abordar una problemática crítica en el ámbito financiero a través de la aplicación de técnicas avanzadas de aprendizaje automático. Con el fin de mejorar los resultados y la eficiencia en la toma de decisiones crediticias, se propone una solución innovadora que combina el conocimiento teórico y la aplicación práctica en un marco integral de estudio.

La estructura de esta tesis se organiza de la siguiente manera. En el Capítulo 2, se abordará la presentación de los datos utilizados para el desarrollo del modelo junto con el tratamiento de los mismos. El Capítulo 3 describe la metodología estadística implementada en la construcción del modelo propuesto, profundizando en el modelo *Random Forest* y en las modificaciones implementadas para que dicho modelo se ajuste al objetivo del presente trabajo. A continuación, en el Capítulo 4, se presentarán y analizarán los resultados obtenidos, detallando cómo el modelo ha impactado en la rentabilidad y eficacia de las decisiones crediticias. Finalmente, el Capítulo 5 estará dedicada a extraer las conclusiones derivadas de este estudio. Los Apéndices presentan los detalles de las variables utilizadas en el modelo y del código de Python.

CAPÍTULO 2

Presentación y tratamiento de los datos

En este capítulo, se presenta una visión general de los datos utilizados en el presente trabajo, junto con un enfoque detallado sobre cómo se separaron las bases de datos en conjuntos de entrenamiento, prueba y fuera de muestra. Dada la riqueza de los datos y al gran número de variables involucradas, se recurre a la estadística descriptiva para proporcionar una descripción exhaustiva de las características claves de los datos.

2.1. Descripción general de los datos

Para la construcción y evaluación del modelo, se dispone de una rica fuente de datos compuesta por 404,244 créditos otorgados en el período mayo 2021 - mayo 2022 (excluyendo el mes de marzo 2022 debido a que se perdió la información completa de ese mes en las bases de la entidad). La información se obtuvo detallada en 215 variables con atributos relacionados a días de atraso en los pagos, patrones de recurrencia en el producto, proporción del ingreso comprometido, historial de ventas, utilización y comportamiento en otros productos ofrecidos por la entidad, *score* de *Bureau* crediticio, *score* de comportamiento en otros productos, entre otros. La tasa de incumplimiento promedio de los créditos es de 12.6 %.

La división temporal se estructuró de la siguiente manera: el período que abarca desde mayo 2021 hasta abril 2022 se destinó a constituir la base "In Sample" (o "In Time"), mientras que mayo 2022 se estableció como referencia para la base "Out of Sample" (o "Out of Time"). Esta partición se lleva a cabo de esta manera debido a que la aplicación del modelo es para un futuro, lo que implica la necesidad de evaluar los resultados en un período que difiere y sigue a aquel utilizado para entrenar el modelo. Dado los bajos niveles de inflación en el país donde se otorgan los créditos, no resulta necesario ajustar los saldos mediante deflactación para lograr homogeneidad.

La mencionada estructuración proporcionó un conjunto de datos "In Sample" compuesto por 365,353 créditos, utilizado en el entrenamiento y ajuste de parámetros del modelo. Esta base encapsula una parte considerable de la información, permitiendo que el modelo asimile patrones y relaciones latentes. Por otro lado, la base "Out of Sample" compuesta por 38,891 créditos, tiene como finalidad someter el modelo a una prueba de generalización. Estos datos

son completamente ajenos al proceso de desarrollo y entrenamiento del modelo, representando situaciones realistas y desconocidas. La tasa de incumplimiento promedio de la base "In Sample" es de 12.5 % mientras que en la base "Out of Sample" es de 14 %.

2.2. Estadística descriptiva

Para comprender a fondo los datos, se emplea el análisis de estadística descriptiva, que permite resumir y visualizar las características esenciales.

Análisis univariado

En primer lugar examinaremos, las distribuciones y características marginales de las variables en forma univariada. En este enfoque, se investigan las características y propiedades específicas de una sola variable a la vez, lo que permite comprender su distribución, tendencia central, dispersión y cualquier patrón característico que pueda estar presente. El análisis univariado es esencial para obtener información preliminar sobre las variables y puede proporcionar ideas valiosas sobre su comportamiento fundamental antes de explorar relaciones más complejas en un análisis multivariado. Este análisis se realizó sobre la base "In Sample".

Se procedió a calcular una serie de estadísticas descriptivas para cada variable presente en el conjunto de datos. Entre las métricas calculadas se incluyen el valor mínimo, máximo, promedio, percentiles, desviación estándar y porcentaje de valores nulos. Con base en los resultados obtenidos, se tomó la decisión de eliminar aquellas variables con un alto porcentaje de valores nulos y aquellas con una única instancia. Asimismo, se identificaron las variables que contenían valores nulos, y se decidió aplicar un tratamiento de imputación de datos para abordar esta situación, más adelante explicaremos este punto en detalle.

Adicionalmente, se examinó la tasa de incumplimiento promedio en función de los períodos de originación, con el objetivo de identificar patrones atípicos. En la Tabla 2.1 se destaca el mes de mayo de 2021, que presenta una tasa de incumplimiento considerablemente menor en comparación con los demás meses. Este comportamiento inusual puede atribuirse a políticas de crédito distintas aplicadas en ese período. Como resultado, se optó por excluir los datos correspondientes al mes de mayo de 2021 de la base de datos.

En un último análisis, se enfocó en la distribución de la variable "Monto del crédito", cuya representación gráfica se observa en la Figura 2.1. La distribución exhibe asimetría, con una concentración significativa de créditos de montos bajos y una presencia limitada de créditos con montos elevados. A modo de complemento, en la Figura 2.2 se puede ver el gráfico boxplot que muestra la distribución de dicha variable en función del período de otorgamiento, donde se puede observar que la cola pesada de la distribución en los créditos de montos altos aparece en todos los períodos. Es importante mencionar, que se realizaron análisis con la variable original y el logaritmo de la misma, al no tener diferencias significativas en los resultados se optó por utilizar la variable sin transformaciones.

La relevancia de este hallazgo radica en su impacto en los resultados monetarios derivados de la aplicación del modelo desarrollado. Dado que la

Período	Tasa de incumplimiento
$05-2021$	$4.8\,\%$
$06-2021$	$12.5\,\%$
$07-2021$	$10.6\,\%$
$08-2021$	$8.0\,\%$
$09-2021$	$8.7\,\%$
$10-2021$	$7.9\,\%$
$11-2021$	$12.8\,\%$
$12-2021$	$12.3\,\%$
$01-2022$	$13.4\,\%$
$02-2021$	$14.9\,\%$
$04-2022$	$14.3\,\%$

Cuadro 2.1: Tasa de incumplimiento por mes de originación

mejora de los resultados financieros es uno de los objetivos primordiales del modelo, se abordará este aspecto con mayor profundidad en secciones posteriores.

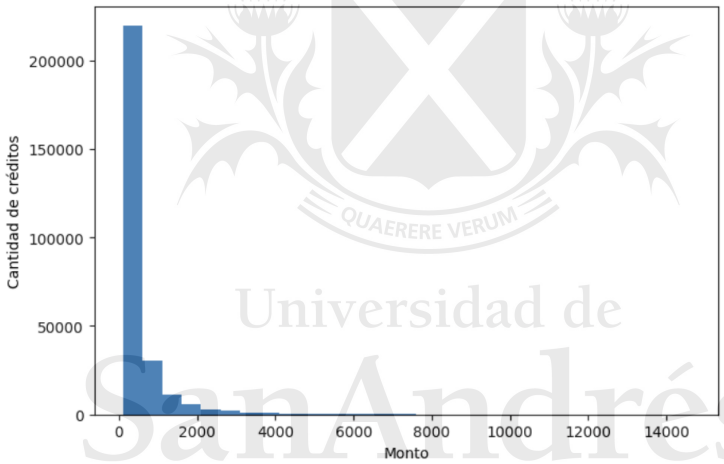


Figura 2.1: Histograma de la variable "Monto del crédito"

Análisis bivariado

El análisis bivariado es un enfoque estadístico que se utiliza para examinar la relación entre dos variables en un conjunto de datos. A través de este método, se investiga cómo los valores de una variable pueden cambiar en función de los valores de otra variable. El análisis bivariado puede revelar patrones, correlaciones y asociaciones entre las variables, lo que ayuda a comprender cómo interactúan y se influyen mutuamente. Al comparar las distribuciones, tendencias y comportamientos de dos variables, es posible identificar conexiones significativas y obtener información valiosa sobre su interacción.

En el contexto de este estudio, se analizó la interacción entre la variable que identifica el incumplimiento de los pagos con el resto de las variables. Este análisis inicial permitió obtener una comprensión preliminar de aquellas variables que podrían resultar relevantes para explicar la variable objetivo del

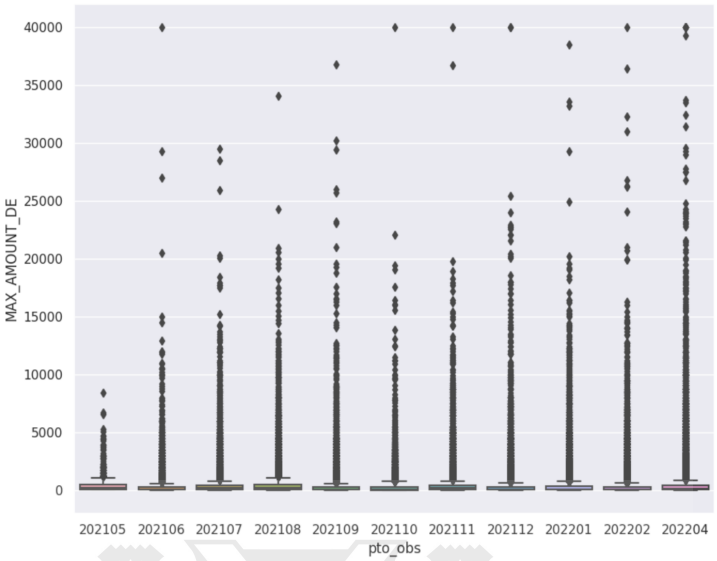


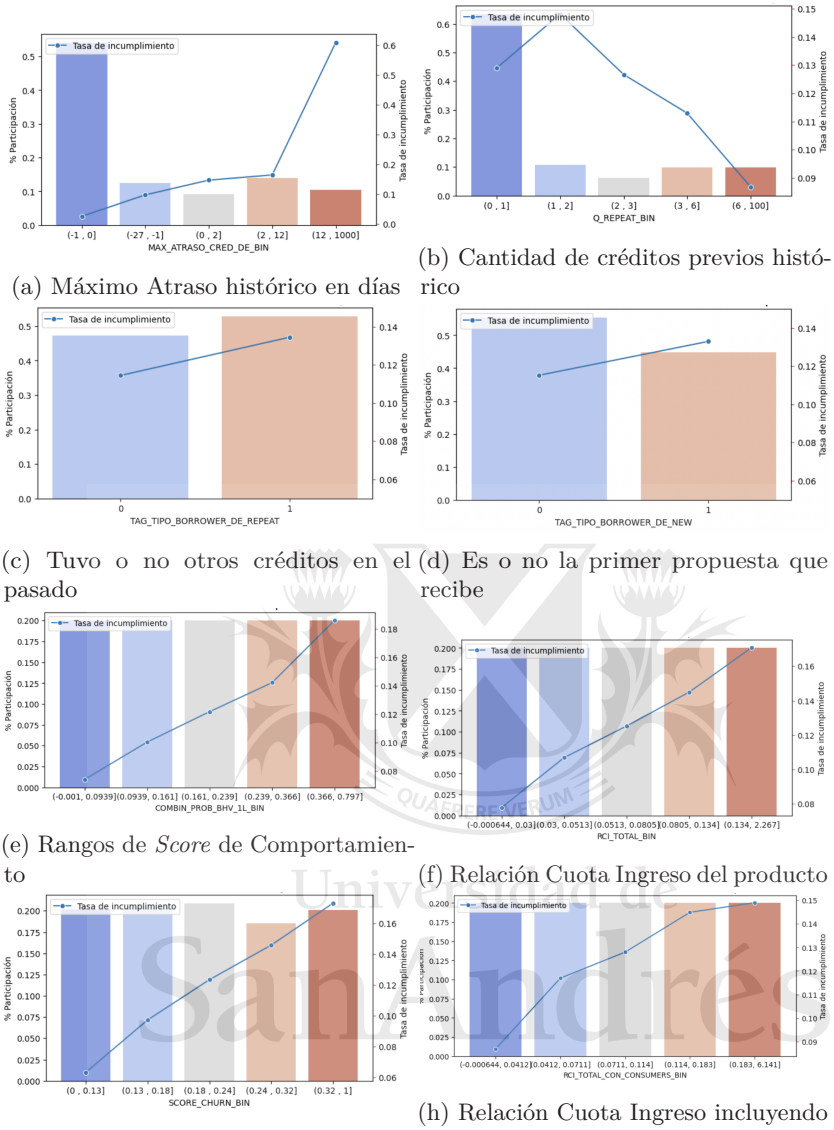
Figura 2.2: Boxplot "Monto del crédito" en función del período de otorgamiento

modelo. Además, posibilitó evaluar la congruencia de los resultados del modelo propuesto. Este análisis se realizó sobre la base "In Sample".

La Figura 2.3 presenta los gráficos bivariados entre la variable objetivo del modelo y aquellas variables que demostraron una interacción significativa con la misma. Allí se puede ver que al aumentar la cantidad de días de atraso, el incumplimiento aumenta marcadamente; simultáneamente se destaca que más del 80 % de la base presentan un incumplimiento menor a 2 días. También se puede apreciar que a medida que aumenta la cantidad de créditos que el cliente tuvo históricamente, menor es la tasa de incumplimiento y que cerca del 50 % de la base no tuvo otro crédito anteriormente. Por otro lado, en aquellos casos donde el crédito se corresponde con la primer propuesta de crédito ofrecida al cliente, el incumplimiento es mayor que en los casos donde tuvo otras propuestas en el pasado. Adicionalmente, se puede ver que a medida que aumenta el valor del *Score* de comportamiento desarrollado por la entidad para otro producto, mayor es el incumplimiento. El mismo comportamiento se ve en el caso del *Score* de *Churn* desarrollado por la entidad. También se destaca que el incumplimiento es creciente a medida que aumenta la Relación Cuota Ingreso del producto (RCI). El mismo comportamiento se ve cuando se incluye en el cálculo de la RCI, los compromisos que tiene el cliente en otros productos.

2.3. Tratamiento de valores nulos

Durante el análisis univariado, se identificó un desafío relacionado con la presencia de un notable número de variables con valores nulos en los datos. Al profundizar en este aspecto, se llegó a la conclusión de que la codificación como valores nulos representaba características intrínsecas de los datos. La existencia de estos valores faltantes era completamente justificable, ya que reflejaban la ausencia de ciertas características. Por ejemplo, si consideramos una variable



(a) Máximo Atraso histórico en días

(b) Cantidad de créditos previos histórico

(c) Tuvo o no otros créditos en el pasado

(d) Es o no la primer propuesta que recibe

(e) Rangos de *Score* de Comportamiento

(f) Relación Cuota Ingreso del producto

(g) Rangos de *Score* de *Churn*

(h) Relación Cuota Ingreso incluyendo productos de Consumer

Figura 2.3: Gráficos bivariados correspondientes a variables que suelen ser informativas en modelos crediticios. En todos los casos la línea azul muestra la tasa de incumplimiento.

que indica el atraso en pagos de un cliente en un producto distinto al que se está modelando, es plausible que algunos clientes no posean dicho producto, lo que resultaría en valores nulos para esa variable. Esta circunstancia requiere un enfoque especial para su manejo.

El primer paso consiste en crear una variable *dummy* que represente la presencia o ausencia de la característica en cuestión. Siguiendo el ejemplo anterior, esta variable *dummy* indicaría si el cliente tiene o no ese producto específico. En segundo lugar, se genera una nueva variable que combina la

variable *dummy* creada en el paso previo con la variable original. En otras palabras, cuando la variable *dummy* refleja la existencia de la característica, la variable combinada adopta los valores de la variable original; de lo contrario, toma el valor de cero. Finalmente, en el algoritmo del modelo seleccionado se incorporarán las nuevas variables generadas, mientras que la variable original será excluida.

2.4. División de la base de datos

La subdivisión de los datos en conjuntos de entrenamiento, prueba y fuera de muestra es un paso crucial en el proceso de desarrollo y evaluación del modelo. Esta etapa tiene como finalidad validar la habilidad del modelo para generalizar y hacer predicciones en datos no previamente observados.

Como se mencionó anteriormente, se llevó a cabo la división de la base en los segmentos "In Sample" y "Out of sample". Además, dentro del conjunto "In Sample", se procedió a realizar una partición adicional entre la Base de Entrenamiento (o *Train*) y la Base de Prueba (o *Test*). La Base de Entrenamiento se emplea con el propósito de entrenar el modelo y ajustar sus parámetros, mientras que la Base de Prueba se reserva para afinar los hiperparámetros y evaluar el rendimiento del modelo durante su desarrollo.

La selección de observaciones para cada uno de estos conjuntos se realizó mediante un proceso aleatorio, asignando un 80 % de los datos a la Base de Entrenamiento y el 20 % restante a la Base de Prueba. Esta estrategia asegura que el modelo sea entrenado y evaluado en segmentos distintos de los datos, lo que contribuye a una evaluación más confiable y realista de su rendimiento y capacidad de generalización.

CAPÍTULO 3

Metodología estadística implementada

En este capítulo, se describe la metodología estadística empleada en el desarrollo del modelo predictivo de probabilidad de incumplimiento de los clientes. El enfoque se basa en el aprendizaje supervisado y se seleccionó el algoritmo *Random Forest* para la construcción del modelo. Además de explicar la elección de *Random Forest*, se destacarán las adaptaciones realizadas al modelo tradicional y se expondrán alternativas exploradas que, pese a haber sido probadas, no obtuvieron los resultados deseados.

3.1. Aprendizaje Supervisado

Rokach [5] define los métodos supervisados como aquellos que intentan descubrir la relación entre atributos de entrada (a veces llamados variables independientes) y un atributo objetivo (a veces denominado variable dependiente). Asimismo establece que la relación que se descubre se representa en una estructura llamada "Modelo" y que, por lo general, los modelos describen y explican fenómenos que están ocultos en el conjunto de datos; y se pueden utilizar para predecir el valor del atributo objetivo siempre que se conozcan los valores de los atributos de entrada. Es útil distinguir entre dos modelos principales de aprendizaje supervisado: Modelos de Clasificación (Clasificadores) y Modelos de Regresión. Los modelos de regresión mapean el espacio de entrada en un dominio de valores reales mientras que los clasificadores mapean el espacio de entrada en clases predefinidas.

Dentro del marco de este trabajo, se ha creado un Modelo de Clasificación que se ajusta a la naturaleza de la variable objetivo, la cual distingue entre las clases "cumple" e "incumple". El propósito principal de este modelo es predecir estas clases con base en una serie de atributos variados que están a nuestra disposición.

3.2. Árbol de Decisión

Rokach [4] define un árbol de decisión como un conjunto de pruebas de decisión estructuradas en forma de árbol que funcionan de manera de *dividir y conquistar*. Postula que cada nodo no terminal está asociado con una prueba de característica también llamada división y que los datos que caen en el nodo se

dividirán en diferentes subconjuntos según sus diferentes valores en la prueba de característica. Agregan que cada nodo hoja está asociado con una etiqueta, que se asignará a las instancias que caigan en este nodo. En la predicción, se realiza una serie de pruebas de características comenzando desde el nodo raíz y se obtiene el resultado cuando se llega a un nodo hoja.

Rokach [4] postula que los algoritmos de aprendizaje de árboles de decisión son procesos recursivos. En cada paso, se proporciona un conjunto de datos y se selecciona una división, luego esta división se utiliza para dividir el conjunto de datos en subconjuntos, y cada subconjunto se considera como el conjunto de datos dado para el siguiente paso. Afirma que la clave de un algoritmo de árbol de decisión es cómo seleccionar las divisiones.

Rokach [4] establece que dado un conjunto de entrenamiento D, la entropía de D se define como:

$$Ent(D) = -\sum_{y \in Y} P(y \mid D) \log P(y \mid D)$$

Si el conjunto de entrenamiento D se divide en subconjuntos $D_1, \ldots, D_k$ la entropía puede ser reducida, y la cantidad de reducción es la ganancia de información, es decir:

$$G(D, D_1, \ldots, D_k) = Ent(D) - \sum_{i=1}^{k} \frac{|D_k|}{|D|} Ent(D_k),$$

donde $|A|$ indica el cardinal del conjunto A.

El autor concluye que, el par de valor-característica que causará la mayor ganancia de información es seleccionado para la división.

Según Rokach [5], los profesionales y tomadores de decisiones prefieren un árbol de decisión menos complejo, ya que se considera más comprensible. Por lo general, los árboles grandes se caracterizan por su gran variabilidad, por lo tanto, exhiben una capacidad deficiente de generalización. Sin embargo, un árbol de decisión grande puede generalizar bien a nuevos patrones, si su construcción se realiza utilizando en forma adecuada la muestra de entrenamiento, validación y test y mediante técnicas de validación cruzada. Las medidas comunes de la complejidad del árbol incluyen las siguientes métricas: (a) número total de nodos; (b) número total de hojas; (c) profundidad del árbol; y (d) número de atributos utilizados.

Rokach [5] establece que es bien conocido que el error de un modelo de predicción se puede descomponer en tres componentes aditivos: el error intrínseco, el sesgo y la varianza. El error intrínseco, también conocido como error irreducible, es el componente generado debido al ruido. Esta cantidad es el límite inferior de cualquier modelo de predicción. El error de sesgo de un modelo es el error persistente o sistemático que se espera que cometa el modelo. La varianza captura la variación aleatoria en el algoritmo de un conjunto de entrenamiento a otro. Más específicamente, mide la sensibilidad del algoritmo al conjunto de entrenamiento real, o el error debido al tamaño finito del conjunto de entrenamiento. En general los modelos presentan un compromiso entre el sesgo y la varianza.

Rokach [5] afirma que en un árbol de decisión simple (es decir, un árbol con un pequeño número de nodos), el componente de sesgo del error tiende a

ser alto y el componente de varianza del error tiende a ser pequeño. A medida que aumenta la complejidad del árbol, el componente de varianza del error se vuelve más grande y el componente de error de sesgo se vuelve más pequeño.

Los procedimientos de clasificación emsamblados, buscan obtener clasificadores que tengan bajo sesgo y baja varianza, el procedimiento de bosques aleatorios, *random forest,* es uno de los más utilizados dentro de estas propuestas. Un bosque aleatorio se construye a partir de múltiples árboles de decisión extensos, que presentan bajo sesgo y alta varianza, los mismos se combinan en una salida final. Rokach [5] establece que, evidencia empírica y teórica muestra que algunas técnicas de bosque de aleatorios actúan como mecanismos de reducción de la varianza, es decir, reducen el componente de varianza del error. Otras técnicas de ensamblado, toman como punto de partida estimadores con baja varianza y alto sesgo y los agregan obteniendo clasificadores que preserven el bajo sesgo y disminuyan la varianza.

3.3. Random Forest

Boateng et al. [1] señalan que en los últimos tiempos, ha habido un gran interés en el *Ensemble Learning*, es decir, métodos que generan muchos clasificadores y agregan sus resultados. Dos métodos conocidos son el *Boosting* y el *Bagging* de árboles de clasificación. En el procedimiento *Boosting*, se genera un sucesión de árboles aleatorios, sencillos, con alto sesgo y cada varianza. Los mismos se generan de forma tal, cada árbol tiene en cuenta la clasificación dada por los árboles que se encontraban previamente en el modelo, otorgándole un mayor peso a las observacions que fueron incorrectamente clasificadas en las etapas anteriores. Al final, la clasificación se realiza, mediante el voto ponderado que se dio a los clasificadores débiles generados a lo largo del proceso iterativo, compensando de este modelo el sesgo de los estimado. Por otra parte, el procedimiento *Bagging*, consiste en construir un conjunto de árboles de clasificación $T_1, \ldots, T_B$ (los mismos son árboles complejos, caracterizados por alta varianza y bajo sesgo) basado en muestras bootstrap obtenidas en forma independiente de la muestra de entrenamiento. Luego, para cada observación x se tiene una clasificación dada por cada árbol, $T_i(x)$, para $i = 1, \ldots, B$. Finalmente, se considera el voto mayoritario simple para la clasificación *Boosting*, de este modo se compensa la alta variabilidad. La principal desventaja que presenta el método de *Boosting* es que los árboles que consideran, suelen tener alta correlación debido a que típicamente las ramas iniciales utilizan las mismas variables y cortes, utilizando las variables con mejor poder predictivo. La alta correlación da lugar a mayor varianza en las estimaciones. Con el objetivo de solucionar este problema se propone el procedimiento de *Random Forest*, extensión de *Bagging*, donde la principal diferencia es la incorporación de la selección aleatoria de características en cada uno de los pasos de la construcción de los árboles. En cada paso de selección de corte óptimo en cada rama, en *Random Forest*, primero selecciona aleatoriamente un subconjunto de características, y luego se lleva a cabo el procedimiento convencional de selección de corte óptimo dentro del subconjunto de características seleccionadas.

Luego, como señalan Boateng et al. [1], el clasificador *Random Forest* agrega una capa adicional de aleatoriedad a *Bagging*. Además de construir cada árbol utilizando una muestra *bootstrap* diferente de los datos, *Random Forest* modifica

cómo se construyen los árboles de clasificación o regresión. En los árboles estándar, cada nodo se divide utilizando la mejor división entre todas las variables, mientras que en un *Random Forest*, cada nodo se divide utilizando la mejor división entre un subconjunto de predictores elegido al azar en ese nodo. Según Boateng et al. [1], esta estrategia algo contraintuitiva resulta ser muy efectiva en comparación con muchos otros clasificadores. Boateng et al. [1] agregan que *Random Forest* es muy fácil de usar en el sentido de que tiene pocos parámetros y generalmente no es muy sensible a sus valores. Este algoritmo puede manejar muy bien características categóricas y también puede manejar espacios de alta dimensionalidad, así como con muestras de entrenamiento de gran tamaño.

3.4. Modelo seleccionado

Como se mencionó anteriormente, se optó por el algoritmo *Random Forest* para la construcción del modelo destinado a predecir el cumplimiento o incumplimiento de los pagos del producto. Esta elección se basa en varios motivos clave que hacen que el *Random Forest* sea una opción apropiada para abordar el problema en cuestión. Uno de los factores determinantes es la abundancia de datos y variables disponibles en este contexto. Dado que se dispone de una gran cantidad de información, incluyendo múltiples atributos relacionados con los clientes y su historial de pagos, *Random Forest* se destaca por su capacidad para manejar conjuntos de datos complejos y de alta dimensionalidad de manera eficiente. Además, la naturaleza aleatoria de la construcción del *Random Forest* permite mitigar el riesgo de sobreajuste en situaciones en las que la cantidad de variables es significativa en comparación con el tamaño de la muestra.

Otro motivo importante es la flexibilidad y adaptabilidad que ofrece el *Random Forest*. Dada la naturaleza heterogénea de los datos, incluyendo variables categóricas y numéricas, el algoritmo puede manejar de manera efectiva esta diversidad sin requerir una preprocesamiento extensivo. Además, *Random Forest* es capaz de capturar relaciones no lineales y complejas entre las variables, lo que es crucial para un problema de predicción de incumplimiento de pagos, donde las relaciones entre los atributos pueden ser sutiles y no lineales.

En resumen, la elección del algoritmo *Random Forest* se justifica por su habilidad para manejar conjuntos de datos grandes y complejos, su adaptabilidad a diversas naturaleza de los datos y su capacidad para capturar relaciones complejas entre variables. Estas características hacen que sea una herramienta poderosa y adecuada para el desarrollo de un modelo predictivo preciso y confiable para el problema de predicción de incumplimiento de pagos.

Ajustes al Random Forest tradicional

Como se indicó previamente, si bien el objetivo fundamental de la tesis es detectar el cumplimiento/incumplimiento en el pago de los creditos, si no se otorga un rol relevante a los montos de los creditos se pierde de vista el rendimiento de la empresa. Considerando esta meta, junto con la significativa asimetría que caracteriza a la variable “Monto del crédito”, la cual ejerce una influencia contundente en los resultados finales, surge la necesidad de explorar ajustes al enfoque tradicional del *Random Forest* con el propósito de lograr mejoras sustanciales en los resultados obtenidos.

Al implementar el proceso de *Random Forest* se llevó a cabo una adaptación en el procedimiento de validación cruzada (*Cross Validation*) destinada a la selección de los hiperparámetros. Esta adaptación se orientó hacia una alineación más directa con el objetivo subyacente del modelo. En este caso no se buscó elegir los parámetros de *Random Forest* maximizando el *accuracy* o alguna otra métrica estándar sino que se definió una métrica ad hoc para este problema específico. A continuación explicaremos el proceso.

Consideramos un esquema de validación cruzada en V-folds. Sean $\Lambda = \{\lambda_1, \ldots, \lambda_r\}$ los r subconjuntos posibles de hiperparámetros a ajustar. Dividimos la muestra de entrenamiento, en forma aleatoria, en V folds disjuntas de igual tamaño (o que difieran a los sumo en una observación), $G_1, \ldots, G_V$, donde el cardinal de G_i es n_i. Fijamos $\lambda \in \Lambda$ y para cada fold G_i con $i \in \{1, \ldots, V\}$, ajustamos el modelo *Random Forest*, considerando todas las observaciones que no pertenecen a G_i, es decir que pertencen a $G_i^c = \{1, \ldots, n\} \setminus G_i$. Luego, para cada $X_j \in G_i$ tenemos el resultado esperado a este modelo dado por,

$$R_{\lambda,i}^j = I_j \times (1 - p_{\lambda,i}^j) - X_j \times p_{\lambda,i}^j,$$

donde $p_{\lambda,i}^j$ es la proporción de árboles del modelo *Random Forest* (ajustado en base a la muestra G_i^c con hiperparámetros λ), donde se predice que el j-ésimo individuo del fold G_i va a defaultear el préstamo y X_j es el monto asociado a cada crédito. Por otra parte, I_j es el interés que se cobrará si el j-ésimo individuo del fold G_i no va a defaultear el préstamo. Por ejemplo si $p_{\lambda,i}^j = 0$ el resultado es I_j, mientras que si $p_{\lambda,i}^j = 1$ el resultado es $-X_j$.

