 Logo

Description automatically generated

**UNIVERSIDAD DE BUENOS AIRES**

**Facultad de Ciencias Económicas**

Uso de modelos estocásticos y técnicas de aprendizaje automático en la proyección de mortalidad.

Alumna: Franco Olivero

Registro N.º 884813

e-mail: franco.n.olivero@gmail.com

Tutor: Cristian Sciaccaluga

Asignatura: Teoría del Equilibrio Actuarial (Cód. 757)

e-mail: cristian\_sciaccaluga@yahoo.com

Firma del tutor:

**1er Cuatrimestre 2025**

# Introducción

Los usuarios Premium son los que más valor traen a la empresa por lo que es necesario retenerlos. Para ello el equipo de marketing necesita predecir con anticipación de dos meses qué usuarios se darán de baja para poder enviarles un estímulo para que no se fuguen. En ese sentido, el objetivo de esta tesina es ver qué modelo de clasificación (de *clustering*) se adapta mejor a las necesidades del negocio de predecir qué usuarios continuarán en el banco y que usuarios solicitarán la baja.

En este estudio son utilizadas una serie de técnicas estadísticas y de ciencias de datos para ir generando modelos, a saber, en primer lugar, una regresión logística y luego diversas técnicas de árboles de decisión (Árbol de decisión simple, *Random Forest, LightGBM*).

Para comparar los modelos se crea un estimador que representa la ganancia del programa de retención para el banco. Para ello se utiliza el valor promedio futuro de ingresos que representa un usuario para el banco y un valor de estímulo monetario con el cual se espera retener al 50% de los usuarios a los que sea enviado.

La ganancia del programa de retención para cada modelo es resultado de la cantidad de aciertos o errores en la predicción de cuántos usuarios pedirán irse del banco en dos meses, siendo que si se envía estímulos a usuarios que no tenían ninguna intención de dejar la compañía, el banco pierde plata.

Se calcula cuánta sería la ganancia para el banco de haber implementado el programa de estímulos con el modelo predictivo propuesto en cada caso y con ello se propone calcular la diferencia de ganancias/pérdidas entre los distintos modelos para ver cuál predice mejor.

Índice

[I. Fundamentación 5](#_Toc107748736)

[II. Marco conceptual 5](#_Toc107748737)

[2.1. Utilización de modelado predictivo en Instituciones Financieras y de Seguros 5](#_Toc107748738)

[2.2. El uso de la ciencia de datos por parte de los Actuarios 6](#_Toc107748739)

[2.3. Técnicas de Clustering: Regresión logística 6](#_Toc107748740)

[2.4. Técnicas de Clustering: Árboles de decisión 8](#_Toc107748741)

[2.5. Técnicas de Clustering: Randon Forest 10](#_Toc107748742)

[2.6. Métodos de ensamble de Árboles 11](#_Toc107748743)

[2.7. Técnicas de Clustering: LightGBM 11](#_Toc107748744)

[2.8. Overfitting 12](#_Toc107748745)

[2.9. Cross-Validation 13](#_Toc107748746)

[2.10. Optimizacion Bayesiana 14](#_Toc107748747)

[III. Problema de Negocio 15](#_Toc107748748)

[3.1. Planteo de la situación inicial 15](#_Toc107748749)

[3.2. Políticas de Retención de clientes 16](#_Toc107748750)

[3.3. Diseño de proyecto para retención proactiva de clientes 16](#_Toc107748751)

[3.4. Datos disponibles 17](#_Toc107748752)

[3.5. Creación de indicadores para medir la efectividad del proyecto 18](#_Toc107748753)

[IV. Desarrollo de modelos predictivos para dar respuesta a problema comercial 18](#_Toc107748754)

[4.1. Generación de modelos predictivos 18](#_Toc107748755)

[4.2 Primera aproximación al data set. 19](#_Toc107748756)

[4.3. Dualidad. Clase ternaria 20](#_Toc107748757)

[4.4. Estimación de la predicción de los modelos con data sets desbalanceados 21](#_Toc107748758)

[4.5. Probabilidad de corte 21](#_Toc107748759)

[4.6. Regresión logística 22](#_Toc107748760)

[4.7. Árboles de decisión 26](#_Toc107748761)

[4.8. Random Forest 27](#_Toc107748762)

[4.9. LightGBM 28](#_Toc107748763)

[V. Resultados de los modelos entrenados con datos de noviembre para enero. 29](#_Toc107748764)

[VI. Conclusiones 29](#_Toc107748765)

[VII. Líneas futuras de investigación. 30](#_Toc107748766)

[Referencias bibliográficas 31](#_Toc107748767)

[Anexos 32](#_Toc107748768)

[Anexo 1. Diccionario de datos: descripción de las variables disponibles en el dataset. 32](#_Toc107748769)

[Anexo 2. Script de R Studio: primera aproximación a la data. 55](#_Toc107748770)

[Anexo 3. Script de R Studio: análisis exploratorio. 56](#_Toc107748771)

[Anexo 4. Script de R Studio: Regresión logística y regresión logística penalizada. 57](#_Toc107748772)

[Anexo 5. Script de R Studio: modelo de Árboles de decisión simples. 59](#_Toc107748773)

[Anexo 6. Script de R Studio: modelo de Random Forest. 64](#_Toc107748774)

[Anexo 7. Script de R Studio: modelo de LightGbm. 70](#_Toc107748775)

# I. Fundamentación

Motiva esta tesis el abordaje del modelado actuarial de riesgos de pérdida de clientes que consiste en detectar qué clientes son propensos a abandonar una compañía o a cancelar la suscripción a un servicio. Se trata de una predicción crítica para muchas empresas ya que adquirir nuevos clientes suele costar más que retener a los existentes, de esta forma el estudiar qué usuarios pedirán la baja y poder accionar políticas de retención proactivas de los mismos representa ventajas comparativas para el negocio. Para ello se estudian herramientas estadísticas y de *machine learning* que generan modelos con los cuales estimar el riesgo de salida de cada cliente. Se espera que los modelos conseguidos a través de las distintas técnicas brinden predicciones lo suficientemente acertadas como para que los programas de retención de clientes permitan aumentar las ganancias de las organizaciones.

# II. **Marco conceptual**

## 2.1. Utilización de modelado predictivo en Instituciones Financieras y de Seguros

Las instituciones financieras y de seguros se sirven de las técnicas actuariales de modelado predictivo para cuantificar y disminuir el riesgo de las operaciones en que participan. Se utilizan modelos de riesgo crediticio para ver a qué usuarios darles un crédito, modelos para predecir conductas fraudulentas de clientes, modelos de predicción del *lapse rate* de pólizas, y muchos otros más. Para ello se utilizan usualmente los modelos lineales generalizados (GLM) que son muy útiles cuando se trabaja con variables continuas, por ejemplo, un modelo generalizado de Poisson para predecir cantidades o ventas, o un modelo de regresión logística (Logit) para modelos de *scoring* crediticio.

Esta última, la regresión logística, también puede ser incluida en el grupo de los modelos de clasificación (o *clustering*) puesto que busca predecir el resultado de una variable categórica binaria.

Otras formas no lineales y no paramétricas de modelos de clasificación se basan en árboles de decisión en los que se toma el conjunto de la información y se la divide por alguna variable que resulte significativa reiteradas veces hasta conseguir una distribución de probabilidades que nos lleve a tomar una decisión acerca de la categoría de la variable dependiente. Además, con la llegada del *machine learning*, es decir, de poder enseñarle con datos al modelo, se han conseguido ajustes muy buenos con estas técnicas de árbol. Esta interacción entre el *machine learning* y los beneficios que trajo a la profesión actuarial en la estimación del riesgo es desarrollado en el apartado siguiente.

## 2.2. El uso de la ciencia de datos por parte de los Actuarios

La ciencia de datos es aquella que, sirviéndose de herramientas informáticas y estadísticas, técnicas de machine learning, de inteligencia artificial, modelos de probabilidad y de visualización de datos, busca administrar, analizar e interpretar los datos. Es así como se logra discernir tendencias, hacer predicciones y tomar decisiones en función de ello.

Actualmente esta ciencia se encuentra en auge debido a que se cuenta con cantidades inmensas de datos con los que antes no se contaba, capacidades de procesamiento muy superiores a con las que se tenía anteriormente y profesionales tales como los actuarios capaces de servirse de estas técnicas avanzadas para poder obtener información importante de estos datos.

De esta forma los datos se han convertido en un insumo muy apreciado por los actuarios a la hora de estimar el riesgo de una cartera, o de hacer cálculos de supuestos, o bien, realizar estudios de análisis de experiencia, entre otras cosas.

## 2.3. Técnicas de Clustering: Regresión logística

La Regresión Logística Simple, desarrollada por David Cox en 1958, es un método de regresión que permite estimar la probabilidad de una variable cualitativa binaria en función de una o más variable cuantitativas y/o cualitativas. Una de las principales aplicaciones de la regresión logística es la de clasificación, en el que las observaciones se clasifican en un grupo u otro dependiendo del valor que tome la o las variables empleadas como predictoras.

Además, se caracteriza por:

-La media condicional de la ecuación de regresión debe ser formulada para estar comprendida entre cero y uno.

-La binomial describe el comportamiento de los errores y el análisis estadístico deberá basarse en ella.

-Los principios que guían las regresiones lineales seguirán vigentes en la regresión logística.

-El método de ajuste es de máxima verosimilitud (Hosmer, 2000).

En los modelos de regresión logística consideramos, además de la variable dicotómica de respuesta, un grupo de variables predictoras, de modo que, si existen variables explicativas de tipo nominal, éstas deben ser incluidas en el modelo señalando que tienen esa condición. Para ello es necesario transformarlas en variables *dummy* (Epidat 4, 2014), por ejemplo, supóngase que cierta variable es nominal (por ejemplo la raza, o la religión profesada, o el grupo sanguíneo, etc.) que consta de k categorías; se crean en esos casos k-1 variables dicotómicas, que son las llamadas variables *dummy* asociadas a esta variable nominal y que se denotarán por Z1, Z2, ..., Zk-1. A cada categoría o clase de la variable nominal le corresponde un conjunto de valores de los Zi con el cual se identifica dicha clase.

Cabe consignar que estos modelos con variables *dummy* implican cierta complejidad en su elaboración e interpretación. Máxime cuando se trata la interacción entre los factores del modelo. Es este uno de los puntos que hace que los otros modelos abordados en esta tesina sean más atractivos ya que no hay que generar transformaciones en la data para poder utilizarlos, sin embargo, en el caso particular de esta tesina no se cuenta con variables nominales por ende esta transformación no es requerida.

Un reloj de aguja

Descripción generada automáticamente con confianza mediaCon respecto a los residuos, se suele utilizar los residuos de deviance que pueden escribirse como una suma donde las di son los que se conocen como componentes de la deviance (Departamento de Matemáticas, Facultad de exactas, Universidad de Buenos Aires, 2005). Esta medida, suele usarse como una medida de discrepancia en un GLM y en ese sentido di es la contribución de cada dato.

Dos modelos en competencia pueden ser comparados mediante la deviance cuando tienen la misma distribución y función link y sólo difieren en el número de parámetros, es decir, cuando se trata de modelos anidados.

Si D es grande, el modelo ajustado proporciona un ajuste pobre. Un valor pequeño de D es indicador de un buen ajuste. De esta forma estos residuos dan cuenta de la bondad del ajuste del modelo. (Morales, 2021)

## 2.4. Técnicas de Clustering: Árboles de decisión

Desde que comenzó a hablarse de inteligencia artificial ha existido la ambición de comprender y descubrir relaciones complejas en la data, esto es encontrar modelos que no solo produzcan predicciones precisas, sino que extraigan conocimiento de una forma inteligible. De esta forma se realizó en el campo de *machine learning* un importante cúmulo de investigación en diversas direcciones. Una de ellas, los métodos basados en árboles de decisión, resultó en resultados efectivos y de fácil usabilidad, así fue capaz de producir modelos entendibles y confiables en casi cualquier tipo de set de datos.

Se atribuye a Morgan and Sonquist [1963], la creación del método de los árboles de decisión, quienes primero propusieron un método llamado de *automatic interaction detector* (AID) es decir detector de interacciones automáticas, para distinguir efectos no aditivos y multivariados en el contexto de data de encuestas.

Basándose en el AID, los principales investigadores que han impulsado la investigación sobre los principios metodológicos modernos son Breiman, Friedman y Quinlan que propusieron simultánea e independientemente algoritmos muy cercanos para la inducción de modelos basados en árboles. En particular, el trabajo unificador de Breiman (Breiman, 1994), ha establecido los árboles de decisión en un marco metodológico simple y consistente, que ha contribuido en gran medida a hacerlos fáciles de entender y de utilizar.

