NSTITUTO TECNOLÓGICO DE BUENOS AIRES – ITBA
ESCUELA DE (INGENIERÍA Y TECNOLOGÍA – INGENIERÍA Y GESTIÓN - POSTGRADO)

Desarrollo de Modelo de Customer Lifetime Value (CLTV) para industria bancaria con técnicas de Machine Learning.

AUTOR/ES: Martins, Nicolás Horacio (Leg. Nº 104245)

DOCENTE/S TITULAR/ES O TUTOR/ES: Pampliega, Juan Martín

TRABAJO FINAL PRESENTADO PARA LA OBTENCIÓN DEL TÍTULO DE ESPECIALISTA EN CIENCIA DE DATOS

BUENOS AIRES

PRIMER CUATRIMESTRE, 2021

1. Introducción

Una gran cantidad de organizaciones de tamaño medio en adelante poseen sistemas de información que les permiten conocer la rentabilidad por cliente, o un valor aproximado de la misma. Algunas de ellas han avanzado, según la rama de la industria, en el desarrollo de modelos predictivos y de segmentación más profundos para entender la principalidad o participación de sus productos en la cartera de sus clientes (*share of wallet*)[1], el nivel de lealtad y el grado de satisfacción de sus clientes. Otras, inclusive, han intentado calcular o predecir los ingresos futuros por cliente con mayor o menor nivel de sistematización. El foco de este trabajo se centra en el estudio y desarrollo de modelos para abordar esta última etapa analítica de la firma, conocida como valor del ciclo de vida de un cliente*, customer lifetime value* o CLTV (Borle et al 2008) en la industria financiera.

2. Estado del arte

2.1 Marco teórico y conceptual: cómo definir adecuadamente el *CLTV*.

El valor del ciclo de vida o de vida útil del cliente (*customer lifetime value o CLTV*) es una métrica utilizada en muchas industrias que se basa en el flujo de efectivo descontado que un cliente acumula durante su vida para la firma. Las empresas han utilizado el CLTV como base para formular e implementar estrategias específicas para los clientes, dado que estos son activos clave de una firma, cambiando el foco de ganancias trimestrales hacia relaciones enfocadas en la salud a largo plazo de la relación con un cliente.

Dado que el CLTV se basa en una predicción futura, por oposición a la rentabilidad histórica de un cliente, su estimación es más compleja. Sin embargo, su utilidad radica en brindar mayor información a las empresas para segmentar mejor a sus clientes y enfocarse en estrategias de fidelización sobre los más rentables y de optimización en los menos rentables. En el límite, esto puede implicar "deshacerse" *(price out)* de los clientes no rentables. Es un herramental poderoso para alocar eficientemente los recursos según los grupos de clientes y poder discriminar el *marketing mix* a aplicar a cada uno de ellos.

El marco de trabajo adoptado parte de entender que las acciones de una firma influyen sobre el comportamiento del cliente a lo largo de su ciclo de vida (adquisición, venta cruzada, retención), lo que a su vez afecta el CLTV y consecuentemente el valor de la firma. De esta forma, el CLTV de los clientes actuales y futuros, que se denomina patrimonio neto de los clientes, *customer equity* o *CE* (Blattberg & Deighton 1996) constituye un *proxy* del valor de la firma o de su precio de mercado. Un esquema conceptual de este proceso se puede expresar de la sigu iente forma:

[1] El *share of wallet* es un indicador que ayuda a conocer cuánto gasta el cliente o, en forma agregada, el segmento objetivo, en la categoría de productos en la que compite la firma, y así poder determinar la participación de la empresa en la misma.

Fig. 1. Esquema conceptual: estrategia de marketing, valor del cliente y valor de la empresa (Gupta *et al* 2006)

Como se observa en la Figura 1, los programas de marketing se desarrollan con distintas estrategias: desde la captación o adquisición de nuevos clientes, la venta de productos adicionales o complementarios a los clientes actuales, hasta técnicas de retención de clientes en forma reactiva (un ejemplo clásico es ofrecer una bonificación para evitar la terminación de un servicio contratado por el cliente) o bien proactiva (por ejemplo, comunicando promociones para fomentar el uso de productos o servicios de clientes que reducen el nivel de compras en el tiempo). Si estos programas son efectivos, pueden generar mayor rentabilidad por cliente para la firma, como así también extender la relación de los consumidores con la empresa. Esto impacta positivamente en el CLTV y, por agregación, en el CE. Por lo tanto, es de vital importancia contar con modelos que permitan estimar el valor del ciclo de vida del cliente para calibrar y optimizar los programas de marketing.