Luego, para cada fold G_i y para cada subconjunto de hiperparámetros λ la ganancia esperada está dada por la siguiente ecuación,

$$L_\lambda^i = \frac{1}{n - n_i} \sum_{j \in G_i^c} X_j + R_{\lambda,i}^j.$$

Es decir, que para cada individuo j del fold i, la ganancia esperada es 0 si la predicción es que incumpla el pago del crédito con probabilidad uno; y es $X_j + I_j$ si el modelo predice que va a devolver el préstamo con probabilidad uno.

Finalmente, calculamos la ganancia global media para cada subconjunto de hiperparámetros λ,

$$L_\lambda = \frac{1}{V} \sum_{i=1}^{V} L_\lambda^i.$$

El subconjunto de parámetros seleccionado $\widehat{\lambda}$, es aquel que maximiza la ganancia global L_λ, es decir

$$\widehat{\lambda} = \arg\max_{\lambda \in \Lambda} L_\lambda.$$

Para concluir, se ajusta el modelo *Random Forest* con hiperparámetros $\widehat{\lambda}$ sobre la muestra de entrenamiento.

Es importante destacar, que este fue el modelo que tuvo mejor desempeño.

Los parámetros que se eligieron por validación cruzada fueron los siguientes hiperparámetros óptimos: profundidad, cantidad de variables y cantidad de árboles. La grilla utilizada es la siguiente para los parámetros profundidad de

los árboles, cantidad de variables utilizadas en cada corte y número de árboles utilizados fueron:

$$profundidad = (20, 30, 40, 50, 60, 70, 80),$$

$$variables = (10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60),$$

y

$$árboles = (10, 20, 30, 40, 50, 60, 70, 80, 90, 100),$$

respectivamente. Los parámetros óptimos resultantes fueron 20, 30 y 50 respectivamente.

A continuación describimos brevemente otras ideas que en este problema específico no tuvieron mejores resultados pero que en otros casos podrían ser beneficios.

Dado que los incumplimientos en créditos de montos elevados ejercen una influencia significativamente mayor en los resultados monetarios finales que los incumplimientos en créditos de montos más bajos, se decidió explorar una asignación ponderada distinta para cada observación:

$$w_i = \frac{X_i}{\sum_{j=1}^{n} X_j}, \tag{3.1}$$

donde w_i representa el nuevo peso de cada observación y X_i el monto asociado a cada uno de los créditos y n es la cantidad total de observaciones en el conjunto de datos de entrenamiento. El modelo estándar asigna igual peso, $1/n$, a cada observación de la muestra.

La implementación de esta modificación en el modelo no condujo a los resultados anticipados. Al profundizar en los análisis de los resultados, se reveló que el desempeño del modelo no lograba capturar adecuadamente el comportamiento de los créditos de mayor cuantía. A partir de esta observación, se procedió a realizar una nueva adaptación en los pesos dados por la Ecuación (3.1) buscando darle mayor peso aún, para esto se consideraron pesos exponenciales, en busca de una mejora sustancial:

$$\widetilde{w}_i = \frac{exp(X_i)}{\sum_{j=1}^{n} exp(X_j)}.$$

Sin embargo, aún esta segunda modificación no arrojó los resultados deseados.

Luego, se llevó a cabo un intento con la ejecución de dos modelos distintos, uno destinado a los créditos de alta cuantía y otro para los demás. La porción que compone el primer segmento representa menos del 0.1 % de los casos en cantidad, pero alcanza un 6 % en términos de monto total. La finalidad de esta estrategia era tratar de capturar de manera más precisa el comportamiento específico de los créditos de mayor envergadura, con el propósito de lograr mejoras en los resultados financieros. A pesar de ello, esta alternativa tampoco resultó en los desenlaces anticipados.

Finalmente, se intentó ajustar un *Random Forest* de regresión con el "Revenue" de cada crédito como variable objetivo. Esta alternativa tampoco arrojó los resultados esperados.

Elección de punto de corte

El modelo *Random Forest* proporciona como salida, la proporción de árboles donde la predicción es el incumplimiento. No obstante, al llevar este modelo a la aplicación práctica, es esencial poder traducir esta proporción en una decisión concreta de otorgar o no otorgar el crédito. En otras palabras, resulta crucial establecer un umbral en la proporción de incumplimiento generada por el modelo, el cual permitirá distinguir entre la concesión o denegación del crédito. En esta fase, también consideramos el objetivo de mejorar los resultados económicos.

En la Ecuación (3.2), se puede ver que para la determinación del punto de corte utilizamos una función de pérdida que será la suma de los intereses perdidos por aquellos casos en los que el modelo predijo "incumple", pero la realidad fue de cumplimiento, más la suma del monto del crédito para aquellos en los que el modelo predijo "cumple" pero hubo incumplimiento.

$$perdida = \sum_{j=1}^{n} P_j \times (1 - R_j) \times I_j + (1 - P_j) \times R_j \times X_j \tag{3.2}$$

donde

$$P_j = \begin{cases} 1 & si \quad p \geq th \\ 0 & si \quad p < th \end{cases}$$

y

$$R_j = \begin{cases} 1 & si \quad cliente \ incumplió \ realmente \\ 0 & si \quad cliente \ cumplió \ realmente, \end{cases}$$

donde n cantidad de créditos en la base; I_j los intereses generados en cada crédito; X_j el capital de cada crédito; p la salida del *Random Forest*; y th el valor del umbral que se quiere buscar.

La Ecuación (3.2) se compone por dos términos, uno que considera el costo de oportunidad perdido al no haber podido cobrar los intereses en caso de no haber otorgado el crédito a un cliente que hubiera cumplido;y otro por la pérdida de capital a un cliente que incumple con el préstamo otorgado.

Para un rango de valores de th entre 0 y 1, se calcula la pérdida resultante de la Ecuación (3.2) y se selecciona el valor óptimo que minimiza dicha pérdida. La determinación del punto de corte se llevó a cabo utilizando la muestra de prueba "In sample".

La Figura 3.1 exhibe los diversos valores de pérdida que permitieron elegir el umbral 0,47 como punto de corte más apropiado. Esta elección, nos brinda un enfoque más efectivo para tomar decisiones respecto a la concesión de créditos. Este punto de corte es el que se utilizará más adelante en el armado de las matrices de confusión, el cálculo de las métricas de calidad predictiva del modelo y el cálculo de los resultados económicos.

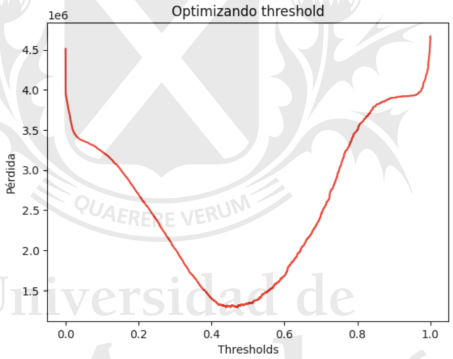


Figura 3.1: Valor de la pérdida en función de distintos puntos de corte

CAPÍTULO 4

Resultados

En este capítulo, se presentan los resultados obtenidos a través del desarrollo y la evaluación del modelo Random Forest. Se describirán las métricas de rendimiento utilizadas para evaluar el modelo, los resultados monetarios y la importancia de las características identificadas por el mismo. Además, se discutirá la interpretación de los resultados en relación con los objetivos de la investigación.

4.1. Métricas de rendimiento

Las métricas de rendimiento seleccionadas para evaluar el modelo Random Forest en la predicción de la probabilidad de incumplimiento de los clientes incluyen: la matriz de confusión, el accuracy, el recall, la precisión y la curva ROC. Las distintas métricas fueron aplicadas tanto en la base de test "In Sample" (o "In Time") como "Out of Sample" (o "Out of Time").

Matriz de confusión

La matriz de confusión proporciona una descripción detallada del rendimiento del modelo al predecir las clases positivas y negativas. En el caso de este estudio, las clases se definen como "incumple" y "cumple". Como se puede ver en la Tabla 4.1, la cantidad de observaciones predichas correctamente en la base de test "In Sample" son 66.286, mientras que las predichas incorrectamente son 6.000 :

		Predicción	
		Cumple	Incumple
Real	Cumple	58,031	5,109
	Incumple	891	8,255

Cuadro 4.1: Matriz de confusión para la base de test "In Sample"

Por su parte, como se puede inferir en la Tabla 4.2, la cantidad de observaciones predichas correctamente en la base de test "Out of sample" son 31.812, mientras que las predichas incorrectamente son 6.591:

Como era de preveer, los resultados en la base de test "Out of Sample" muestran un desempeño inferior en comparación con la base "In Sample". Esta disparidad puede atribuirse a diversos factores que afectan la capacidad del modelo para generalizar y adaptarse a datos nuevos y desconocidos.

		Predicción	
		Cumple	Incumple
Real	Cumple	27,969	5,045
	Incumple	1,546	3,843

Cuadro 4.2: Matriz de confusión para la base de test "Out of Sample"

En primer lugar, al haber entrenado el modelo con la base "In Sample", este ha adquirido conocimiento específico de los patrones contenidos en esos datos. Sin embargo, esta especialización puede dificultar su habilidad para generalizar adecuadamente y aplicar estos patrones a un conjunto de datos diferente. En segundo lugar, es plausible que se haya producido un fenómeno de sobreajuste (*Overfitting*), especialmente en segmentos de la muestra que están subrepresentados. Cuando el modelo se ajusta en exceso a los detalles particulares de los datos de entrenamiento, puede no ser capaz de adaptarse eficazmente a datos nuevos y desconocidos, lo que impacta negativamente en su rendimiento general. En tercer lugar, la base de test "Out of Sample" podría presentar un mayor nivel de ruido o variabilidad en comparación con la base de entrenamiento. Esta mayor incertidumbre puede confundir al modelo y reducir su precisión en la predicción. Finalmente, los cambios en la distribución de los datos entre las bases "In Sample" y "Out of Sample" también pueden contribuir a la brecha en el rendimiento. Si el modelo no ha sido expuesto a las variaciones presentes en los datos de prueba, es natural que su capacidad para realizar predicciones precisas se vea disminuida.

Recall, Precision y Accuracy

El recall, también conocido como sensibilidad, mide la proporción de casos positivos que el modelo ha identificado correctamente respecto al total de casos positivos reales. En nuestro caso, de los 9,146 casos de real incumplimiento en la base de test "In Sample", el modelo ha identificado como incumplimiento a 8,255, es decir al 90 %. En el caso de la base de test "Out of Sample", de los 5,389 casos de real incumplimiento, el modelo ha identificado como incumpliento a 3,843, es decir al 71 %.

Por otro lado, la precisión mide la proporción de casos positivos identificados correctamente en relación con todos los casos clasificados como positivos. En nuestro caso, de los 13,364 predichos como incumplimiento en la base de test "In Sample", el 62 % realmente incumplieron. En la base de test "Out of Sample" por su parte, de los 8,888 créditos predichos como incumplimiento, el 43 % realmente incumplieron.

Adicionalmente, se calculó el accuracy del modelo, que es la proporción de predicciones correctas en relación con el total de predicciones realizadas. En la base de test "In Sample", del total de 72,286 créditos, se predijeron correctamente el 92 %; mientras que en la base de test "Out of Sample", del total de 38,403 créditos, se predijeron correctamente el 83 %.

En la Tabla 4.3, pueden observarse los valores de las distintas métricas mencionadas, calculadas tanto en la base de test "In Sample" como en la "Out of Sample".

Es importante mencionar que el modelo arroja buenos resultado tanto de accuracy como de recall. Es decir, que hay un alto porcentaje de los créditos

Base	Recall	Precision	Accuracy
"In Sample"	90 %	62 %	92 %
"Out of Sample"	71 %	43 %	83 %

Cuadro 4.3: Recall, Precisicion y Accuracy para bases de test "In Sample" y "Out of Sample"

que incumplieron que el modelo los identifica como tal y también hay un alto porcentaje de los créditos que cumplieron que el modelo los identifica como tal. Sin embargo, el valor de la precisión es bajo, lo que significa que el modelo está identificando como incumplimiento a gran cantidad de créditos que en realidad cumplieron con sus pagos. Esto último en principio puede no ser algo bueno y es muy importante, en consecuencia, entender cuál es el impacto monetario de estos resultados. Más adelante se aborda este tema en profundidad.

Curva ROC y AUC

Fawcett [3] define la curva ROC como una herramienta gráfica que ilustra el desempeño de un modelo en diferentes umbrales de clasificación. Visualiza la relación entre la tasa de verdaderos positivos y la tasa de falsos positivos. Es posible trazar una diagonal en el gráfico que representa la estrategia de elección al azar. Los puntos por debajo de esta diagonal indican un rendimiento inferior al azar, mientras que aquellos por encima sugieren un rendimiento superior. El punto (0,1) simboliza una clasificación perfecta, donde no hay falsos positivos y la tasa de verdaderos positivos alcanza el 100 %. En resumen, en la búsqueda de un modelo de clasificación óptimo, nos esforzamos por acercarnos a la esquina superior izquierda de esta curva.

Por otro lado, una métrica usual para caracterizar la curva ROC es medir el Área Bajo la Curva (AUC) es una métrica que nos permite comparar diferentes modelos de clasificación. Representa el área bajo la curva ROC y es un indicador esencial de la capacidad discriminativa del modelo. Variando entre 0 y 1, un valor más alto de AUC indica un mejor rendimiento. Si bien el AUC no será menor a 0,5 (el área bajo la diagonal), un modelo razonable debería aspirar a tener un AUC cercano a 1 (ver Fawcett [3]).

En la Figura 4.1, se pueden visualizar tanto la curva ROC como los valores de AUC en ambas bases de prueba (0,97 para la base "In Sample" y 0,89 para la base "Out of Sample"). Estos valores, notablemente cercanos a 1, subrayan el buen rendimiento de los modelos. En conjunto, la curva ROC y el AUC proporcionan una perspectiva completa y cuantitativa del poder predictivo del modelo.

4.2. Resultados monetarios

En el contexto del presente trabajo, más allá de los resultados de las métricas previamente mencionadas, cobra importancia analizar si el modelo tiene el potencial de generar mayores beneficios para la compañía. Como se ha mencionado en la Sección 4.1 y como puede verse en la Tabla 4.1, expuesta anteriormente, resulta evidente que el valor de la precisión del modelo es bajo, lo cual en principio podría parecer un panorama desfavorable. No

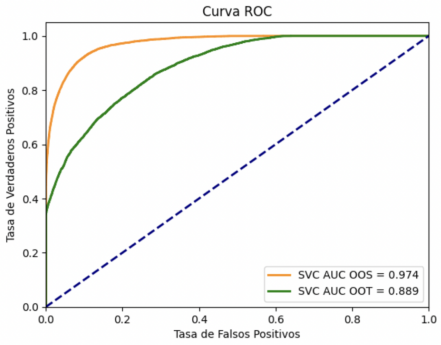


Figura 4.1: Curvas ROC y AUC para bases de test "In Sample" y "Out of Sample"

obstante, esta perspectiva puede alterarse significativamente al considerar la distribución asimétrica de los montos de los créditos, detallada en el Capítulo 2. Es precisamente esta distribución la que establece un fundamento esencial para evaluar la situación financiera en su totalidad y no solo considerar la calidad predictiva del modelo medida con las métricas habituales. En ese sentido, se define "ganancia" como la acumulación de intereses obtenidos por aquellos créditos donde los pagos fueron cumplidos, ajustada por la suma del capital e intereses de los créditos en los que no se cumplió con los pagos. Con el fin de cuantificar estos efectos financieros, se llevó a cabo el cálculo de las ganancias del producto tanto antes como después de la aplicación del modelo desarrollado. Además se calculó el *Loss Ratio*, métrica que se utiliza en la entidad para gestionar el producto. El *Loss Ratio* se calcula mediante la siguiente fórmula:

$$LossRatio = \frac{Capital + Intereses\ \ créditos\ \ que\ \ incumplieron + Costos}{Revenues}, \tag{4.1}$$

donde *Capital + Intereses créditos que incumplieron* es el monto de aquellos créditos que no cumplieron con su pago; *Costos* refiere a todos los costos del producto, incluye costos de fondeo, costos de cobranza, costos de bureau de crédito, entre otros y *Revenues* hace referencia a las ganancias por intereses del producto.

Un *Loss Ratio* mayor a 1 indica que el producto no está siendo rentable, ya que los revenues generados en el producto no alcanzan para cubrir los costos y el monto de aquellos créditos que incumplieron con su pago.

Este enfoque financiero se ilustra con mayor claridad en la Tabla 4.4, que muestra los resultados previos y posteriores a la implementación del modelo en la base de test "In Sample". Antes de la aplicación del modelo, los créditos que se habían otorgado eran 101,357 por un monto igual a 37,864,882. Al aplicar el modelo en la base de test "In Sample", aquellos créditos con una salida del *Random Forest* menor al umbral elegido y que por lo tanto, se asume se hubiesen otorgado, sería de 85,058, con un monto del crédito que suma 28,095,369. Lógicamente, como en la práctica, como no se estaba aplicando ningún modelo para decidir el otorgamiento de los créditos, al aplicar el modelo,

la cantidad de créditos otorgados y su monto son inferiores al original. Es importante destacar que los resultados originales del producto eran negativos. No obstante, la adopción del modelo desarrollado posibilitaría a la entidad generar ganancias positivas y reorientar el rendimiento financiero hacia un escenario favorable. Esto es así debido a que un gran porcentaje de los créditos que imcumplieron están siendo capturados por el modelo y por lo tanto, al no considerar su otorgamiento, es posible generar ganancias positivas. Si bien al identificar el modelo una gran cantidad de créditos como incumplimiento cuando en realidad no lo son (baja precisión del modelo) se está perdiendo la posibilidad de otorgar más créditos, se considera en esta instancia más importante poder identificar una gran cantidad de los créditos que incumplen para poder pasar de resultados negativos a positivos.

Métrica	Sin el modelo	Con el modelo
Créditos originados	101,357	85,058
Monto originado	\$37,864,882	\$28,095,369
Revenues	$\$ - 448{,}830$	\$3,223,579
Loss Ratio	1,09	0,13

Cuadro 4.4: Resultados monetarios en la base de test "In Sample"

Al observar los resultados en la base de test "Out of sample", representados en la Tabla 4.5, arribamos a las mismas conclusiones, aunque en este caso los resultados son más magros como era de esperarse.

Métrica	Sin el modelo	Con el modelo
Créditos originados	42,147	32,511
Monto originado	\$21,272,908	\$13,772,991
Revenues	$\$ - 1{,}436{,}506$	\$753,094
Loss Ratio	1,43	0,69

Cuadro 4.5: Resultados monetarios en la base de test "Out of sample"

4.3. Resultados por rangos

Como se ha mencionado anteriormente, la distribución del monto de los créditos es asimétrica, hay una gran concentración de créditos con montos chicos y muy pocos créditos con montos grandes. Si bien en el procedimiento de cross validación y de selección del punto de corte se tuvieron en cuenta los resultados de las compañía, los mismos no se tuvieron en cuenta en el output de la construcción del modelo. Es por esto que se espera que el modelo tenga un mejor ajuste en los créditos de montos bajos que en los altos.

Por lo tanto, resulta interesante hacer un estudio de la sensibilidad del modelo para diferentes percentiles de los créditos. Por un lado, se analizan los créditos *grandes*, mayores a \$10,000[1], que representan menos del 0.1 % de los casos en cantidad, pero alcanza un 6 % en términos de monto total. Los creditos restantes se separan en quintiles respecto de su monto siendo 'qi' el

[1]Se eligió ese umbral ya que se corresponde con el segmento de "Grandes Vendedores", segmento que tiene gran relevancia para la entidad y al que se le asignan políticas diferenciadas.

i-ésimo quintil. Los valores en la escala de la moneda son $50, $100, $200, $400 y $10,000.

Las Tablas 4.6 y 4.8 muestran las métricas usuales para los datos "In Sample" y "Out of Sample" respectivamente. Como es esperable, los resultados "In Sample" son mejores. Por otra parte, en ambos casos, se puede ver cómo la precisión y el accuracy empeoran de manera drástica para los quintiles mayores y que esta pérdida es más marcada para los créditos *grandes*. El recall prácticamente no varía a lo largo de los quintiles, es decir que el modelo logra identificar correctamente a un alto porcentaje de los créditos en incumplimiento en todos los rangos.

En las Tablas 4.7 y 4.9 se muestran las ganancias y el *Loss Ratio* (ver Equación 4.1) para los rangos mencionados en el párrafo anterior. Con "sin el modelo" se hace referencia a los resultados reales obtenidos sobre las bases de test "In Sample" y "Out of Sample"; mientras que con "con el modelo", a los resultados una vez aplicado el modelo desarrollado. En primer lugar, cabe destacar que, como era de esperarse, la mejora en el *Loss Ratio* una vez aplicado el modelo es mayor en la base de test "In Sample" que en la "Out of Sample". De todos modos, los resultados en la base de test "Out of Sample" son más que satisfactorios. Salvo en el segmento de créditos *grandes* que presenta un *Loss Ratio* mayor a 1. Este resultado resulta razonable ya que en primer lugar, como ya se ha mencionado, el modelo tiene un peor ajuste en este segmento; y en segundo lugar, son pocos casos y con montos grandes lo cual hace del *ratio* un valor más volátil. En segundo lugar, si bien con la implementación del modelo se otorgan menos créditos y menos monto total que el original, en todos los segmentos y en las dos bases las ganancias mejoran. Esto se explica por la habilidad del modelo de capturar los casos de incumplimiento. De todos modos, el segmento de créditos *grandes* sigue teniendo resultados negativos; pero la pérdida se reduce a casi el tercio de la original.

Rango	Recall	Precision	Accuracy
q1	92 %	78 %	96 %
q2	90 %	73 %	95 %
q3	89 %	62 %	91 %
q4	89 %	56 %	89 %
q5	89 %	50 %	84 %
$> 10{,}000$	89 %	56 %	75 %

Cuadro 4.6: Métricas segmentadas en rangos de monto del crédito en la base de test "In Sample"

4.4. Importancia de las variables

Para finalizar con el capítulo de resultados, se expondrán las variables que ejercen un mayor impacto en el modelo, en el Anexo 1 se puede encuentra una descripción de las variables analizadas. Es relevante comprender cuáles son las variables más influyentes del modelo ya que esto puede proporcionar ideas valiosas para futuras estrategias crediticias. Tal como se evidencia en la Figura 4.2, se puede ver que la importancia de las variables está centrada en prácticamente 4 variables, con un decaimiento exponencial en la incidencia en

Rango	**Ganancias sin el modelo**	**Ganancias con el modelo**	**LR sin el modelo**	**LR con el modelo**
q1	\$32,840	\$195,609	0,86	0,08
q2	\$170,139	\$385,192	0,61	0,07
q3	\$90,053	\$309,696	0,76	0,09
q4	\$92,119	\$438,363	0,84	0,11
q5	$\$ - 353{,}253$	\$1,789,144	1,13	0,15
$> 10{,}000$	$\$ - 405{,}837$	\$2,017,438	3,25	0,72

Cuadro 4.7: Resultados monetarios segmentados en rangos de monto del crédito en la base de test "In Sample"

Rango	Recall	Precision	Accuracy
q1	74 %	58 %	92 %
q2	72 %	52 %	90 %
q3	74 %	42 %	83 %
q4	73 %	40 %	78 %
q5	69 %	39 %	73 %
$> 10{,}000$	60 %	46 %	63 %

Cuadro 4.8: Métricas segmentadas en rangos de monto del crédito en la base de test "Out of Sample"

Rango	**Ganancias sin el modelo**	**Ganancias con el modelo**	**LR sin el modelo**	**LR con el modelo**
q1	\$5,776	\$67,084	0,96	0,42
q2	\$100,585	\$158,965	0,51	0,16
q3	\$65,556	\$141,076	0,70	0,22
q4	\$46,527	\$183,736	0,87	0,31
q5	$\$ - 1{,}014{,}475$	\$447,361	1,46	0,71
$> 10{,}000$	$\$ - 640{,}675$	$\$ - 231{,}189$	3,68	2,75

Cuadro 4.9: Resultados monetarios segmentados en rangos de monto del crédito en la base de test "Out of Sample"

el modelo. Las que han demostrado tener un peso significativo son aquellas que contienen información de mora pasada, características de recurrencia en los créditos, relaciones cuota/ingreso, score bureau y variaciones en las ventas. La elección de estas variables guardan estrecha relación con las operaciones del negocio en cuestión.

Además, como mostramos en el Capítulo 2 al realizar el análisis bivariado de datos, estas varibles mostraban amplio poder discriminador en relación a la variable de estudio, confirmando su asociación con el porcentaje de incumplimiento.

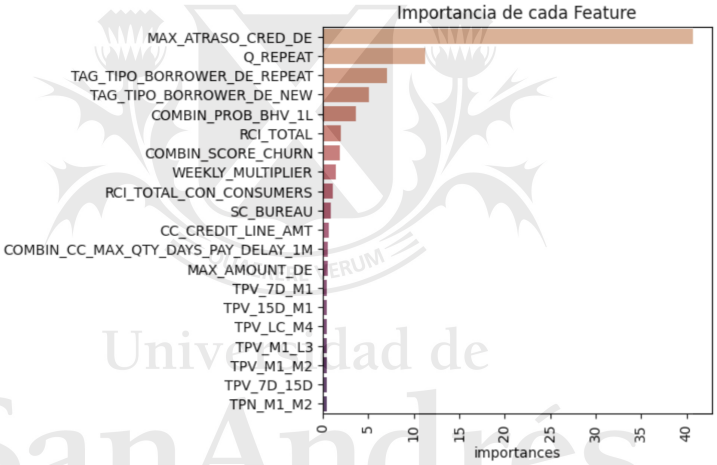


Figura 4.2: Importancia de las variables

CAPÍTULO 5

Conclusión

A modo de conclusión, se ha logrado cumplir con éxito el objetivo de desarrollar un modelo para la toma de decisiones en el otorgamiento de créditos a corto plazo, con el propósito de mejorar los resultados financieros de la compañía. A lo largo del proceso de desarrollo y evaluación del modelo, se exploraron diversas estrategias y enfoques, todos ellos orientados a maximizar la precisión de los resultados y la rentabilidad.

La consideración de la asimetría inherente a la distribución de los montos de los créditos desempeñó un papel crucial en la búsqueda de soluciones efectivas. Mediante una exhaustiva evaluación de diferentes opciones, se exploró y analizó en detalle cómo modificar el algoritmo *Random Forest* con el fin de obtener los mejores resultados.

Las distintas métricas de calidad predictiva evaluadas arrojaron niveles consistentemente satisfactorios en cuanto al desempeño del modelo. A pesar de que la precisión del modelo se ubicó en un nivel inferior, lo cual limita la capacidad de otorgar créditos en algunos casos, los valores de recall y accuracy demostraron ser altamente prometedores. Esta destacada capacidad del modelo para identificar de manera precisa los créditos en incumplimiento resulta fundamental, ya que es precisamente esta habilidad la que respalda la transición de las ganancias de negativas a positivas al incorporar el modelo en el proceso de toma de decisiones. Es importante mencionar que en el segmento de créditos *grandes*, el ajuste del modelo no es muy satisfactorio y por lo tanto, genera volatilidad en los resultados y que estos sean peores a los deseados.

Bibliografía

[1] Boateng, E. Y., Otoo, J., Abaye1, D. A. (2020) Basic Tenets of Classification Algorithms K-Nearest-Neighbor, Support Vector Machine, Random Forest and Neural Network: A Review, Journal of Data Analysis and Information Processing, **8**, 341-357.

[2] Chopra, A., Bhilare, P. (2018) Application of Ensemble Models in Credit Scoring Models, Business Perspectives and Research, **6**(2), 129–141.

[3] Fawcett, T. (2006). An introduction to ROC analysis, Pattern Recognition Letters, **27**(8), 861-874.

[4] Rokach, L. (2012) Machine Learning & Pattern Recognition Series, Chapman & Hall Cambridge, UK.

[5] Rokach, L. (2019) Ensemble Learning: Pattern Classification Using Ensemble Methods, Second Edition. Negev, Israel.

[6] Schmitt, M. (2022) Deep Learning vs. Gradient Boosting: Benchmarking stateof-the-art machine learning algorithms for credit scoring, arXiv:2205.10535.

[7] Tripathi, D., Shukla, A.K., Reddy, B.R. et al. (2022). Credit Scoring Models Using Ensemble Learning and Classification Approaches: A Comprehensive Survey, Wireless Personal Communications, **123**, 785–812.