El éxito de los árboles de decisión (y por extensión, de todos los métodos basados en árboles) se explica por varios factores que los hacen bastante atractivos en la práctica:

- Los árboles de decisión son no paramétricos. Pueden modelar relaciones arbitrariamente complejas entre entradas y salidas, sin ninguna suposición a priori;

- Los árboles de decisión manejan datos heterogéneos (variables ordenadas o categóricas, o una mezcla de ambas);

- Los árboles de decisión implementan intrínsecamente la selección de características, haciéndolos robustos a variables irrelevantes o ruidosas (al menos hasta cierto punto);

- Los árboles de decisión son robustos a los valores atípicos o a ciertos errores en las etiquetas;

- Los árboles de decisión son fácilmente interpretables, incluso para usuarios no orientados a la estadística. Y lo que es más importante, los árboles de decisión son la base de muchos algoritmos modernos y de vanguardia. Por lo tanto, comprender todos los detalles algorítmicos de los árboles de decisión individuales es un requisito previo obligatorio para un análisis en profundidad de estos métodos.

La idea de los árboles de decisión es tomar un data set y dividirlo en función a algún valor de alguna variable, luego a esa base resultante dividirla nuevamente en función de otra variable, esto daría como resultado una base dividida en cuatro, donde la probabilidad de pertenecer a cada rama del árbol se da por la cantidad de casos de la rama sobre la cantidad total de casos de la data. Este proceso permite ir alterando la cantidad de casos positivos sobre casos totales en cada rama del árbol y de esta forma es cómo se alcanzan probabilidades segmentadas para cada una de ellas de ser un caso positivo.

Este proceso se puede realizar k veces y así obtener árboles de distintos grados.

Para una mayor claridad mostraremos un ejemplo.

Se propone una muestra de 100 casos de los cuales 10 son clase 1 y 90 son clase 0. Se quiere predecir la clase 1 y se decide dividir al data set por un valor de una variable. Luego de generar la separación obtengo dos nodos donde cada uno tiene una cantidad tal que al sumar ambos dan 100. Estos tienen una concentración de clase 1 sobre el total distinta a la de la muestra original. Dando uno de los nodos una concentración mayor y otro una concentración menor. En la figura se observa que en uno de los nodos pasa a haber 70 casos de los cuales 2 son clase 1 y en el otro nodo 30 casos restantes de los cuales 8 pertenecen a la clase 1.

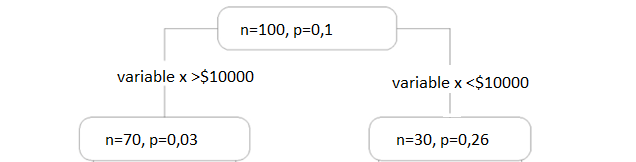


Figura 1. Árbol de decisión. Ejemplo.

Si se sigue realizando este procedimiento segmentando la muestra de acuerdo a distintas variables se obtendrá una cierta submuestra con probabilidad lo suficientemente alta de pertenecer a la clase 1, que es justamente lo que se desea conocer. No es conveniente dividir el árbol de forma infinita ya el árbol se estaría ajustando demasiado al dataset de entrenamiento y sería poco eficaz para su uso en nuevos datos. Para ampliar esto se desarrolla el concepto de *overfitting* en la sección 2.6*.*

Estos modelos se perfeccionan con distintas técnicas para ver cuales son las variables de corte más acertadas para cada data set y con qué valor de corte.

Otro punto destacable para tomar en cuenta son los llamados hiperparámetros que se determinan *a priori* antes de generar el modelo, que son: la dimensión del árbol máxima, el parecido que se pide entre los datasets divididos resultantes, la cantidad mínima de casos por rama del árbol, etc. Para la determinación de los mejores hiperparámetros, en esta tesina se utiliza el enfoque de la optimización bayesiana, que es uno de los múltiples enfoques posibles, esto será ampliado en la sección 2.8.

## 2.5. Técnicas de Clustering: Randon Forest

El método de *Random Forest* introducido por Leo Breiman​ y Adele Cutler tuvo una alta aceptación entre los científicos de datos puesto que supera en precisión a los árboles de decisión individuales y suele tener menos problemas de *overfitting*.

Con este método ya no se habla de un único árbol de decisión, sino de muchos, aquí ya hay un *forest*, es decir un bosque, donde hay diversos árboles con características distintivas, distintas variables divisoras, distintos hiperparámetros, etc.

Con respecto a la parte de *random*, este método de Machine Learning se basa en entrenar árboles de decisión en paralelo utilizando varios subconjuntos aleatorios del conjunto de la data. El proceso de generación de muestras simuladas y aleatorias dentro de un conjunto de datos se conoce como *bootsrapping* y con ello es asegurado que cada árbol de decisión individual resultante es único. Luego estos árboles son agregados para conseguir el mejor modelo predictor posible, lo cual lleva el nombre de *bagging* (Breiman, 1994).

## 2.6. Métodos de ensamble de Árboles

Los métodos de ensamble combinan múltiples modelos en uno nuevo con el objetivo de lograr mejores predicciones que cualquiera de los modelos individuales originales. Dos de los tipos de ensamble más utilizados son:

-**Bagging**: Se ajustan múltiples modelos, cada uno con un subconjunto distinto de los datos de entrenamiento. Para predecir, todos los modelos que forman el agregado participan aportando su predicción. Como valor final, se toma la media de todas las predicciones (variables continuas) o la clase más frecuente (variables categóricas). Como ya fue mencionado los modelos *Random Forest* están dentro de esta categoría.

-**Boosting**: Se ajustan secuencialmente múltiples modelos sencillos, llamados *weak learners*, de forma que cada modelo aprende de los errores del anterior. Como valor final, al igual que en bagging, se toma la media de todas las predicciones (variables continuas) o la clase más frecuente (variables cualitativas). Como su nombre lo indica este será el método que seguirá el modelo *Light Gradient Boosting Machine* (*LightGBM*) (Rodrigo Amat, 2020).

## 2.7. Técnicas de Clustering: LightGBM

*LightGBM*, abreviatura de *Light Gradient Boosting Machine*, es un modelo de estructura rápida, apropiada e [impulso de gradiente](https://datascience.eu/es/aprendizaje-automatico/impulso-de-gradiente-lo-que-necesitas-saber/). Es gratuito y de código abierto, para el aprendizaje automático, desarrollado originalmente por Microsoft que se basa en algoritmos de árboles de decisión y se utiliza para la clasificación y otras tareas de aprendizaje automático. El desarrollo se centra en el rendimiento y la escalabilidad. Mientras que los árboles de otros algoritmos crecen horizontalmente, el algoritmo *LightGBM* crece verticalmente, es decir, crece por hojas.

Este método tiene gran popularidad ya que a los algoritmos tradicionales les resulta difícil dar resultados con rapidez, porque el tamaño de los datos aumenta rápidamente día a día, en cambio, *LightGBM*, se llama justamente "Light" por su capacidad de cálculo y por dar resultados más rápidamente. Requiere menos memoria para ejecutarse y es capaz de manejar grandes cantidades de datos.

Es una variante del algoritmo *AdaBoost* (Freund) que fue el primero que comenzó a aplicar la estrategia de *boosting* a problemas de clasificación binaria.

La idea general de estos modelos de *boosting* es partiendo de un modelo *weak learner o base learner*, que sea capaz de predecir la variable respuesta con un porcentaje de acierto ligeramente superior a lo esperado por azar, iniciar un proceso iterativo. En la primera iteración, se ajusta el *weak learner* empleando los datos de entrenamiento y los pesos iniciales (que deben ser todos iguales). Con el *weak learner* ajustado y almacenado, se predicen las observaciones de entrenamiento y se identifican aquellas bien y mal clasificadas. Con esta información se actualizan los pesos relativos de las observaciones.

Se asigna un peso total al *weak learner*, proporcional al total de aciertos. Cuantos más aciertos consiga el modelo, mayor su influencia en el conjunto del ensemble.

En la siguiente iteración, se llama de nuevo al *weak learner* y se vuelve a ajustar, esta vez, empleando los pesos actualizados en la iteración anterior. El nuevo modelo se almacena, para su posterior ensamble. Este proceso se repite n veces, generando un total de n *weak learner*. Para clasificar nuevas observaciones, se obtiene la predicción de cada uno de los *weak learners* que forman el ensamble y se agregan sus resultados, ponderando el peso de cada uno acorde al peso que se le ha asignado en el ajuste. El objetivo detrás de esta estrategia es que cada nuevo *weak learner* se centre en predecir correctamente las observaciones que los anteriores no han sido capaces (Rodrigo Amat, 2020).

## 2.8. Overfitting

El *overfitting* es el sobreajuste de un modelo a los datos con los cuales se genera, de esta forma representa una deficiencia que dificulta la precisión y el rendimiento del modelo en su aplicación a nuevos datos.

Detectar el *overfitting* es casi imposible antes de probar los datos. El proceso para detectar el sobreajuste es separar los datos en diferentes subconjuntos para facilitar el entrenamiento y las pruebas. Los datos se dividen en dos partes principales, es decir, un conjunto de prueba y un conjunto de entrenamiento.

El conjunto de entrenamiento representa la mayoría de los datos disponibles y entrena el modelo. El conjunto de prueba representa una pequeña parte del conjunto de datos y se utiliza para probar la precisión de los datos con los que nunca antes se interactuó. Al segmentar el conjunto de datos, podemos examinar el rendimiento del modelo en cada conjunto de datos para detectar el sobreajuste cuando ocurre, así como ver cómo funciona el proceso de entrenamiento. Se da, entonces, un *overfitting* cuando el modelo es perfecto prediciendo en los datos donde se entrena, pero al aplicarlo a datos nuevos da un ajuste pésimo.

Para ver esto gráficamente se propone la siguiente figura:

Gráfico

Descripción generada automáticamente

Figura 2. Gráfico de representación de *overfitting*. Ejemplo. (Wikipedia, s.f.)

En la figura se observa un modelo que busca separar los puntos rojos de los azules. En este data set la línea verde muestra un *overfitting* ya que el modelo trata de clasificar bien a todos los puntos, en cambio la línea negra es a lo que se aspira ya que generaliza mejor, obteniendo un mayor error de predicción en el conjunto de datos de entrenamiento, pero un sentido más claro de la diferenciación global de las clases permitiendo aplicarlo a nueva información y esperando mejores resultados.

Esto tiene tamaña importancia ya que el objetivo fundamental de cualquier modelo predictivo es brindar una buena performance en datos con los cuales no se entrenó.

## 2.9. Cross-Validation

La validación cruzada es una poderosa medida preventiva contra el sobreajuste o *overfitting* (Gumpta).

La idea es utilizar los datos de entrenamiento inicial para generar múltiples divisiones de prueba. Luego se utiliza estas divisiones para ajustar el modelo. En la validación cruzada estándar de k veces, dividimos los datos en k subconjuntos, llamados pliegues o *folds*. Luego, entrenamos iterativamente el algoritmo en k-1 pliegues mientras usamos el pliegue restante como conjunto de prueba (llamado “*Validation*”).

La validación cruzada permite ajustar los hiperparámetros de los modelos solo con su conjunto de entrenamiento original. Esto permite mantener el conjunto de prueba como un conjunto de datos realmente no antes utilizado para probar el modelo resultante.

Se ejemplifica lo explicado con la siguiente figura:

Imagen de la pantalla de un celular con letras

Descripción generada automáticamente con confianza baja

Figura 3. Gráfico del proceso de partición de la data en folds para realizar una Validación Cruzada.

## 2.10. Optimizacion Bayesiana

La mayoría de las modelos de *machine learning* tienen una serie de parámetros que no pueden aprenderse de los datos, sino que tienen que ser establecidos previo entrenamiento. A estos parámetros se les suele conocer como hiperparámetros.

Dado que no existe forma de conocer de antemano cuáles son los valores adecuados, la única forma de identificarlos es probando diferentes combinaciones y evaluándolas mediante métodos de validación por *resampling* (*cross-validation, bootstrapping*). A este proceso se le conoce como *model tuning* (Rodrigo Amat, 2020).

Dos de las estrategias más empleadas son *grid search* y *random search*. En la primera, los valores estudiados (de hiperparámetros) se distribuyen de forma uniforme dentro de un rango delimitado por el analista. En el segundo caso, los datos son aleatorios dentro de ese rango.