Normalmente, el CLTV de un cliente se calcula a partir de tres componentes: frecuencia de compra, margen de contribución y costos de comercialización, ajustados por la tasa de retención de los clientes en el tiempo. Esto puede expresarse bajo la siguiente forma matemática:

$$CLTV = \sum_{t=0}^{T} \frac{(p_t - c_t)\, r_t}{(1+i)^i} - CA$$

Donde:

p_t = precio pagado por un consumidor en el momento t,
c_t = costo directo de servicio al cliente en el momento t,
i = tasa de descuento o costo de capital para la empresa,
r_t = probabilidad de que el cliente repita la compra o esté "vivo" en el momento t,
CA = costo de adquisición, y
T = horizonte de tiempo para estimar CLTV.

Si se asumen márgenes constantes de los flujos de rentabilidad y de las tasas de retención de un cliente, el cálculo se puede aproximar a la siguiente forma más sencilla:

$$CLTV = \frac{Margen * Tasa\ de\ Retención}{1 + Tasa\ de\ Descuento - Tasa\ de\ Retención}$$

Para calcular correctamente el CLTV se deben tener en cuenta las características de la base de clientes que se está analizando en términos de la tipología de las transacciones de productos y de la relación entre la firma y el cliente. Tal como se presenta en la Figura 2, siguiendo a Fader *et al* (2006), las primeras se pueden clasificar en continuas y discretas, mientras que la segunda se clasifica en contractual y no contractual. Por transacciones "continuas", nos referimos a que las transacciones pueden ocurrir en cualquier momento. Esto contrasta con el tiempo discreto, donde las transacciones pueden ocurrir sólo en un punto específico en el tiempo. Por ejemplo, las relaciones comerciales que van desde la asistencia a eventos especiales hasta el reabastecimiento de recetas y las renovaciones de contratos a menudo se restringen durante períodos de tiempo específicos. En estas situaciones, la pregunta gerencial clave no es cuándo se realizarán las transacciones, sino si una transacción se llevará a cabo durante el período especificado.

La forma "no contractual" se refiere a un aspecto diferente en la relación entre la empresa y sus clientes, y obviamente contrasta con una configuración contractual. En este último caso, se observa el momento en que el cliente se vuelve inactivo (por ejemplo, cuando el cliente no renueva su suscripción o se comunica con la empresa para cancelar su contrato). En el primer caso, el punto en que el cliente se vuelve inactivo no es observado por la empresa. Esto significa que la empresa no puede diferenciar entre un cliente que ha finalizado su relación con la empresa y uno que se encuentra simplemente en medio de una larga pausa entre las transacciones. En entornos no contractuales, no podemos utilizar de manera significativa nociones como "tasas de retención" o aplicar herramientas como el análisis de supervivencia.

Fig. 2. Clasificación de la relación comercial empresa-cliente (Fader *et al* 2006)

2.2 Antecedentes, evolución y enfoques para desarrollar y modelar el *CLTV.*

El estado del arte en este dominio utiliza un gran número de características hechas a mano y regresores de conjunto para pronosticar el valor, predecir la rotación y evaluar la lealtad del cliente. Recientemente, los dominios de *machine learning* que incluyen lenguaje, visión y habla han mostrado avances dramáticos al reemplazar las características artesanales de los conjuntos de datos utilizados para la predicción del CLTV con características que se aprenden automáticamente de los datos (Chamberlain *et al* 2017).

Siguiendo a Harman (2016), la evolución de gran parte de la literatura de marketing para modelos CLTV está dedicada a los modelos de proceso estocástico, que comienzan con los modelos probabilísticos del tipo Pareto / NBD (Distribución Binomial Negativa). Lo que hace que estos modelos estocásticos sean valiosos es que son capaces de proporcionar estimaciones prospectivas de los patrones de compra y la rotación de clientes cuando el evento de un cliente que abandona la empresa no es observable, lo cual es típico de la mayoría de los bienes y servicios no contractuales de consumo. En estas configuraciones, los modelos de procesos estocásticos todavía nos permiten modelar una probabilidad de si un cliente permanece activo.