APÉNDICE A

Detalle de las variables utilizadas

1. FLAG_SCORE_CHURN: indica si el cliente tenía *Score* de *Churn* al momento del otorgamiento del crédito
2. FLAG_PPV_CF: indica si el cliente tiene además del producto en cuestión, otros productos llamados PPV y CF
3. FLAG_ACTIVE_PPV_CF: indica si el cliente tiene una propuesta activa de los prodcutos PPV o CF
4. FLAG_SCORE_BHV: indica si el cliente tenía *Score* de comportamiento del producto CF al momento del otorgamiento del crédito
5. FLAG_LINE_CC_USE: indica si el cliente estaba utilizando su línea de consumo al momento del otorgamiento del crédito
6. FLAG_LINE_CC_USE_1M: indica si el cliente estaba utilizando su línea de consumo el mes anterior al del otorgamiento del crédito
7. FLAG_LINE_CC_USE_3M: indica si el cliente estaba utilizando su línea de consumo en los 3 meses anteriores al del otorgamiento del crédito
8. FLAG_LINE_CC_USE_6M: indica si el cliente estaba utilizando su línea de consumo en los 6 meses anteriores al del otorgamiento del crédito
9. FLAG_LINE_CC_USE_12M: indica si el cliente estaba utilizando su línea de consumo en los 12 meses anteriores al del otorgamiento del crédito
10. FLAG_LINE_CC_USE_vs_1: indica si el cliente estaba utilizando su línea de consumo tanto en el mes de otorgamiento del crédito como en el mes anterior
11. FLAG_LINE_CC_USE_vs_3: indica si el cliente estaba utilizando su línea de consumo tanto en el mes de otorgamiento del crédito como en los 3 meses anteriores
12. FLAG_LINE_CC_USE_vs_6: indica si el cliente estaba utilizando su línea de consumo tanto en el mes de otorgamiento del crédito como en los 6 meses anteriores

13. FLAG_LINE_CC_USE_vs_12: indica si el cliente estaba utilizando su línea de consumo tanto en el mes de otorgamiento del crédito como en los 12 meses anteriores

14. FLAG_UPSELL: indica si el cliente recibió un *Upsell* en su línea de consumo

15. FLAG_DOWNSELL: indica si el cliente recibió un *Downsell* en su línea de consumo

16. FLAG_LINE_CC: indica si el cliente tiene una línea de consumo en el mes de otorgamiento del crédito

17. FLAG_LINE_CC_1M: indica si el cliente tenía una línea de consumo en el mes anterior al del otorgamiento del crédito

18. FLAG_LINE_CC_3M: indica si el cliente tenía una línea de consumo en los 3 meses anteriores al del otorgamiento del crédito

19. FLAG_LINE_CC_6M: indica si el cliente tenía una línea de consumo en los 6 meses anteriores al del otorgamiento del crédito

20. FLAG_LINE_CC_12M: indica si el cliente tenía una línea de consumo en los 12 meses anteriores al del otorgamiento del crédito

21. COMBIN_SCORE_CHURN: variable combinada entre FLAG_SCORE_CHURN y el valor del *Score* de *Churn* del cliente al momento del otorgamiento del crédito

22. CRD_PROP_TOTAL_AMOUNT: monto de la propuesta

23. CONSEC_ACTIVITY_LY: cantidad de meses con ventas consecutivas al momento del otorgamiento del crédito

24. ACTIVITY: cantidad de meses con ventas al momento del otorgamiento del crédito

25. WEEKLY_ACTIVITY_L3M: cantidad de semnas con ventas en los últimos 3 meses al momento del otorgamiento del crédito

26. FLAG_POINT_PRO: indica si el cliente tiene el dispositivo de pago "PRO"

27. Q_DEVICES: indica cuántos dispositivos de venta tiene el cliente

28. MAX_AMOUNT_DE: monto del crédito

29. RATIO_CLAIMS: porcentaje de ventas con reclamos

30. Q_FINISHED_CRED: cantidad de créditos finalizados históricamente

31. COMBIN_SUM_AMOUNT_FINISHED_CREDIT: variable combinada entre FLAG_PPV_CF y el monto del crédito de los productos PPV o CF finalizados

32. COMBIN_MAX_ATRASO_FINISHED_CRED: variable combinada entre FLAG_PPV_CF y el atraso máximo que tuvieron históricamente los créditos de CF y PPV

33. COMBIN_AVG_ATRASO_FINISHED_CRED: variable combinada entre FLAG_PPV_CF y el atraso promedio que tuvieron históricamente los créditos de CF y PPV

34. COMBIN_PROP_AMOUNT_LAST_FINISHED_CRED: variable combinada entre
FLAG_PPV_CF y el monto de la propuesta de los productos PPV o CF finalizados

35. Q_ACTIVE_CRED: cantidad de créditos activos

36. COMBIN_SUM_AMOUNT_ACTIVE_CRED: variable combinada entre
FLAG_ACTIVE_PPV_CF y el monto de los créditos activos PPV y CF

37. DEBT_BALANCE: deuda total del cliente en todos sus productos

38. COMBIN_PORCENTAJE_PAGO: variable combinada entre FLAG_PPV_CF y el porcentaje de pago de los créditos PPV y CF

39. INST_TOTAL_DEBT_BALANCE: cantidad de cuotas remanentes

40. COMBIN_MAX_ATRASO_ACTIVE_CRED: combinación entre FLAG_ACTIVE_PPV_CF y el atraso máximo en los créditos activos de CF y PPV

41. COMBIN_AVG_ATRASO_ACTIVE_CRED: combinación entre FLAG_ACTIVE_PPV_CF y el atraso promedio en los créditos activos de CF y PPV

42. MAX_ATRASO_CRED_DE: cantidad máxima de días de atraso históricamente en días

43. NO_APROBABLE_CF: indica si el cliente no es aprobable para el producto CF

44. COMBIN_Q_CUOTAS_PAGAS_1L_ACTIVE: combinación entre variable FLAG_ACTIVE_PPV_CF y la cantidad de cuotas pagas

45. COMBIN_Q_CUOTAS_IMPAGAS_1L_ACTIVE: combinación entre variable FLAG_ACTIVE_PPV_CF y la cantidad de cuotas impagas

46. FLAG_MIA: indica si el cliente tiene el producto adelanto

47. SC_BUREAU: valor del *Score* de *Bureau* de crédito

48. COMBIN_PROB_BHV_1L: combinación entre variable FLAG_SCORE_BHV y el valor de la probabilidad que arroja el modelo

49. Q_CUENTAS_VINCULADAS: indica la cantidad de cuentas vinculadas

50. Q_REPEAT: cantidad de créditos que sacó el cliente históricamente

51. TPN_7D: cantidad de ventas en los últimos 7 días anteriores al otorgamiento del crédito
52. TPN_15D: cantidad de ventas en los últimos 15 días anteriores al otorgamiento del crédito
53. TPN_M1: cantidad de ventas en el último mes anterior al otorgamiento del crédito
54. TPN_M2: cantidad de ventas en los últimos dos meses anteriores al otorgamiento del crédito
55. TPN_M3: cantidad de ventas en los últimos tres meses anteriores al otorgamiento del crédito
56. TPN_M4: cantidad de ventas en los últimos cuatro meses anteriores al otorgamiento del crédito
57. TPN_M5: cantidad de ventas en los últimos cinco meses anteriores al otorgamiento del crédito
58. TPN_M6: cantidad de ventas en los últimos seis meses anteriores al otorgamiento del crédito
59. TPN_AVG_3M: cantidad de ventas promedio en los últimos 3 meses anteriores al otorgamiento del crédito
60. TPN_AVG_6M: cantidad de ventas promedio en los últimos 6 meses anteriores al otorgamiento del crédito
61. TPV_LC_7D: monto de las ventas en los últimos 7 días anteriores al otorgamiento del crédito
62. TPV_LC_15D: monto de las ventas en los últimos 15 días anteriores al otorgamiento del crédito
63. TPV_LC_M1: monto de las ventas en el último mes anterior al otorgamiento del crédito
64. TPV_LC_M2: monto de las ventas en los últimos dos meses anteriores al otorgamiento del crédito
65. TPV_LC_M3: monto de las ventas en los últimos tres meses anteriores al otorgamiento del crédito
66. TPV_LC_M4: monto de las ventas en los últimos cuatro meses anteriores al otorgamiento del crédito
67. TPV_LC_M5: monto de las ventas en los últimos cinco meses anteriores al otorgamiento del crédito
68. TPV_LC_M6: monto de las ventas en los últimos seis meses anteriores al otorgamiento del crédito
69. TPV_LC_AVG_3M: monto promedio de ventas en los últimos 3 meses anteriores al otorgamiento del crédito

70. TPV_LC_AVG_6M: monto promedio de ventas en los últimos 6 meses anteriores al otorgamiento del crédito

71. TPV_LC_SMOOTHED_ADJ_15D: monto de ventas suavizadas y ajustadas con la tendencia de los últimos 15 días anteriores al otorgamiento del crédito

72. MONTHLY_TPV_LC: promedio de ventas de los últimos 6 meses sin el máximo ni el mínimo anteriores al otorgamiento del crédito

73. WEEKLY_TPV_LC: promedio de ventas semanales de los últimos 6 meses sin el máximo ni el mínimo anteriores al otorgamiento del crédito

74. TPV_M1_L3: variación de ventas entre el mes de otorgamiento del crédito y los últimos 3 meses

75. TPV_M1_M2: variación de ventas entre el mes de otorgamiento del crédito y el mes anterior

76. TPV_15D_M1: variación de ventas entre los últimos 15 días y el mes completo anterior al otorgamiento del crédito

77. TPV_7D_M1: variación de ventas entre los últimos 7 días y el mes completo anterior al otorgamiento del crédito

78. TPV_7D_15D: variación de ventas entre los últimos 7 y 15 días anteriores al otorgamiento del crédito

79. TPN_M1_L3: variación entre la cantidad de ventas en el mes de otorgamiento del crédito y los 3 meses anteriores

80. TPN_M1_M2: variación de la cantidad de ventas entre el mes de otorgamiento del crédito y el mes anterior

81. TPN_15D_M1: variación en la cantidad de ventas entre los últimos 15 días y el mes completo anterior al otorgamiento del crédito

82. TPN_7D_M1: variación en la cantidad de ventas entre los últimos 7 días y el mes completo anterior al otorgamiento del crédito

83. TPN_7D_15D: variación en la cantidad de ventas entre los últimos 7 y 15 días anteriores al otorgamiento del crédito

84. FLAG_CHURNEO_EVER: indica si el cliente alguna vez dejó de vender históricamente

85. WEEKLY_MULTIPLIER: multiplicador que se aplica a las ventas semanales

86. RCI_TOTAL: relación cuota ingreso

87. CC_INST_AVG_AMOUNT: promedio ponderado por monto de las cuotas remanentes de los productos de consumer

88. RCI_TOTAL_CON_CONSUMERS: relación cuota ingreso incluyendo los productos de consumer

89. TAG_OFERTA_DE_1ST_RULE: indica si es un crédito con política flexible
90. TAG_OFERTA_DE_2ND_RULE: indica si es un crédito con política flexible categoría 2
91. TAG_OFERTA_DE_REGLA_BAU: indica si es un crédito con política habitual
92. TIPO_PERSONA_PF: indica si es persona física
93. TIPO_PERSONA_PJ: indica si es persona jurídica
94. TIPO_PERSONA_nan: indica si no tiene el dato de persona física o jurídica
95. TIPO_DEVICE_OTRO: indica si el cliente tiene dispositivos no habituales
96. TIPO_DEVICE_POINT_MINI_H_BLUE: indica si el cliente tiene un dispositivo del tipo mini, H, blue
97. TIPO_DEVICE_POINT_SMART: indica si el client tiene el dispositivo point smart
98. TIPO_DEVICE_nan: indica si no está el dato del tipo de dispositivo
99. FLAG_PPV_ACTIVE_CRED_0.0: indica si el cliente no tiene activo un producto PPV
100. FLAG_PPV_ACTIVE_CRED_1.0: indica si el cliente tiene activo un producto PPV
101. FLAG_PPV_ACTIVE_CRED_nan: indica si no se tiene el dato del producto PPV
102. KILLER_TRAVAS_RECIVIBLES_0.0: indica si el cliente no tiene travados sus ingresos futuros (es una herramienta que se puede utilizar en Brasil)
103. KILLER_TRAVAS_RECIVIBLES_1.0: indica si el cliente tiene travados sus ingresos futuros (es una herramienta que se puede utilizar en Brasil)
104. KILLER_TRAVAS_RECIVIBLES_nan: indica si si no se tiene el dato de travas recibibles
105. TAG_TIPO_BORROWER_DE_NEW: indica si el crédito que sacó el cliente se corresponde con su primer oferta de crédito en este producto
106. TAG_TIPO_BORROWER_DE_OLD: indica que el cliente sacó un crédito después de mucho tiempo de no haber sacado
107. TAG_TIPO_BORROWER_DE_REPEAT: indica que el cliente viene tomando varios créditos
108. TAG_TIPO_BORROWER_DE_nan: indica que no se tiene el dato de si la propuesta es new, old o repeat

109. TAG_POINT_POINT: indica que el cliente tiene un dispositivo point

110. TAG_POINT_POINT_PRO: indica el cliente tiene un dispositivo point pro

111. TAG_POINT_QR: indica que el cliente tiene un dispositivo QR

112. TIPO_DEVICE_POINT_I_STANDALONE: indica que el cliente tiene un dispositivo point standalone

113. TAG_POINT_nan: indica que no se tiene el dato del tipo de dispositivo que tiene el cliente

114. CC_CREDIT_LINE_AMT: monto de la línea de los productos de consumo

115. COMBIN_CC_RAT_CREDIT_LINE_USE: variable combinada entre FLAG_LINE_CC_USE y el porcentaje del uso de la línea de consumo en el mes de otorgamiento del cédito

116. COMBIN_CC_RAT_CREDIT_LINE_USE_1M: variable combinada entre
FLAG_LINE_CC_USE_1M y el porcentaje del uso de la línea de consumo en el mes anterior al de otorgamiento del cédito

117. COMBIN_CC_RAT_CREDIT_LINE_USE_3M: variable combinada entre
FLAG_LINE_CC_USE_3M y el porcentaje del uso de la línea de consumo en los 3 meses anteriores al de otorgamiento del cédito

118. COMBIN_CC_RAT_CREDIT_LINE_USE_6M: variable combinada entre
FLAG_LINE_CC_USE_6M y el porcentaje del uso de la línea de consumo en los 6 meses anteriores al de otorgamiento del cédito

119. COMBIN_CC_RAT_CREDIT_LINE_USE_12M: variable combinada entre
FLAG_LINE_CC_USE_12M y el porcentaje del uso de la línea de consumo en los 12 meses anteriores al de otorgamiento del cédito

120. COMBIN_CC_DIF_CREDIT_LINE_USE_VS_1M: variable combinada entre FLAG_LINE_CC_USE_vs_1 y la variación entre el ratio de uso entre el mes de otorgamiento del crédito y el mes anterior

121. COMBIN_CC_DIF_CREDIT_LINE_USE_VS_3M: variable combinada entre FLAG_LINE_CC_USE_vs_3 y la variación entre el ratio de uso entre el mes de otorgamiento del crédito y los tres meses anteriores

122. COMBIN_CC_DIF_CREDIT_LINE_USE_VS_6M: variable combinada entre FLAG_LINE_CC_USE_vs_6 y la variación entre el ratio de uso entre el mes de otorgamiento del crédito y los seis meses anteriores

123. COMBIN_CC_DIF_CREDIT_LINE_USE_VS_12M: variable combinada entre FLAG_LINE_CC_USE_vs_12 y la variación entre el ratio de uso entre el mes de otorgamiento del crédito y los doce meses anteriores

124. COMBIN_CC_DAYS_SINCE_LAST_UPSELL: variable combinada entre FLAG_UPSELL y la cantidad de días que pasaron desde el último upsell

125. COMBIN_CC_LAST_UPSELL_RAT_AMT: variable combinada entre FLAG_UPSELL y el monto del incremento del último upsell

126. COMBIN_CC_UPSELL_MIN_RAT_AMT: variable combinada entre FLAG_UPSELL y el mínimo incremento que se le hizo a la línea de consumo

127. COMBIN_CC_UPSELL_AVG_RAT_AMT: variable combinada entre FLAG_UPSELL y el máximo incremento que se le hizo a la línea de consumo

128. COMBIN_CC_UPSELL_MAX_RAT_AMT: variable combinada entre FLAG_UPSELL y el promedio de los incrementos que se le hicieron a la línea de consumo

129. COMBIN_CC_DAYS_SINCE_LAST_DOWNSELL: variable combinada entre FLAG_DOWNSELL y la cantidad de días que pasaron desde el último downsell

130. COMBIN_CC_LAST_DOWNSELL_RAT_AMT: variable combinada entre FLAG_DOWNSELL y el monto del incremento del último downsell

131. COMBIN_CC_DOWNSELL_MIN_RAT_AMT: variable combinada entre FLAG_DOWNSELL y el mínimo decremento que se le hizo a la línea de consumo

132. COMBIN_CC_DOWNSELL_AVG_RAT_AMT: variable combinada entre FLAG_DOWNSELL y el promedio de los decrementos que se le hicieron a la línea de consumo

133. COMBIN_CC_DOWNSELL_MAX_RAT_AMT: variable combinada entre FLAG_DOWNSELL y el máximo decremento que se le hizo a la línea de consumo

134. COMBIN_CC_MAX_QTY_DAYS_PAY_DELAY: variable combinada entre FLAG_LINE_CC y el día de atraso máximo en su línea de consumo en el mes de otorgamiento del crédito

135. COMBIN_CC_MAX_QTY_DAYS_PAY_DELAY_1M: variable combinada entre FLAG_LINE_CC_1M y el día de atraso máximo en su línea de consumo en el mes anterior al del otorgamiento del crédito

136. COMBIN_CC_MAX_QTY_DAYS_PAY_DELAY_3M: variable combinada entre FLAG_LINE_CC_3M y el día de atraso máximo en su línea de consumo en los 3 meses anteriores al del otorgamiento del crédito

137. COMBIN_CC_MAX_QTY_DAYS_PAY_DELAY_6M: variable combinada entre FLAG_LINE_CC_6M y el día de atraso máximo en su línea de consumo en los 6 meses anteriores al del otorgamiento del crédito

138. COMBIN_CC_MAX_QTY_DAYS_PAY_DELAY_12M: variable combinada entre FLAG_LINE_CC_12M y el día de atraso máximo en su línea de consumo en los 12 meses anteriores al del otorgamiento del crédito

139. COMBIN_CC_RAT_AMT_OVERDUE: variable combinada entre FLAG_LINE_CC y el el ratio de su deuda en mora en el mes de otorgamiento del crédito

140. COMBIN_CC_RAT_AMT_OVERDUE_1M: variable combinada entre FLAG_LINE_CC_1M y el el ratio de su deuda en mora en el mes anterior al del otorgamiento del crédito

141. COMBIN_CC_RAT_AMT_OVERDUE_3M: variable combinada entre FLAG_LINE_CC_3M y el el ratio de su deuda en mora en los 3 meses anteriores al del otorgamiento del crédito

142. COMBIN_CC_RAT_AMT_OVERDUE_6M: variable combinada entre FLAG_LINE_CC_6M y el el ratio de su deuda en mora en los 3 meses anteriores al del otorgamiento del crédito

143. COMBIN_CC_RAT_AMT_OVERDUE_12M: variable combinada entre FLAG_LINE_CC_12M y el el ratio de su deuda en mora en los 12 meses anteriores al del otorgamiento del crédito

144. COMBIN_CC_RAT_AMT_OVERDUE_5_10_DPD: variable combinada entre FLAG_LINE_CC y el ratio de deuda con atraso entre 5 y 10 días de mora en el mes de otorgamiento del crédito

145. COMBIN_CC_RAT_AMT_OVERDUE_5_10_DPD_1M: variable combinada entre FLAG_LINE_CC_1M y el ratio de deuda con atraso entre 5 y 10 días de mora en el mes anterior al de otorgamiento del crédito

146. COMBIN_CC_RAT_AMT_OVERDUE_5_10_DPD_3M: variable combinada entre FLAG_LINE_CC_3M y el ratio de deuda con atraso entre 5 y 10 días de mora en los 3 meses anteriores al de otorgamiento del crédito

147. COMBIN_CC_RAT_AMT_OVERDUE_5_10_DPD_6M: variable combinada entre FLAG_LINE_CC_6M y el ratio de deuda con atraso entre 5 y 10 días de mora en los 6 meses anteriores al de otorgamiento del crédito

148. COMBIN_CC_RAT_AMT_OVERDUE_5_10_DPD_12M: variable combinada entre FLAG_LINE_CC_12M y el ratio de deuda con atraso entre 5 y 10 días de mora en los 12 meses anteriores al de otorgamiento del crédito

149. COMBIN_CC_RAT_AMT_OVERDUE_10_20_DPD: variable combinada entre FLAG_LINE_CC y el ratio de deuda con atraso entre 10 y 20 días de mora en el mes de otorgamiento del crédito

150. COMBIN_CC_RAT_AMT_OVERDUE_10_20_DPD_1M: variable combinada entre FLAG_LINE_CC_1M y el ratio de deuda con atraso entre 10 y 20 días de mora en el mes anterior al de otorgamiento del crédito

151. COMBIN_CC_RAT_AMT_OVERDUE_10_20_DPD_3M: variable combinada entre FLAG_LINE_CC_3M y el ratio de deuda con atraso entre 10 y 20 días de mora en los 3 meses anteriores al de otorgamiento del crédito

152. COMBIN_CC_RAT_AMT_OVERDUE_10_20_DPD_6M: variable combinada entre FLAG_LINE_CC_6M y el ratio de deuda con atraso entre 10 y 20 días de mora en los 6 meses anteriores al de otorgamiento del crédito

153. COMBIN_CC_RAT_AMT_OVERDUE_10_20_DPD_12M: variable combinada entre FLAG_LINE_CC_12M y el ratio de deuda con atraso entre 10 y 20 días de mora en los 12 meses anteriores al de otorgamiento del crédito

154. COMBIN_CC_RAT_AMT_OVERDUE_20_30_DPD: variable combinada entre FLAG_LINE_CC y el ratio de deuda con atraso entre 20 y 30 días de mora en el mes de otorgamiento del crédito

155. COMBIN_CC_RAT_AMT_OVERDUE_20_30_DPD_1M: variable combinada entre FLAG_LINE_CC_1M y el ratio de deuda con atraso entre 20 y 30 días de mora en el mes anterior al de otorgamiento del crédito

156. COMBIN_CC_RAT_AMT_OVERDUE_20_30_DPD_3M: variable combinada entre FLAG_LINE_CC_3M y el ratio de deuda con atraso entre 20 y 30 días de mora en los 3 meses anteriores al de otorgamiento del crédito

157. COMBIN_CC_RAT_AMT_OVERDUE_20_30_DPD_6M: variable combinada entre FLAG_LINE_CC_6M y el ratio de deuda con atraso entre 20 y 30 días de mora en los 6 meses anteriores al de otorgamiento del crédito

158. COMBIN_CC_RAT_AMT_OVERDUE_20_30_DPD_12M: variable combinada entre FLAG_LINE_CC_12M y el ratio de deuda con atraso entre 20 y 30 días de mora en los 12 meses anteriores al de otorgamiento del crédito

159. COMBIN_CC_RAT_AMT_OVERDUE_OVER_30_DPD: variable combinada entre FLAG_LINE_CC y el ratio de deuda con atraso mayor a 30 días de mora en el mes de otorgamiento del crédito

160. COMBIN_CC_RAT_AMT_OVERDUE_OVER_30_DPD_1M: variable combinada entre FLAG_LINE_CC_1M y el ratio de deuda con mayor a 30 días de mora en el mes anterior al de otorgamiento del crédito

161. COMBIN_CC_RAT_AMT_OVERDUE_OVER_30_DPD_3M: variable combinada entre FLAG_LINE_CC_3M y el ratio de deuda con atraso mayor a 30 días de mora en los 3 meses anteriores al de otorgamiento del crédito

162. COMBIN_CC_RAT_AMT_OVERDUE_OVER_30_DPD_6M: variable combinada entre FLAG_LINE_CC_6M y el ratio de deuda con atraso mayor a 30 días de mora en los 6 meses anteriores al de otorgamiento del crédito

163. COMBIN_CC_RAT_AMT_OVERDUE_OVER_30_DPD_12M: variable combinada entre FLAG_LINE_CC_12M y el ratio de deuda con atraso mayor a 30 días de mora en los 12 meses anteriores al de otorgamiento del crédito

164. COMBIN_CC_RAT_INST_QTY_OVERDUE: variable combinada entre
FLAG_LINE_CC y el el ratio de cuotas en mora en el mes de otorgamiento del crédito

165. COMBIN_CC_RAT_INST_QTY_OVERDUE_1M: variable combinada entre FLAG_LINE_CC_1M y el el ratio de cuotas en mora en el mes anterior al del otorgamiento del crédito

166. COMBIN_CC_RAT_INST_QTY_OVERDUE_3M: variable combinada entre FLAG_LINE_CC_3M y el el ratio de cuotas en mora en los 3 meses anteriores al del otorgamiento del crédito

167. COMBIN_CC_RAT_INST_QTY_OVERDUE_6M: variable combinada entre FLAG_LINE_CC_6M y el el ratio de cuotas en mora en los 3 meses anteriores al del otorgamiento del crédito

168. COMBIN_CC_RAT_INST_QTY_OVERDUE_12M: variable combinada entre FLAG_LINE_CC_12M y el el ratio de cuotas en mora en los 12 meses anteriores al del otorgamiento del crédito

169. COMBIN_CC_RAT_INST_QTY_OVERDUE_5_10_DPD: variable combinada entre FLAG_LINE_CC y el ratio de cuotas con atraso entre 5 y 10 días de mora en el mes de otorgamiento del crédito

170. COMBIN_CC_RAT_INST_QTY_OVERDUE_5_10_DPD_1M: variable combinada entre FLAG_LINE_CC_1M y el ratio de cuotas con atraso entre 5 y 10 días de mora en el mes anterior al de otorgamiento del crédito

171. COMBIN_CC_RAT_INST_QTY_OVERDUE_5_10_DPD_3M: variable combinada entre FLAG_LINE_CC_3M y el ratio de cuotas con atraso entre 5 y 10 días de mora en los 3 meses anteriores al de otorgamiento del crédito

172. COMBIN_CC_RAT_INST_QTY_OVERDUE_5_10_DPD_6M: variable combinada entre FLAG_LINE_CC_6M y el ratio de cuotas con atraso entre 5 y 10 días de mora en los 6 meses anteriores al de otorgamiento del crédito

173. COMBIN_CC_RAT_INST_QTY_OVERDUE_5_10_DPD_12M: variable combinada entre FLAG_LINE_CC_12M y el ratio de cuotas con atraso entre 5 y 10 días de mora en los 12 meses anteriores al de otorgamiento del crédito

174. COMBIN_CC_RAT_INST_QTY_OVERDUE_10_20_DPD: variable combinada entre FLAG_LINE_CC y el ratio de cuotas con atraso entre 10 y 20 días de mora en el mes de otorgamiento del crédito

175. COMBIN_CC_RAT_INST_QTY_OVERDUE_10_20_DPD_1M: variable combinada entre FLAG_LINE_CC_1M y el ratio de cuotas con atraso entre 10 y 20 días de mora en el mes anterior al de otorgamiento del crédito

176. COMBIN_CC_RAT_INST_QTY_OVERDUE_10_20_DPD_3M: variable combinada entre FLAG_LINE_CC_3M y el ratio de cuotas con atraso entre 10 y 20 días de mora en los 3 meses anteriores al de otorgamiento del crédito

177. COMBIN_CC_RAT_INST_QTY_OVERDUE_10_20_DPD_6M: variable combinada entre FLAG_LINE_CC_6M y el ratio de cuotas con atraso entre 10 y 20 días de mora en los 6 meses anteriores al de otorgamiento del crédito

178. COMBIN_CC_RAT_INST_QTY_OVERDUE_10_20_DPD_12M: variable combinada entre FLAG_LINE_CC_12M y el ratio de cuotas con atraso entre 10 y 20 días de mora en los 12 meses anteriores al de otorgamiento del crédito

179. COMBIN_CC_RAT_INST_QTY_OVERDUE_20_30_DPD: variable combinada entre FLAG_LINE_CC y el ratio de cuotas con atraso entre 20 y 30 días de mora en el mes de otorgamiento del crédito

180. COMBIN_CC_RAT_INST_QTY_OVERDUE_20_30_DPD_1M: variable combinada entre FLAG_LINE_CC_1M y el ratio de cuotas con atraso entre 20 y 30 días de mora en el mes anterior al de otorgamiento del crédito

181. COMBIN_CC_RAT_INST_QTY_OVERDUE_20_30_DPD_3M: variable combinada entre FLAG_LINE_CC_3M y el ratio de cuotas con atraso entre 20 y 30 días de mora en los 3 meses anteriores al de otorgamiento del crédito

182. COMBIN_CC_RAT_INST_QTY_OVERDUE_20_30_DPD_6M: variable combinada entre FLAG_LINE_CC_6M y el ratio de cuotas con atraso entre 20 y 30 días de mora en los 6 meses anteriores al de otorgamiento del crédito

183. COMBIN_CC_RAT_INST_QTY_OVERDUE_20_30_DPD_12M: variable combinada entre FLAG_LINE_CC_12M y el ratio de cuotas con atraso entre 20 y 30 días de mora en los 12 meses anteriores al de otorgamiento del crédito

184. COMBIN_CC_RAT_INST_QTY_OVERDUE_OVER_30_DPD: variable combinada entre FLAG_LINE_CC y el ratio de cuotas con atraso mayor a 30 días de mora en el mes de otorgamiento del crédito

185. COMBIN_CC_RAT_INST_QTY_OVERDUE_OVER_30_DPD_1M: variable combinada entre FLAG_LINE_CC_1M y el ratio de cuotas con mayor a 30 días de mora en el mes anterior al de otorgamiento del crédito

186. COMBIN_CC_RAT_INST_QTY_OVERDUE_OVER_30_DPD_3M: variable combinada entre FLAG_LINE_CC_3M y el ratio de cuotas con atraso mayor a 30 días de mora en los 3 meses anteriores al de otorgamiento del crédito

187. COMBIN_CC_RAT_INST_QTY_OVERDUE_OVER_30_DPD_6M: variable combinada entre FLAG_LINE_CC_6M y el ratio de cuotas con atraso mayor a 30 días de mora en los 6 meses anteriores al de otorgamiento del crédito

188. COMBIN_CC_RAT_INST_QTY_OVERDUE_OVER_30_DPD_12M: variable combinada entre FLAG_LINE_CC_12M y el ratio de cuotas con atraso mayor a 30 días de mora en los 12 meses anteriores al de otorgamiento del crédito