Aunque estas dos estrategias son totalmente válidas y generan buenos resultados, fundamentalmente cuando se posee algún criterio para acotar el rango de búsqueda, comparten una carencia común: ninguna tiene en cuenta los resultados obtenidos hasta el momento, lo que les impide focalizar la búsqueda en las regiones de mayor interés y evitar regiones innecesarias.

En términos generales, la optimización bayesiana de hiperparámetros consiste en crear un modelo probabilístico en el que el valor de la función objetivo es la métrica de validación del modelo. Con esta estrategia se consigue que la búsqueda se vaya redirigiendo en cada iteración hacia las regiones de mayor interés. El objetivo final es reducir el número de combinaciones de hiperparámetros con las que se evalúa el modelo, eligiendo únicamente los mejores candidatos. Esto significa que, la ventaja frente a las otras estrategias mencionadas se maximiza cuando el espacio de búsqueda es muy amplio (Rodrigo Amat, 2020).

# III. Problema de Negocio

## 3.1. Planteo de la situación inicial

La pérdida de clientes es un problema común en las empresas de muchos sectores. Si se quiere crecer como empresa, hay que invertir en la captación de nuevos clientes. Cada vez que un cliente se va, representa una importante inversión perdida (Avaus, s.f.). Hay que dedicar tiempo y esfuerzo a sustituirlos. Ser capaz de predecir cuándo es probable que un cliente se vaya, y ofrecerle incentivos para que se quede, puede suponer un gran ahorro para una empresa.

Por tanto, entender qué es lo que mantiene a los clientes comprometidos es un conocimiento extremadamente valioso, ya que puede ayudarle a desarrollar sus estrategias de retención y a poner en marcha prácticas operativas destinadas a evitar que los clientes se marchen.

En este caso,el banco tiene una cartera de usuarios Premium que son los que más valor traen a la empresa y es por ello que es de suma importancia que estos clientes no se fuguen. Actualmente la empresa no hace campañas proactivas de retención de clientes, simplemente una vez que el cliente manifiesta que se quiere ir, tratan de persuadirlo. Es por todo lo antes explicado que el equipo de marketing necesita predecir con anticipación de dos meses qué usuarios se darán de baja para poder enviarles un estímulo para evitar que dejen la empresa.

Para ello se realiza un modelo predictivo para elegir a los clientes a los que se hará una campaña de marketing preventiva de retención, antes que manifiesten su voluntad de darse de baja.

La capacidad de predecir que un cliente abandonará la empresa con alta probabilidad cuando aún se está a tiempo de hacer algo al respecto, representa una enorme fuente potencial de ahorro y una ventaja comparativa en relación con el resto de la industria, en este caso, bancaria.

## 3.2. Políticas de Retención de clientes

Dentro del mundo del marketing se reconocen distintas políticas de retención de clientes. En el banco actualmente se utiliza la política reactiva de retención donde el cliente manifiesta su voluntad de irse, y entonces, aunque se le ofrezcan facilidades o beneficios para permanecer como cliente, generalmente, ya es tarde.

La campaña que se describe en esta tesina es de retención proactiva, donde se le ofrece un estímulo para que se quede en la compañía a un cliente que aún no manifiesta el querer irse pero que con probabilidad alta lo hará en dos meses o en un tiempo relativamente corto. Este es el target de clientes que se busca identificar y, el por qué de su decisión de dejar el banco para poder hacer campañas de marketing específicas para retenerlos.

Además de ver las causas o características que nos pueden llevar a pensar que un cliente dejará el banco con alta probabilidad, se busca, en un universo ideal, prevenir que alcancen esos valores o situaciones o características que vuelven más altamente probable la decisión del usuario de irse. Aquí ya no hablamos de políticas de retención reactivas ni proactivas sino de las preventivas.

## 3.3. Diseño de proyecto para retención proactiva de clientes

La campaña de retención proactiva de clientes consiste en enviar un estímulo, que tiene un costo de $ 1,250, éste es un monto determinado por el equipo de Marketing.

El equipo de marketing comparte que ha estado haciendo pequeños experimentos con los que ha encontrado que de las personas que se iban a ir en el mes 2 y reciben el estímulo, el 50% decide quedarse y el otro 50% decide irse igualmente.

Un cliente que se queda deja una ganancia futura de $ 100,000, este es el valor actual de la renta futura que cada cliente le deja al banco en promedio durante todo el tiempo en que opera con ellos. Esta información fue compartida por el equipo de Finanzas.

Se conocen los datos del mes 0 y se quiere hacer una campaña de marketing de retención proactiva a clientes que el modelo prediga tienen alta probabilidad de darse de baja durante el mes 2.

Sería interesante probar qué ocurriría si en vez de enviar $1,250 se enviara algún otro monto, que ocurriría con el porcentaje de efectividad de la campaña, y mejorar las predicciones tomando el valor futuro de ganancia futura para el banco para cada cliente en vez de utilizar un promedio. Todo ello se podría desarrollar para mejorar la campaña, pero en este caso como es una tesis actuarial nos centramos en la estimación del riesgo de dejar la compañía de cada cliente y para ello utilizamos la serie de hipótesis o simplificaciones antes mencionadas.

## 3.4. Datos disponibles

Se posee la base de datos de clientes premium de un banco argentino durante tres meses: noviembre y diciembre de 2020 y enero de 2021. Cada registro (o fila) es un cliente, y además tenemos 158 variables.

Una de ellas, la variable dependiente, es la clase ternaria, que contiene información futura:

|  |  |
| --- | --- |
| **Clase ternaria** | **Descripción** |
| BAJA +1 | Se dará de baja durante el mes próximo. |
| BAJA+2 | Se dará de baja durante el mes siguiente al mes próximo. |
| CONTINÚA | No pedirá la baja. |

Figura 4. Explicación de la variable dependiente: Clase ternaria.

Como para calcular la clase ternaria debo conocer dos meses del futuro, los meses de diciembre de 2020 y de enero de 2021 no contienen los valores de esta variable completa.

El resto de las variables disponibles están detalladas junto a su definición específica en el Anexo.

## 3.5. Creación de indicadores para medir la efectividad del proyecto

Para poder comparar los modelos se crea un estimador que representa la ganancia del programa de retención para el banco. Para ello se utiliza el valor promedio futuro de ingresos que representa un usuario al banco y un valor de estímulo monetario con el cual se espera retener al 50% de los usuarios a los que sea enviado.

La ganancia del programa de retención para cada modelo es resultado de la cantidad de aciertos o errores en la predicción de cuántos usuarios pedirán irse del banco en dos meses, siendo que si se envía estímulos a usuarios que no tenían ninguna intención de dejar la compañía, el banco pierde plata.

Se calcula cuánta sería la ganancia para el banco de haber implementado el programa de estímulos con el modelo predictivo propuesto en cada caso y con ello se propone calcular la diferencia de ganancias/pérdidas entre los distintos modelos para ver cuál predice mejor.

Ganancia de la campaña:

Donde “aciertos” son aquellos clientes que reciben el estímulo e iban a irse en el mes 2.

Para evaluar los modelos generamos las predicciones y conociendo el “futuro”, es decir sabiendo realmente qué cliente pertenece a qué clase {BAJA+1, BAJA+2, CONTINUA} se suma la ganancia de cada modelo.

# IV. Desarrollo de modelos predictivos para dar respuesta a problema comercial

## 4.1. Generación de modelos predictivos

Como fue comentado antes, para mejorar las ganancias del banco a través de retener a los clientes que piden la baja enviando un incentivo es de vital importancia contar con un modelo capaz de predecir con una anticipación adecuada a través de la información de cuenta del cliente, sus movimientos, sus saldos, etc. quiénes son los que decidirán irse. Para ello se busca asignarle una probabilidad de salida de la compañía a cada cliente de la cartera. Se estudia a continuación a partir de qué probabilidad de salida es conveniente enviar un estímulo y los distintos modelos trabajados para asignar dichas probabilidades. Durante el desarrollo del problema se mencionan los problemas y desafíos que tiene la problemática y de qué forma se decide abordarlos en cada caso. En el próximo apartado se verán los resultados de los distintos modelos al aplicarlos en un data set nuevo y se podrá comparar el poder predictivo de cada uno para este caso particular.

## 4.2 Primera aproximación al data set.

El primer enfoque que se toma en esta tesina es el de estudiar con qué variables contamos y qué nos pueden contar acerca de la variable a predecir, la clase ternaria.

Texto

Descripción generada automáticamente con confianza mediaPara ello se agrupa el universo por clase ternaria para ver cuántos casos de BAJA +2 (o éxitos para nuestro modelo) existen en este set de datos.

Figura 5. Cantidad de casos por clase ternaria.

Se observa que la cantidad de personas que se darán de baja en dos meses son un porcentaje muy bajo del total de usuarios de la cartera premium del banco.

Se vuelve a hacer un summary pero segregado por clase ternaria. Se observan algunas diferencias en los estimadores de las variables en función de la clase ternaria.

Se propone observar gráficamente la siguiente variable en particular para ejemplificar.

Gráfico, Gráfico de cajas y bigotes

Descripción generada automáticamente

Figura 6. Saldo de cuentas dividida por clase ternaria. Fuente: Elaboración propia.

En este caso se observa la variable mcuentas\_saldo que es el saldo total de todas las cuentas del cliente, cajas de ahorro, cuentas corrientes, en pesos y en dólares. El valor está convertido a pesos. En el caso de esta variable se observan diferencias significativas entre las clases BAJA+1 y BAJA +2 con respecto a la clase CONTINUA.

## 4.3. Dualidad. Clase ternaria

El modelo busca hallar los usuarios que, queriendo irse, aún pueden ser convencidos de permanecer en la compañía, por ello los BAJA +1 representan una pérdida para la campaña, pues su decisión ya no es reversible. Los modelos predictivos con los que trabajaremos requieren una agrupación entre las clases ternarias para generar dos grupos.

Si se toma en consideración el criterio de casos positivos versus casos negativos para la campaña de retención, la agrupación natural sería CONTINUA y BAJA +1 y por otra parte BAJA +2. Este criterio presenta una desventaja clara que tiene que ver con el sentido del evento a predecir, se cree que los eventos de BAJA tienen más en común entre sí que con los eventos CONTINUA, por ende, puede ser confuso para el modelo a la hora de estimar de qué forma debe tomar los valores de variable. Como analogía si un cliente que decide darse de baja es un cliente “enfermo”: el BAJA+1 es el paciente terminal, el BAJA + 2 el que aún puede ser salvado, pero se encuentra en riesgo y el CONTINUA una persona sana, por ende los pacientes sanos y los terminales no deben ser agrupados ya que es poco probable que el modelo pueda hallar la categoría intermedia. Por todo esto se decidió agrupar BAJA+ 1 con BAJA+2 para realizar todo el desarrollo a continuación, esto, sin embargo, puede ser puesto en discusión en futuros estudios pero no es objeto de esta tesina.

## 4.4. Estimación de la predicción de los modelos con data sets desbalanceados

En los modelos de clasificación suele ser común encontrar que en el conjunto de datos de entrenamiento existen algunas de las clases de muestra que son “minoritaria” es decir, de la cual se tienen muy poquitas muestras. Esto *provoca que el dataset este desbalanceado.*

Un ejemplo claro es en el área de Salud en donde suele haber conjuntos de datos con miles de registros con pacientes “negativos” y unos pocos casos positivos es decir, que padecen la enfermedad que queremos clasificar.

El criterio que suele utilizarse a la hora de evaluar modelos predictivos es el de ver cuantos aciertos se han dado en función de los casos totales. Este criterio de *accuracy* o de precisión no sirve para los data sets que están tan desbalanceados como el nuestro:

En primer lugar, si se decidiera no hacer la campaña, es decir no enviar el estímulo a nadie, se tendría un nivel de *accuracy* o de acierto del 99,6% ya que únicamente no se estaría “acertando” a quien se envía el estímulo en el caso de los 975 casos de BAJA+2.

Por ello el enfoque que se toma para evaluar los modelos predictivos no es en función de los aciertos sino de la ganancia que le traería potencialmente a la compañía calculada a través de la función Ganancia enunciada en el apartado anterior.

## 4.5. Probabilidad de corte

Los modelos que se generan en esta tesina por los distintos métodos dan como resultado una estimación del riesgo que tiene cada usuario de abandonar la compañía, es decir que la salida del modelo es una probabilidad por cada cliente. Pero cómo saber si esa probabilidad de salida es lo suficientemente significativa como para enviarle un estímulo. Para ello se busca una probabilidad de corte, que es independiente de la metodología por la cual son generadas las probabilidades de cada uno de los usuarios.