189. FLAG_FDV_0.0: indica si el cliente no es de fuerza de venta

190. FLAG_FDV_1.0: indica si el cliente es de fuerza de venta

191. FLAG_FDV_nan: no se tiene el dato que indique si el cliente es o no de fuerza de venta

APÉNDICE B

Código de Python

```
import os

import pandas as pd
from pandas import DataFrame
pd.set_option('display.max_rows', 500)
pd.set_option('display.max_columns', 500)
import numpy as np
import numpy.ma as ma
import pickle

from sklearn import preprocessing, metrics
from sklearn.metrics import make_scorer, accuracy_score, confusion_matrix,
    roc_curve, auc, recall_score, classification_report,fbeta_score,
    precision_recall_curve, roc_auc_score
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn import tree

import matplotlib.pyplot as plt
import seaborn as sns
from numpy import random

pd.set_option('display.max_rows', 500)
pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 500)

import getpass as gp
from matplotlib.axes import Axes
from matplotlib import pyplot as plt, font_manager as fm
import os


from sqlalchemy.dialects import registry

#Levantamos la base de datos

!pip install -U -q PyDrive
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials

auth.authenticate_user()
gauth = GoogleAuth()
```

```
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)

from google.colab import drive
drive.mount('/content/drive')

data_in=pd.read_csv('/content/drive/MyDrive/Tesis/BASE_IN_TIME_MLB_INSTORE_EO.
    csv')
data_in.shape

data_out=pd.read_csv('/content/drive/MyDrive/Tesis/BASE_OOT_MLB_INSTORE_EO.csv
    ')
data_out.shape

data_in = data_in[data_in.FLAG_DEFAULT_MES.notnull()]

data_in = data_in[data_in.CC_INST_AVG_AMOUNT.notnull()]

#Gr ficos univariados

data_in['pto_obs'] = pd.to_datetime(data_in['CRD_PROP_CREATION_DATE_ID']).
    astype(str).str[:7]

data_in['pto_obs'] = data_in['pto_obs'].replace({'-':''}, regex=True)

import seaborn as sns
data_plot=data_in[['pto_obs','MAX_AMOUNT_DE']]
sns.set(rc={"figure.figsize":(10, 8)})
sns.boxplot(data=data_plot, x='pto_obs', y='MAX_AMOUNT_DE',order=["202105", "
    202106","202107","202108","202109","202110","202111","202112","202201",
"202202", "202204"])

bins_amount=range(100,15000,100)

# Creating histogram
fig, axs = plt.subplots(1, 1,
                        figsize =(7, 5),
                        tight_layout = True)

# Creating histogram
N, bins, patches = axs.hist(data_in['MAX_AMOUNT_DE'], bins = bins_amount)

# Adding extra features
plt.xlabel("Monto")
plt.ylabel("Cantidad de cr ditos")
plt.legend(legend)
plt.title('MONTO DEL CR DITO \n HISTOGRAMA')

#Show plot
plt.show()

#Accionables univariado
data_in.groupby(['pto_obs'])['FLAG_DEFAULT_MES'].mean()

#Elimino per odo con BR muy distinta
data_in = data_in[data_in['pto_obs'] != '202105']

#Elimino variables con todo null
data_in.drop(columns = ['PROB_BHV_2L', 'FLAG_MONOTRIBUTO', 'PLAZO_L2','
    OLD_Q_DAYS_DE','SEMAFORO'], inplace = True)
data_out.drop(columns = ['PROB_BHV_2L', 'FLAG_MONOTRIBUTO', 'PLAZO_L2','
    OLD_Q_DAYS_DE','SEMAFORO'], inplace = True)
```

```
# encuentro columnas que tienen el mismo valor en
# todas las filas
n_valores_unicos = data_in.nunique()
valor_unico = n_valores_unicos[n_valores_unicos == 1]
valor_unico

# elimino columnas con un unico valor
data_in = data_in.drop(columns = valor_unico.index)

#An lisis bivariado

data_in['MAX_ATRASO_CRED_DE_BIN'] =np.where(data_in['MAX_ATRASO_CRED_DE']<=-1,
    '(-27 , -1]',
                                    np.where((data_in['MAX_ATRASO_CRED_DE'
                                        ]>-1)&(data_in['MAX_ATRASO_CRED_DE'
                                        ]<=0),'(-1 , 0]',
                                    np.where((data_in['MAX_ATRASO_CRED_DE'
                                        ]>0)&(data_in['MAX_ATRASO_CRED_DE'
                                        ]<=2),'(0 , 2]',
                                    np.where((data_in['MAX_ATRASO_CRED_DE'
                                        ]>2)&(data_in['MAX_ATRASO_CRED_DE'
                                        ]<=12),'(2 , 12]',
                                    np.where((data_in['MAX_ATRASO_CRED_DE'
                                        ]>12)&(data_in['MAX_ATRASO_CRED_DE'
                                        ]<=1000),'(12 , 1000]',None)))))

biva_1=data_in.groupby('MAX_ATRASO_CRED_DE_BIN')['FLAG_DEFAULT_MES'].agg(['
    mean','sum','count'])
biva_1['% Participacin']=biva_1['count']/biva_1['count'].sum()
biva_1=biva_1.reset_index().sort_values(by='mean')

#Create combo chart
fig, ax1 = plt.subplots(figsize=(7,4))
color = 'tab:green'
#bar plot creation
ax1.set_title('Bivariado MAX ATRASO - Incumplimiento', fontsize=16)
ax1 = sns.barplot(x='MAX_ATRASO_CRED_DE_BIN', y='% Participacin', data =
    biva_1, palette='coolwarm')
ax1.tick_params(axis='y')
#specify we want to share the same x-axis
ax2 = ax1.twinx()
color = 'tab:red'
#line plot creation
ax2.set_ylabel('Tasa de incumplimiento', fontsize=10)
ax2 = sns.lineplot(x= biva_1['MAX_ATRASO_CRED_DE_BIN'].astype(str), y=biva_1['
    mean'],marker='o')
#for x, y in zip(biva_1['SCORE_CHURN_BIN'], biva_1['mean']):
#    ax2.text(x, y,f'{y:.3f}', color="Black")
ax2.tick_params(axis='y', color=color)
plt.legend(loc='upper left', labels=['Tasa de incumplimiento'])
#show plot
plt.show()

data_in['Q_REPEAT_BIN'] =np.where(data_in['Q_REPEAT']<=1,'(0 , 1]',
                                    np.where((data_in['Q_REPEAT']>1)&(data_in
                                        ['Q_REPEAT']<=2),'(1 , 2]',
                                    np.where((data_in['Q_REPEAT']>2)&(data_in
                                        ['Q_REPEAT']<=3),'(2 , 3]',
                                    np.where((data_in['Q_REPEAT']>3)&(data_in
                                        ['Q_REPEAT']<=6),'(3 , 6]',
```

```
                                        np.where((data_in['Q_REPEAT']>6)&(data_in
                                            ['Q_REPEAT']<=100),'(6 , 100]',None))
                                            )))

biva_1=data_in.groupby('Q_REPEAT_BIN')['FLAG_DEFAULT_MES'].agg(['mean','sum','
    count'])
biva_1['% Participacion']=biva_1['count']/biva_1['count'].sum()
biva_1=biva_1.reset_index()

#Create combo chart
fig, ax1 = plt.subplots(figsize=(7,4))
color = 'tab:green'
#bar plot creation
ax1.set_title('Bivariado CANTIDAD DE CREDITOS PREVIOS - Incumplimiento',
    fontsize=16)
ax1 = sns.barplot(x='Q_REPEAT_BIN', y='% Participacion', data = biva_1,
    palette='coolwarm')
ax1.tick_params(axis='y')
#specify we want to share the same x-axis
ax2 = ax1.twinx()
color = 'tab:red'
#line plot creation
ax2.set_ylabel('Tasa de incumplimiento', fontsize=10)
ax2 = sns.lineplot(x= biva_1['Q_REPEAT_BIN'].astype(str), y=biva_1['mean'],
    marker='o')
#for x, y in zip(biva_1['SCORE_CHURN_BIN'], biva_1['mean']):
#    ax2.text(x, y,f'{y:.3f}', color="Black")
ax2.tick_params(axis='y', color=color)
plt.legend(loc='upper left', labels=['Tasa de incumplimiento'])
#show plot

biva_1=training_set_dummies.groupby('TAG_TIPO_BORROWER_DE_REPEAT')['
    FLAG_DEFAULT_MES'].agg(['mean','sum','count'])
biva_1['% Participacion']=biva_1['count']/biva_1['count'].sum()
biva_1=biva_1.reset_index().sort_values(by='mean')

#Create combo chart
fig, ax1 = plt.subplots(figsize=(7,4))
color = 'tab:green'
#bar plot creation
ax1.set_title('Bivariado TUVO OTROS CREDITOS O NO - Incumplimiento', fontsize
    =16)
ax1 = sns.barplot(x='TAG_TIPO_BORROWER_DE_REPEAT', y='% Participacion', data
    = biva_1, palette='coolwarm')
ax1.tick_params(axis='y')
#specify we want to share the same x-axis
ax2 = ax1.twinx()
color = 'tab:red'
#line plot creation
ax2.set_ylabel('Tasa de incumplimiento', fontsize=10)
ax2 = sns.lineplot(x= biva_1['TAG_TIPO_BORROWER_DE_REPEAT'].astype(str), y=
    biva_1['mean'],marker='o')
#for x, y in zip(biva_1['SCORE_CHURN_BIN'], biva_1['mean']):
#    ax2.text(x, y,f'{y:.3f}', color="Black")
ax2.tick_params(axis='y', color=color)
ax2.set_ylim([0.05, 0.15])
plt.legend(loc='upper left', labels=['Tasa de incumplimiento'])
#show plot
plt.show()

biva_1=training_set_dummies.groupby('TAG_TIPO_BORROWER_DE_NEW')['
    FLAG_DEFAULT_MES'].agg(['mean','sum','count'])
```

```
biva_1['% Participacion']=biva_1['count']/biva_1['count'].sum()
biva_1=biva_1.reset_index().sort_values(by='mean')

#Create combo chart
fig, ax1 = plt.subplots(figsize=(7,4))
color = 'tab:green'
#bar plot creation
ax1.set_title('Bivariado ES O NO LA PRIMER PROPUESTA - Incumplimiento',
     fontsize=16)
ax1 = sns.barplot(x='TAG_TIPO_BORROWER_DE_NEW', y='% Participacion', data =
     biva_1, palette='coolwarm')
ax1.tick_params(axis='y')
#specify we want to share the same x-axis
ax2 = ax1.twinx()
color = 'tab:red'
#line plot creation
ax2.set_ylabel('Tasa de incumplimiento', fontsize=10)
ax2 = sns.lineplot(x= biva_1['TAG_TIPO_BORROWER_DE_NEW'].astype(str), y=biva_1
     ['mean'],marker='o')
#for x, y in zip(biva_1['SCORE_CHURN_BIN'], biva_1['mean']):
#    ax2.text(x, y,f'{y:.3f}', color="Black")
ax2.tick_params(axis='y', color=color)
ax2.set_ylim([0.05, 0.15])
plt.legend(loc='upper left', labels=['Tasa de incumplimiento'])
#show plot
plt.show()

training_set_dummies['COMBIN_PROB_BHV_1L_BIN'] =  pd.qcut(training_set_dummies
     ['COMBIN_PROB_BHV_1L'], 5)

biva_1=training_set_dummies.groupby('COMBIN_PROB_BHV_1L_BIN')['
     FLAG_DEFAULT_MES'].agg(['mean','sum','count'])
biva_1['% Participacion']=biva_1['count']/biva_1['count'].sum()
biva_1=biva_1.reset_index().sort_values(by='mean')

#Create combo chart
fig, ax1 = plt.subplots(figsize=(7.5,4))
color = 'tab:green'
#bar plot creation
ax1.set_title('Bivariado SCORE BHV - Incumplimiento', fontsize=16)
ax1 = sns.barplot(x='COMBIN_PROB_BHV_1L_BIN', y='% Participacion', data =
     biva_1, palette='coolwarm')
ax1.tick_params(axis='y')
#specify we want to share the same x-axis
ax2 = ax1.twinx()
color = 'tab:red'
#line plot creation
ax2.set_ylabel('Tasa de incumplimiento', fontsize=10)
ax2 = sns.lineplot(x= biva_1['COMBIN_PROB_BHV_1L_BIN'].astype(str), y=biva_1['
     mean'],marker='o')
#for x, y in zip(biva_1['SCORE_CHURN_BIN'], biva_1['mean']):
#    ax2.text(x, y,f'{y:.3f}', color="Black")
ax2.tick_params(axis='y', color=color)
plt.legend(loc='upper left', labels=['Tasa de incumplimiento'])
#show plot
plt.show()

data_in['RCI_TOTAL_BIN'] =  pd.qcut(data_in['RCI_TOTAL'], 5)

biva_1=data_in.groupby('RCI_TOTAL_BIN')['FLAG_DEFAULT_MES'].agg(['mean','sum',
     'count'])
biva_1['% Participacion']=biva_1['count']/biva_1['count'].sum()
```

```
biva_1=biva_1.reset_index().sort_values(by='mean')

#Create combo chart
fig, ax1 = plt.subplots(figsize=(7.5,4))
color = 'tab:green'
#bar plot creation
ax1.set_title('Bivariado RCI TOTAL - Incumplimiento', fontsize=16)
ax1 = sns.barplot(x='RCI_TOTAL_BIN', y='% Participacion', data = biva_1,
    palette='coolwarm')
ax1.tick_params(axis='y')
#specify we want to share the same x-axis
ax2 = ax1.twinx()
color = 'tab:red'
#line plot creation
ax2.set_ylabel('Tasa de incumplimiento', fontsize=10)
ax2 = sns.lineplot(x= biva_1['RCI_TOTAL_BIN'].astype(str), y=biva_1['mean'],
    marker='o')
#for x, y in zip(biva_1['SCORE_CHURN_BIN'], biva_1['mean']):
#    ax2.text(x, y,f'{y:.3f}', color="Black")
ax2.tick_params(axis='y', color=color)
plt.legend(loc='upper left', labels=['Tasa de incumplimiento'])
#show plot
plt.show()

data_in['SCORE_CHURN_BIN'] =np.where(data_in['SCORE_CHURN']<=0.13,'(0 , 0.13]'
    ,
                                np.where((data_in['SCORE_CHURN']>0.13)&(
                                    data_in['SCORE_CHURN']<=0.18),'(0.13
                                    , 0.18]',
                                np.where((data_in['SCORE_CHURN']>0.18)&(
                                    data_in['SCORE_CHURN']<=0.24),'(0.18
                                    , 0.24]',
                                np.where((data_in['SCORE_CHURN']>0.24)&(
                                    data_in['SCORE_CHURN']<=0.32),'(0.24
                                    , 0.32]',
                                np.where((data_in['SCORE_CHURN']>0.32)&(
                                    data_in['SCORE_CHURN']<=1),'(0.32 ,
                                    1]',None)))))

biva_1=data_in.groupby('SCORE_CHURN_BIN')['FLAG_DEFAULT_MES'].agg(['mean','sum
    ','count'])
biva_1['% Participacion']=biva_1['count']/biva_1['count'].sum()
biva_1=biva_1.reset_index().sort_values(by='mean')

#Create combo chart
fig, ax1 = plt.subplots(figsize=(7,4))
color = 'tab:green'
#bar plot creation
ax1.set_title('Bivariado Score Churn - Incumplimiento', fontsize=16)
ax1 = sns.barplot(x='SCORE_CHURN_BIN', y='% Participacion', data = biva_1,
    palette='coolwarm')
ax1.tick_params(axis='y')
#specify we want to share the same x-axis
ax2 = ax1.twinx()
color = 'tab:red'
#line plot creation
ax2.set_ylabel('Tasa de incumplimiento', fontsize=10)
ax2 = sns.lineplot(x= biva_1['SCORE_CHURN_BIN'].astype(str), y=biva_1['mean'],
    marker='o')
#for x, y in zip(biva_1['SCORE_CHURN_BIN'], biva_1['mean']):
#    ax2.text(x, y,f'{y:.3f}', color="Black")
ax2.tick_params(axis='y', color=color)
```

```
plt.legend(loc='upper left', labels=['Tasa de incumplimiento'])
#show plot
plt.show()

data_in['RCI_TOTAL_CON_CONSUMERS_BIN'] =  pd.qcut(data_in['
    RCI_TOTAL_CON_CONSUMERS'], 5)

biva_1=data_in.groupby('RCI_TOTAL_CON_CONSUMERS_BIN')['FLAG_DEFAULT_MES'].agg
    (['mean','sum','count'])
biva_1['% Participaci n']=biva_1['count']/biva_1['count'].sum()
biva_1=biva_1.reset_index().sort_values(by='mean')

#Create combo chart
fig, ax1 = plt.subplots(figsize=(8.5,4))
color = 'tab:green'
#bar plot creation
ax1.set_title('Bivariado RCI TOTAL CON CONSUMERS - Incumplimiento', fontsize
    =16)
ax1 = sns.barplot(x='RCI_TOTAL_CON_CONSUMERS_BIN', y='% Participaci n', data
    = biva_1, palette='coolwarm')
ax1.tick_params(axis='y')
#specify we want to share the same x-axis
ax2 = ax1.twinx()
color = 'tab:red'
#line plot creation
ax2.set_ylabel('Tasa de incumplimiento', fontsize=10)
ax2 = sns.lineplot(x= biva_1['RCI_TOTAL_CON_CONSUMERS_BIN'].astype(str), y=
    biva_1['mean'],marker='o')
#for x, y in zip(biva_1['SCORE_CHURN_BIN'], biva_1['mean']):
#    ax2.text(x, y,f'{y:.3f}', color="Black")
ax2.tick_params(axis='y', color=color)
plt.legend(loc='upper left', labels=['Tasa de incumplimiento'])
#show plot
plt.show()

data_in['SC_BUREAU_BIN'] =  pd.qcut(data_in['SC_BUREAU'], 5)

biva_1=data_in.groupby('SC_BUREAU_BIN')['FLAG_DEFAULT_MES'].agg(['mean','sum',
    'count'])
biva_1['% Participaci n']=biva_1['count']/biva_1['count'].sum()
biva_1=biva_1.reset_index().sort_values(by='mean')

#Create combo chart
fig, ax1 = plt.subplots(figsize=(8.5,4))
color = 'tab:green'
#bar plot creation
ax1.set_title('Bivariado SCORE BUREAU - Incumplimiento', fontsize=16)
ax1 = sns.barplot(x='SC_BUREAU_BIN', y='% Participaci n', data = biva_1,
    palette='coolwarm')
ax1.tick_params(axis='y')
#specify we want to share the same x-axis
ax2 = ax1.twinx()
color = 'tab:red'
#line plot creation
ax2.set_ylabel('Tasa de incumplimiento', fontsize=10)
ax2 = sns.lineplot(x= biva_1['SC_BUREAU_BIN'].astype(str), y=biva_1['mean'],
    marker='o')
#for x, y in zip(biva_1['SCORE_CHURN_BIN'], biva_1['mean']):
#    ax2.text(x, y,f'{y:.3f}', color="Black")
ax2.tick_params(axis='y', color=color)
plt.legend(loc='upper left', labels=['Tasa de incumplimiento'])
#show plot
```

```
plt.show()

data_in['CC_CREDIT_LINE_AMT_BIN'] =np.where(data_in['CC_CREDIT_LINE_AMT']<=50,
     '(0 , 50]',
                                     np.where((data_in['CC_CREDIT_LINE_AMT'
                                          ]>50)&(data_in['CC_CREDIT_LINE_AMT'
                                          ]<=200),'(50 , 200]',
                                     np.where((data_in['CC_CREDIT_LINE_AMT'
                                          ]>200)&(data_in['CC_CREDIT_LINE_AMT'
                                          ]<=300),'(200 , 300]',
                                     np.where((data_in['CC_CREDIT_LINE_AMT'
                                          ]>300)&(data_in['CC_CREDIT_LINE_AMT'
                                          ]<=500),'(300 , 500]',
                                     np.where((data_in['CC_CREDIT_LINE_AMT'
                                          ]>500)&(data_in['CC_CREDIT_LINE_AMT'
                                          ]<=15000),'(500 , 15000]',None)))))

biva_1=data_in.groupby('CC_CREDIT_LINE_AMT_BIN')['FLAG_DEFAULT_MES'].agg(['
     mean','sum','count'])
biva_1['% Participacin']=biva_1['count']/biva_1['count'].sum()
biva_1=biva_1.reset_index()

#Create combo chart
fig, ax1 = plt.subplots(figsize=(7,4))
color = 'tab:green'
#bar plot creation
ax1.set_title('Bivariado MONTO L NEA CONSUMER - Bad Rate', fontsize=16)
ax1 = sns.barplot(x='CC_CREDIT_LINE_AMT_BIN', y='% Participacin', data =
     biva_1, palette='coolwarm')
ax1.tick_params(axis='y')
#specify we want to share the same x-axis
ax2 = ax1.twinx()
color = 'tab:red'
#line plot creation
ax2.set_ylabel('Bad Rate', fontsize=10)
ax2 = sns.lineplot(x= biva_1['CC_CREDIT_LINE_AMT_BIN'].astype(str), y=biva_1['
     mean'],marker='o')
#for x, y in zip(biva_1['SCORE_CHURN_BIN'], biva_1['mean']):
#    ax2.text(x, y,f'{y:.3f}', color="Black")
ax2.tick_params(axis='y', color=color)
plt.legend(loc='upper left', labels=['Bad Rate'])
#show plot
plt.show()

data_in['CRD_PROP_TOTAL_AMOUNT_BIN'] =np.where(data_in['CRD_PROP_TOTAL_AMOUNT'
     ]<=50,'(25 , 50]',
                                     np.where((data_in['CRD_PROP_TOTAL_AMOUNT'
                                          ]>50)&(data_in['CRD_PROP_TOTAL_AMOUNT
                                          ']<=100),'(50 , 100]',
                                     np.where((data_in['CRD_PROP_TOTAL_AMOUNT'
                                          ]>100)&(data_in['
                                          CRD_PROP_TOTAL_AMOUNT']<=200),'(100 ,
                                           200]',
                                     np.where((data_in['CRD_PROP_TOTAL_AMOUNT'
                                          ]>200)&(data_in['
                                          CRD_PROP_TOTAL_AMOUNT']<=500),'(200 ,
                                           500]',
                                     np.where((data_in['CRD_PROP_TOTAL_AMOUNT'
                                          ]>500)&(data_in['
                                          CRD_PROP_TOTAL_AMOUNT']<=50000),'(500
                                           , 50000]',None)))))
```

```
biva_1=data_in.groupby('CRD_PROP_TOTAL_AMOUNT_BIN')['FLAG_DEFAULT_MES'].agg(['
    mean','sum','count'])
biva_1['% Participaci n']=biva_1['count']/biva_1['count'].sum()
biva_1=biva_1.reset_index().sort_values(by='mean')

#Create combo chart
fig, ax1 = plt.subplots(figsize=(7,4))
color = 'tab:green'
#bar plot creation
ax1.set_title('Bivariado MONTO PROPUESTA - Bad Rate', fontsize=16)
ax1 = sns.barplot(x='CRD_PROP_TOTAL_AMOUNT_BIN', y='% Participaci n', data =
    biva_1, palette='coolwarm')
ax1.tick_params(axis='y')
#specify we want to share the same x-axis
ax2 = ax1.twinx()
color = 'tab:red'
#line plot creation
ax2.set_ylabel('Bad Rate', fontsize=10)
ax2 = sns.lineplot(x= biva_1['CRD_PROP_TOTAL_AMOUNT_BIN'].astype(str), y=
    biva_1['mean'],marker='o')
#for x, y in zip(biva_1['SCORE_CHURN_BIN'], biva_1['mean']):
#    ax2.text(x, y,f'{y:.3f}', color="Black")
ax2.tick_params(axis='y', color=color)
plt.legend(loc='upper left', labels=['Bad Rate'])
#show plot
plt.show()

#Tratamiento de nulos
data_in['WEEKLY_MULTIPLIER'] = np.where(data_in['WEEKLY_MULTIPLIER'].isnull(),
     2, data_in['WEEKLY_MULTIPLIER'])
data_out['WEEKLY_MULTIPLIER'] = np.where(data_out['WEEKLY_MULTIPLIER'].isnull
    (), 2, data_out['WEEKLY_MULTIPLIER'])

data_in['FLAG_SCORE_CHURN']=np.where(data_in['SCORE_CHURN'].isnull(),0,1)
data_in['FLAG_PPV_CF']=np.where(data_in['SUM_AMOUNT_FINISHED_CREDIT'].isnull()
    ,0,1)
data_in['FLAG_ACTIVE_PPV_CF']=np.where(data_in['SUM_AMOUNT_ACTIVE_CRED'].
    isnull(),0,1)
data_in['FLAG_SCORE_BHV']=np.where(data_in['PROB_BHV_1L'].isnull(),0,1)
data_in['MAX_ATRASO_CRED_DE']=np.where((data_in['MAX_ATRASO_CRED_DE'].isnull()