Para hallar esta probabilidad óptima de corte se realiza la siguiente derivación matemática:

Para que Ganancia >0

Dado que

Es decir que la probabilidad de ser BAJA+2 debe ser mayor a 0,025 para que genere una esperanza positiva de ganancias el enviarles un estímulo.

## 4.6. Regresión logística

Para comenzar el modelo logístico se recodifica la variable clase ternaria tomando valor 1 cuando es BAJA+1 y BAJA+2 y 0 cuando es CONTINUA.

Luego se realiza un primer modelo utilizando el paquete de R *“dplyr”.*

El link canónico utilizado para la estimación es el *logit* que se deriva de la distribución de probabilidades binomial. A continuación, la demostración (Duncan Anderson, 2007.):

Se genera un primer modelo de regresión logística con todas las variables contenidas en el data set, se analizan los p-value de los coeficientes para cada una de estas variables y se descubre en este caso un modelo con un problema de *overfitting* donde el mensaje que se obtiene es de advertencia:

1: glm.fit: el algoritmo no converge

2: glm.fit: las probabilidades ajustadas son numéricamente 0 o 1

Donde se observa que muchas variables no son significativas.

Sumado a esto se decide hacer un estudio de los errores del modelo donde se evidencian problemas importantes de asimetría.

Gráfico

Descripción generada automáticamente con confianza media

Figura.7. Residuos de Deviance de regresión logística generada con todo el dataset. Fuente: Elaboración propia.

En el anexo se puede ver la salida completa del modelo.

Para solucionar estos problemas del modelo se decidió realizar una regresión logística penalizada, en la cual se impone una “penalización” al modelo logístico por tener demasiadas variables. Esto consiste en hacer que los coeficientes de las variables menos contributivas se reduzcan a cero. Esto también se conoce como regularización. (Kassambara, 2018)

En esta tesina se utiliza la forma de penalización más común: la de Lasso donde los coeficientes de algunas variables de menor contribución se fuerzan para que sean exactamente cero. Sólo se mantienen en el modelo final las variables más significativas.

El modelo resultante es:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Coefficients: |  |  |  |  |
|  | Estimate | Std. Error | z value | Pr(>|z|) |
| (Intercept) | 5,30E+02 | 2,33E+02 | 2.272 | 0.023082 |
| active\_quarter | -5,06E+02 | 8,96E+01 | -5.645 | 1.65e-08 |
| cliente\_antiguedad | 5,26E-01 | 3,59E-01 | 1.467 | 0.142265 |
| cproductos | -1,47E+02 | 3,26E+01 | -4.512 | 6.41e-06 |
| mcuenta\_corriente | 2,31E-03 | 2,71E-04 | 8.499 | < 2e-16 |
| mcaja\_ahorro | -3,31E-03 | 9,33E-04 | -3.553 | 0.000381 |
| mcaja\_ahorro\_dolares | 1,09E-03 | 2,37E-04 | 4.588 | 4.47e-06 |
| mcuentas\_saldo | -2,06E-03 | 3,45E-04 | -5.981 | 2.22e-09 |
| mautoservicio | -1,52E-02 | 4,43E-03 | -3.421 | 0.000624 |
| ctarjeta\_visa | -6,31E+02 | 2,35E+02 | -2.688 | 0.007189 |
| ctarjeta\_visa\_transacciones | -8,11E+01 | 8,36E+00 | -9.698 | < 2e-16 |
| mtarjeta\_visa\_consumo | 6,93E-04 | 2,44E-03 | 0.284 | 0.776410 |
| ctarjeta\_master | -1,65E+03 | 2,36E+02 | -6.988 | 2.79e-12 |
| ctarjeta\_master\_transacciones | -2,25E+01 | 1,70E+01 | -1.329 | 0.183818 |
| mtarjeta\_master\_consumo | -8,28E-03 | 4,22E-03 | -1.960 | 0.050001 |
| mprestamos\_personales | -1,27E-02 | 2,14E-03 | -5.911 | 3.39e-09 |
| mprestamos\_prendarios | 5,35E-05 | 4,31E-04 | 0.124 | 0.901191 |
| mprestamos\_hipotecarios | -1,10E-02 | 9,32E-03 | -1.181 | 0.237614 |
| mplazo\_fijo\_dolares | -1,89E-03 | 5,33E-04 | -3.538 | 0.000404 |
| mplazo\_fijo\_pesos | -1,98E-01 | 5,61E+00 | -0.035 | 0.971809 |
| mpayroll | -8,92E-03 | 1,27E-03 | -7.009 | 2.40e-12 |
| mcomisiones\_mantenimiento | 3,08E-01 | 5,21E-02 | 5.901 | 3.61e-09 |
| mcomisiones\_otras | -9,28E-02 | 4,49E-02 | -2.068 | 0.038658 |
| mcheques\_depositados\_rechazados | -2,80E+00 | 9,85E+01 | -0.028 | 0.977293 |
| mcheques\_emitidos\_rechazados | -2,40E+00 | 6,20E+01 | -0.039 | 0.969171 |
| ctrx\_quarter | -1,28E+01 | 1,21E+00 | -10.527 | < 2e-16 |
| Master\_delinquency | -2,04E+03 | 5,06E+02 | -4.025 | 5.70e-05 |
| Master\_msaldototal | 4,94E-03 | 1,82E-03 | 2.713 | 0.006663 |
| Master\_mlimitecompra | 4,42E-06 | 1,32E-04 | 0.034 | 0.973241 |
| Master\_mpagado | 3,18E-03 | 3,14E-03 | 1.011 | 0.311810 |
| Visa\_delinquency | -1,31E+02 | 1,88E+02 | -0.694 | 0.487828 |
| Visa\_msaldototal | -1,36E-03 | 1,86E-03 | -0.732 | 0.464049 |
| Visa\_mlimitecompra | 7,81E-05 | 1,41E-04 | 0.553 | 0.580123 |
| Visa\_mpagado | 5,30E-03 | 1,84E-03 | 2.880 | 0.003980 |

A este modelo se intentó mejorarlo quitando las variables no significativas, pero se obtuvieron ajustes peores por lo que se decidió quedarse con este modelo.

Los resultados de este modelo se encuentran en la sección de Resultados.

## 4.7. Árboles de decisión

Se comienza a generar árboles de decisión en R Studio utilizando la librería “*rpart*”.

El proceso por el que se arma el modelo predictivo consiste en particionar la data del mes de prueba en 5 *folds* y generar un árbol de decisión simple con la información contenida en 4 de ellos para luego probarlo en el *fold* restante. Se realiza un cálculo de la ganancia de aplicar el programa de incentivos a ese conjunto de data de prueba y se multiplica por 5 para que sea representativo de el set de datos sin particionar. Luego se repite este proceso análogamente 4 veces más para tener una ganancia estimada para cada conjunto de folds y se promedia.

Los hiperparámetros de estos árboles son determinados mediante la maximización de la función de ganancia por un proceso de iteración. Para ello se toma todo el conjunto de operaciones antes descripto (un árbol entrenado con validación cruzada) y se itera 150 veces a través de las funciones de la librería de R "*mlrMBO*" que es una opción práctica operacionalmente para realizar el proceso de optimización bayesiana de hiperparámetros. La búsqueda que realiza el programa sigue la lógica de generar una pequeña base de datos con hiperparámetros y ganancias y luego generar modelos predictivos que permita optimizar dónde buscar los hiperparámetros del siguiente árbol en función de incrementar las ganancias obtenidas.

A continuación, se muestran los primeros dos niveles del árbol generado con las variables más importantes seleccionadas:

ctrx\_quarter: Cantidad de movimientos voluntarios en las cuentas bancarias (no tarjeta de crédito) que el cliente realizó en los últimos 90 días.

mpasivos\_margen: Monto total de la ganancia que el banco ha obtenido por el dinero/inversiones que el cliente tiene en el banco.

Diagrama

Descripción generada automáticamente

Esta figura busca mostrar de una forma gráfica y entendible los primeros niveles de cómo se compone el árbol de decisión.

Si un cliente realiza voluntariamente más de 19 operaciones es altamente improbable que pida la baja en dos meses.

Luego de generar el modelo, y de clasificar a los usuarios en función de este se utiliza la misma probabilidad de corte que en el caso de la logística, es decir que con más 0,025.

## 4.8. Random Forest

Se comienza a generar árboles de decisión en R Studio utilizando la librería “*ranger*”. El proceso de generación de este modelo es operacionalmente análogo al de los árboles de decisión del segmento anterior. Se genera un modelo de *random* *forest*, se lo entrena con validación cruzada y luego se maximizan las ganancias mediante la optimización de los hiperparámetros relacionados a este método utilizando los mismos métodos de optimización bayesiana ya descriptos.

Los hiperparámetros que se maximizan son:

-*num.trees*: número de árboles a incluir en el modelo de *Random Forest.*

-*mtry*: número de variables predictoras como candidatas en cada ramificación. Por defecto, para problemas de clasificación viene dado por la raíz cuadrada del número de variables, y para la regresión, el número de variables dividido entre 3.

- *min.node.size:* mínimo número de muestras dentro de los nodos terminales. Para casos de clasificación, por defecto será 2, y para regresión será 5.

-*max.depth*: máximo número de nodos terminales. Por defecto, no se planifica un proceso de poda, dejando crecer los árboles hasta su límite máximo.

El modelo generado no puede ser visualizado gráficamente con un árbol como en el modelo anterior ni a través de la formación de una ecuación con coeficientes por cada variable predictora como es el caso de la regresión logística debido a su gran complejidad. Es por ello se adjuntan directamente los resultados obtenidos en la sección de resultados.

## 4.9. LightGBM

Como en los otros modelos utilizados en R Studio, se instalan las librerías necesarias, en particular se instala “*lightgbm*” con la que se comienza a generar los modelos. Nuevamente se define la función ganancia a maximizar a través de la elección de hiperparámetros. Estos están limitados entre ciertos rangos. Luego se diseña el proceso de tal forma que se alcance una ganancia máxima iterando en el conjunto del dataset de prueba, pero de forma fragmentada mediante el proceso de *cross validation*.

Estos hiperparámetros a optimizar son:

-Num\_leaves: Controla el número de hojas de un árbol. LightGBM utiliza un algoritmo de crecimiento de árboles por hojas, por lo que num\_leaves es el parámetro principal para controlar la complejidad del árbol.

-Min\_data\_in\_leaf: Determina el número mínimo de muestras necesarias para estar en una hoja, lo cual es muy importante para el control del sobreajuste.

-Feature\_fraction: Es la proporción de características que se seleccionan al azar en cada nodo.

Finalmente, la salida del modelo para el conjunto de datos de prueba determina el conjunto de casos (clientes) con una alta probabilidad de fuga de la compañía, y, por el contrario, al conjunto de casos con una baja probabilidad de fuga, es decir, de continuidad en la misma. A estos datos resultantes se le aplica la función ganancia, generando así un valor potencial de ingresos del programa de estímulos.

Al igual que en el caso de *Random Forest,* ver gráficamente o con estimadores o fórmulas este modelo predictivo no es posible, por ello directamente se muestran los resultados en la sección siguiente.

Con respecto a las características de la corrida de este modelo se destaca una mayor rapidez con respecto a los otros modelos predictivos, sin embargo, con estos volúmenes de datos no representa una mejora de tiempos significativa.

# V. Resultados de los modelos entrenados con datos de noviembre para enero.

La salida de los cuatro modelos predictivos es un archivo con el id del cliente y si el modelo predice que ese usuario va a ser un BAJA o no. Se genera la ganancia potencial como -$1,250 por todos los usuarios que son BAJA y por ende son receptores del estímulo y $48750 por todos aquellos cuya clase ternaria real efectivamente era BAJA+2.

Con todo esto sumamos las ganancias por modelo y obtenemos:



Se observa que LightGBM genera ganancias 17% mayores a las que provee la regresión logística penalizada.

# VI. Conclusiones

De acuerdo con lo observado, realizar un análisis de riesgo de salida de un cliente genera efectivamente ventajas para el negocio, independientemente del modelo que se elija para generar el modelo predictivo.

Las posibilidades de generar modelos, de mejorar su ajuste a través de optimización de parámetros, de mejoras en la toma de muestras de los datos y derivados son diversas y numerosas. A lo largo del proceso de selección de modelos para su exposición surgieron una gran cantidad de posibilidades, pero se tomaron en cuenta aquellas que se consideraron convenientes, sin embargo, se comprueba que el universo de herramientas para una correcta estimación del riesgo es variado y que está en continuo crecimiento.