    )&(data_in['FLAG_DEFAULT_MES']==0),0,data_in['MAX_ATRASO_CRED_DE'])
data_in['FLAG_LINE_CC_USE']=np.where(data_in['CC_RAT_CREDIT_LINE_USE'].isnull
    (),0,1)
data_in['FLAG_LINE_CC_USE_1M']=np.where(data_in['CC_RAT_CREDIT_LINE_USE_1M'].
    isnull(),0,1)
data_in['FLAG_LINE_CC_USE_3M']=np.where(data_in['CC_RAT_CREDIT_LINE_USE_3M'].
    isnull(),0,1)
data_in['FLAG_LINE_CC_USE_6M']=np.where(data_in['CC_RAT_CREDIT_LINE_USE_6M'].
    isnull(),0,1)
data_in['FLAG_LINE_CC_USE_12M']=np.where(data_in['CC_RAT_CREDIT_LINE_USE_12M'
    ].isnull(),0,1)
data_in['FLAG_LINE_CC_USE_vs_1']=np.where(data_in['
    CC_DIF_CREDIT_LINE_USE_VS_1M'].isnull(),0,1)
data_in['FLAG_LINE_CC_USE_vs_3']=np.where(data_in['
    CC_DIF_CREDIT_LINE_USE_VS_3M'].isnull(),0,1)
data_in['FLAG_LINE_CC_USE_vs_6']=np.where(data_in['
    CC_DIF_CREDIT_LINE_USE_VS_6M'].isnull(),0,1)
data_in['FLAG_LINE_CC_USE_vs_12']=np.where(data_in['
    CC_DIF_CREDIT_LINE_USE_VS_12M'].isnull(),0,1)
data_in['FLAG_UPSELL']=np.where(data_in['CC_DAYS_SINCE_LAST_UPSELL'].isnull()
    ,0,1)
```

```
data_in['FLAG_DOWNSELL']=np.where(data_in['CC_DAYS_SINCE_LAST_DOWNSELL'].
    isnull(),0,1)
data_in['FLAG_LINE_CC']=np.where(data_in['CC_MAX_QTY_DAYS_PAY_DELAY'].isnull()
    ,0,1)
data_in['FLAG_LINE_CC_1M']=np.where(data_in['CC_MAX_QTY_DAYS_PAY_DELAY_1M'].
    isnull(),0,1)
data_in['FLAG_LINE_CC_3M']=np.where(data_in['CC_MAX_QTY_DAYS_PAY_DELAY_3M'].
    isnull(),0,1)
data_in['FLAG_LINE_CC_6M']=np.where(data_in['CC_MAX_QTY_DAYS_PAY_DELAY_6M'].
    isnull(),0,1)
data_in['FLAG_LINE_CC_12M']=np.where(data_in['CC_MAX_QTY_DAYS_PAY_DELAY_12M'].
    isnull(),0,1)

data_out['FLAG_SCORE_CHURN']=np.where(data_out['SCORE_CHURN'].isnull(),0,1)
data_out['FLAG_PPV_CF']=np.where(data_out['SUM_AMOUNT_FINISHED_CREDIT'].isnull
    (),0,1)
data_out['FLAG_ACTIVE_PPV_CF']=np.where(data_out['SUM_AMOUNT_ACTIVE_CRED'].
    isnull(),0,1)
data_out['FLAG_SCORE_BHV']=np.where(data_out['PROB_BHV_1L'].isnull(),0,1)
data_out['MAX_ATRASO_CRED_DE']=np.where((data_out['MAX_ATRASO_CRED_DE'].isnull
    ())&(data_out['FLAG_DEFAULT_MES']==0),0,data_out['MAX_ATRASO_CRED_DE'])
data_out['FLAG_LINE_CC_USE']=np.where(data_out['CC_RAT_CREDIT_LINE_USE'].
    isnull(),0,1)
data_out['FLAG_LINE_CC_USE_1M']=np.where(data_out['CC_RAT_CREDIT_LINE_USE_1M'
    ].isnull(),0,1)
data_out['FLAG_LINE_CC_USE_3M']=np.where(data_out['CC_RAT_CREDIT_LINE_USE_3M'
    ].isnull(),0,1)
data_out['FLAG_LINE_CC_USE_6M']=np.where(data_out['CC_RAT_CREDIT_LINE_USE_6M'
    ].isnull(),0,1)
data_out['FLAG_LINE_CC_USE_12M']=np.where(data_out['CC_RAT_CREDIT_LINE_USE_12M
    '].isnull(),0,1)
data_out['FLAG_LINE_CC_USE_vs_1']=np.where(data_out['
    CC_DIF_CREDIT_LINE_USE_VS_1M'].isnull(),0,1)
data_out['FLAG_LINE_CC_USE_vs_3']=np.where(data_out['
    CC_DIF_CREDIT_LINE_USE_VS_3M'].isnull(),0,1)
data_out['FLAG_LINE_CC_USE_vs_6']=np.where(data_out['
    CC_DIF_CREDIT_LINE_USE_VS_6M'].isnull(),0,1)
data_out['FLAG_LINE_CC_USE_vs_12']=np.where(data_out['
    CC_DIF_CREDIT_LINE_USE_VS_12M'].isnull(),0,1)
data_out['FLAG_UPSELL']=np.where(data_out['CC_DAYS_SINCE_LAST_UPSELL'].isnull
    (),0,1)
data_out['FLAG_DOWNSELL']=np.where(data_out['CC_DAYS_SINCE_LAST_DOWNSELL'].
    isnull(),0,1)
data_out['FLAG_LINE_CC']=np.where(data_out['CC_MAX_QTY_DAYS_PAY_DELAY'].isnull
    (),0,1)
data_out['FLAG_LINE_CC_1M']=np.where(data_out['CC_MAX_QTY_DAYS_PAY_DELAY_1M'].
    isnull(),0,1)
data_out['FLAG_LINE_CC_3M']=np.where(data_out['CC_MAX_QTY_DAYS_PAY_DELAY_3M'].
    isnull(),0,1)
data_out['FLAG_LINE_CC_6M']=np.where(data_out['CC_MAX_QTY_DAYS_PAY_DELAY_6M'].
    isnull(),0,1)
data_out['FLAG_LINE_CC_12M']=np.where(data_out['CC_MAX_QTY_DAYS_PAY_DELAY_12M'
    ].isnull(),0,1)

data_in['COMBIN_SCORE_CHURN']=np.where(data_in['FLAG_SCORE_CHURN']==0,0,
    data_in['SCORE_CHURN'])

data_in['COMBIN_SUM_AMOUNT_FINISHED_CREDIT']=np.where(data_in['FLAG_PPV_CF'
    ]==0,0,data_in['SUM_AMOUNT_FINISHED_CREDIT'])
data_in['COMBIN_LAST_DELAY_PPV_CF']=np.where(data_in['FLAG_PPV_CF']==0,0,
    data_in['LAST_DELAY_PPV_CF'])
```

```
data_in['COMBIN_MAX_ATRASO_FINISHED_CRED']=np.where(data_in['FLAG_PPV_CF'
    ]==0,0,data_in['MAX_ATRASO_FINISHED_CRED'])
data_in['COMBIN_AVG_ATRASO_FINISHED_CRED']=np.where(data_in['FLAG_PPV_CF'
    ]==0,0,data_in['AVG_ATRASO_FINISHED_CRED'])
data_in['COMBIN_PROP_AMOUNT_LAST_FINISHED_CRED']=np.where(data_in['FLAG_PPV_CF
    ']==0,0,data_in['PROP_AMOUNT_LAST_FINISHED_CRED'])

data_in['COMBIN_SUM_AMOUNT_ACTIVE_CRED']=np.where(data_in['FLAG_ACTIVE_PPV_CF'
    ]==0,0,data_in['SUM_AMOUNT_ACTIVE_CRED'])
data_in['COMBIN_PORCENTAJE_PAGO']=np.where(data_in['FLAG_ACTIVE_PPV_CF']==0,0,
    data_in['PORCENTAJE_PAGO'])
data_in['COMBIN_MAX_ATRASO_ACTIVE_CRED']=np.where(data_in['FLAG_ACTIVE_PPV_CF'
    ]==0,0,data_in['MAX_ATRASO_ACTIVE_CRED'])
data_in['COMBIN_AVG_ATRASO_ACTIVE_CRED']=np.where(data_in['FLAG_ACTIVE_PPV_CF'
    ]==0,0,data_in['AVG_ATRASO_ACTIVE_CRED'])
data_in['COMBIN_Q_CUOTAS_PAGAS_1L_ACTIVE']=np.where(data_in['
    FLAG_ACTIVE_PPV_CF']==0,0,data_in['Q_CUOTAS_PAGAS_1L_ACTIVE'])
data_in['COMBIN_Q_CUOTAS_IMPAGAS_1L_ACTIVE']=np.where(data_in['
    FLAG_ACTIVE_PPV_CF']==0,0,data_in['Q_CUOTAS_IMPAGAS_1L_ACTIVE'])
data_in['COMBIN_PLAZO_L1']=np.where(data_in['FLAG_ACTIVE_PPV_CF']==0,0,data_in
    ['PLAZO_L1'])
data_in['COMBIN_Q_CUOTAS_PAGAS_2L_ACTIVE']=np.where(data_in['
    FLAG_ACTIVE_PPV_CF']==0,0,data_in['Q_CUOTAS_PAGAS_2L_ACTIVE'])
data_in['COMBIN_Q_CUOTAS_IMPAGAS_2L_ACTIVE']=np.where(data_in['
    FLAG_ACTIVE_PPV_CF']==0,0,data_in['Q_CUOTAS_IMPAGAS_2L_ACTIVE'])

data_in['COMBIN_PROB_BHV_1L']=np.where(data_in['FLAG_SCORE_BHV']==0,0,data_in[
    'PROB_BHV_1L'])

data_in['COMBIN_CC_RAT_CREDIT_LINE_USE']=np.where(data_in['FLAG_LINE_CC_USE'
    ]==0,0,data_in['CC_RAT_CREDIT_LINE_USE'])
data_in['COMBIN_CC_RAT_CREDIT_LINE_USE_1M']=np.where(data_in['
    FLAG_LINE_CC_USE_1M']==0,0,data_in['CC_RAT_CREDIT_LINE_USE_1M'])
data_in['COMBIN_CC_RAT_CREDIT_LINE_USE_3M']=np.where(data_in['
    FLAG_LINE_CC_USE_3M']==0,0,data_in['CC_RAT_CREDIT_LINE_USE_3M'])
data_in['COMBIN_CC_RAT_CREDIT_LINE_USE_6M']=np.where(data_in['
    FLAG_LINE_CC_USE_6M']==0,0,data_in['CC_RAT_CREDIT_LINE_USE_6M'])
data_in['COMBIN_CC_RAT_CREDIT_LINE_USE_12M']=np.where(data_in['
    FLAG_LINE_CC_USE_12M']==0,0,data_in['CC_RAT_CREDIT_LINE_USE_12M'])
data_in['COMBIN_CC_DIF_CREDIT_LINE_USE_VS_1M']=np.where(data_in['
    FLAG_LINE_CC_USE_vs_1']==0,0,data_in['CC_DIF_CREDIT_LINE_USE_VS_1M'])
data_in['COMBIN_CC_DIF_CREDIT_LINE_USE_VS_3M']=np.where(data_in['
    FLAG_LINE_CC_USE_vs_3']==0,0,data_in['CC_DIF_CREDIT_LINE_USE_VS_3M'])
data_in['COMBIN_CC_DIF_CREDIT_LINE_USE_VS_6M']=np.where(data_in['
    FLAG_LINE_CC_USE_vs_6']==0,0,data_in['CC_DIF_CREDIT_LINE_USE_VS_6M'])
data_in['COMBIN_CC_DIF_CREDIT_LINE_USE_VS_12M']=np.where(data_in['
    FLAG_LINE_CC_USE_vs_12']==0,0,data_in['CC_DIF_CREDIT_LINE_USE_VS_12M'])

data_in['COMBIN_CC_DAYS_SINCE_LAST_UPSELL']=np.where(data_in['FLAG_UPSELL'
    ]==0,0,data_in['CC_DAYS_SINCE_LAST_UPSELL'])
data_in['COMBIN_CC_LAST_UPSELL_RAT_AMT']=np.where(data_in['FLAG_UPSELL']==0,0,
    data_in['CC_LAST_UPSELL_RAT_AMT'])
data_in['COMBIN_CC_UPSELL_MIN_RAT_AMT']=np.where(data_in['FLAG_UPSELL']==0,0,
    data_in['CC_UPSELL_MIN_RAT_AMT'])
data_in['COMBIN_CC_UPSELL_AVG_RAT_AMT']=np.where(data_in['FLAG_UPSELL']==0,0,
    data_in['CC_UPSELL_AVG_RAT_AMT'])
data_in['COMBIN_CC_UPSELL_MAX_RAT_AMT']=np.where(data_in['FLAG_UPSELL']==0,0,
    data_in['CC_UPSELL_MAX_RAT_AMT'])

data_in['COMBIN_CC_DAYS_SINCE_LAST_DOWNSELL']=np.where(data_in['FLAG_DOWNSELL'
    ]==0,0,data_in['CC_DAYS_SINCE_LAST_DOWNSELL'])
```

```
data_in['COMBIN_CC_LAST_DOWNSELL_RAT_AMT']=np.where(data_in['FLAG_DOWNSELL'
    ]==0,0,data_in['CC_LAST_DOWNSELL_RAT_AMT'])
data_in['COMBIN_CC_DOWNSELL_MIN_RAT_AMT']=np.where(data_in['FLAG_DOWNSELL'
    ]==0,0,data_in['CC_DOWNSELL_MIN_RAT_AMT'])
data_in['COMBIN_CC_DOWNSELL_AVG_RAT_AMT']=np.where(data_in['FLAG_DOWNSELL'
    ]==0,0,data_in['CC_DOWNSELL_AVG_RAT_AMT'])
data_in['COMBIN_CC_DOWNSELL_MAX_RAT_AMT']=np.where(data_in['FLAG_DOWNSELL'
    ]==0,0,data_in['CC_DOWNSELL_MAX_RAT_AMT'])

data_in['COMBIN_CC_MAX_QTY_DAYS_PAY_DELAY']=np.where(data_in['FLAG_LINE_CC'
    ]==0,0,data_in['CC_MAX_QTY_DAYS_PAY_DELAY'])
data_in['COMBIN_CC_RAT_AMT_OVERDUE']=np.where(data_in['FLAG_LINE_CC']==0,0,
    data_in['CC_RAT_AMT_OVERDUE'])
data_in['COMBIN_CC_RAT_AMT_OVERDUE_5_10_DPD']=np.where(data_in['FLAG_LINE_CC'
    ]==0,0,data_in['CC_RAT_AMT_OVERDUE_5_10_DPD'])
data_in['COMBIN_CC_RAT_AMT_OVERDUE_10_20_DPD']=np.where(data_in['FLAG_LINE_CC'
    ]==0,0,data_in['CC_RAT_AMT_OVERDUE_10_20_DPD'])
data_in['COMBIN_CC_RAT_AMT_OVERDUE_20_30_DPD']=np.where(data_in['FLAG_LINE_CC'
    ]==0,0,data_in['CC_RAT_AMT_OVERDUE_20_30_DPD'])
data_in['COMBIN_CC_RAT_AMT_OVERDUE_OVER_30_DPD']=np.where(data_in['
    FLAG_LINE_CC']==0,0,data_in['CC_RAT_AMT_OVERDUE_OVER_30_DPD'])
data_in['COMBIN_CC_RAT_INST_QTY_OVERDUE']=np.where(data_in['FLAG_LINE_CC'
    ]==0,0,data_in['CC_RAT_INST_QTY_OVERDUE'])
data_in['COMBIN_CC_RAT_INST_QTY_OVERDUE_5_10_DPD']=np.where(data_in['
    FLAG_LINE_CC']==0,0,data_in['CC_RAT_INST_QTY_OVERDUE_5_10_DPD'])
data_in['COMBIN_CC_RAT_INST_QTY_OVERDUE_10_20_DPD']=np.where(data_in['
    FLAG_LINE_CC']==0,0,data_in['CC_RAT_INST_QTY_OVERDUE_10_20_DPD'])
data_in['COMBIN_CC_RAT_INST_QTY_OVERDUE_20_30_DPD']=np.where(data_in['
    FLAG_LINE_CC']==0,0,data_in['CC_RAT_INST_QTY_OVERDUE_20_30_DPD'])
data_in['COMBIN_CC_RAT_INST_QTY_OVERDUE_OVER_30_DPD']=np.where(data_in['
    FLAG_LINE_CC']==0,0,data_in['CC_RAT_INST_QTY_OVERDUE_OVER_30_DPD'])

data_in['COMBIN_CC_MAX_QTY_DAYS_PAY_DELAY_1M']=np.where(data_in['
    FLAG_LINE_CC_1M']==0,0,data_in['CC_MAX_QTY_DAYS_PAY_DELAY_1M'])
data_in['COMBIN_CC_RAT_AMT_OVERDUE_1M']=np.where(data_in['FLAG_LINE_CC_1M'
    ]==0,0,data_in['CC_RAT_AMT_OVERDUE_1M'])
data_in['COMBIN_CC_RAT_AMT_OVERDUE_5_10_DPD_1M']=np.where(data_in['
    FLAG_LINE_CC_1M']==0,0,data_in['CC_RAT_AMT_OVERDUE_5_10_DPD_1M'])
data_in['COMBIN_CC_RAT_AMT_OVERDUE_10_20_DPD_1M']=np.where(data_in['
    FLAG_LINE_CC_1M']==0,0,data_in['CC_RAT_AMT_OVERDUE_10_20_DPD_1M'])
data_in['COMBIN_CC_RAT_AMT_OVERDUE_20_30_DPD_1M']=np.where(data_in['
    FLAG_LINE_CC_1M']==0,0,data_in['CC_RAT_AMT_OVERDUE_20_30_DPD_1M'])
data_in['COMBIN_CC_RAT_AMT_OVERDUE_OVER_30_DPD_1M']=np.where(data_in['
    FLAG_LINE_CC_1M']==0,0,data_in['CC_RAT_AMT_OVERDUE_OVER_30_DPD_1M'])
data_in['COMBIN_CC_RAT_INST_QTY_OVERDUE_1M']=np.where(data_in['FLAG_LINE_CC_1M
    ']==0,0,data_in['CC_RAT_INST_QTY_OVERDUE_1M'])
data_in['COMBIN_CC_RAT_INST_QTY_OVERDUE_5_10_DPD_1M']=np.where(data_in['
    FLAG_LINE_CC_1M']==0,0,data_in['CC_RAT_INST_QTY_OVERDUE_5_10_DPD_1M'])
data_in['COMBIN_CC_RAT_INST_QTY_OVERDUE_10_20_DPD_1M']=np.where(data_in['
    FLAG_LINE_CC_1M']==0,0,data_in['CC_RAT_INST_QTY_OVERDUE_10_20_DPD_1M'])
data_in['COMBIN_CC_RAT_INST_QTY_OVERDUE_20_30_DPD_1M']=np.where(data_in['
    FLAG_LINE_CC_1M']==0,0,data_in['CC_RAT_INST_QTY_OVERDUE_20_30_DPD_1M'])
data_in['COMBIN_CC_RAT_INST_QTY_OVERDUE_OVER_30_DPD_1M']=np.where(data_in['
    FLAG_LINE_CC_1M']==0,0,data_in['CC_RAT_INST_QTY_OVERDUE_OVER_30_DPD_1M'])

data_in['COMBIN_CC_MAX_QTY_DAYS_PAY_DELAY_3M']=np.where(data_in['
    FLAG_LINE_CC_3M']==0,0,data_in['CC_MAX_QTY_DAYS_PAY_DELAY_3M'])
data_in['COMBIN_CC_RAT_AMT_OVERDUE_3M']=np.where(data_in['FLAG_LINE_CC_3M'
    ]==0,0,data_in['CC_RAT_AMT_OVERDUE_3M'])
data_in['COMBIN_CC_RAT_AMT_OVERDUE_5_10_DPD_3M']=np.where(data_in['
    FLAG_LINE_CC_3M']==0,0,data_in['CC_RAT_AMT_OVERDUE_5_10_DPD_3M'])
```

```
data_in['COMBIN_CC_RAT_AMT_OVERDUE_10_20_DPD_3M']=np.where(data_in['
    FLAG_LINE_CC_3M']==0,0,data_in['CC_RAT_AMT_OVERDUE_10_20_DPD_3M'])
data_in['COMBIN_CC_RAT_AMT_OVERDUE_20_30_DPD_3M']=np.where(data_in['
    FLAG_LINE_CC_3M']==0,0,data_in['CC_RAT_AMT_OVERDUE_20_30_DPD_3M'])
data_in['COMBIN_CC_RAT_AMT_OVERDUE_OVER_30_DPD_3M']=np.where(data_in['
    FLAG_LINE_CC_3M']==0,0,data_in['CC_RAT_AMT_OVERDUE_OVER_30_DPD_3M'])
data_in['COMBIN_CC_RAT_INST_QTY_OVERDUE_3M']=np.where(data_in['FLAG_LINE_CC_3M
    ']==0,0,data_in['CC_RAT_INST_QTY_OVERDUE_3M'])
data_in['COMBIN_CC_RAT_INST_QTY_OVERDUE_5_10_DPD_3M']=np.where(data_in['
    FLAG_LINE_CC_3M']==0,0,data_in['CC_RAT_INST_QTY_OVERDUE_5_10_DPD_3M'])
data_in['COMBIN_CC_RAT_INST_QTY_OVERDUE_10_20_DPD_3M']=np.where(data_in['
    FLAG_LINE_CC_3M']==0,0,data_in['CC_RAT_INST_QTY_OVERDUE_10_20_DPD_3M'])
data_in['COMBIN_CC_RAT_INST_QTY_OVERDUE_20_30_DPD_3M']=np.where(data_in['
    FLAG_LINE_CC_3M']==0,0,data_in['CC_RAT_INST_QTY_OVERDUE_20_30_DPD_3M'])
data_in['COMBIN_CC_RAT_INST_QTY_OVERDUE_OVER_30_DPD_3M']=np.where(data_in['
    FLAG_LINE_CC_3M']==0,0,data_in['CC_RAT_INST_QTY_OVERDUE_OVER_30_DPD_3M'])

data_in['COMBIN_CC_MAX_QTY_DAYS_PAY_DELAY_6M']=np.where(data_in['
    FLAG_LINE_CC_6M']==0,0,data_in['CC_MAX_QTY_DAYS_PAY_DELAY_6M'])
data_in['COMBIN_CC_RAT_AMT_OVERDUE_6M']=np.where(data_in['FLAG_LINE_CC_6M'
    ]==0,0,data_in['CC_RAT_AMT_OVERDUE_6M'])
data_in['COMBIN_CC_RAT_AMT_OVERDUE_5_10_DPD_6M']=np.where(data_in['
    FLAG_LINE_CC_6M']==0,0,data_in['CC_RAT_AMT_OVERDUE_5_10_DPD_6M'])
data_in['COMBIN_CC_RAT_AMT_OVERDUE_10_20_DPD_6M']=np.where(data_in['
    FLAG_LINE_CC_6M']==0,0,data_in['CC_RAT_AMT_OVERDUE_10_20_DPD_6M'])
data_in['COMBIN_CC_RAT_AMT_OVERDUE_20_30_DPD_6M']=np.where(data_in['
    FLAG_LINE_CC_6M']==0,0,data_in['CC_RAT_AMT_OVERDUE_20_30_DPD_6M'])
data_in['COMBIN_CC_RAT_AMT_OVERDUE_OVER_30_DPD_6M']=np.where(data_in['
    FLAG_LINE_CC_6M']==0,0,data_in['CC_RAT_AMT_OVERDUE_OVER_30_DPD_6M'])
data_in['COMBIN_CC_RAT_INST_QTY_OVERDUE_6M']=np.where(data_in['FLAG_LINE_CC_6M
    ']==0,0,data_in['CC_RAT_INST_QTY_OVERDUE_6M'])
data_in['COMBIN_CC_RAT_INST_QTY_OVERDUE_5_10_DPD_6M']=np.where(data_in['
    FLAG_LINE_CC_6M']==0,0,data_in['CC_RAT_INST_QTY_OVERDUE_5_10_DPD_6M'])
data_in['COMBIN_CC_RAT_INST_QTY_OVERDUE_10_20_DPD_6M']=np.where(data_in['
    FLAG_LINE_CC_6M']==0,0,data_in['CC_RAT_INST_QTY_OVERDUE_10_20_DPD_6M'])
data_in['COMBIN_CC_RAT_INST_QTY_OVERDUE_20_30_DPD_6M']=np.where(data_in['
    FLAG_LINE_CC_6M']==0,0,data_in['CC_RAT_INST_QTY_OVERDUE_20_30_DPD_6M'])
data_in['COMBIN_CC_RAT_INST_QTY_OVERDUE_OVER_30_DPD_6M']=np.where(data_in['
    FLAG_LINE_CC_6M']==0,0,data_in['CC_RAT_INST_QTY_OVERDUE_OVER_30_DPD_6M'])

data_in['COMBIN_CC_MAX_QTY_DAYS_PAY_DELAY_12M']=np.where(data_in['
    FLAG_LINE_CC_12M']==0,0,data_in['CC_MAX_QTY_DAYS_PAY_DELAY_12M'])
data_in['COMBIN_CC_RAT_AMT_OVERDUE_12M']=np.where(data_in['FLAG_LINE_CC_12M'
    ]==0,0,data_in['CC_RAT_AMT_OVERDUE_12M'])
data_in['COMBIN_CC_RAT_AMT_OVERDUE_5_10_DPD_12M']=np.where(data_in['
    FLAG_LINE_CC_12M']==0,0,data_in['CC_RAT_AMT_OVERDUE_5_10_DPD_12M'])
data_in['COMBIN_CC_RAT_AMT_OVERDUE_10_20_DPD_12M']=np.where(data_in['
    FLAG_LINE_CC_12M']==0,0,data_in['CC_RAT_AMT_OVERDUE_10_20_DPD_12M'])
data_in['COMBIN_CC_RAT_AMT_OVERDUE_20_30_DPD_12M']=np.where(data_in['
    FLAG_LINE_CC_12M']==0,0,data_in['CC_RAT_AMT_OVERDUE_20_30_DPD_12M'])
data_in['COMBIN_CC_RAT_AMT_OVERDUE_OVER_30_DPD_12M']=np.where(data_in['
    FLAG_LINE_CC_12M']==0,0,data_in['CC_RAT_AMT_OVERDUE_OVER_30_DPD_12M'])
data_in['COMBIN_CC_RAT_INST_QTY_OVERDUE_12M']=np.where(data_in['
    FLAG_LINE_CC_12M']==0,0,data_in['CC_RAT_INST_QTY_OVERDUE_12M'])
data_in['COMBIN_CC_RAT_INST_QTY_OVERDUE_5_10_DPD_12M']=np.where(data_in['
    FLAG_LINE_CC_12M']==0,0,data_in['CC_RAT_INST_QTY_OVERDUE_5_10_DPD_12M'])
data_in['COMBIN_CC_RAT_INST_QTY_OVERDUE_10_20_DPD_12M']=np.where(data_in['
    FLAG_LINE_CC_12M']==0,0,data_in['CC_RAT_INST_QTY_OVERDUE_10_20_DPD_12M'])
data_in['COMBIN_CC_RAT_INST_QTY_OVERDUE_20_30_DPD_12M']=np.where(data_in['
    FLAG_LINE_CC_12M']==0,0,data_in['CC_RAT_INST_QTY_OVERDUE_20_30_DPD_12M'])
data_in['COMBIN_CC_RAT_INST_QTY_OVERDUE_OVER_30_DPD_12M']=np.where(data_in['
    FLAG_LINE_CC_12M']==0,0,data_in['CC_RAT_INST_QTY_OVERDUE_OVER_30_DPD_12M'
```

```
    ])

data_out['COMBIN_SCORE_CHURN']=np.where(data_out['FLAG_SCORE_CHURN']==0,0,
    data_out['SCORE_CHURN'])

data_out['COMBIN_SUM_AMOUNT_FINISHED_CREDIT']=np.where(data_out['FLAG_PPV_CF'
    ]==0,0,data_out['SUM_AMOUNT_FINISHED_CREDIT'])
data_out['COMBIN_LAST_DELAY_PPV_CF']=np.where(data_out['FLAG_PPV_CF']==0,0,
    data_out['LAST_DELAY_PPV_CF'])
data_out['COMBIN_MAX_ATRASO_FINISHED_CRED']=np.where(data_out['FLAG_PPV_CF'
    ]==0,0,data_out['MAX_ATRASO_FINISHED_CRED'])
data_out['COMBIN_AVG_ATRASO_FINISHED_CRED']=np.where(data_out['FLAG_PPV_CF'
    ]==0,0,data_out['AVG_ATRASO_FINISHED_CRED'])
data_out['COMBIN_PROP_AMOUNT_LAST_FINISHED_CRED']=np.where(data_out['
    FLAG_PPV_CF']==0,0,data_out['PROP_AMOUNT_LAST_FINISHED_CRED'])

data_out['COMBIN_SUM_AMOUNT_ACTIVE_CRED']=np.where(data_out['
    FLAG_ACTIVE_PPV_CF']==0,0,data_out['SUM_AMOUNT_ACTIVE_CRED'])
data_out['COMBIN_PORCENTAJE_PAGO']=np.where(data_out['FLAG_ACTIVE_PPV_CF'
    ]==0,0,data_out['PORCENTAJE_PAGO'])
data_out['COMBIN_MAX_ATRASO_ACTIVE_CRED']=np.where(data_out['
    FLAG_ACTIVE_PPV_CF']==0,0,data_out['MAX_ATRASO_ACTIVE_CRED'])
data_out['COMBIN_AVG_ATRASO_ACTIVE_CRED']=np.where(data_out['
    FLAG_ACTIVE_PPV_CF']==0,0,data_out['AVG_ATRASO_ACTIVE_CRED'])
data_out['COMBIN_Q_CUOTAS_PAGAS_1L_ACTIVE']=np.where(data_out['
    FLAG_ACTIVE_PPV_CF']==0,0,data_out['Q_CUOTAS_PAGAS_1L_ACTIVE'])
data_out['COMBIN_Q_CUOTAS_IMPAGAS_1L_ACTIVE']=np.where(data_out['
    FLAG_ACTIVE_PPV_CF']==0,0,data_out['Q_CUOTAS_IMPAGAS_1L_ACTIVE'])
data_out['COMBIN_PLAZO_L1']=np.where(data_out['FLAG_ACTIVE_PPV_CF']==0,0,
    data_out['PLAZO_L1'])
data_out['COMBIN_Q_CUOTAS_PAGAS_2L_ACTIVE']=np.where(data_out['
    FLAG_ACTIVE_PPV_CF']==0,0,data_out['Q_CUOTAS_PAGAS_2L_ACTIVE'])
data_out['COMBIN_Q_CUOTAS_IMPAGAS_2L_ACTIVE']=np.where(data_out['
    FLAG_ACTIVE_PPV_CF']==0,0,data_out['Q_CUOTAS_IMPAGAS_2L_ACTIVE'])

data_out['COMBIN_PROB_BHV_1L']=np.where(data_out['FLAG_SCORE_BHV']==0,0,
    data_out['PROB_BHV_1L'])

data_out['COMBIN_CC_RAT_CREDIT_LINE_USE']=np.where(data_out['FLAG_LINE_CC_USE'
    ]==0,0,data_out['CC_RAT_CREDIT_LINE_USE'])
data_out['COMBIN_CC_RAT_CREDIT_LINE_USE_1M']=np.where(data_out['
    FLAG_LINE_CC_USE_1M']==0,0,data_out['CC_RAT_CREDIT_LINE_USE_1M'])
data_out['COMBIN_CC_RAT_CREDIT_LINE_USE_3M']=np.where(data_out['
    FLAG_LINE_CC_USE_3M']==0,0,data_out['CC_RAT_CREDIT_LINE_USE_3M'])
data_out['COMBIN_CC_RAT_CREDIT_LINE_USE_6M']=np.where(data_out['
    FLAG_LINE_CC_USE_6M']==0,0,data_out['CC_RAT_CREDIT_LINE_USE_6M'])
data_out['COMBIN_CC_RAT_CREDIT_LINE_USE_12M']=np.where(data_out['
    FLAG_LINE_CC_USE_12M']==0,0,data_out['CC_RAT_CREDIT_LINE_USE_12M'])
data_out['COMBIN_CC_DIF_CREDIT_LINE_USE_VS_1M']=np.where(data_out['
    FLAG_LINE_CC_USE_vs_1']==0,0,data_out['CC_DIF_CREDIT_LINE_USE_VS_1M'])
data_out['COMBIN_CC_DIF_CREDIT_LINE_USE_VS_3M']=np.where(data_out['
    FLAG_LINE_CC_USE_vs_3']==0,0,data_out['CC_DIF_CREDIT_LINE_USE_VS_3M'])
data_out['COMBIN_CC_DIF_CREDIT_LINE_USE_VS_6M']=np.where(data_out['
    FLAG_LINE_CC_USE_vs_6']==0,0,data_out['CC_DIF_CREDIT_LINE_USE_VS_6M'])
data_out['COMBIN_CC_DIF_CREDIT_LINE_USE_VS_12M']=np.where(data_out['
    FLAG_LINE_CC_USE_vs_12']==0,0,data_out['CC_DIF_CREDIT_LINE_USE_VS_12M'])

data_out['COMBIN_CC_DAYS_SINCE_LAST_UPSELL']=np.where(data_out['FLAG_UPSELL'
    ]==0,0,data_out['CC_DAYS_SINCE_LAST_UPSELL'])
data_out['COMBIN_CC_LAST_UPSELL_RAT_AMT']=np.where(data_out['FLAG_UPSELL'
    ]==0,0,data_out['CC_LAST_UPSELL_RAT_AMT'])
```

```
data_out['COMBIN_CC_UPSELL_MIN_RAT_AMT']=np.where(data_out['FLAG_UPSELL'
    ]==0,0,data_out['CC_UPSELL_MIN_RAT_AMT'])
data_out['COMBIN_CC_UPSELL_AVG_RAT_AMT']=np.where(data_out['FLAG_UPSELL'
    ]==0,0,data_out['CC_UPSELL_AVG_RAT_AMT'])
data_out['COMBIN_CC_UPSELL_MAX_RAT_AMT']=np.where(data_out['FLAG_UPSELL'
    ]==0,0,data_out['CC_UPSELL_MAX_RAT_AMT'])

data_out['COMBIN_CC_DAYS_SINCE_LAST_DOWNSELL']=np.where(data_out['
    FLAG_DOWNSELL']==0,0,data_out['CC_DAYS_SINCE_LAST_DOWNSELL'])
data_out['COMBIN_CC_LAST_DOWNSELL_RAT_AMT']=np.where(data_out['FLAG_DOWNSELL'
    ]==0,0,data_out['CC_LAST_DOWNSELL_RAT_AMT'])
data_out['COMBIN_CC_DOWNSELL_MIN_RAT_AMT']=np.where(data_out['FLAG_DOWNSELL'
    ]==0,0,data_out['CC_DOWNSELL_MIN_RAT_AMT'])
data_out['COMBIN_CC_DOWNSELL_AVG_RAT_AMT']=np.where(data_out['FLAG_DOWNSELL'
    ]==0,0,data_out['CC_DOWNSELL_AVG_RAT_AMT'])