Concretamente se observa que de los métodos estudiados el que mejores resultados arroja para este problema en específico y con estos datos es el de *LightGBM* de Microsoft debido a que predice las mayores ganancias. Estos resultados resultan provisorios, a la espera de nuevas puestas a prueba para la estimación de otros riesgos o con la utilización de otros datos, que la confirmen o pongan nuevamente en discusión.

# VII. Líneas futuras de investigación.

Queda pendiente para próximas líneas de investigación el estudio de nuevos algoritmos de aprendizaje automático diferentes a los aquí estudiados, tales como los algoritmos XGBoost.

También podría ser interesante para próximos trabajos la utilización de máquinas virtuales con más capacidad de procesamiento que permitirían más iteraciones a las optimizaciones bayesianas para mejorar la robustez de los modelos.

Por último, otra posibilidad de ampliar el estudio de la problemática de retención de clientes es el poder probar la efectividad del estímulo y la relación entre la cantidad de dinero propuesta para ese fin y cuantos usuarios efectivamente toman la decisión de quedarse. Sería conveniente personalizar el programa por cliente o por grupos de clientes para enviar el mínimo valor necesario para que éstos decidan quedarse y maximizando de esta forma las ganancias aún más.

# Referencias bibliográficas

Avaus. (s.f.). Obtenido de https://www.avaus.com/blog/predicting-customer-churn/

Breiman, L. (Septiembre de 1994). Bagging Predictors. University of California. Obtenido de https://www.stat.berkeley.edu/~breiman/bagging.pdf

Departamento de Matemáticas, Facultad de exactas, Universidad de Buenos Aires. (2005). Modelos Lineales Generalizados. Obtenido de http://www.dm.uba.ar/materias/modelos\_lineales\_generalizados\_Mae/2005/1/glm5.pdf

Duncan Anderson, F. S. (2007.). *A Practitioner's Guide to Generalized Linear Models.Third Edition.* CAS.

Epidat 4. (2014). *Regresión logística.* Obtenido de https://www.sergas.es/Saude-publica/Documents/1898/Ayuda\_Epidat4\_Regresion\_logistica\_Octubre2014.pdf

Freund, R. E. (s.f.). *Boosting. Foundations and Algorithms.* Cambridge: The MIT Press.

Gumpta, P. (s.f.). *towardsdatascience.* Obtenido de https://towardsdatascience.com/cross-validation-in-machine-learning-72924a69872f

Hosmer, D. a. (2000). *Applied logistic regression. 2nd Edition.* New York : John Wiley & Sons, Inc.

Kassambara. (11 de marzo de 2018). Penalized Logistic Regression Essentials in R: Ridge, Lasso and Elastic Net. Obtenido de http://www.sthda.com/english/articles/36-classification-methods-essentials/149-penalized-logistic-regression-essentials-in-r-ridge-lasso-and-elastic-net/

Morales, J. (2021). *Modelos Estadísticos.* Universidad Miguel Hernández de Elche.

Rodrigo Amat, J. (Abril de 2020). Obtenido de https://www.cienciadedatos.net/documentos/62\_optimizacion\_bayesiana\_hiperparametros.html

*Wikipedia*. (s.f.). Obtenido de https://commons.wikimedia.org/wiki/File:Overfitting.svg