data_out['COMBIN_CC_DOWNSELL_MAX_RAT_AMT']=np.where(data_out['FLAG_DOWNSELL'
    ]==0,0,data_out['CC_DOWNSELL_MAX_RAT_AMT'])

data_out['COMBIN_CC_MAX_QTY_DAYS_PAY_DELAY']=np.where(data_out['FLAG_LINE_CC'
    ]==0,0,data_out['CC_MAX_QTY_DAYS_PAY_DELAY'])
data_out['COMBIN_CC_RAT_AMT_OVERDUE']=np.where(data_out['FLAG_LINE_CC']==0,0,
    data_out['CC_RAT_AMT_OVERDUE'])
data_out['COMBIN_CC_RAT_AMT_OVERDUE_5_10_DPD']=np.where(data_out['FLAG_LINE_CC
    ']==0,0,data_out['CC_RAT_AMT_OVERDUE_5_10_DPD'])
data_out['COMBIN_CC_RAT_AMT_OVERDUE_10_20_DPD']=np.where(data_out['
    FLAG_LINE_CC']==0,0,data_out['CC_RAT_AMT_OVERDUE_10_20_DPD'])
data_out['COMBIN_CC_RAT_AMT_OVERDUE_20_30_DPD']=np.where(data_out['
    FLAG_LINE_CC']==0,0,data_out['CC_RAT_AMT_OVERDUE_20_30_DPD'])
data_out['COMBIN_CC_RAT_AMT_OVERDUE_OVER_30_DPD']=np.where(data_out['
    FLAG_LINE_CC']==0,0,data_out['CC_RAT_AMT_OVERDUE_OVER_30_DPD'])
data_out['COMBIN_CC_RAT_INST_QTY_OVERDUE']=np.where(data_out['FLAG_LINE_CC'
    ]==0,0,data_out['CC_RAT_INST_QTY_OVERDUE'])
data_out['COMBIN_CC_RAT_INST_QTY_OVERDUE_5_10_DPD']=np.where(data_out['
    FLAG_LINE_CC']==0,0,data_out['CC_RAT_INST_QTY_OVERDUE_5_10_DPD'])
data_out['COMBIN_CC_RAT_INST_QTY_OVERDUE_10_20_DPD']=np.where(data_out['
    FLAG_LINE_CC']==0,0,data_out['CC_RAT_INST_QTY_OVERDUE_10_20_DPD'])
data_out['COMBIN_CC_RAT_INST_QTY_OVERDUE_20_30_DPD']=np.where(data_out['
    FLAG_LINE_CC']==0,0,data_out['CC_RAT_INST_QTY_OVERDUE_20_30_DPD'])
data_out['COMBIN_CC_RAT_INST_QTY_OVERDUE_OVER_30_DPD']=np.where(data_out['
    FLAG_LINE_CC']==0,0,data_out['CC_RAT_INST_QTY_OVERDUE_OVER_30_DPD'])

data_out['COMBIN_CC_MAX_QTY_DAYS_PAY_DELAY_1M']=np.where(data_out['
    FLAG_LINE_CC_1M']==0,0,data_out['CC_MAX_QTY_DAYS_PAY_DELAY_1M'])
data_out['COMBIN_CC_RAT_AMT_OVERDUE_1M']=np.where(data_out['FLAG_LINE_CC_1M'
    ]==0,0,data_out['CC_RAT_AMT_OVERDUE_1M'])
data_out['COMBIN_CC_RAT_AMT_OVERDUE_5_10_DPD_1M']=np.where(data_out['
    FLAG_LINE_CC_1M']==0,0,data_out['CC_RAT_AMT_OVERDUE_5_10_DPD_1M'])
data_out['COMBIN_CC_RAT_AMT_OVERDUE_10_20_DPD_1M']=np.where(data_out['
    FLAG_LINE_CC_1M']==0,0,data_out['CC_RAT_AMT_OVERDUE_10_20_DPD_1M'])
data_out['COMBIN_CC_RAT_AMT_OVERDUE_20_30_DPD_1M']=np.where(data_out['
    FLAG_LINE_CC_1M']==0,0,data_out['CC_RAT_AMT_OVERDUE_20_30_DPD_1M'])
data_out['COMBIN_CC_RAT_AMT_OVERDUE_OVER_30_DPD_1M']=np.where(data_out['
    FLAG_LINE_CC_1M']==0,0,data_out['CC_RAT_AMT_OVERDUE_OVER_30_DPD_1M'])
data_out['COMBIN_CC_RAT_INST_QTY_OVERDUE_1M']=np.where(data_out['
    FLAG_LINE_CC_1M']==0,0,data_out['CC_RAT_INST_QTY_OVERDUE_1M'])
data_out['COMBIN_CC_RAT_INST_QTY_OVERDUE_5_10_DPD_1M']=np.where(data_out['
    FLAG_LINE_CC_1M']==0,0,data_out['CC_RAT_INST_QTY_OVERDUE_5_10_DPD_1M'])
data_out['COMBIN_CC_RAT_INST_QTY_OVERDUE_10_20_DPD_1M']=np.where(data_out['
    FLAG_LINE_CC_1M']==0,0,data_out['CC_RAT_INST_QTY_OVERDUE_10_20_DPD_1M'])
data_out['COMBIN_CC_RAT_INST_QTY_OVERDUE_20_30_DPD_1M']=np.where(data_out['
    FLAG_LINE_CC_1M']==0,0,data_out['CC_RAT_INST_QTY_OVERDUE_20_30_DPD_1M'])
```

```
data_out['COMBIN_CC_RAT_INST_QTY_OVERDUE_OVER_30_DPD_1M']=np.where(data_out['
    FLAG_LINE_CC_1M']==0,0,data_out['CC_RAT_INST_QTY_OVERDUE_OVER_30_DPD_1M'])

data_out['COMBIN_CC_MAX_QTY_DAYS_PAY_DELAY_3M']=np.where(data_out['
    FLAG_LINE_CC_3M']==0,0,data_out['CC_MAX_QTY_DAYS_PAY_DELAY_3M'])
data_out['COMBIN_CC_RAT_AMT_OVERDUE_3M']=np.where(data_out['FLAG_LINE_CC_3M'
    ]==0,0,data_out['CC_RAT_AMT_OVERDUE_3M'])
data_out['COMBIN_CC_RAT_AMT_OVERDUE_5_10_DPD_3M']=np.where(data_out['
    FLAG_LINE_CC_3M']==0,0,data_out['CC_RAT_AMT_OVERDUE_5_10_DPD_3M'])
data_out['COMBIN_CC_RAT_AMT_OVERDUE_10_20_DPD_3M']=np.where(data_out['
    FLAG_LINE_CC_3M']==0,0,data_out['CC_RAT_AMT_OVERDUE_10_20_DPD_3M'])
data_out['COMBIN_CC_RAT_AMT_OVERDUE_20_30_DPD_3M']=np.where(data_out['
    FLAG_LINE_CC_3M']==0,0,data_out['CC_RAT_AMT_OVERDUE_20_30_DPD_3M'])
data_out['COMBIN_CC_RAT_AMT_OVERDUE_OVER_30_DPD_3M']=np.where(data_out['
    FLAG_LINE_CC_3M']==0,0,data_out['CC_RAT_AMT_OVERDUE_OVER_30_DPD_3M'])
data_out['COMBIN_CC_RAT_INST_QTY_OVERDUE_3M']=np.where(data_out['
    FLAG_LINE_CC_3M']==0,0,data_out['CC_RAT_INST_QTY_OVERDUE_3M'])
data_out['COMBIN_CC_RAT_INST_QTY_OVERDUE_5_10_DPD_3M']=np.where(data_out['
    FLAG_LINE_CC_3M']==0,0,data_out['CC_RAT_INST_QTY_OVERDUE_5_10_DPD_3M'])
data_out['COMBIN_CC_RAT_INST_QTY_OVERDUE_10_20_DPD_3M']=np.where(data_out['
    FLAG_LINE_CC_3M']==0,0,data_out['CC_RAT_INST_QTY_OVERDUE_10_20_DPD_3M'])
data_out['COMBIN_CC_RAT_INST_QTY_OVERDUE_20_30_DPD_3M']=np.where(data_out['
    FLAG_LINE_CC_3M']==0,0,data_out['CC_RAT_INST_QTY_OVERDUE_20_30_DPD_3M'])
data_out['COMBIN_CC_RAT_INST_QTY_OVERDUE_OVER_30_DPD_3M']=np.where(data_out['
    FLAG_LINE_CC_3M']==0,0,data_out['CC_RAT_INST_QTY_OVERDUE_OVER_30_DPD_3M'])

data_out['COMBIN_CC_MAX_QTY_DAYS_PAY_DELAY_6M']=np.where(data_out['
    FLAG_LINE_CC_6M']==0,0,data_out['CC_MAX_QTY_DAYS_PAY_DELAY_6M'])
data_out['COMBIN_CC_RAT_AMT_OVERDUE_6M']=np.where(data_out['FLAG_LINE_CC_6M'
    ]==0,0,data_out['CC_RAT_AMT_OVERDUE_6M'])
data_out['COMBIN_CC_RAT_AMT_OVERDUE_5_10_DPD_6M']=np.where(data_out['
    FLAG_LINE_CC_6M']==0,0,data_out['CC_RAT_AMT_OVERDUE_5_10_DPD_6M'])
data_out['COMBIN_CC_RAT_AMT_OVERDUE_10_20_DPD_6M']=np.where(data_out['
    FLAG_LINE_CC_6M']==0,0,data_out['CC_RAT_AMT_OVERDUE_10_20_DPD_6M'])
data_out['COMBIN_CC_RAT_AMT_OVERDUE_20_30_DPD_6M']=np.where(data_out['
    FLAG_LINE_CC_6M']==0,0,data_out['CC_RAT_AMT_OVERDUE_20_30_DPD_6M'])
data_out['COMBIN_CC_RAT_AMT_OVERDUE_OVER_30_DPD_6M']=np.where(data_out['
    FLAG_LINE_CC_6M']==0,0,data_out['CC_RAT_AMT_OVERDUE_OVER_30_DPD_6M'])
data_out['COMBIN_CC_RAT_INST_QTY_OVERDUE_6M']=np.where(data_out['
    FLAG_LINE_CC_6M']==0,0,data_out['CC_RAT_INST_QTY_OVERDUE_6M'])
data_out['COMBIN_CC_RAT_INST_QTY_OVERDUE_5_10_DPD_6M']=np.where(data_out['
    FLAG_LINE_CC_6M']==0,0,data_out['CC_RAT_INST_QTY_OVERDUE_5_10_DPD_6M'])
data_out['COMBIN_CC_RAT_INST_QTY_OVERDUE_10_20_DPD_6M']=np.where(data_out['
    FLAG_LINE_CC_6M']==0,0,data_out['CC_RAT_INST_QTY_OVERDUE_10_20_DPD_6M'])
data_out['COMBIN_CC_RAT_INST_QTY_OVERDUE_20_30_DPD_6M']=np.where(data_out['
    FLAG_LINE_CC_6M']==0,0,data_out['CC_RAT_INST_QTY_OVERDUE_20_30_DPD_6M'])
data_out['COMBIN_CC_RAT_INST_QTY_OVERDUE_OVER_30_DPD_6M']=np.where(data_out['
    FLAG_LINE_CC_6M']==0,0,data_out['CC_RAT_INST_QTY_OVERDUE_OVER_30_DPD_6M'])

data_out['COMBIN_CC_MAX_QTY_DAYS_PAY_DELAY_12M']=np.where(data_out['
    FLAG_LINE_CC_12M']==0,0,data_out['CC_MAX_QTY_DAYS_PAY_DELAY_12M'])
data_out['COMBIN_CC_RAT_AMT_OVERDUE_12M']=np.where(data_out['FLAG_LINE_CC_12M'
    ]==0,0,data_out['CC_RAT_AMT_OVERDUE_12M'])
data_out['COMBIN_CC_RAT_AMT_OVERDUE_5_10_DPD_12M']=np.where(data_out['
    FLAG_LINE_CC_12M']==0,0,data_out['CC_RAT_AMT_OVERDUE_5_10_DPD_12M'])
data_out['COMBIN_CC_RAT_AMT_OVERDUE_10_20_DPD_12M']=np.where(data_out['
    FLAG_LINE_CC_12M']==0,0,data_out['CC_RAT_AMT_OVERDUE_10_20_DPD_12M'])
data_out['COMBIN_CC_RAT_AMT_OVERDUE_20_30_DPD_12M']=np.where(data_out['
    FLAG_LINE_CC_12M']==0,0,data_out['CC_RAT_AMT_OVERDUE_20_30_DPD_12M'])
data_out['COMBIN_CC_RAT_AMT_OVERDUE_OVER_30_DPD_12M']=np.where(data_out['
    FLAG_LINE_CC_12M']==0,0,data_out['CC_RAT_AMT_OVERDUE_OVER_30_DPD_12M'])
```

```
data_out['COMBIN_CC_RAT_INST_QTY_OVERDUE_12M']=np.where(data_out['
    FLAG_LINE_CC_12M']==0,0,data_out['CC_RAT_INST_QTY_OVERDUE_12M'])
data_out['COMBIN_CC_RAT_INST_QTY_OVERDUE_5_10_DPD_12M']=np.where(data_out['
    FLAG_LINE_CC_12M']==0,0,data_out['CC_RAT_INST_QTY_OVERDUE_5_10_DPD_12M'])
data_out['COMBIN_CC_RAT_INST_QTY_OVERDUE_10_20_DPD_12M']=np.where(data_out['
    FLAG_LINE_CC_12M']==0,0,data_out['CC_RAT_INST_QTY_OVERDUE_10_20_DPD_12M'])
data_out['COMBIN_CC_RAT_INST_QTY_OVERDUE_20_30_DPD_12M']=np.where(data_out['
    FLAG_LINE_CC_12M']==0,0,data_out['CC_RAT_INST_QTY_OVERDUE_20_30_DPD_12M'])
data_out['COMBIN_CC_RAT_INST_QTY_OVERDUE_OVER_30_DPD_12M']=np.where(data_out['
    FLAG_LINE_CC_12M']==0,0,data_out['CC_RAT_INST_QTY_OVERDUE_OVER_30_DPD_12M'
    ])

mean_nulos = data_in[data_in['COMBIN_MAX_ATRASO_ACTIVE_CRED'].isnull()]['
    FLAG_DEFAULT_MES'].mean()
decil_80 = data_in['COMBIN_MAX_ATRASO_ACTIVE_CRED'].quantile(.80)
prob_80 = data_in[data_in['COMBIN_MAX_ATRASO_ACTIVE_CRED']>decil_80]['
    FLAG_DEFAULT_MES'].mean()
decil_60 = data_in['COMBIN_MAX_ATRASO_ACTIVE_CRED'].quantile(.60)
prob_60 = data_in[(data_in['COMBIN_MAX_ATRASO_ACTIVE_CRED']>decil_60)&(data_in
    ['COMBIN_MAX_ATRASO_ACTIVE_CRED']<decil_80)]['FLAG_DEFAULT_MES'].mean()
decil_40 = data_in['COMBIN_MAX_ATRASO_ACTIVE_CRED'].quantile(.40)
prob_40 = data_in[(data_in['COMBIN_MAX_ATRASO_ACTIVE_CRED']>decil_40)&(data_in
    ['COMBIN_MAX_ATRASO_ACTIVE_CRED']<decil_60)]['FLAG_DEFAULT_MES'].mean()
decil_20 = data_in['COMBIN_MAX_ATRASO_ACTIVE_CRED'].quantile(.20)
prob_20 = data_in[(data_in['COMBIN_MAX_ATRASO_ACTIVE_CRED']>decil_20)&(data_in
    ['COMBIN_MAX_ATRASO_ACTIVE_CRED']<decil_40)]['FLAG_DEFAULT_MES'].mean()
prob_0 = data_in[(data_in['COMBIN_MAX_ATRASO_ACTIVE_CRED']<decil_20)]['
    FLAG_DEFAULT_MES'].mean()

print('Default nulos:' , mean_nulos)
print('Percentil 80:', decil_80)
print('Default percentil 80 - 100:', prob_80)
print('Percentil 60:', decil_60)
print('Default percentil 60 - 80:', prob_60)
print('Percentil 40:', decil_40)
print('Default percentil 40 - 60:', prob_40)
print('Percentil 20:', decil_20)
print('Probabilidad percentil 20 - 40:' , prob_20)
print('Probabilidad percentil 0 - 20:' , prob_0)

data_in['COMBIN_MAX_ATRASO_ACTIVE_CRED'] = np.where(data_in['
    COMBIN_MAX_ATRASO_ACTIVE_CRED'].isnull(), 0, data_in['
    COMBIN_MAX_ATRASO_ACTIVE_CRED'])
data_out['COMBIN_MAX_ATRASO_ACTIVE_CRED'] = np.where(data_out['
    COMBIN_MAX_ATRASO_ACTIVE_CRED'].isnull(), 0, data_out['
    COMBIN_MAX_ATRASO_ACTIVE_CRED'])

mean_nulos = data_in[data_in['COMBIN_AVG_ATRASO_ACTIVE_CRED'].isnull()]['
    FLAG_DEFAULT_MES'].mean()
decil_80 = data_in['COMBIN_AVG_ATRASO_ACTIVE_CRED'].quantile(.80)
prob_80 = data_in[data_in['COMBIN_AVG_ATRASO_ACTIVE_CRED']>decil_80]['
    FLAG_DEFAULT_MES'].mean()
decil_60 = data_in['COMBIN_AVG_ATRASO_ACTIVE_CRED'].quantile(.60)
prob_60 = data_in[(data_in['COMBIN_AVG_ATRASO_ACTIVE_CRED']>decil_60)&(data_in
    ['COMBIN_AVG_ATRASO_ACTIVE_CRED']<decil_80)]['FLAG_DEFAULT_MES'].mean()
decil_40 = data_in['COMBIN_AVG_ATRASO_FINISHED_CRED'].quantile(.40)
prob_40 = data_in[(data_in['COMBIN_AVG_ATRASO_ACTIVE_CRED']>decil_40)&(data_in
    ['COMBIN_AVG_ATRASO_ACTIVE_CRED']<decil_60)]['FLAG_DEFAULT_MES'].mean()
decil_20 = data_in['COMBIN_AVG_ATRASO_ACTIVE_CRED'].quantile(.20)
prob_20 = data_in[(data_in['COMBIN_AVG_ATRASO_ACTIVE_CRED']>decil_20)&(data_in
    ['COMBIN_AVG_ATRASO_ACTIVE_CRED']<decil_40)]['FLAG_DEFAULT_MES'].mean()
```

```
prob_0 = data_in[(data_in['COMBIN_AVG_ATRASO_ACTIVE_CRED']<decil_20)]['
    FLAG_DEFAULT_MES'].mean()

print('Default nulos:' , mean_nulos)
print('Percentil 80:', decil_80)
print('Default percentil 80 - 100:', prob_80)
print('Percentil 60:', decil_60)
print('Default percentil 60 - 80:', prob_60)
print('Percentil 40:', decil_40)
print('Default percentil 40 - 60:', prob_40)
print('Percentil 20:', decil_20)
print('Probabilidad percentil 20 - 40:' , prob_20)
print('Probabilidad percentil 0 - 20:' , prob_0)

data_in['COMBIN_AVG_ATRASO_ACTIVE_CRED'] = np.where(data_in['
    COMBIN_AVG_ATRASO_ACTIVE_CRED'].isnull(), 0, data_in['
    COMBIN_AVG_ATRASO_ACTIVE_CRED'])
data_out['COMBIN_AVG_ATRASO_ACTIVE_CRED'] = np.where(data_out['
    COMBIN_AVG_ATRASO_ACTIVE_CRED'].isnull(), 0, data_out['
    COMBIN_AVG_ATRASO_ACTIVE_CRED'])

mean_nulos = data_in[data_in['MAX_ATRASO_CRED_DE'].isnull()]['FLAG_DEFAULT_MES
    '].mean()
decil_80 = data_in['MAX_ATRASO_CRED_DE'].quantile(.80)
prob_80 = data_in[data_in['MAX_ATRASO_CRED_DE']>decil_80]['FLAG_DEFAULT_MES'].
    mean()
decil_60 = data_in['MAX_ATRASO_CRED_DE'].quantile(.60)
prob_60 = data_in[(data_in['MAX_ATRASO_CRED_DE']>decil_60)&(data_in['
    MAX_ATRASO_CRED_DE']<decil_80)]['FLAG_DEFAULT_MES'].mean()
decil_40 = data_in['MAX_ATRASO_CRED_DE'].quantile(.40)
prob_40 = data_in[(data_in['MAX_ATRASO_CRED_DE']>decil_40)&(data_in['
    MAX_ATRASO_CRED_DE']<decil_60)]['FLAG_DEFAULT_MES'].mean()
decil_20 = data_in['MAX_ATRASO_CRED_DE'].quantile(.20)
prob_20 = data_in[(data_in['MAX_ATRASO_CRED_DE']>decil_20)&(data_in['
    MAX_ATRASO_CRED_DE']<decil_40)]['FLAG_DEFAULT_MES'].mean()
prob_0 = data_in[(data_in['MAX_ATRASO_CRED_DE']<decil_20)]['FLAG_DEFAULT_MES'
    ].mean()

print('Default nulos:' , mean_nulos)
print('Percentil 80:', decil_80)
print('Default percentil 80 - 100:', prob_80)
print('Percentil 60:', decil_60)
print('Default percentil 60 - 80:', prob_60)
print('Percentil 40:', decil_40)
print('Default percentil 40 - 60:', prob_40)
print('Percentil 20:', decil_20)
print('Probabilidad percentil 20 - 40:' , prob_20)
print('Probabilidad percentil 0 - 20:' , prob_0)

data_in['MAX_ATRASO_CRED_DE'] = np.where(data_in['MAX_ATRASO_CRED_DE'].isnull
    (), 500, data_in['MAX_ATRASO_CRED_DE'])
data_out['MAX_ATRASO_CRED_DE'] = np.where(data_out['MAX_ATRASO_CRED_DE'].
    isnull(), 500, data_out['MAX_ATRASO_CRED_DE'])

#filtrar obs nulas
data_in = data_in[data_in['COMBIN_MAX_ATRASO_FINISHED_CRED'].notnull()]
data_out = data_out[data_out['COMBIN_MAX_ATRASO_FINISHED_CRED'].notnull()]

data_in = data_in[data_in['COMBIN_AVG_ATRASO_FINISHED_CRED'].notnull()]
data_out = data_out[data_out['COMBIN_AVG_ATRASO_FINISHED_CRED'].notnull()]

data_in = data_in[data_in['TIPO_PERSONA'].notnull()]
```

```
data_out = data_out[data_out['TIPO_PERSONA'].notnull()]

data_in = data_in[data_in['MAX_AMOUNT_DE'].notnull()]
data_out = data_out[data_out['MAX_AMOUNT_DE'].notnull()]

data_in = data_in[data_in['SC_BUREAU'].notnull()]
data_out = data_out[data_out['SC_BUREAU'].notnull()]

data_in = data_in[data_in['TPV_M1_M2'].notnull()]
data_out = data_out[data_out['TPV_M1_M2'].notnull()]

data_in = data_in[data_in['TPV_15D_M1'].notnull()]
data_out = data_out[data_out['TPV_15D_M1'].notnull()]

data_in = data_in[data_in['TPV_7D_M1'].notnull()]
data_out = data_out[data_out['TPV_7D_M1'].notnull()]

data_in = data_in[data_in['TPV_7D_15D'].notnull()]
data_out = data_out[data_out['TPV_7D_15D'].notnull()]

data_in = data_in[data_in['TPN_M1_M2'].notnull()]
data_out = data_out[data_out['TPN_M1_M2'].notnull()]

data_in = data_in[data_in['TPN_7D_15D'].notnull()]
data_out = data_out[data_out['TPN_7D_15D'].notnull()]

#Armamos base de train y test

data_in['FLAG_TEST']=random.binomial(1, 0.2, size=len(data_in.
    CUS_CUST_ID_BORROWER)) ##A un 20% random le pongo el flag_test=1
data_test=data_in[data_in.FLAG_TEST==1].copy()
data_train=data_in[data_in.FLAG_TEST==0].copy()
data_test.to_pickle("Data_Test_Base_MLB_Instore_EO_2.pkl")
data_train.to_pickle("Data_Train_Base_MLB_Instore_EO_2.pkl")

data_test= pd.read_pickle("Data_Test_Base_MLB_Instore_EO_2.pkl")

data_test.shape

data_train= pd.read_pickle("Data_Train_Base_MLB_Instore_EO_2.pkl")

data_train.shape

print(data_in.CREDITOS.sum())

data_in_target_count= data_in.loc[data_in['FLAG_DEFAULT_MES'] == 1, '
    FLAG_DEFAULT_MES'].sum() #para todas las filas que tengan flag_default_mes
    =1, me traigo ese valor y lo sumo
data_in_target_count
data_in_target_count/(data_in.shape[0])*100
# Cantidad de usuarios por mes de foto
#print(data_in['CRD_PROP_CREATION_DATE_ID'].value_counts().sort_index())

print(data_train.shape)
# Esta base fue compuesta por 408k cr ditos
print(data_train.CREDITOS.sum())

data_train_target_count= data_train.loc[data_train['FLAG_DEFAULT_MES'] == 1, '
    FLAG_DEFAULT_MES'].sum()
data_train_target_count
data_train_target_count/(data_train.shape[0])*100
```

```
# Cantidad de usuarios por mes de foto
#print(data_train['CRD_PROP_CREATION_DATE_ID'].value_counts().sort_index())

# IN SAMPLE: TEST OUT OF SAMPLE (20%)
# Tenemos 217 variables y 73k usuarios.mes
print(data_test.shape)
# Esta base fue compuesta por 102k cr ditos
print(data_test.CREDITOS.sum())

data_test_target_count= data_test.loc[data_test['FLAG_DEFAULT_MES'] == 1, '
    FLAG_DEFAULT_MES'].sum()
data_test_target_count
data_test_target_count/(data_test.shape[0])*100
# Cantidad de usuarios por mes de foto
#print(data_test['CRD_PROP_CREATION_DATE_ID'].value_counts().sort_index())

# OUT OF TIME
# Tenemos 216 variables y 39k usuarios.mes
print(data_out.shape)
# Esta base fue compuesta por 42k cr ditos
print(data_out.CREDITOS.sum())

data_out_target_count= data_out.loc[data_out['FLAG_DEFAULT_MES'] == 1, '
    FLAG_DEFAULT_MES'].sum()
data_out_target_count
data_out_target_count/(data_out.shape[0])*100
# Cantidad de usuarios por mes de foto
#print(data_out['CRD_PROP_CREATION_DATE_ID'].value_counts().sort_index())

#Dummies

training_set_dummies = pd.get_dummies(data_train, columns = ["TAG_OFERTA_DE",
    "TIPO_PERSONA", "TIPO_DEVICE", "FLAG_PPV_ACTIVE_CRED", "
    FLAG_ACTIVE_LAST_PROPOSAL", "KILLER_TRAVAS_RECIVIBLES", "
    TAG_TIPO_BORROWER_DE","TAG_POINT", "TAG_PRODUCTO", "FLAG_FDV"], dummy_na =
     True)

test_set_dummies = pd.get_dummies(data_test, columns = ["TAG_OFERTA_DE", "
    TIPO_PERSONA", "TIPO_DEVICE", "FLAG_PPV_ACTIVE_CRED", "
    FLAG_ACTIVE_LAST_PROPOSAL", "KILLER_TRAVAS_RECIVIBLES","
    TAG_TIPO_BORROWER_DE","TAG_POINT", "TAG_PRODUCTO", "FLAG_FDV"], dummy_na =
     True)

oot_set_dummies = pd.get_dummies(data_out, columns = ["TAG_OFERTA_DE", "
    TIPO_PERSONA", "TIPO_DEVICE", "FLAG_PPV_ACTIVE_CRED", "
    FLAG_ACTIVE_LAST_PROPOSAL", "KILLER_TRAVAS_RECIVIBLES","
    TAG_TIPO_BORROWER_DE","TAG_POINT", "TAG_PRODUCTO", "FLAG_FDV"], dummy_na =
     True)

training_set_dummies_pre = training_set_dummies.copy()

mask_test = [not feature in list(test_set_dummies.columns) for feature in list
    (training_set_dummies.columns)]
print(training_set_dummies_pre.columns[mask_test])

mask_oot = [not feature in list(oot_set_dummies.columns) for feature in list(
    training_set_dummies.columns)]
print(training_set_dummies_pre.columns[mask_oot])

oot_set_dummies['TAG_OFERTA_DE_NO OFERTABLE'] = 0

#Armamos bases finales
```

```
features = [
'FLAG_SCORE_CHURN',
'FLAG_PPV_CF',
'FLAG_ACTIVE_PPV_CF',
'FLAG_SCORE_BHV',
'FLAG_LINE_CC_USE',
'FLAG_LINE_CC_USE_1M',
'FLAG_LINE_CC_USE_3M',
'FLAG_LINE_CC_USE_6M',
'FLAG_LINE_CC_USE_12M',
'FLAG_LINE_CC_USE_vs_1',
'FLAG_LINE_CC_USE_vs_3',
'FLAG_LINE_CC_USE_vs_6',
'FLAG_LINE_CC_USE_vs_12',
'FLAG_UPSELL',
'FLAG_DOWNSELL',
'FLAG_LINE_CC',
'FLAG_LINE_CC_1M',
'FLAG_LINE_CC_3M',
'FLAG_LINE_CC_6M',
'FLAG_LINE_CC_12M',
'COMBIN_SCORE_CHURN',
#'SCORE_CHURN',
'CRD_PROP_TOTAL_AMOUNT',
'CONSEC_ACTIVITY_LY',
'ACTIVITY',
'WEEKLY_ACTIVITY_L3M',
'FLAG_POINT_PRO',
'Q_DEVICES',
#'OLD_Q_DAYS_DE',
'MAX_AMOUNT_DE',
'RATIO_CLAIMS',
'Q_FINISHED_CRED',
'COMBIN_SUM_AMOUNT_FINISHED_CREDIT',
#'SUM_AMOUNT_FINISHED_CREDIT',
'COMBIN_LAST_DELAY_PPV_CF',
#'LAST_DELAY_PPV_CF',
'COMBIN_MAX_ATRASO_FINISHED_CRED',
#'MAX_ATRASO_FINISHED_CRED',
'COMBIN_AVG_ATRASO_FINISHED_CRED',
#'AVG_ATRASO_FINISHED_CRED',
'COMBIN_PROP_AMOUNT_LAST_FINISHED_CRED',
#'PROP_AMOUNT_LAST_FINISHED_CRED',
'Q_ACTIVE_CRED',
'COMBIN_SUM_AMOUNT_ACTIVE_CRED',
#'SUM_AMOUNT_ACTIVE_CRED',
'DEBT_BALANCE',
#'PORCENTAJE_PAGO',
'COMBIN_PORCENTAJE_PAGO',
'INST_TOTAL_DEBT_BALANCE',
'COMBIN_MAX_ATRASO_ACTIVE_CRED',
#'MAX_ATRASO_ACTIVE_CRED',
'COMBIN_AVG_ATRASO_ACTIVE_CRED',
#'AVG_ATRASO_ACTIVE_CRED',
'MAX_ATRASO_CRED_DE',
'NO_APROBABLE_CF',
'COMBIN_Q_CUOTAS_PAGAS_1L_ACTIVE',
#'Q_CUOTAS_PAGAS_1L_ACTIVE',
'COMBIN_Q_CUOTAS_IMPAGAS_1L_ACTIVE',
#'Q_CUOTAS_IMPAGAS_1L_ACTIVE',
#'PLAZO_L1',
```

```
'COMBIN_PLAZO_L1',
'COMBIN_Q_CUOTAS_PAGAS_2L_ACTIVE',
#'Q_CUOTAS_PAGAS_2L_ACTIVE',
'COMBIN_Q_CUOTAS_IMPAGAS_2L_ACTIVE',
#'Q_CUOTAS_IMPAGAS_2L_ACTIVE',
'FLAG_MIA',
'SC_BUREAU',
'COMBIN_PROB_BHV_1L',
#'PROB_BHV_1L',
'Q_CUENTAS_VINCULADAS',
'Q_REPEAT',
'TPN_7D',
'TPN_15D',
'TPN_M1',
'TPN_M2',
'TPN_M3',
'TPN_M4',
'TPN_M5',
'TPN_M6',
'TPN_AVG_3M',
'TPN_AVG_6M',
'TPV_LC_7D',
'TPV_LC_15D',
'TPV_LC_M1',
'TPV_LC_M2',
'TPV_LC_M3',
'TPV_LC_M4',
'TPV_LC_M5',
'TPV_LC_M6',
'TPV_LC_AVG_3M',
'TPV_LC_AVG_6M',
'TPV_LC_SMOOTHED_ADJ_15D',
'MONTHLY_TPV_LC',
'WEEKLY_TPV_LC',
'TPV_M1_L3',
'TPV_M1_M2',
'TPV_15D_M1',
'TPV_7D_M1',
'TPV_7D_15D',
'TPN_M1_L3',
'TPN_M1_M2',
'TPN_15D_M1',
'TPN_7D_M1',
'TPN_7D_15D',
'FLAG_CHURNEO_EVER',