# Anexos

## Anexo 1. Diccionario de datos: descripción de las variables disponibles en el dataset.

|  |  |  |  |
| --- | --- | --- | --- |
| # | campo | unidad | Significado |
| 1 | numero\_de\_cliente | id | ID único que se asigna a cada cliente. Clientes mas nuevos tienen Ids mayores. Si un ex cliente regresa se le asigna el numero\_de\_cliente que ya tuvo asignado la primera vez. |
| 2 | foto\_mes | fecha | mes correspondiente a la foto de los datos del datawarehouse. La fecha 201907 indica que la foto fue obtenida el 31-julio-2019 a las 23:59:59 |
| 3 | active\_quarter | -- | Indica si el cliente ha realizado transacciones voluntarias en el ultimo timestre . Que el banco cobre la comisión de mantenimiento de cuenta NO es un movimiento voluntario por mas que es un debito en la cuenta. Que se cobre una cuota de una compra con tarjeta de crédito, *tampoco* es un movimiento voluntario. |
| 4 | cliente\_vip | -- | Indica si marketing considera a esa cliente un cliente vip al momento de obtención de la foto. A pesar que todos estos son clientes de Paquete Premium, el area de marketing solo considera vip a menos del 2% de los paquete premium , y da a estos clientes VIP un trato a puertas cerradas. |
| 5 | internet | -- | indica si el cliente usa servicios de HomeBanking o tiene instalada la app |
| 6 | cliente\_edad | años | Edad en años del cliente. |
| 7 | cliente\_antiguedad | meses | Antiguedad medida en meses de el cliente. Es de la ultima vez que esa persona reingresó como cliente del banco. |
| 8 | mrentabilidad | pesos | Ganancia total que ha obtenido el banco de ese cliente, en ese mes. |
| 9 | mrentabilidad\_annual | pesos | Ganancia total que el banco ha obtenido de ese cliente en el ultimo año de relacion cliente-banco, o meses desde que ingresó si es cliente reciente. |
| 10 | mcomisiones | pesos | Monto total de las comisiones que ha ganado el banco por ese cliente. |
| 11 | mactivos\_margen | pesos | Monto total de la ganancia que el banco ha obtenido en concepto de intereses que ha cobrado al cliente. |
| 12 | mpasivos\_margen | pesos | Monto total de la ganancia que el banco ha obtenido por el dinero/inversiones que el cliente tiene en el banco. |
| 13 | cproductos | -- | Cantidad de productos que el cliente posee con el banco. Se cuentan las familias de productos. |
| 14 | tpaquete2 | -- |  |
| 15 | tpaquete3 | -- |  |
| 16 | tpaquete4 | -- |  |
| 17 | tcuentas | -- | Cantidad de cuentas que el cliente tiene, vale 0 si no tiene ninguna, 1 si solo tiene cajas de ahorro o solo cuentas corrientes, 2 si tiene al menos una caja de ahorro y tambien al menos una cuenta corriente. |
| 18 | ccuenta\_corriente | -- | Cantidad de cuentas corrientes que tiene el cliente. Hay muy pocos clientes con mas de una cuenta corriente. |
| 19 | mcuenta\_corriente\_adicional | pesos | Monto total de las cuentas corrientes adicionales que no forman parte del paquete. |
| 20 | mcuenta\_corriente | pesos | Monto total de las cuenta corriente del paquete premium |
| 21 | ccaja\_ahorro | -- | Cantidad de cajas de ahorro que tiene el cliente. |
| 22 | mcaja\_ahorro | pesos | Monto total de la caja de ahorro del Paquete Premium |
| 23 | mcaja\_ahorro\_adicional | pesos | Monto total de las cajas otras cajas de ahorro que no forman parte del paquete. |
| 24 | mcaja\_ahorro\_dolares | pesos | Monto total de las cajas de ahorro en dólares. El valor esta expresado en pesos, y se considera el valor del dolar de cierre del último dia hábil del mes. |
| 25 | mdescubierto\_preacordado | pesos | Monto total de los descubiertos acordados en las cuentas corrientes. |
| 26 | mcuentas\_saldo | pesos | Saldo total de TODAS las cuentas del cliente, cajas de ahorro, cuentas corrientes, pesos y dolares. El valor esta convertido a pesos. |
| 27 | ctarjeta\_debito | -- | Cantidad de tarjetas de débito que posee el cliente. |
| 28 | ctarjeta\_debito\_transacciones | -- | Cantidad de transacciones que hizo el cliente con su tarjeta de débito durante el mes. |
| 29 | mautoservicio | pesos | Monto total de las transacciones que hizo el cliente con su tarjeta de débito duirante el mes. |
| 30 | ctarjeta\_visa | -- | Cantidad de cuentas VISA que posee el cliente. Una cuenta posee un titular y puede llegar a tener varios adicionales. El titular y/o adicionales pueden tener ( o no ) un plástico vigente. Aunque no es común, un cliente SI puede ser titular de más de una cuenta VISA . |
| 31 | ctarjeta\_visa\_transacciones | -- | Cantidad de transacciones efectuadas durante el mes con la tarjeta de crédito VISA |
| 32 | mtarjeta\_visa\_consumo | pesos | Monto total de los consumos efectuados durante el mes con la tarjeta de crédito VISA |
| 33 | ctarjeta\_master | -- | Cantidad de cuentas Mastercard que tiene la persona. Una cuenta puede tener un plástico titular y varias plásticos adicionales. |
| 34 | ctarjeta\_master\_transacciones | -- | Cantidad de transacciones efectuadas durante el mes con la tarjeta de crédito MasterCard |
| 35 | mtarjeta\_master\_consumo | pesos | Monto total de los consumos efectuados durante el mes con la tarjeta de crédito MasterCard |
| 36 | cprestamos\_personales | -- | Cantidad de préstamos personales vigentes del cliente |
| 37 | mprestamos\_personales | pesos | Monto total de la deuda restante de todos los préstamos personales del cliente. |
| 38 | cprestamos\_prendarios | -- | Cantidad de préstamos prendarios vigentes del cliente |
| 39 | mprestamos\_prendarios | pesos | Monto total de la deuda restante de todos los préstamos prendarios del cliente. |
| 40 | cprestamos\_hipotecarios | -- | Cantidad de préstamos hipotecarios vigentes del cliente |
| 41 | mprestamos\_hipotecarios | pesos | Monto total de la deuda restante de todos los préstamos hipotecarios del cliente. |
| 42 | cplazo\_fijo | -- | Cantidad de plazos fijos que tiene el cliente Se suma la cantidad de plazos fijos en pesos mas la cantidad de plazos fijos en dolares. |
| 43 | mplazo\_fijo\_dolares | pesos | Monto total de todos los plazos fijos vigentes el dólares. El valor está convertido a pesos considerando el valor de cierre del ultimo dia habil del mes. |
| 44 | mplazo\_fijo\_pesos | pesos | Monto total de todos los plazos fijos vigentes en pesos. |
| 45 | cinversion1 | -- | Cantidad de inversiones de tipo 1 que posee el cliente vigentes. |
| 46 | minversion1\_pesos | pesos | Monto total de inversiones tipo1 , convertidos en pesos |
| 47 | minversion1\_dolares | pesos | Monto total de inversiones tipo1 en dolares , expresado en pesos al ultimo dia habil del mes |
| 48 | cinversion2 | -- | Cantidad de inversiones de tipo2 que posee el cliente vigentes. |
| 49 | minversion2 | pesos | Monto total de la inversión de tipo2 expresada en pesos. |
| 50 | cseguro\_vida | -- | Cantidad de seguros de vida contratados por el cliente. No se tienen en cuenta los seguros de vida obligatorios de los prestamos hipotecarios, prendarios, personales, etc |
| 51 | cseguro\_auto | -- | Cantidad de seguros automotores que el cliente posee vigentes. |
| 52 | cseguro\_vivienda | -- | Cantidad de seguros de vivienda que tiene el cliente. |
| 53 | cseguro\_accidentes\_personales | -- | Cantidad de seguros de accidentes personales |
| 54 | ccaja\_seguridad | -- | Si el cliente tiene cajas de seguridad. Un 0 si no tiene ninguna, un 1 si tiene al menos una caja de seguridad. { 0, 1 } son los unicos dos valores posibles . |
| 55 | cpayroll\_trx | -- | Cantidad de Acreditaciones de Haberes en relación de depencia que le hicieron al cliente en ese mes. Un cliente puede estar en relacion de dependencia con mas de una empresa. Una empresa puede hacerle VARIOS depósitos al mismo empleado durante el mes. Soalmente se consideran las acreditaciones de empresas que tienen un contrato con el banco. |
| 56 | mpayroll | pesos | Monto total que le acreditaron los empleadores “acreditados” al cliente durante el mes. |
| 57 | mpayroll2 | pesos | Monto total que le acreditaron fuera de archivo de los empleadores “acreditados” al cliente durante el mes. |
| 58 | cpayroll2\_trx | -- | Cantidad de transacciones de acreditacion de haberes en el mes |
| 59 | ccuenta\_debitos\_automaticos | -- | Cantidad de débitos automáticos debitados durante el mes en las cuentas ( no tarjetas de crédito ) |
| 60 | mcuenta\_debitos\_automaticos | pesos | Monto total de débitos automáticos debitados durante el mes en las cuentas ( no tarjetas de crédito ) . Todo está convertido a pesos argentinos, al valor de la fecha de cierre del utlimo dia habil del mes. |
| 61 | ctarjeta\_visa\_debitos\_automaticos | -- | Cantidad de débitos automáticos debitados durante el mes en las tarjeta de crédito VISA |
| 62 | mtarjeta\_visa\_debitos\_automaticos | pesos | Monto total de débitos automáticos debitados durante el mes en la tarjeta de crédito VISA . Todo está convertido a pesos argentinos, al valor de la fecha de cierre del utlimo dia habil del mes. |
| 63 | ctarjeta\_master\_debitos\_automaticos | -- | Cantidad de débitos automáticos debitados durante el mes en las tarjeta de crédito MasterCard |
| 64 | mttarjeta\_master\_debitos\_automaticos | pesos | Monto total de débitos automáticos debitados durante el mes en la tarjeta de crédito MasterCard . Todo está convertido a pesos argentinos, al valor de la fecha de cierre del utlimo dia habil del mes. |
| 65 | cpagodeservicios | -- | Cantidad de pagos de servicios efectuados durante el mes. |
| 66 | mpagodeservicios | pesos | Monto total de pagos de servicios efectuados durante el mes. |
| 67 | cpagomiscuentas | -- | Cantidad de pagos efectuados durante el mes por el canal PagoMisCuentas |
| 68 | mpagomiscuentas | pesos | Monto total en pesos argentinos de pagos efectuados durante el mes por el canal PagoMisCuentas |
| 69 | ccajeros\_propios\_descuentos | -- | Cantidad de descuentos con los que se benefició el cliente al utilizar su tarjeta de débito durante el mes. |
| 70 | mcajeros\_propios\_descuentos | pesos | Monto total en pesos de descuentos con los que se benefició el cliente al utilizar su tarjeta de débito durante el mes. |
| 71 | ctarjeta\_visa\_descuentos | -- | Cantidad de descuentos con los que se benefició el cliente al utilizar su tarjeta de crédito VISA durante el mes. |
| 72 | mtarjeta\_visa\_descuentos | pesos | Monto total en pesos de descuentos con los que se benefició el cliente al utilizar su tarjeta de crédito VISA durante el mes. |
| 73 | ctarjeta\_master\_descuentos | -- | Cantidad de descuentos con los que se benefició el cliente al utilizar su tarjeta de crédito MasterCard durante el mes. |
| 74 | mtarjeta\_master\_descuentos | pesos | Monto total en pesos de descuentos con los que se benefició el cliente al utilizar su tarjeta de crédito MasterCard durante el mes. |
| 75 | ccomisiones\_mantenimiento | -- | Cantidad de comisiones de mantenimiento de productos cobrardas durante el banco durante el mes. |
| 76 | mcomisiones\_mantenimiento | pesos | Monto total en pesos de comisiones de mantenimiento de productos cobrardas durante el banco durante el mes. |
| 77 | ccomisiones\_otras | -- | Cantidad de *otras comisiones* que se cobraron al cliente durante el mes. |
| 78 | mcomisiones\_otras | pesos | Monto total en pesos de *otras comisiones* que se cobraron al cliente durante el mes. |
| 79 | cforex | -- | Cantidad de transacciones de cambio de moneda que realizó el cliente durante el mes. |
| 80 | cforex\_buy | -- | Cantidad de transacciones de compra de moneda extranjera que realizó el cliente durante el mes |
| 81 | mforex\_buy | pesos | Monto total medido en pesos de transacciones de compra de moneda extranjera que realizó el cliente durante el mes |
| 82 | cforex\_sell |  | Cantidad de transacciones de venta de moneda extranjera que realizó el cliente durante el mes |
| 83 | mforex\_sell | pesos | Monto total medido en pesos de transacciones de venta de moneda extranjera que realizó el cliente durante el mes |
| 84 | ctransferencias\_recibidas | -- | Cantidad de transferencias recibidas en todas las cuentas durante el mes. Puede ser transferencias propias o de terceros. |
| 85 | mtransferencias\_recibidas | pesos | Monto total de transferencias recibidas en todas las cuentas durante el mes. Puede ser transferencias propias o de terceros. |
| 86 | ctransferencias\_emitidas | -- | Cantidad de transferencias emitidas en todas las cuentas durante el mes. Puede ser transferencias propias o de terceros. |
| 87 | mtransferencias\_emitidas | pesos | Monto total de transferencias emitidas en todas las cuentas durante el mes. Puede ser transferencias propias o de terceros. |
| 88 | cextraccion\_autoservicio | -- | Cantidad de extracciones en cajeros automáticos durante el mes |
| 89 | mextraccion\_autoservicio | pesos | Monto total de extracciones en cajeros automáticos durante el mes |
| 90 | ccheques\_depositados | -- | Cantidad de cheques que se depositaron en las cuentas del cliente durante el mes. |
| 91 | mcheques\_depositados | pesos | Monto total de los cheques que se depositaron y se pudo hacer el cobro , en las cuentas del cliente durante el mes. |
| 92 | ccheques\_emitidos | -- | Cantidad de cheques del cliente que se cobraron ( ya sea por el cliente mismo o por terceros ) durante el mes . |
| 93 | mcheques\_emitidos | pesos | Monto total de cheques del cliente que se cobraron ( ya sea por el cliente mismo o por terceros ) durante el mes . |
| 94 | ccheques\_depositados\_rechazados | -- | Cantidad de cheques que se depositaron en cuentas del cliente y fueron rechazados durante el mes. |
| 95 | mcheques\_depositados\_rechazados | pesos | Monto total de los cheques que se depositaron en cuentas del cliente y fueron rechazados durante el mes. |
| 96 | ccheques\_emitidos\_rechazados | -- | Cantidad de cheques emitidos por el cliente que fueron rechazados durante el mes. |
| 97 | mcheques\_emitidos\_rechazados | pesos | Monto total de cheques emitidos por el cliente que fueron rechazados durante el mes. |
| 98 | tcallcenter | -- | {0,1} indica si la persona está adherida al canal de banca telefónica . |
| 99 | ccallcenter\_transacciones | -- | Cantidad de transacciones que el cliente efectuó durante el mes por el canal de banca telefónica. |
| 100 | thomebanking | -- | {0, 1} indica si el cliente está adherido al Home Banking |
| 101 | chomebanking\_transacciones | -- | Cantidad de transacciones por home banking que el cliente hizo durante el mes. |
| 102 | ccajas\_transacciones | -- | Cantidad de transacciones que el cliente realizó durante el mes en la linea de cajas de las sucursales del banco. |
| 103 | ccajas\_consultas | -- | Cantidad de consultas que el cliente realizó durante el mes en la linea de cajas de las sucursales del banco. |
| 104 | ccajas\_depositos | -- | Cantidad de depóstitos que el cliente realizó durante el mes en la linea de cajas de las sucursales del banco. |
| 105 | ccajas\_extracciones | -- | Cantidad de extracciones que el cliente realizó durante el mes en la linea de cajas de las sucursales del banco. |
| 106 | ccajas\_otras | -- | Cantidad de otros tipos de transacciones que el cliente realizó durante el mes en la linea de cajas de las sucursales del banco. |
| 107 | catm\_trx | -- | Cantidad de transacciones que el cliente realizó durante el mes en cajeros automáticos propiedad del banco. |
| 108 | matm | pesos | Monto total en pesos de transacciones que el cliente realizó durante el mes en cajeros automáticos propiedad del banco. |
| 109 | catm\_trx\_other | -- | Cantidad de transacciones que el cliente realizó durante el mes en cajeros automáticos que no son propiedad del banco. |
| 110 | matm\_other | pesos | Monto total en pesos de transacciones que el cliente realizó durante el mes en cajeros automáticos que no son propiedad del banco. |
| 111 | ctrx\_quarter | -- | Cantidad de movimientos voluntarios en las cuentas bancarias ( no tarjeta de credito ) que el cliente realizó en los ultimos 90 dias. |
| 112 | tmobile\_app | -- | { 0, 1} indica si la persona se instaló alguna vez la aplicación movil |
| 113 | cmobile\_app\_trx | -- | indica la cantidad de transacciones realizadas ese mes desde la aplicacion móvil |
| 114 | Master\_delinquency | -- | { 0, 1 } indica si el cliente no llegó a completar el pago mínimo y esta moroso. Estos clientes NO pueden utilizar mas la tarjeta hasta tanto regularicen su situacion. Obviamente si les caen las cuotas de compras anteriores, intereses, punitorios, etc |
| 115 | Master\_status | -- | { 0, 6, 7, 9 } indica el estado de la cuenta de la tarjeta de crédito. 0 abierta, 6 en proceso de cierre, 7 en proceso avanzado de cierre, 9 cuenta cerrada. Una cuenta cerrada puede volver a abrirse !! |
| 116 | Master\_mfinanciacion\_limite | pesos | Límite de financiacion de la tarjeta de crédito, expresado en pesos. |
| 117 | Master\_Fvencimiento | dias | Dias para el vencimiento del plastico de la tarjeta de crédito, contados a la fecha de la foto. |
| 118 | Master\_Finiciomora | dias | Dias desde el inicio de la mora ( el dia siguiente al vencimiento ), contados a la fecha de la foto. |
| 119 | Master\_msaldototal | pesos | Saldo total de la tarjeta, para ese mes. |
| 120 | Master\_msaldopesos | pesos | Saldo total el pesos de la tarjeta, para ese mes. |
| 121 | Master\_msaldodolares | pesos | Saldo total el dólares de la tarjeta, para ese mes. |
| 122 | Master\_mconsumospesos | pesos | Monto total de los consumos en pesos efectuados por el cliente durante ese mes. |
| 123 | Master\_mconsumosdolares | pesos | Monto total de los consumos en dolares efectuados por el cliente durante ese mes. |
| 124 | Master\_mlimitecompra | pesos | Límite de compra, valor muy importante. |
| 125 | Master\_madelantopesos | pesos | Extracciones de pesos realizadas con la tarjeta de crédito durante ese mes. |
| 126 | Master\_madelantodolares | pesos | Extracciones de dolares realizadas con la tarjeta de crédito durante ese mes. |
| 127 | Master\_fultimo\_cierre | dias | Dias desde del utlimo cierre de la tarjeta de crédito, contados a la fecha de la foto. |
| 128 | Master\_mpagado | pesos | Monto total de todos los pagos efectuados por el cliente |
| 129 | Master\_mpagospesos | pesos | Monto total de todos los pagos en pesos efectuados por el cliente |
| 130 | Master\_mpagosdolares | pesos | Monto total de todos los pagos en dólares efectuados por el cliente |
| 131 | Master\_fechaalta | dias | Dias desde de alta de la cuenta de la tarjeta de crédito, contados a la fecha de la foto. |
| 132 | Master\_mconsumototal | pesos | Monto total expresado en pesos, de todos los consumos ( pesos y dolares ) efectuados por el cliente durante ese mes. |
| 133 | Master\_cconsumos | -- | Cantidad de consumos con la tarjeta de credito efectuados por el cliente durante el mes. |
| 134 | Master\_cadelantosefectivo | -- | Cantidad de adelantos en efectivo que el cliente realizó durante ese mes. |
| 135 | Master\_mpagominimo | pesos | Monto del pago minimo necesario para no ser moroso de la tarjeta de crédito |
| 136 | Visa\_delinquency | -- | { 0, 1 } indica si el cliente no llegó a completar el pago mínimo y esta moroso. Estos clientes NO pueden utilizar mas la tarjeta hasta tanto regularicen su situacion. Obviamente si les caen las cuotas de compras anteriores, intereses, punitorios, etc |
| 137 | Visa\_status | -- | { 0, 6, 7, 9 } indica el estado de la cuenta de la tarjeta de crédito. 0 abierta, 6 en proceso de cierre, 7 en proceso avanzado de cierre, 9 cuenta cerrada. Una cuenta cerrada puede volver a abrirse !! |
| 138 | Visa\_mfinanciacion\_limite | pesos | Límite de financiacion de la tarjeta de crédito, expresado en pesos. |
| 139 | Visa\_Fvencimiento | dias | Dias para el vencimiento del plastico de la tarjeta de crédito, contados a la fecha de la foto |
| 140 | Visa\_Finiciomora | dias | Dias para el inicio de la mora ( el dia siguiente al vencimiento ), contados a la fecha de la foto |
| 141 | Visa\_msaldototal | pesos | Saldo total de la tarjeta, para ese mes. |
| 142 | Visa\_msaldopesos | pesos | Saldo total el pesos de la tarjeta, para ese mes. |
| 143 | Visa\_msaldodolares | pesos | Saldo total el dólares de la tarjeta, para ese mes. |
| 144 | Visa\_mconsumospesos | pesos | Monto total de los consumos en pesos efectuados por el cliente durante ese mes. |
| 145 | Visa\_mconsumosdolares | pesos | Monto total de los consumos en dolares efectuados por el cliente durante ese mes. |
| 146 | Visa\_mlimitecompra | pesos | Límite de compra, valor muy importante. |
| 147 | Visa\_madelantopesos | pesos | Extracciones de pesos realizadas con la tarjeta de crédito durante ese mes. |
| 148 | Visa\_madelantodolares | pesos | Extracciones de dolares realizadas con la tarjeta de crédito durante ese mes. |
| 149 | Visa\_fultimo\_cierre | dias | Dias del utlimo cierre de la tarjeta de crédito, contados a la fecha de la foto. |
| 150 | Visa\_mpagado | pesos | Monto total de todos los pagos efectuados por el cliente |

## Anexo 2. Script de R Studio: primera aproximación a la data.

library("data.table")

install.packages("ROCR")

#Contiene funciones para evaluar performance de la clasificación

library("ROCR")

dataset <- fread("C:/Users/fiore/Documents/fiore facultad/TESINA/datasetsOri/paquete\_premium\_202011.csv")

#El universo agrupado por clase\_ternaria

ftable(dataset$clase\_ternaria)

#Resumen segregado por clase ternaria

Archivosegregado <-by(dataset, dataset$clase\_ternaria, summary)

Archivosegregado

fwrite( Archivosegregado, file="C:/Users/fiore/Documents/fiore facultad/TESINA/Archivosegregado.csv")

#Permite generar graficos y visualizaciones

summary(dataset)

summary(dataset[,1:10]) #Primeras 10 columnas

#Corte por mcuentas\_saldo

hist(dataset[ ,mcuentas\_saldo] )

boxplot(mcuentas\_saldo ~ clase\_ternaria, data=dataset)

boxplot(mcuentas\_saldo ~ clase\_ternaria, data=dataset, outline=FALSE)

hist(dataset$cliente\_antiguedad)

#calculo la ganancia de incentivar cada caso

dataset[ , ganancia:= kFPGain ] #-1250

dataset[ clase\_ternaria=='BAJA+2' , ganancia:= kTPGain] #48750

sum( dataset$ganancia )

## Anexo 3. Script de R Studio: análisis exploratorio.

library("data.table")

library(ggplot2)

carpeta\_datasets <- "~/datasets/"

#Archivo con datos etiquetados para entrenamiento

dataset <- paste0(carpeta\_datasets, "paquete\_premium\_202011.csv")

dataset <- fread(dataset)

#Corte por mcuentas\_saldo

hist(dataset[ ,mcuentas\_saldo] )

boxplot(mcuentas\_saldo ~ clase\_ternaria, data=dataset)

boxplot(mcuentas\_saldo ~ clase\_ternaria, data=dataset, outline=FALSE)

boxplot(Visa\_mconsumototal ~ clase\_ternaria, data=dataset, outline=FALSE)

hist(dataset$ccallcenter\_transacciones)

for (feature in names(correlaciones[order(-correlaciones[,1]),][c(3,4,6,11)])) {

plot <- ggplot(dataset, aes(x=as.factor(clase), y=get(feature), fill=as.factor(clase)))

geom\_boxplot(fill="slateblue", alpha=0.2)+

geom\_violin(alpha=0.2)+

ggtitle(feature)+

scale\_y\_log10()

print(plot)

}

dataset$TC\_cactividad = dataset$Visa\_cconsumos + dataset$Master\_cconsumos

ggplot(dataset, aes(x=as.factor(clase), y=TC\_cactividad, fill=as.factor(clase)))

geom\_boxplot(fill="slateblue", alpha=0.2)+

geom\_violin(alpha = 0.2)+

ylim(0, 15)

## Anexo 4. Script de R Studio: Regresión logística y regresión logística penalizada.

rm(list=ls()) #remove all objects

gc()

#install.packages("dplyr")

library("dplyr")

#install.packages("car")

library(car)

require("data.table")

require("rpart")

library(glmnet)

library(tidyverse)

library(caret)

train<-fread("C:/Users/fiore/Documents/fiore facultad/TESINA/datasetsOri/paquete\_premium\_202011.csv")

test<-fread("C:/Users/fiore/Documents/fiore facultad/TESINA/datasetsOri/paquete\_premium\_202101.csv")

train$clase\_ternaria<- recode(train$clase\_ternaria,"'BAJA+1'='1' ; 'BAJA+2'='1'; 'CONTINUA'='0'")

ftable(train$clase\_ternaria)

modelo\_logistico <- glm(formula=clase\_ternaria ~ ., data = train, family = "binomial"(link="logit"))

summary(modelo\_logistico)

Find the best lambda using cross-validation

cv.lasso <- cv.glmnet(formula=clase\_ternaria ~ ., alpha = 1, family = "binomial")

# Fit the final model on the training data

modelo\_logistico\_lasso <- glmnet(formula=clase\_ternaria ~ ., alpha = 1, family = "binomial",

lambda = cv.lasso$lambda.min)

residuals.glm(modelo\_logistico, type="deviance")

par(mfrow=c(1,4))

plot(residuals.glm(modelo\_logistico, type="deviance"), main="Residuals")

hist(residuals.glm(modelo\_logistico, type="deviance"), main="Residuals")

boxplot(residuals.glm(modelo\_logistico, type="deviance"), main="Residuals")

qqnorm(residuals.glm(modelo\_logistico, type="deviance"))

qqline(residuals.glm(modelo\_logistico, type="deviance"))

summary(residuals.glm(modelo\_logistico, type="deviance"))

prediccion <- predict( modelo\_logistico2, test )

#predicciones <- ifelse(test = modelo\_logistico2$fitted.values > 0.025, yes = 1, no = 0)

entrega <- test [ , list( numero\_de\_cliente, prediccion) ]

fwrite( entrega, file="C:/Users/fiore/Documents/fiore facultad/TESINA/Logistica.csv", sep="," )

## Anexo 5. Script de R Studio: modelo de Árboles de decisión simples.

#limpio la memoria

rm( list=ls() ) #remove all objects

gc() #garbage collection

require("data.table")

require("rlist")

require("yaml")

require("rpart")

require("parallel")

#paquetes necesarios para la Bayesian Optimization

require("DiceKriging")

require("mlrMBO")

#defino la carpeta donde trabajo

setwd( "~/buckets/b1/" )

kscript <- "rpart"

karch\_generacion <- "./datasetsOri/paquete\_premium\_202011.csv"

karch\_aplicacion <- "./datasetsOri/paquete\_premium\_202101.csv"

kBO\_iter <- 150 #cantidad de iteraciones de la Optimizacion Bayesiana

hs <- makeParamSet(

makeNumericParam("cp" , lower= -1 , upper= 0.1),

makeIntegerParam("minsplit" , lower= 1L , upper= 8000L),

makeIntegerParam("minbucket", lower= 1L , upper= 2000L),

makeIntegerParam("maxdepth" , lower= 3L , upper= 20L),

forbidden = quote( minbucket > 0.5\*minsplit ) )

ksemilla\_azar <- 102191 #Aqui poner la propia semilla

#------------------------------------------------------------------------------

#funcion para crossvalidation

particionar <- function( data, division, agrupa="", campo="fold", start=1, seed=NA )

{

if( !is.na( seed) ) set.seed( seed )

bloque <- unlist( mapply( function(x,y) { rep( y, x ) }, division, seq( from=start, length.out=length(division) ) ) )

data[ , (campo) := sample( rep( bloque, ceiling(.N/length(bloque))) )[1:.N],

by= agrupa ]

}

#------------------------------------------------------------------------------

ArbolSimple <- function( fold\_test, data, param )

{

#genero el modelo

modelo <- rpart("clase\_ternaria ~ .",

data= data[ fold != fold\_test, ],

xval= 0,

control= param )

#aplico el modelo a los datos de testing, fold==2

prediccion <- predict( modelo, data[ fold==fold\_test, ], type = "prob")

prob\_baja2 <- prediccion[, "BAJA+2"]

ganancia\_testing <- sum( data[ fold==fold\_test ][ prob\_baja2 >0.025, ifelse( clase\_ternaria=="BAJA+2", 48750, -1250 ) ] ) #unicamente BAJA+2 porque es testing

return( ganancia\_testing )

}

#------------------------------------------------------------------------------

ArbolesCrossValidation <- function( data, param, qfolds, pagrupa, semilla )

{

divi <- rep( 1, qfolds )

particionar( data, divi, seed=semilla, agrupa=pagrupa )

ganancias <- mcmapply( ArbolSimple,

seq(qfolds), # 1 2 3 4 5

MoreArgs= list( data, param),

SIMPLIFY= FALSE,

mc.cores= 1 )

data[ , fold := NULL ]

#devuelvo la primer ganancia y el promedio

return( mean( unlist( ganancias )) \* qfolds ) #aqui normalizo la ganancia

}

#------------------------------------------------------------------------------

#esta funcion solo puede recibir los parametros que se estan optimizando

#el resto de los parametros se pasan como variables globales

EstimarGanancia <- function( x )

{

GLOBAL\_iteracion <<- GLOBAL\_iteracion + 1

xval\_folds <- 5

ganancia <- ArbolesCrossValidation( dataset, param=x, qfolds= xval\_folds, pagrupa="clase\_ternaria", semilla=ksemilla\_azar )

#si tengo una ganancia superadora, genero el un archivo nuevo

if( ganancia > GLOBAL\_ganancia\_max )

{

GLOBAL\_ganancia\_max <<- ganancia #asigno la nueva maxima ganancia

modelo <- rpart("clase\_ternaria ~ .",

data= dataset,

xval= 0,

control= x )

#calculo la importancia de variables

tb\_importancia <- as.data.table( list( "Feature"= names(modelo$variable.importance),

"Importance"= modelo$variable.importance ) )

fwrite( tb\_importancia,

file= paste0(kimp, "imp\_", GLOBAL\_iteracion, ".txt"),

sep="\t" )

#genero el vector con la prediccion, la probabilidad de ser positivo

prediccion <- predict( modelo, dapply)

prob\_baja2 <- prediccion[, "BAJA+2"]

Predicted <- ifelse( prob\_baja2 > 0.025, 1, 0 )

entrega <- as.data.table( list( "numero\_de\_cliente"=dapply$numero\_de\_cliente, "Predicted"=Predicted) )

#genero el archivo con las predicciones

fwrite( entrega,

file= paste0(kfinal, GLOBAL\_iteracion, ".csv" ),

sep= "," )

}

#logueo

xx <- x

xx$xval\_folds <- xval\_folds

xx$ganancia <- ganancia

loguear( xx, arch= klog )

return( ganancia )

}

#------------------------------------------------------------------------------

#Aqui empieza el programa

#en estos archivos quedan los resultados

kbayesiana <- paste0("./work/E", "\_", kscript, ".RDATA" )

klog <- paste0("./work/E", "\_", kscript, ".txt" )

kimp <- paste0("./work/E", "\_", kscript, "\_" )

kfinal <- paste0("./final/E", "\_", kscript, "\_" )

GLOBAL\_ganancia\_max <- -Inf

GLOBAL\_iteracion <- 0

#cargo los datasets

dataset <- fread(karch\_generacion) #donde entreno

#creo la clase\_binaria2 1={ BAJA+2,BAJA+1} 0={CONTINUA}

dataset[ , clase01:= ifelse( clase\_ternaria=="CONTINUA", 0, 1 ) ]

dapply <- fread(karch\_aplicacion) #donde aplico el modelo

#Aqui comienza la configuracion de la Bayesian Optimization

configureMlr( show.learner.output = FALSE)

funcion\_optimizar <- EstimarGanancia

obj.fun <- makeSingleObjectiveFunction(

fn= funcion\_optimizar,

minimize= FALSE, #estoy Maximizando la ganancia

noisy= TRUE,

par.set= hs,

has.simple.signature = FALSE

)

ctrl <- makeMBOControl()

ctrl <- setMBOControlTermination(ctrl, iters= kBO\_iter )

ctrl <- setMBOControlInfill(ctrl, crit= makeMBOInfillCritEI())

surr.km <- makeLearner("regr.km", predict.type= "se", covtype= "matern3\_2", control= list(trace= TRUE))

#inicio la optimizacion bayesiana

if(!file.exists(kbayesiana)) {

run <- mbo(obj.fun, learner = surr.km, control = ctrl)