'WEEKLY_MULTIPLIER',
'RCI_TOTAL',
'CC_INST_AVG_AMOUNT',
'RCI_TOTAL_CON_CONSUMERS',
'TAG_OFERTA_DE_1ST RULE',
'TAG_OFERTA_DE_2ND RULE',
'TAG_OFERTA_DE_NO OFERTABLE',
'TAG_OFERTA_DE_REGLA BAU',
'TAG_OFERTA_DE_nan',
'TIPO_PERSONA_PF',
'TIPO_PERSONA_PJ',
'TIPO_PERSONA_nan',
'TIPO_DEVICE_OTRO',
'TIPO_DEVICE_POINT_MINI / H / BLUE',
'TIPO_DEVICE_POINT_SMART',
'TIPO_DEVICE_nan',
'FLAG_PPV_ACTIVE_CRED_0.0',
```

```
'FLAG_PPV_ACTIVE_CRED_1.0',
'FLAG_PPV_ACTIVE_CRED_nan',
#'FLAG_ACTIVE_LAST_PROPOSAL_0.0',
#'FLAG_ACTIVE_LAST_PROPOSAL_1.0',
#'FLAG_ACTIVE_LAST_PROPOSAL_nan',
'KILLER_TRAVAS_RECIVIBLES_0.0',
'KILLER_TRAVAS_RECIVIBLES_1.0',
'KILLER_TRAVAS_RECIVIBLES_nan',
'TAG_TIPO_BORROWER_DE_NEW',
'TAG_TIPO_BORROWER_DE_OLD',
'TAG_TIPO_BORROWER_DE_REPEAT',
'TAG_TIPO_BORROWER_DE_nan',
'TAG_POINT_POINT',
'TAG_POINT_POINT_PRO',
'TAG_POINT_QR',
'TIPO_DEVICE_POINT_I_STANDALONE',
'TIPO_DEVICE_POINT_PRO',
'TAG_POINT_nan',
'CC_CREDIT_LINE_AMT',
'COMBIN_CC_RAT_CREDIT_LINE_USE',
#'CC_RAT_CREDIT_LINE_USE',
'COMBIN_CC_RAT_CREDIT_LINE_USE_1M',
#'CC_RAT_CREDIT_LINE_USE_1M',
'COMBIN_CC_RAT_CREDIT_LINE_USE_3M',
#'CC_RAT_CREDIT_LINE_USE_3M',
'COMBIN_CC_RAT_CREDIT_LINE_USE_6M',
#'CC_RAT_CREDIT_LINE_USE_6M',
'COMBIN_CC_RAT_CREDIT_LINE_USE_12M',
#'CC_RAT_CREDIT_LINE_USE_12M',
'COMBIN_CC_DIF_CREDIT_LINE_USE_VS_1M',
#'CC_DIF_CREDIT_LINE_USE_VS_1M',
'COMBIN_CC_DIF_CREDIT_LINE_USE_VS_3M',
#'CC_DIF_CREDIT_LINE_USE_VS_3M',
'COMBIN_CC_DIF_CREDIT_LINE_USE_VS_6M',
#'CC_DIF_CREDIT_LINE_USE_VS_6M',
'COMBIN_CC_DIF_CREDIT_LINE_USE_VS_12M',
#'CC_DIF_CREDIT_LINE_USE_VS_12M',
'COMBIN_CC_DAYS_SINCE_LAST_UPSELL',
#'CC_DAYS_SINCE_LAST_UPSELL',
'COMBIN_CC_LAST_UPSELL_RAT_AMT',
#'CC_LAST_UPSELL_RAT_AMT',
'COMBIN_CC_UPSELL_MIN_RAT_AMT',
#'CC_UPSELL_MIN_RAT_AMT',
'COMBIN_CC_UPSELL_AVG_RAT_AMT',
#'CC_UPSELL_AVG_RAT_AMT',
'COMBIN_CC_UPSELL_MAX_RAT_AMT',
#'CC_UPSELL_MAX_RAT_AMT',
'COMBIN_CC_DAYS_SINCE_LAST_DOWNSELL',
#'CC_DAYS_SINCE_LAST_DOWNSELL',
'COMBIN_CC_LAST_DOWNSELL_RAT_AMT',
#'CC_LAST_DOWNSELL_RAT_AMT',
'COMBIN_CC_DOWNSELL_MIN_RAT_AMT',
#'CC_DOWNSELL_MIN_RAT_AMT',
'COMBIN_CC_DOWNSELL_AVG_RAT_AMT',
#'CC_DOWNSELL_AVG_RAT_AMT',
'COMBIN_CC_DOWNSELL_MAX_RAT_AMT',
#'CC_DOWNSELL_MAX_RAT_AMT',
#'CC_RAT_CRED_AMT_PAID',
'COMBIN_CC_MAX_QTY_DAYS_PAY_DELAY',
#'CC_MAX_QTY_DAYS_PAY_DELAY',
'COMBIN_CC_MAX_QTY_DAYS_PAY_DELAY_1M',
#'CC_MAX_QTY_DAYS_PAY_DELAY_1M',
```

```
'COMBIN_CC_MAX_QTY_DAYS_PAY_DELAY_3M',
#'CC_MAX_QTY_DAYS_PAY_DELAY_3M',
'COMBIN_CC_MAX_QTY_DAYS_PAY_DELAY_6M',
#'CC_MAX_QTY_DAYS_PAY_DELAY_6M',
'COMBIN_CC_MAX_QTY_DAYS_PAY_DELAY_12M',
#'CC_MAX_QTY_DAYS_PAY_DELAY_12M',
'COMBIN_CC_RAT_AMT_OVERDUE',
#'CC_RAT_AMT_OVERDUE',
'COMBIN_CC_RAT_AMT_OVERDUE_1M',
#'CC_RAT_AMT_OVERDUE_1M',
'COMBIN_CC_RAT_AMT_OVERDUE_3M',
#'CC_RAT_AMT_OVERDUE_3M',
'COMBIN_CC_RAT_AMT_OVERDUE_6M',
#'CC_RAT_AMT_OVERDUE_6M',
'COMBIN_CC_RAT_AMT_OVERDUE_12M',
#'CC_RAT_AMT_OVERDUE_12M',
'COMBIN_CC_RAT_AMT_OVERDUE_5_10_DPD',
#'CC_RAT_AMT_OVERDUE_5_10_DPD',
'COMBIN_CC_RAT_AMT_OVERDUE_5_10_DPD_1M',
#'CC_RAT_AMT_OVERDUE_5_10_DPD_1M',
'COMBIN_CC_RAT_AMT_OVERDUE_5_10_DPD_3M',
#'CC_RAT_AMT_OVERDUE_5_10_DPD_3M',
'COMBIN_CC_RAT_AMT_OVERDUE_5_10_DPD_6M',
'COMBIN_CC_RAT_AMT_OVERDUE_5_10_DPD_12M',
#'CC_RAT_AMT_OVERDUE_5_10_DPD_6M',
#'CC_RAT_AMT_OVERDUE_5_10_DPD_12M',
'COMBIN_CC_RAT_AMT_OVERDUE_10_20_DPD',
#'CC_RAT_AMT_OVERDUE_10_20_DPD',
'COMBIN_CC_RAT_AMT_OVERDUE_10_20_DPD_1M',
#'CC_RAT_AMT_OVERDUE_10_20_DPD_1M',
'COMBIN_CC_RAT_AMT_OVERDUE_10_20_DPD_3M',
#'CC_RAT_AMT_OVERDUE_10_20_DPD_3M',
'COMBIN_CC_RAT_AMT_OVERDUE_10_20_DPD_6M',
#'CC_RAT_AMT_OVERDUE_10_20_DPD_6M',
'COMBIN_CC_RAT_AMT_OVERDUE_5_10_DPD_12M',
#'CC_RAT_AMT_OVERDUE_10_20_DPD_12M',
'COMBIN_CC_RAT_AMT_OVERDUE_20_30_DPD',
#'CC_RAT_AMT_OVERDUE_20_30_DPD',
'COMBIN_CC_RAT_AMT_OVERDUE_20_30_DPD_1M',
#'CC_RAT_AMT_OVERDUE_20_30_DPD_1M',
'COMBIN_CC_RAT_AMT_OVERDUE_20_30_DPD_3M',
#'CC_RAT_AMT_OVERDUE_20_30_DPD_3M',
'COMBIN_CC_RAT_AMT_OVERDUE_20_30_DPD_6M',
#'CC_RAT_AMT_OVERDUE_20_30_DPD_6M',
'COMBIN_CC_RAT_AMT_OVERDUE_20_30_DPD_12M',
#'CC_RAT_AMT_OVERDUE_20_30_DPD_12M',
'COMBIN_CC_RAT_AMT_OVERDUE_OVER_30_DPD',
#'CC_RAT_AMT_OVERDUE_OVER_30_DPD',
'COMBIN_CC_RAT_AMT_OVERDUE_OVER_30_DPD_1M',
#'CC_RAT_AMT_OVERDUE_OVER_30_DPD_1M',
'COMBIN_CC_RAT_AMT_OVERDUE_OVER_30_DPD_3M',
#'CC_RAT_AMT_OVERDUE_OVER_30_DPD_3M',
'COMBIN_CC_RAT_AMT_OVERDUE_OVER_30_DPD_6M',
#'CC_RAT_AMT_OVERDUE_OVER_30_DPD_6M',
'COMBIN_CC_RAT_AMT_OVERDUE_OVER_30_DPD_12M',
#'CC_RAT_AMT_OVERDUE_OVER_30_DPD_12M',
'COMBIN_CC_RAT_INST_QTY_OVERDUE',
#'CC_RAT_INST_QTY_OVERDUE',
'COMBIN_CC_RAT_INST_QTY_OVERDUE_1M',
#'CC_RAT_INST_QTY_OVERDUE_1M',
'COMBIN_CC_RAT_INST_QTY_OVERDUE_3M',
#'CC_RAT_INST_QTY_OVERDUE_3M',
```

```
'COMBIN_CC_RAT_INST_QTY_OVERDUE_6M',
#'CC_RAT_INST_QTY_OVERDUE_6M',
'COMBIN_CC_RAT_INST_QTY_OVERDUE_12M',
#'CC_RAT_INST_QTY_OVERDUE_12M',
'COMBIN_CC_RAT_INST_QTY_OVERDUE_5_10_DPD',
#'CC_RAT_INST_QTY_OVERDUE_5_10_DPD',
'COMBIN_CC_RAT_INST_QTY_OVERDUE_5_10_DPD_1M',
#'CC_RAT_INST_QTY_OVERDUE_5_10_DPD_1M',
'COMBIN_CC_RAT_INST_QTY_OVERDUE_5_10_DPD_3M',
#'CC_RAT_INST_QTY_OVERDUE_5_10_DPD_3M',
'COMBIN_CC_RAT_INST_QTY_OVERDUE_5_10_DPD_6M',
#'CC_RAT_INST_QTY_OVERDUE_5_10_DPD_6M',
'COMBIN_CC_RAT_INST_QTY_OVERDUE_5_10_DPD_12M',
#'CC_RAT_INST_QTY_OVERDUE_5_10_DPD_12M',
'COMBIN_CC_RAT_INST_QTY_OVERDUE_10_20_DPD',
#'CC_RAT_INST_QTY_OVERDUE_10_20_DPD',
'COMBIN_CC_RAT_INST_QTY_OVERDUE_10_20_DPD_1M',
#'CC_RAT_INST_QTY_OVERDUE_10_20_DPD_1M',
'COMBIN_CC_RAT_INST_QTY_OVERDUE_10_20_DPD_3M',
#'CC_RAT_INST_QTY_OVERDUE_10_20_DPD_3M',
'COMBIN_CC_RAT_INST_QTY_OVERDUE_10_20_DPD_6M',
#'CC_RAT_INST_QTY_OVERDUE_10_20_DPD_6M',
'COMBIN_CC_RAT_INST_QTY_OVERDUE_10_20_DPD_12M',
#'CC_RAT_INST_QTY_OVERDUE_10_20_DPD_12M',
'COMBIN_CC_RAT_INST_QTY_OVERDUE_20_30_DPD',
#'CC_RAT_INST_QTY_OVERDUE_20_30_DPD',
'COMBIN_CC_RAT_INST_QTY_OVERDUE_20_30_DPD_1M',
#'CC_RAT_INST_QTY_OVERDUE_20_30_DPD_1M',
'COMBIN_CC_RAT_INST_QTY_OVERDUE_20_30_DPD_3M',
#'CC_RAT_INST_QTY_OVERDUE_20_30_DPD_3M',
'COMBIN_CC_RAT_INST_QTY_OVERDUE_20_30_DPD_6M',
#'CC_RAT_INST_QTY_OVERDUE_20_30_DPD_6M',
'COMBIN_CC_RAT_INST_QTY_OVERDUE_20_30_DPD_12M',
#'CC_RAT_INST_QTY_OVERDUE_20_30_DPD_12M',
'COMBIN_CC_RAT_INST_QTY_OVERDUE_OVER_30_DPD',
#'CC_RAT_INST_QTY_OVERDUE_OVER_30_DPD',
'COMBIN_CC_RAT_INST_QTY_OVERDUE_OVER_30_DPD_1M',
#'CC_RAT_INST_QTY_OVERDUE_OVER_30_DPD_1M',
'COMBIN_CC_RAT_INST_QTY_OVERDUE_OVER_30_DPD_3M',
#'CC_RAT_INST_QTY_OVERDUE_OVER_30_DPD_3M',
#'CC_RAT_INST_QTY_OVERDUE_OVER_30_DPD_6M',
'COMBIN_CC_RAT_INST_QTY_OVERDUE_OVER_30_DPD_6M',
'COMBIN_CC_RAT_INST_QTY_OVERDUE_OVER_30_DPD_12M',
#'CC_RAT_INST_QTY_OVERDUE_OVER_30_DPD_12M',
'FLAG_FDV_0.0',
'FLAG_FDV_1.0',
'FLAG_FDV_nan'
]

X_train = training_set_dummies.loc[:,features]
y_train = training_set_dummies['FLAG_DEFAULT_MES']

X_test = test_set_dummies.loc[:,features]
y_test = test_set_dummies['FLAG_DEFAULT_MES']

X_oot = oot_set_dummies.loc[:,features]
y_oot = oot_set_dummies['FLAG_DEFAULT_MES']

#Entrenamiento

k_folds = 5
n_samples = len(training_set_dummies)
```

```
fold_size = n_samples // k_folds

hyperparameter_space_RFC =  {
                            'n_estimators': [50, 100],
                            'max_features': [30, 50],
                            'max_depth': [20, 30, 40],
 #                           'min_samples_leaf':[10],
                            'min_samples_split': [100],
                            'class_weight': ['balanced'],
                            'random_state': [42]
                            }

best_resultados = 0.0
best_params = {}

for n_estimators in hyperparameter_space_RFC['n_estimators']:
    for max_depth in hyperparameter_space_RFC['max_depth']:
      for max_features in hyperparameter_space_RFC['max_features']:
          # Inicializar el clasificador RandomForest con los hiperpar metros
              actuales
          rf_classifier = RandomForestClassifier(n_estimators=n_estimators,
              max_depth=max_depth,max_features=max_features,class_weight='
              balanced',
                                                 random_state=42,
                                                     min_samples_split=100)

          # Almacenar las puntuaciones de precisi n para cada partici n
          resultados = []

          for fold in range(k_folds):
              # Obtener los  ndices  de los datos para el fold actual
              start_idx = fold * fold_size
              end_idx = start_idx + fold_size

              # Dividir los datos en datos de entrenamiento y prueba
              X_train_cv = X_train.loc[0:start_idx,]
              X_train_cv=X_train_cv.append(X_train.loc[end_idx:,])
              X_test_cv = X_train.loc[start_idx:end_idx,]
              y_train_cv = y_train.loc[0:start_idx,]
              y_train_cv=y_train_cv.append(y_train.loc[end_idx:,])
              y_test_cv = y_train.loc[start_idx:end_idx,]

              base_train_cv=training_set_dummies.loc[0:start_idx,]
              base_train_cv=base_train_cv.append(training_set_dummies.loc[
                  end_idx:,])
              base_test_cv = training_set_dummies.loc[start_idx:end_idx,]

              # Entrenar el clasificador
              modelo=rf_classifier.fit(X_train_cv, y_train_cv)

              # Evaluar el rendimiento en el fold actual
              y_probas = modelo.predict_proba(X_test_cv) #devuelve
                  probabilidad (y)
              df_y_probas= pd.DataFrame(y_probas)
              base_test_cv['prediccion']=df_y_probas[1]
              resultado_esperado=(1-base_test_cv['prediccion'])*base_test_cv['
                  INTERESES']-base_test_cv['prediccion']*base_test_cv['
                  MONTO_DEFAULT']
              resultado_esperado_total=np.sum(resultado_esperado)
              resultados.append(resultado_esperado_total)
          # Calcular la precisi n promedio para los hiperpar metros actuales
          avg_resultados = np.mean(resultados)
```

```
            # Actualizar los mejores hiperparmetros si encontramos un mejor
                rendimiento
            if avg_resultados > best_resultados:
                best_resultados = avg_resultados
                best_params = {'n_estimators': n_estimators, 'max_depth':
                    max_depth,'max_features': max_features}

# Paso 5: Imprimir los mejores hiperparmetros y su rendimiento
print("Mejores hiperparmetros:", best_params)
print("Mejor revenue promedio: {:.2f}".format(best_resultados))

print("Mejores hiperparmetros:", best_params)
print("Mejor revenue promedio: {:.2f}".format(best_resultados))

RFC=RandomForestClassifier(criterion='entropy',max_depth=20,max_features=30,
    n_estimators=50,random_state=42,class_weight='balanced',min_samples_split
    =100)
RFC_fitted=RFC.fit(X_train,y_train)

pickle.dump(RFC_fitted,open('RFC_fitted[MLB_INSTORE_EO_5_MELI].sav','wb'))

RFC_fitted_5 = pickle.load(open('/content/drive/MyDrive/Tesis/RFC_fitted[
    MLB_INSTORE_EO_5_MELI].sav', 'rb'))

#Predecimos

y_probas_train_RFC_3 = RFC_fitted_3.predict_proba(X_train) #devuelve
    probabilidad (y)
df_y_probas_train_RFC_3 = pd.DataFrame(y_probas_train_RFC_3)

y_probas_test_RFC_3 = RFC_fitted_3.predict_proba(X_test)  #devuelve
    probabilidad (y)
df_y_probas_test_RFC_3 = pd.DataFrame(y_probas_test_RFC_3)

y_probas_oot_RFC_3 = RFC_fitted_3.predict_proba(X_oot)  #devuelve probabilidad
     (y)
df_y_probas_oot_RFC_3 = pd.DataFrame(y_probas_oot_RFC_3)

y_probas_train_RFC_5 = RFC_fitted_5.predict_proba(X_train) #devuelve
    probabilidad (y)
df_y_probas_train_RFC_5 = pd.DataFrame(y_probas_train_RFC_5)

y_probas_test_RFC_5 = RFC_fitted_5.predict_proba(X_test)  #devuelve
    probabilidad (y)
df_y_probas_test_RFC_5 = pd.DataFrame(y_probas_test_RFC_5)

y_probas_oot_RFC_5 = RFC_fitted_5.predict_proba(X_oot)  #devuelve probabilidad
     (y)
df_y_probas_oot_RFC_5 = pd.DataFrame(y_probas_oot_RFC_5)

#Elegimos el punto de corte

def to_labels(probabilidades, threshold):
    """Esta funci n convierte un array de probabilidades en una predicci n
        binaria, dada una probabilidad threshold."""
    return (probabilidades >= threshold).astype('int')

def perdida(y_true, y_pred):
    """Esta funci n, dados un array de predicci n binaria y el array de
        etiquetas verdaderas, computa la p rdida monetaria
```

```
    que sufriramos de hacer esa prediccin. La prdida se computa como los
         intereses perdidos con los falsos positivos ms
    el saldo de deuda con los falsos negativos."""
    orden_def = np.where(y_pred==0,False,True)
    index_def = X_test.loc[orden_def,:].index
    orden_nodef = np.where(y_pred==1,False,True)
    index_nodef = X_test.loc[orden_nodef,:].index
    perdida_fp = sum(test_set_dummies['INTERESES'].loc[index_def])
    perdida_fn = sum(test_set_dummies['MONTO_DEFAULT'].loc[index_nodef])
    perdida = perdida_fp + perdida_fn
    return perdida

def to_labels(probabilidades, threshold):
    """Esta funcin convierte un array de probabilidades en una prediccin
         binaria, dada una probabilidad threshold."""
    return (probabilidades >= threshold).astype('int')

def perdida(y_true, y_pred):
    """Esta funcin, dados un array de prediccin binaria y el array de
         etiquetas verdaderas, computa la prdida monetaria
    que sufriramos de hacer esa prediccin. La prdida se computa como los
         intereses perdidos con los falsos positivos ms
    el saldo de deuda con los falsos negativos."""
    orden_def = np.where(y_pred==0,False,True)
    index_def = X_test.loc[orden_def,:].index
    orden_nodef = np.where(y_pred==1,False,True)
    index_nodef = X_test.loc[orden_nodef,:].index
    perdida_fp = sum(test_set_dummies['INTERESES'].loc[index_nodef])
    perdida_fn = sum(test_set_dummies['MONTO_DEFAULT'].loc[index_def])
    perdida = perdida_fp - perdida_fn
    return perdida

thresholds = np.arange(0, 1, 0.0001)
y_prob_test_RFC_5 = y_probas_test_RFC_5[:, 1]

perdida_fp_fn_RFC_5 = [perdida(y_test, to_labels(y_prob_test_RFC_5, t)) for t
    in thresholds]

ix_RFC_5 = np.argmin(perdida_fp_fn_RFC_5)
print('Threshold=%.3f, perdida=%.5f' % (thresholds[ix_RFC_5],
    perdida_fp_fn_RFC_5[ix_RFC_5]))

plt.title("Optimizando threshold")
plt.xlabel("Thresholds")
plt.ylabel("P rdida")
plt.plot(thresholds, perdida_fp_fn_RFC_5, color = "red")

#Evaluacin e impacto

y_prob_train_RFC_5 = y_probas_train_RFC_5[:, 1]
y_pred_train_RFC_5 = to_labels(y_prob_train_RFC_5,0.471)

y_prob_test_RFC_5 = y_probas_test_RFC_5[:, 1]
y_pred_test_RFC_5 = to_labels(y_prob_test_RFC_5,0.471)

y_prob_oot_RFC_5 = y_probas_oot_RFC_5[:, 1]
y_pred_oot_RFC_5 = to_labels(y_prob_oot_RFC_5,0.471)

metrics.confusion_matrix(y_test,y_pred_test_RFC_5)

print(classification_report(y_test, y_pred_test_RFC_5))
```

```
metrics.confusion_matrix(y_oot,y_pred_oot_RFC_5)

print(classification_report(y_oot, y_pred_oot_RFC_5))

fpr_RFC_train, tpr_RFC_train, _ = roc_curve(y_train, y_prob_train_RFC_5)
roc_auc_RFC_train = round(auc(fpr_RFC_train, tpr_RFC_train),3)
fpr_RFC_test, tpr_RFC_test, _ = roc_curve(y_test, y_prob_test_RFC_5)
roc_auc_RFC_test = round(auc(fpr_RFC_test, tpr_RFC_test),3)
fpr_RFC_oot, tpr_RFC_oot, _ = roc_curve(y_oot, y_prob_oot_RFC_5)
roc_auc_RFC_oot = round(auc(fpr_RFC_oot, tpr_RFC_oot),3)


plt.figure()
lw = 2
plt.plot(fpr_RFC_test, tpr_RFC_test, color='darkorange', lw=lw, label='SVC AUC
     OOS = %0.3f' % roc_auc_RFC_test)
plt.plot(fpr_RFC_oot, tpr_RFC_oot, color='green', lw=lw, label='SVC AUC OOT =
    %0.3f' % roc_auc_RFC_oot)
plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0]) ; plt.ylim([0.0, 1.05])
plt.xlabel('Tasa de Falsos Positivos') ; plt.ylabel('Tasa de Verdaderos
    Positivos')
plt.title('Curva ROC')
plt.legend(loc="lower right")
plt.show()

print('ROC en train:',roc_auc_RFC_train*100,'\nROC en test:',roc_auc_RFC_test
    *100,'\nROC en OOT:',roc_auc_RFC_oot*100)

#Impacto monetario

print('Accuracy Score:', metrics.accuracy_score(y_test,y_pred_test_RFC_5)*100,
     '\nPrecision Score:', metrics.precision_score(y_test,y_pred_test_RFC_5)
    *100, '\nRecall Score:', metrics.recall_score(y_test,y_pred_test_RFC_5)
    *100, '\nF1 Score:', metrics.f1_score(y_test,y_pred_test_RFC_5)*100)

index_test = X_test.index

revenues_test_pre = test_set_dummies['INTERESES'].loc[index_test].sum() -
    test_set_dummies['MONTO_DEFAULT'].loc[index_test].sum()

print('Cantidad de usuarios en test:',len(index_test),'\nCantidad de cr ditos
    :',test_set_dummies['CREDITOS'].loc[index_test].sum(), '\nMonto originado:
    ',test_set_dummies.ORIGINACION.sum(), '\nSu revenue:',revenues_test_pre,'\
    nSus intereses y punitorios: ', test_set_dummies['INTERESES'].loc[
    index_test].sum(),'\nSu default:', test_set_dummies['MONTO_DEFAULT'].loc[
    index_test].sum(), "\nSu loss_ratio:", test_set_dummies['MONTO_DEFAULT'].
    loc[index_test].sum() / test_set_dummies['INTERESES'].loc[index_test].sum
    ())

orden_no_def_test_RFC_5 = np.where(y_pred_test_RFC_5==0,True,False) #target =
    0 (predigo son no defaulteadores)
index_no_def_test_RFC_5 = X_test.loc[orden_no_def_test_RFC_5,:].index #me
    quedo con las filas cuyo predicted target = 0
revenues_test_post_RFC_5 = test_set_dummies['INTERESES'].loc[
    index_no_def_test_RFC_5].sum() - test_set_dummies['MONTO_DEFAULT'].loc[
    index_no_def_test_RFC_5].sum()
print('Cantidad de usuarios en test que predigo negativos:',len(
    index_no_def_test_RFC_5),'\nCantidad de cr ditos:',test_set_dummies['
    CREDITOS'].loc[index_no_def_test_RFC_5].sum(), '\nMonto originado:',
    test_set_dummies['ORIGINACION'][orden_no_def_test_RFC_5].sum(), '\nSu
    revenue:',revenues_test_post_RFC_5,'\nSus intereses y punitorios: ',
```

```
    test_set_dummies['INTERESES'].loc[index_no_def_test_RFC_5].sum(),'\nSu
    default:', test_set_dummies['MONTO_DEFAULT'].loc[index_no_def_test_RFC_5].
    sum(), "\nSu loss_ratio:", test_set_dummies['MONTO_DEFAULT'].loc[
    index_no_def_test_RFC_5].sum() / test_set_dummies['INTERESES'].loc[
    index_no_def_test_RFC_5].sum())

print('Accuracy Score:', metrics.accuracy_score(y_oot,y_pred_oot_RFC_5)*100, '
    \nPrecision Score:', metrics.precision_score(y_oot,y_pred_oot_RFC_5)*100,
    '\nRecall Score:', metrics.recall_score(y_oot,y_pred_oot_RFC_5)*100, '\nF1
     Score:', metrics.f1_score(y_oot,y_pred_oot_RFC_5)*100)

revenues_oot_pre = oot_set_dummies['INTERESES'].sum() - oot_set_dummies['
    MONTO_DEFAULT'].sum()
print('Cantidad de usuarios en oot:',oot_set_dummies.CUS_CUST_ID_BORROWER.
    count(),'\nCantidad de cr ditos:',oot_set_dummies.CREDITOS.sum(), '\
    nMonto originado:',oot_set_dummies.ORIGINACION.sum(),'\nSu revenue:',
    revenues_oot_pre,'\nSus intereses y punitorios: ', oot_set_dummies['
    INTERESES'].sum(),'\nSu default:', oot_set_dummies['MONTO_DEFAULT'].sum(),
     "\nSu loss_ratio:", oot_set_dummies['MONTO_DEFAULT'].sum() /
    oot_set_dummies['INTERESES'].sum())

orden_no_def_oot_RFC_5 = np.where(y_pred_oot_RFC_5==0,True,False) #mask de los
     predichos negativos en test
revenues_oot_post_RFC_5 = oot_set_dummies['INTERESES'][orden_no_def_oot_RFC_5
    ].sum() - oot_set_dummies['MONTO_DEFAULT'][orden_no_def_oot_RFC_5].sum()
print('Cantidad de usuarios que enciende:',sum(orden_no_def_oot_RFC_5),'\
    nCantidad de cr ditos:',oot_set_dummies['CREDITOS'][
    orden_no_def_oot_RFC_5].sum(),'\nMonto originado:',oot_set_dummies['
    ORIGINACION'][orden_no_def_oot_RFC_5].sum(), '\nSu revenue:',
    revenues_oot_post_RFC_5,'\nSus intereses y punitorios: ', oot_set_dummies[
    'INTERESES'][orden_no_def_oot_RFC_5].sum(),'\nSu default:',
    oot_set_dummies['MONTO_DEFAULT'][orden_no_def_oot_RFC_5].sum(), "\nSu
    loss_ratio:", oot_set_dummies['MONTO_DEFAULT'][orden_no_def_oot_RFC_5].sum
    () / oot_set_dummies['INTERESES'][orden_no_def_oot_RFC_5].sum())

#Importancia de las variables

importances_RFC = RFC_fitted_5.feature_importances_[RFC_fitted_5.
    feature_importances_>0]
columns_RFC = X_test.columns[RFC_fitted_5.feature_importances_>0]
feature_importances_RFC = pd.DataFrame(data = importances_RFC, index=
    columns_RFC, columns = ["importances"])
feature_importances_RFC = feature_importances_RFC.sort_values(by='importances'
    , ascending=False).head(20)
columnas = feature_importances_RFC.index