} else run <- mboContinue( kbayesiana ) #retomo en caso que ya exista

## Anexo 6. Script de R Studio: modelo de Random Forest.

#limpio la memoria

rm( list=ls() ) #remove all objects

gc() #garbage collection

require("data.table")

require("rlist")

require("yaml")

require("ranger")

require("randomForest")

require("parallel")

#paquetes necesarios para la Bayesian Optimization

require("DiceKriging")

require("mlrMBO")

#defino la carpeta donde trabajo

setwd( "~/buckets/b1/" )

kscript <- "ranger"

karch\_generacion <- "./datasetsOri/paquete\_premium\_202011.csv"

karch\_aplicacion <- "./datasetsOri/paquete\_premium\_202101.csv"

kBO\_iter <- 150 #cantidad de iteraciones de la Optimizacion Bayesiana

hs <- makeParamSet(

makeIntegerParam("num.trees" , lower= 2L , upper= 500L), #la letra L al final significa ENTERO

makeIntegerParam("max.depth", lower= 0L , upper= 20L), # 0 significa profundidad infinita

makeIntegerParam("min.node.size" , lower= 1L , upper= 500L),

makeIntegerParam("mtry" , lower= 2L , upper= 50L))

ksemilla\_azar <- 102191 #Aqui poner la propia semilla

#------------------------------------------------------------------------------

#funcion para particionar, es la que Andres reemplaza con caret

particionar <- function( data, division, agrupa="", campo="fold", start=1, seed=NA )

{

if( !is.na( seed) ) set.seed( seed )

bloque <- unlist( mapply( function(x,y) { rep( y, x ) }, division, seq( from=start, length.out=length(division) ) ) )

data[ , (campo) := sample( rep( bloque, ceiling(.N/length(bloque))) )[1:.N],

by= agrupa ]

}

#------------------------------------------------------------------------------

ranger\_Simple <- function( fold\_test, pdata, param )

{

#genero el modelo

set.seed(ksemilla\_azar)

modelo <- ranger( formula= "clase\_binaria ~ .",

data= pdata[ fold!= fold\_test],

probability= TRUE, #para que devuelva las probabilidades

num.trees= param$num.trees,

mtry= param$mtry,

min.node.size= param$min.node.size,

max.depth= param$max.depth

)

prediccion <- predict( modelo, pdata[ fold==fold\_test] )

ganancia\_testing <- pdata[ fold==fold\_test,

sum( (prediccion$predictions[ ,"POS" ] > 0.025) \*

ifelse( clase\_binaria=="POS", 48750, -1250) ) ]

return( ganancia\_testing )

}

#------------------------------------------------------------------------------

ranger\_CrossValidation <- function( data, param, pcampos\_buenos, qfolds, pagrupa, semilla )

{

divi <- rep( 1, qfolds )

particionar( data, divi, seed=semilla, agrupa=pagrupa )

ganancias <- mcmapply( ranger\_Simple,

seq(qfolds), # 1 2 3 4 5

MoreArgs= list( data, param),

SIMPLIFY= FALSE,

mc.cores= 1 )

data[ , fold := NULL ]

#devuelvo la primer ganancia y el promedio

return( mean( unlist( ganancias )) \* qfolds ) #aqui normalizo la ganancia

}

#------------------------------------------------------------------------------

#esta funcion solo puede recibir los parametros que se estan optimizando

#el resto de los parametros se pasan como variables globales

EstimarGanancia\_ranger <- function( x )

{

gc()

GLOBAL\_iteracion <<- GLOBAL\_iteracion + 1

xval\_folds <- 5

ganancia <- ranger\_CrossValidation( dataset,

param= x,

qfolds= xval\_folds,

pagrupa= "clase\_binaria",

semilla= ksemilla\_azar )

if( ganancia > GLOBAL\_ganancia\_max )

{

GLOBAL\_ganancia\_max <<- ganancia #asigno la nueva maxima ganancia

set.seed(ksemilla\_azar)

modelo <- ranger( formula= "clase\_binaria ~ .",

data= dataset,

probability= TRUE, #para que devuelva las probabilidades

num.trees= x$num.trees,

mtry= x$mtry,

min.node.size= x$min.node.size,

max.depth= x$max.depth

)

prediccion <- predict( modelo, dapply )

Predicted <- as.integer( prediccion$predictions[ ,"POS" ] > 0.025 )

entrega <- as.data.table( list( "numero\_de\_cliente"=dapply$numero\_de\_cliente,

"Predicted"= Predicted) )

#genero el archivo final

fwrite( entrega,

file= paste0(GLOBAL\_iteracion, ".csv" ),

sep= "," )

}

#logueo

xx <- x

xx$xval\_folds <- xval\_folds

xx$ganancia <- ganancia

loguear( xx, arch= klog )

return( ganancia )

}

#------------------------------------------------------------------------------

#Aqui empieza el programa

#en estos archivos quedan los resultados

kbayesiana <- paste0("./work/E", "\_", kscript, ".RDATA" )

klog <- paste0("./work/E", "\_", kscript, ".txt" )

kimp <- paste0("./work/E", "\_", kscript, "\_" )

kfinal <- paste0("./final/E", "\_", kscript, "\_" )

GLOBAL\_ganancia\_max <- -Inf

GLOBAL\_iteracion <- 0

#cargo los datasets

dataset <- fread(karch\_generacion) #donde entreno

#creo la clase\_binaria2 1={ BAJA+2,BAJA+1} 0={CONTINUA}

dataset[ , clase01:= ifelse( clase\_ternaria=="CONTINUA", 0, 1 ) ]

dapply <- fread(karch\_aplicacion) #donde aplico el modelo

#Aqui comienza la configuracion de la Bayesian Optimization

configureMlr( show.learner.output = FALSE)

funcion\_optimizar <- EstimarGanancia\_ranger

obj.fun <- makeSingleObjectiveFunction(

fn= funcion\_optimizar,

minimize= FALSE, #estoy Maximizando la ganancia

noisy= TRUE,

par.set= hs,

has.simple.signature = FALSE

)

ctrl <- makeMBOControl()

ctrl <- setMBOControlTermination(ctrl, iters= kBO\_iter )

ctrl <- setMBOControlInfill(ctrl, crit= makeMBOInfillCritEI())

surr.km <- makeLearner("regr.km", predict.type= "se", covtype= "matern3\_2", control= list(trace= TRUE))

#inicio la optimizacion bayesiana

if(!file.exists(kbayesiana)) {

run <- mbo(obj.fun, learner = surr.km, control = ctrl)

} else run <- mboContinue( kbayesiana ) #retomo en caso que ya exista

## Anexo 7. Script de R Studio: modelo de LightGbm.

#limpio la memoria

rm( list=ls() ) #remove all objects

require("data.table")

require("rlist")

require("yaml")

require("lightgbm")

#paquetes necesarios para la Bayesian Optimization

require("DiceKriging")

require("mlrMBO")

#defino la carpeta donde trabajo

setwd( directory.root )

kscript <- "lgb\_binaria2"

karch\_generacion <- "./datasetsOri/paquete\_premium\_202011.csv"

karch\_aplicacion <- "./datasetsOri/paquete\_premium\_202101.csv"

kBO\_iter <- 150 #cantidad de iteraciones de la Optimizacion Bayesiana

#Aqui se cargan los hiperparametros

hs <- makeParamSet(

makeNumericParam("learning\_rate", lower= 0.01 , upper= 0.1),

makeNumericParam("feature\_fraction", lower= 0.2 , upper= 1.0),

makeIntegerParam("min\_data\_in\_leaf", lower= 0 , upper= 8000),

makeIntegerParam("num\_leaves", lower=16L , upper= 1024L),

makeNumericParam("prob\_corte", lower= 0.020, upper= 0.055)

)

ksemilla\_azar <- 102191 #Aqui poner la propia semilla

#------------------------------------------------------------------------------

PROB\_CORTE <- 0.025

fganancia\_logistic\_lightgbm <- function(probs, datos)

{

vlabels <- getinfo(datos, "label")

vpesos <- getinfo(datos, "weight")

#aqui esta el inmoral uso de los pesos para calcular la ganancia correcta

gan <- sum( (probs > PROB\_CORTE ) \*

ifelse( vlabels== 1 & vpesos > 1, 48750, -1250 ) )

return( list( "name"= "ganancia",

"value"= gan,

"higher\_better"= TRUE ) )

}

#------------------------------------------------------------------------------

#esta funcion solo puede recibir los parametros que se estan optimizando

EstimarGanancia\_lightgbm <- function( x )

{

GLOBAL\_iteracion <<- GLOBAL\_iteracion + 1

gc()

PROB\_CORTE <<- x$prob\_corte #asigno la variable global

kfolds <- 5 # cantidad de folds para cross validation

param\_basicos <- list( objective= "binary",

metric= "custom",

first\_metric\_only= TRUE,

boost\_from\_average= TRUE,

feature\_pre\_filter= FALSE,

verbosity= -100,

seed= 999983,

max\_depth= -1, # -1 significa no limitar,

min\_gain\_to\_split= 0.0,

lambda\_l1= 0.0,

)

param\_variable <- list( early\_stopping\_rounds= as.integer(50 + 5/x$learning\_rate) )

param\_completo <- c( param\_basicos, param\_variable, x )

set.seed( 999983 )

modelocv <- lgb.cv( data= dtrain,

eval= fganancia\_logistic\_lightgbm,

stratified= TRUE, #sobre el cross validation

nfold= kfolds, #folds del cross validation

param= param\_completo,

verbose= -100

)

ganancia <- unlist(modelocv$record\_evals$valid$ganancia$eval)[ modelocv$best\_iter ]

ganancia\_normalizada <- ganancia\* kfolds

attr(ganancia\_normalizada ,"extras" ) <- list("num\_iterations"= modelocv$best\_iter)

param\_completo$num\_iterations <- modelocv$best\_iter #asigno el mejor num\_iterations

#si tengo una ganancia superadora, genero el archivo final

if( ganancia > GLOBAL\_ganancia\_max )

{

GLOBAL\_ganancia\_max <<- ganancia #asigno la nueva maxima ganancia a una variable GLOBAL

set.seed(ksemilla\_azar)

modelo <- lightgbm( data= dtrain,

param= param\_completo,

verbose= -100

)

#logueo

xx <- param\_completo

xx$ganancia <- ganancia\_normalizada #le agrego la ganancia

loguear( xx, arch= klog )

return( ganancia )

}

}

#------------------------------------------------------------------------------

#Aqui empieza el programa

#en estos archivos quedan los resultados

kbayesiana <- paste0("./work/E", "\_", kscript, ".RDATA" )

klog <- paste0("./work/E", "\_", kscript, ".txt" )

kimp <- paste0("./work/E", "\_", kscript, "\_" )

kfinal <- paste0("./final/E", "\_", kscript, "\_" )

GLOBAL\_ganancia\_max <- -Inf

GLOBAL\_iteracion <- 0

#cargo los datasets

dataset <- fread(karch\_generacion) #donde entreno

#creo la clase\_binaria2 1={ BAJA+2,BAJA+1} 0={CONTINUA}

dataset[ , clase01:= ifelse( clase\_ternaria=="CONTINUA", 0, 1 ) ]

dapply <- fread(karch\_aplicacion) #donde aplico el modelo

#Aqui comienza la configuracion de la Bayesian Optimization

funcion\_optimizar <- EstimarGanancia\_lightgbm #la funcion que voy a maximizar

configureMlr( show.learner.output= FALSE)

obj.fun <- makeSingleObjectiveFunction(

fn= funcion\_optimizar, #la funcion que voy a maximizar

minimize= FALSE, #estoy Maximizando la ganancia

noisy= TRUE,

par.set= hs, #definido al comienzo del programa

has.simple.signature = FALSE #paso los parametros en una lista

)

ctrl <- makeMBOControl( save.on.disk.at.time= 600, save.file.path= kbayesiana) #se graba cada 600 segundos

ctrl <- setMBOControlTermination(ctrl, iters= kBO\_iter ) #cantidad de iteraciones

ctrl <- setMBOControlInfill(ctrl, crit= makeMBOInfillCritEI() )

#establezco la funcion que busca el maximo

surr.km <- makeLearner("regr.km", predict.type= "se", covtype= "matern3\_2", control= list(trace= TRUE))

#inicio la optimizacion bayesiana

if(!file.exists(kbayesiana)) {

run <- mbo(obj.fun, learner= surr.km, control= ctrl)

} else {

run <- mboContinue( kbayesiana ) #retomo en caso que ya exista

}