importances = feature_importances_RFC.importances * 100.00
plt.figure(figsize=(5,5))
sns.barplot(x=importances, y=columnas, orient = 'h', palette = "flare")
plt.xticks(rotation=90)
plt.title('Importancia de cada Feature')
plt.show()

#Resultados por rangos

training_set_dummies['MAX_AMOUNT_DE'].quantile(0.20)

training_set_dummies['MAX_AMOUNT_DE'].quantile(0.40)

training_set_dummies['MAX_AMOUNT_DE'].quantile(0.60)

training_set_dummies['MAX_AMOUNT_DE'].quantile(0.80)
```

```
base_test=test_set_dummies.copy()

base_oot=oot_set_dummies.copy()

base_test['prediccion']=y_pred_test_RFC_5

base_oot['prediccion']=y_pred_oot_RFC_5

base_test_q1=base_test[base_test['MAX_AMOUNT_DE']<=50]
base_test_q2=base_test[(base_test['MAX_AMOUNT_DE']>50)&(base_test['
    MAX_AMOUNT_DE']<=100)]
base_test_q3=base_test[(base_test['MAX_AMOUNT_DE']>100)&(base_test['
    MAX_AMOUNT_DE']<=200)]
base_test_q4=base_test[(base_test['MAX_AMOUNT_DE']>200)&(base_test['
    MAX_AMOUNT_DE']<=400)]
base_test_q5=base_test[(base_test['MAX_AMOUNT_DE']>400)&(base_test['
    MAX_AMOUNT_DE']<10000)]
base_test_q6=base_test[(base_test['MAX_AMOUNT_DE']>=10000)]

base_oot_q1=base_oot[base_oot['MAX_AMOUNT_DE']<=50]
base_oot_q2=base_oot[(base_oot['MAX_AMOUNT_DE']>50)&(base_oot['MAX_AMOUNT_DE'
    ]<=100)]
base_oot_q3=base_oot[(base_oot['MAX_AMOUNT_DE']>100)&(base_oot['MAX_AMOUNT_DE'
    ]<=200)]
base_oot_q4=base_oot[(base_oot['MAX_AMOUNT_DE']>200)&(base_oot['MAX_AMOUNT_DE'
    ]<=400)]
base_oot_q5=base_oot[(base_oot['MAX_AMOUNT_DE']>400)&(base_oot['MAX_AMOUNT_DE'
    ]<10000)]
base_oot_q6=base_oot[(base_oot['MAX_AMOUNT_DE']>=10000)]

print(classification_report(base_test_q1['FLAG_DEFAULT_MES'], base_test_q1['
    prediccion']))

revenues_test_pre_q1 = base_test_q1['INTERESES'].sum() - base_test_q1['
    MONTO_DEFAULT'].sum()
print('Cantidad de usuarios en test q1:',base_test_q1.CUS_CUST_ID_BORROWER.
    count(),'\nCantidad de cr ditos:',base_test_q1.CREDITOS.sum(), '\nMonto
    originado:',base_test_q1.ORIGINACION.sum(),'\nSu revenue:',
    revenues_test_pre_q1,'\nSus intereses y punitorios: ', base_test_q1['
    INTERESES'].sum(),'\nSu default:', base_test_q1['MONTO_DEFAULT'].sum(), "\
    nSu loss_ratio:", base_test_q1['MONTO_DEFAULT'].sum() / base_test_q1['
    INTERESES'].sum())

revenues_test_post_RFC_5_q1 = base_test_q1[base_test_q1['prediccion']==0]['
    INTERESES'].sum() - base_test_q1[base_test_q1['prediccion']==0]['
    MONTO_DEFAULT'].sum()
print('Cantidad de usuarios en test que predigo negativos:',len(base_test_q1[
    base_test_q1['prediccion']==0]['INTERESES']),'\nCantidad de cr ditos:',
    base_test_q1[base_test_q1['prediccion']==0]['CREDITOS'].sum(), '\nMonto
    originado:',base_test_q1[base_test_q1['prediccion']==0]['ORIGINACION'].sum
    (), '\nSu revenue:',revenues_test_post_RFC_5_q1,'\nSus intereses y
    punitorios: ', base_test_q1[base_test_q1['prediccion']==0]['INTERESES'].
    sum(),'\nSu default:', base_test_q1[base_test_q1['prediccion']==0]['
    MONTO_DEFAULT'].sum(), "\nSu loss_ratio:", base_test_q1[base_test_q1['
    prediccion']==0]['MONTO_DEFAULT'].sum() / base_test_q1[base_test_q1['
    prediccion']==0]['INTERESES'].sum())

print(classification_report(base_test_q2['FLAG_DEFAULT_MES'], base_test_q2['
    prediccion']))

revenues_test_pre_q2 = base_test_q2['INTERESES'].sum() - base_test_q2['
    MONTO_DEFAULT'].sum()
```

```
print('Cantidad de usuarios en test q2:',base_test_q2.CUS_CUST_ID_BORROWER.
    count(),'\nCantidad de cr ditos:',base_test_q2.CREDITOS.sum(), '\nMonto
    originado:',base_test_q2.ORIGINACION.sum(),'\nSu revenue:',
    revenues_test_pre_q2,'\nSus intereses y punitorios: ', base_test_q2['
    INTERESES'].sum(),'\nSu default:', base_test_q2['MONTO_DEFAULT'].sum(), "\
    nSu loss_ratio:", base_test_q2['MONTO_DEFAULT'].sum() / base_test_q2['
    INTERESES'].sum())

revenues_test_post_RFC_5_q2 = base_test_q2[base_test_q2['prediccion']==0]['
    INTERESES'].sum() - base_test_q2[base_test_q2['prediccion']==0]['
    MONTO_DEFAULT'].sum()
print('Cantidad de usuarios en test que predigo negativos:',len(base_test_q2[
    base_test_q2['prediccion']==0]['INTERESES']),'\nCantidad de cr ditos:',
    base_test_q2[base_test_q2['prediccion']==0]['CREDITOS'].sum(), '\nMonto
    originado:',base_test_q2[base_test_q2['prediccion']==0]['ORIGINACION'].sum
    (), '\nSu revenue:',revenues_test_post_RFC_5_q2,'\nSus intereses y
    punitorios: ', base_test_q2[base_test_q2['prediccion']==0]['INTERESES'].
    sum(),'\nSu default:', base_test_q2[base_test_q2['prediccion']==0]['
    MONTO_DEFAULT'].sum(), "\nSu loss_ratio:", base_test_q2[base_test_q2['
    prediccion']==0]['MONTO_DEFAULT'].sum() / base_test_q2[base_test_q2['
    prediccion']==0]['INTERESES'].sum())

print(classification_report(base_test_q3['FLAG_DEFAULT_MES'], base_test_q3['
    prediccion']))

revenues_test_pre_q3 = base_test_q3['INTERESES'].sum() - base_test_q3['
    MONTO_DEFAULT'].sum()
print('Cantidad de usuarios en test q3:',base_test_q3.CUS_CUST_ID_BORROWER.
    count(),'\nCantidad de cr ditos:',base_test_q3.CREDITOS.sum(), '\nMonto
    originado:',base_test_q3.ORIGINACION.sum(),'\nSu revenue:',
    revenues_test_pre_q3,'\nSus intereses y punitorios: ', base_test_q3['
    INTERESES'].sum(),'\nSu default:', base_test_q3['MONTO_DEFAULT'].sum(), "\
    nSu loss_ratio:", base_test_q3['MONTO_DEFAULT'].sum() / base_test_q3['
    INTERESES'].sum())

revenues_test_post_RFC_5_q3 = base_test_q3[base_test_q3['prediccion']==0]['
    INTERESES'].sum() - base_test_q3[base_test_q3['prediccion']==0]['
    MONTO_DEFAULT'].sum()
print('Cantidad de usuarios en test que predigo negativos:',len(base_test_q3[
    base_test_q3['prediccion']==0]['INTERESES']),'\nCantidad de cr ditos:',
    base_test_q3[base_test_q3['prediccion']==0]['CREDITOS'].sum(), '\nMonto
    originado:',base_test_q3[base_test_q3['prediccion']==0]['ORIGINACION'].sum
    (), '\nSu revenue:',revenues_test_post_RFC_5_q3,'\nSus intereses y
    punitorios: ', base_test_q3[base_test_q3['prediccion']==0]['INTERESES'].
    sum(),'\nSu default:', base_test_q3[base_test_q3['prediccion']==0]['
    MONTO_DEFAULT'].sum(), "\nSu loss_ratio:", base_test_q3[base_test_q3['
    prediccion']==0]['MONTO_DEFAULT'].sum() / base_test_q3[base_test_q3['
    prediccion']==0]['INTERESES'].sum())

print(classification_report(base_test_q4['FLAG_DEFAULT_MES'], base_test_q4['
    prediccion']))

revenues_test_pre_q4 = base_test_q4['INTERESES'].sum() - base_test_q4['
    MONTO_DEFAULT'].sum()
print('Cantidad de usuarios en test q4:',base_test_q4.CUS_CUST_ID_BORROWER.
    count(),'\nCantidad de cr ditos:',base_test_q4.CREDITOS.sum(), '\nMonto
    originado:',base_test_q4.ORIGINACION.sum(),'\nSu revenue:',
    revenues_test_pre_q4,'\nSus intereses y punitorios: ', base_test_q4['
    INTERESES'].sum(),'\nSu default:', base_test_q4['MONTO_DEFAULT'].sum(), "\
    nSu loss_ratio:", base_test_q4['MONTO_DEFAULT'].sum() / base_test_q4['
    INTERESES'].sum())
```

```
revenues_test_post_RFC_5_q4 = base_test_q4[base_test_q4['prediccion']==0]['
    INTERESES'].sum() - base_test_q4[base_test_q4['prediccion']==0]['
    MONTO_DEFAULT'].sum()
print('Cantidad de usuarios en test que predigo negativos:',len(base_test_q4[
    base_test_q4['prediccion']==0]['INTERESES']),'\nCantidad de cr ditos:',
    base_test_q4[base_test_q4['prediccion']==0]['CREDITOS'].sum(), '\nMonto
    originado:',base_test_q4[base_test_q4['prediccion']==0]['ORIGINACION'].sum
    (), '\nSu revenue:',revenues_test_post_RFC_5_q4,'\nSus intereses y
    punitorios: ', base_test_q4[base_test_q4['prediccion']==0]['INTERESES'].
    sum(),'\nSu default:', base_test_q4[base_test_q4['prediccion']==0]['
    MONTO_DEFAULT'].sum(), "\nSu loss_ratio:", base_test_q4[base_test_q4['
    prediccion']==0]['MONTO_DEFAULT'].sum() / base_test_q4[base_test_q4['
    prediccion']==0]['INTERESES'].sum())

print(classification_report(base_test_q5['FLAG_DEFAULT_MES'], base_test_q5['
    prediccion']))

revenues_test_pre_q5 = base_test_q5['INTERESES'].sum() - base_test_q5['
    MONTO_DEFAULT'].sum()
print('Cantidad de usuarios en test q5:',base_test_q5.CUS_CUST_ID_BORROWER.
    count(),'\nCantidad de cr ditos:',base_test_q5.CREDITOS.sum(), '\nMonto
    originado:',base_test_q5.ORIGINACION.sum(),'\nSu revenue:',
    revenues_test_pre_q5,'\nSus intereses y punitorios: ', base_test_q5['
    INTERESES'].sum(),'\nSu default:', base_test_q5['MONTO_DEFAULT'].sum(), "\
    nSu loss_ratio:", base_test_q5['MONTO_DEFAULT'].sum() / base_test_q5['
    INTERESES'].sum())

revenues_test_post_RFC_5_q5 = base_test_q5[base_test_q5['prediccion']==0]['
    INTERESES'].sum() - base_test_q5[base_test_q5['prediccion']==0]['
    MONTO_DEFAULT'].sum()
print('Cantidad de usuarios en test que predigo negativos:',len(base_test_q5[
    base_test_q5['prediccion']==0]['INTERESES']),'\nCantidad de cr ditos:',
    base_test_q5[base_test_q5['prediccion']==0]['CREDITOS'].sum(), '\nMonto
    originado:',base_test_q5[base_test_q5['prediccion']==0]['ORIGINACION'].sum
    (), '\nSu revenue:',revenues_test_post_RFC_5_q5,'\nSus intereses y
    punitorios: ', base_test_q5[base_test_q5['prediccion']==0]['INTERESES'].
    sum(),'\nSu default:', base_test_q5[base_test_q5['prediccion']==0]['
    MONTO_DEFAULT'].sum(), "\nSu loss_ratio:", base_test_q5[base_test_q5['
    prediccion']==0]['MONTO_DEFAULT'].sum() / base_test_q5[base_test_q5['
    prediccion']==0]['INTERESES'].sum())

print(classification_report(base_test_q6['FLAG_DEFAULT_MES'], base_test_q6['
    prediccion']))

revenues_test_pre_q6 = base_test_q6['INTERESES'].sum() - base_test_q6['
    MONTO_DEFAULT'].sum()
print('Cantidad de usuarios en test q6:',base_test_q6.CUS_CUST_ID_BORROWER.
    count(),'\nCantidad de cr ditos:',base_test_q6.CREDITOS.sum(), '\nMonto
    originado:',base_test_q6.ORIGINACION.sum(),'\nSu revenue:',
    revenues_test_pre_q6,'\nSus intereses y punitorios: ', base_test_q6['
    INTERESES'].sum(),'\nSu default:', base_test_q6['MONTO_DEFAULT'].sum(), "\
    nSu loss_ratio:", base_test_q6['MONTO_DEFAULT'].sum() / base_test_q6['
    INTERESES'].sum())

revenues_test_post_RFC_5_q6 = base_test_q5[base_test_q5['prediccion']==0]['
    INTERESES'].sum() - base_test_q6[base_test_q6['prediccion']==0]['
    MONTO_DEFAULT'].sum()
print('Cantidad de usuarios en test que predigo negativos:',len(base_test_q6[
    base_test_q6['prediccion']==0]['INTERESES']),'\nCantidad de cr ditos:',
    base_test_q6[base_test_q6['prediccion']==0]['CREDITOS'].sum(), '\nMonto
    originado:',base_test_q6[base_test_q6['prediccion']==0]['ORIGINACION'].sum
    (), '\nSu revenue:',revenues_test_post_RFC_5_q6,'\nSus intereses y
```

```
    punitorios: ', base_test_q6[base_test_q6['prediccion']==0]['INTERESES'].
    sum(),'\nSu default:', base_test_q6[base_test_q6['prediccion']==0]['
    MONTO_DEFAULT'].sum(), "\nSu loss_ratio:", base_test_q6[base_test_q6['
    prediccion']==0]['MONTO_DEFAULT'].sum() / base_test_q6[base_test_q6['
    prediccion']==0]['INTERESES'].sum())

print(classification_report(base_oot_q1['FLAG_DEFAULT_MES'], base_oot_q1['
    prediccion']))

revenues_oot_pre_q1 = base_oot_q1['INTERESES'].sum() - base_oot_q1['
    MONTO_DEFAULT'].sum()
print('Cantidad de usuarios en test q1:',base_oot_q1.CUS_CUST_ID_BORROWER.
    count(),'\nCantidad de cr ditos:',base_oot_q1.CREDITOS.sum(), '\nMonto
    originado:',base_oot_q1.ORIGINACION.sum(),'\nSu revenue:',
    revenues_oot_pre_q1,'\nSus intereses y punitorios: ', base_oot_q1['
    INTERESES'].sum(),'\nSu default:', base_oot_q1['MONTO_DEFAULT'].sum(), "\
    nSu loss_ratio:", base_oot_q1['MONTO_DEFAULT'].sum() / base_oot_q1['
    INTERESES'].sum())

revenues_oot_post_RFC_5_q1 = base_oot_q1[base_oot_q1['prediccion']==0]['
    INTERESES'].sum() - base_oot_q1[base_oot_q1['prediccion']==0]['
    MONTO_DEFAULT'].sum()
print('Cantidad de usuarios en test que predigo negativos:',len(base_oot_q1[
    base_oot_q1['prediccion']==0]['INTERESES']),'\nCantidad de cr ditos:',
    base_oot_q1[base_oot_q1['prediccion']==0]['CREDITOS'].sum(), '\nMonto
    originado:',base_oot_q1[base_oot_q1['prediccion']==0]['ORIGINACION'].sum()
    , '\nSu revenue:',revenues_oot_post_RFC_5_q1,'\nSus intereses y punitorios
    : ', base_oot_q1[base_oot_q1['prediccion']==0]['INTERESES'].sum(),'\nSu
    default:', base_oot_q1[base_oot_q1['prediccion']==0]['MONTO_DEFAULT'].sum
    (), "\nSu loss_ratio:", base_oot_q1[base_oot_q1['prediccion']==0]['
    MONTO_DEFAULT'].sum() / base_oot_q1[base_oot_q1['prediccion']==0]['
    INTERESES'].sum())

print(classification_report(base_oot_q2['FLAG_DEFAULT_MES'], base_oot_q2['
    prediccion']))

revenues_oot_pre_q2 = base_oot_q2['INTERESES'].sum() - base_oot_q2['
    MONTO_DEFAULT'].sum()
print('Cantidad de usuarios en test q2:',base_oot_q2.CUS_CUST_ID_BORROWER.
    count(),'\nCantidad de cr ditos:',base_oot_q2.CREDITOS.sum(), '\nMonto
    originado:',base_oot_q2.ORIGINACION.sum(),'\nSu revenue:',
    revenues_oot_pre_q2,'\nSus intereses y punitorios: ', base_oot_q2['
    INTERESES'].sum(),'\nSu default:', base_oot_q2['MONTO_DEFAULT'].sum(), "\
    nSu loss_ratio:", base_oot_q2['MONTO_DEFAULT'].sum() / base_oot_q2['
    INTERESES'].sum())

revenues_oot_post_RFC_5_q2 = base_oot_q2[base_oot_q2['prediccion']==0]['
    INTERESES'].sum() - base_oot_q2[base_oot_q2['prediccion']==0]['
    MONTO_DEFAULT'].sum()
print('Cantidad de usuarios en test que predigo negativos:',len(base_oot_q2[
    base_oot_q2['prediccion']==0]['INTERESES']),'\nCantidad de cr ditos:',
    base_oot_q2[base_oot_q2['prediccion']==0]['CREDITOS'].sum(), '\nMonto
    originado:',base_oot_q2[base_oot_q2['prediccion']==0]['ORIGINACION'].sum()
    , '\nSu revenue:',revenues_oot_post_RFC_5_q2,'\nSus intereses y punitorios
    : ', base_oot_q2[base_oot_q2['prediccion']==0]['INTERESES'].sum(),'\nSu
    default:', base_oot_q2[base_oot_q2['prediccion']==0]['MONTO_DEFAULT'].sum
    (), "\nSu loss_ratio:", base_oot_q2[base_oot_q2['prediccion']==0]['
    MONTO_DEFAULT'].sum() / base_oot_q2[base_oot_q2['prediccion']==0]['
    INTERESES'].sum())

print(classification_report(base_oot_q3['FLAG_DEFAULT_MES'], base_oot_q3['
    prediccion']))
```

```
revenues_oot_pre_q3 = base_oot_q3['INTERESES'].sum() - base_oot_q3['
    MONTO_DEFAULT'].sum()
print('Cantidad de usuarios en test q3:',base_oot_q3.CUS_CUST_ID_BORROWER.
    count(),'\nCantidad de cr ditos:',base_oot_q3.CREDITOS.sum(), '\nMonto
    originado:',base_oot_q3.ORIGINACION.sum(),'\nSu revenue:',
    revenues_oot_pre_q3,'\nSus intereses y punitorios: ', base_oot_q3['
    INTERESES'].sum(),'\nSu default:', base_oot_q3['MONTO_DEFAULT'].sum(), "\
    nSu loss_ratio:", base_oot_q3['MONTO_DEFAULT'].sum() / base_oot_q3['
    INTERESES'].sum())

revenues_oot_post_RFC_5_q3 = base_oot_q3[base_oot_q3['prediccion']==0]['
    INTERESES'].sum() - base_oot_q3[base_oot_q3['prediccion']==0]['
    MONTO_DEFAULT'].sum()
print('Cantidad de usuarios en test que predigo negativos:',len(base_oot_q3[
    base_oot_q3['prediccion']==0]['INTERESES']),'\nCantidad de cr ditos:',
    base_oot_q3[base_oot_q3['prediccion']==0]['CREDITOS'].sum(), '\nMonto
    originado:',base_oot_q3[base_oot_q3['prediccion']==0]['ORIGINACION'].sum()
    , '\nSu revenue:',revenues_oot_post_RFC_5_q3,'\nSus intereses y punitorios
    : ', base_oot_q3[base_oot_q3['prediccion']==0]['INTERESES'].sum(),'\nSu
    default:', base_oot_q3[base_oot_q3['prediccion']==0]['MONTO_DEFAULT'].sum
    (), "\nSu loss_ratio:", base_oot_q3[base_oot_q3['prediccion']==0]['
    MONTO_DEFAULT'].sum() / base_oot_q3[base_oot_q3['prediccion']==0]['
    INTERESES'].sum())

print(classification_report(base_oot_q4['FLAG_DEFAULT_MES'], base_oot_q4['
    prediccion']))

revenues_oot_pre_q4 = base_oot_q4['INTERESES'].sum() - base_oot_q4['
    MONTO_DEFAULT'].sum()
print('Cantidad de usuarios en test q4:',base_oot_q4.CUS_CUST_ID_BORROWER.
    count(),'\nCantidad de cr ditos:',base_oot_q4.CREDITOS.sum(), '\nMonto
    originado:',base_oot_q4.ORIGINACION.sum(),'\nSu revenue:',
    revenues_oot_pre_q4,'\nSus intereses y punitorios: ', base_oot_q4['
    INTERESES'].sum(),'\nSu default:', base_oot_q4['MONTO_DEFAULT'].sum(), "\
    nSu loss_ratio:", base_oot_q4['MONTO_DEFAULT'].sum() / base_oot_q4['
    INTERESES'].sum())

revenues_oot_post_RFC_5_q4 = base_oot_q4[base_oot_q4['prediccion']==0]['
    INTERESES'].sum() - base_oot_q4[base_oot_q4['prediccion']==0]['
    MONTO_DEFAULT'].sum()
print('Cantidad de usuarios en test que predigo negativos:',len(base_oot_q4[
    base_oot_q4['prediccion']==0]['INTERESES']),'\nCantidad de cr ditos:',
    base_oot_q4[base_oot_q4['prediccion']==0]['CREDITOS'].sum(), '\nMonto
    originado:',base_oot_q4[base_oot_q4['prediccion']==0]['ORIGINACION'].sum()
    , '\nSu revenue:',revenues_oot_post_RFC_5_q4,'\nSus intereses y punitorios
    : ', base_oot_q4[base_oot_q4['prediccion']==0]['INTERESES'].sum(),'\nSu
    default:', base_oot_q4[base_oot_q4['prediccion']==0]['MONTO_DEFAULT'].sum
    (), "\nSu loss_ratio:", base_oot_q4[base_oot_q4['prediccion']==0]['
    MONTO_DEFAULT'].sum() / base_oot_q4[base_oot_q4['prediccion']==0]['
    INTERESES'].sum())

print(classification_report(base_oot_q5['FLAG_DEFAULT_MES'], base_oot_q5['
    prediccion']))

revenues_oot_pre_q5 = base_oot_q5['INTERESES'].sum() - base_oot_q5['
    MONTO_DEFAULT'].sum()
print('Cantidad de usuarios en test q5:',base_oot_q5.CUS_CUST_ID_BORROWER.
    count(),'\nCantidad de cr ditos:',base_oot_q5.CREDITOS.sum(), '\nMonto
    originado:',base_oot_q5.ORIGINACION.sum(),'\nSu revenue:',
    revenues_oot_pre_q5,'\nSus intereses y punitorios: ', base_oot_q5['
    INTERESES'].sum(),'\nSu default:', base_oot_q5['MONTO_DEFAULT'].sum(), "\
```

```
    nSu loss_ratio:", base_oot_q5['MONTO_DEFAULT'].sum() / base_oot_q5['
    INTERESES'].sum())

revenues_oot_post_RFC_5_q5 = base_oot_q5[base_oot_q5['prediccion']==0]['
    INTERESES'].sum() - base_oot_q5[base_oot_q5['prediccion']==0]['
    MONTO_DEFAULT'].sum()
print('Cantidad de usuarios en test que predigo negativos:',len(base_oot_q5[
    base_oot_q5['prediccion']==0]['INTERESES']),'\nCantidad de cr ditos:',
    base_oot_q5[base_oot_q5['prediccion']==0]['CREDITOS'].sum(), '\nMonto
    originado:',base_oot_q5[base_oot_q5['prediccion']==0]['ORIGINACION'].sum()
    , '\nSu revenue:',revenues_oot_post_RFC_5_q5,'\nSus intereses y punitorios
    : ', base_oot_q5[base_oot_q5['prediccion']==0]['INTERESES'].sum(),'\nSu
    default:', base_oot_q5[base_oot_q5['prediccion']==0]['MONTO_DEFAULT'].sum
    (), "\nSu loss_ratio:", base_oot_q5[base_oot_q5['prediccion']==0]['
    MONTO_DEFAULT'].sum() / base_oot_q5[base_oot_q5['prediccion']==0]['
    INTERESES'].sum())

print(classification_report(base_oot_q6['FLAG_DEFAULT_MES'], base_oot_q6['
    prediccion']))

revenues_oot_pre_q6 = base_oot_q6['INTERESES'].sum() - base_oot_q6['
    MONTO_DEFAULT'].sum()
print('Cantidad de usuarios en test q6:',base_oot_q6.CUS_CUST_ID_BORROWER.
    count(),'\nCantidad de cr ditos:',base_oot_q6.CREDITOS.sum(), '\nMonto
    originado:',base_oot_q6.ORIGINACION.sum(),'\nSu revenue:',
    revenues_oot_pre_q6,'\nSus intereses y punitorios: ', base_oot_q6['
    INTERESES'].sum(),'\nSu default:', base_oot_q6['MONTO_DEFAULT'].sum(), "\
    nSu loss_ratio:", base_oot_q6['MONTO_DEFAULT'].sum() / base_oot_q6['
    INTERESES'].sum())

revenues_oot_post_RFC_5_q6 = base_oot_q6[base_oot_q6['prediccion']==0]['
    INTERESES'].sum() - base_oot_q6[base_oot_q6['prediccion']==0]['
    MONTO_DEFAULT'].sum()
print('Cantidad de usuarios en test que predigo negativos:',len(base_oot_q6[
    base_oot_q6['prediccion']==0]['INTERESES']),'\nCantidad de cr ditos:',
    base_oot_q6[base_oot_q6['prediccion']==0]['CREDITOS'].sum(), '\nMonto
    originado:',base_oot_q6[base_oot_q6['prediccion']==0]['ORIGINACION'].sum()
    , '\nSu revenue:',revenues_oot_post_RFC_5_q6,'\nSus intereses y punitorios
    : ', base_oot_q6[base_oot_q6['prediccion']==0]['INTERESES'].sum(),'\nSu
    default:', base_oot_q6[base_oot_q6['prediccion']==0]['MONTO_DEFAULT'].sum
    (), "\nSu loss_ratio:", base_oot_q6[base_oot_q6['prediccion']==0]['
    MONTO_DEFAULT'].sum() / base_oot_q6[base_oot_q6['prediccion']==0]['
    INTERESES'].sum())
```