Los investigadores de mercadeo utilizaron la NBD como una herramienta para estimar efectivamente las tasas de transacción, la penetración de mercado y más. Si bien el NBD es un modelo estacionario, los recuentos de la incidencia de la transacción estimada a partir del NBD podrían usarse como una línea de base para comparar contra las desviaciones en los datos. Dichas desviaciones podrían atribuirse a cambios debidos a acciones de marketing o estacionalidad, por ejemplo, los usuarios esporádicos de un alimento de temporada aumentarían su uso cuando estaba en temporada alta, mientras que los usuarios recurrentes no cambiarían su comportamiento de compra. De este modo, la NBD pudo brindar orientación a los gerentes e investigadores sobre los impactos de las acciones de la empresa y ayudarles a reconocer el comportamiento de diferentes segmentos de clientes.

A principios de la década de 1990, la literatura de marketing comenzó a utilizar la función de riesgo de un proceso estocástico para modelar los datos de transacción. Los modelos de riesgo econométricos están diseñados para modelar el tiempo para la ocurrencia de un evento. Se usan comúnmente en la literatura de bioestadística, y dada su capacidad de parametrizar los modelos sobre las posibles causas de la ocurrencia de un evento los convierte en una herramienta útil para los problemas de mercadotecnia relacionados con el tiempo hasta la siguiente transacción del cliente, por ejemplo.

A fines de la década de 1990, a medida que mejoraba el poder de cómputo, comenzamos a ver procesos estocásticos y modelos de riesgo aplicados a eventos de interés para los comercializadores, además de transacciones y elección de marca. Además, se proponen modelos más complejos durante este período de tiempo al combinar más de un proceso estocástico para explicar datos más complejos que analizan los eventos de interés del consumidor, además del tiempo de transacción y la elección de la marca. La BG / NBD de Fader, Hardie y Lee (2005) re-introdujo esta línea de investigación en los modelos CLTV.

Si bien no es un modelo CLTV, el modelo de Recencia, Frecuencia y Monetario RFM (antigüedad de la última compra, frecuencia de transacciones y valor monetario acumulado) propuesto por Colombo y Jiang (1999) ha recibido atención de la literatura de modelos de CLTV, principalmente debido a su modelo de gasto del cliente. Colombo y Jiang modelan la frecuencia de transacciones usando el NBD. Presentan el modelo de proceso estocástico compuesto Gamma / Gamma para estimar el gasto de los clientes. Asumen que la frecuencia y el gasto eran independientes y se estiman por separado. El modelo Gamma / Gamma supone que cada consumidor puede gastar una cantidad que sigue una distribución Gamma. Su modelo permite una gran flexibilidad independientemente de los datos

subyacentes, ya que la distribución Gamma puede modelar muchas formas diferentes. El modelo Gamma / Gamma se utilizó para modelar el gasto del consumidor en aplicaciones CLTV de Fader, Hardie y Lee (2005), así como Glady, Baesens y Croux (2009).

Al igual que los modelos econométricos de CLTV, los modelos de persistencia se centran en modelar el comportamiento de sus componentes, es decir, adquisición, retención y venta cruzada. Cuando se dispone de series suficientemente largas, es posible tratar estos componentes como parte de un sistema dinámico. Los avances en el análisis multivariado de series de tiempo, en particular los modelos vectoriales autorregresivos (VAR), las raíces unitarias y la cointegración, se pueden usar para estudiar cómo un movimiento en una variable (por ejemplo, una campaña de adquisición o una mejora del servicio al cliente) impacta otras variables del sistema en el tiempo. Hasta la fecha, este enfoque, conocido como modelado de persistencia, se ha utilizado en un contexto CLTV para estudiar el impacto de la publicidad, el descuento y la calidad del producto en la equidad del cliente y para examinar las diferencias en el CLTV resultante de diferentes métodos de adquisición de clientes. La principal contribución del modelado de persistencia es que proyecta el comportamiento a largo plazo o de equilibrio de una variable o un grupo de variables de interés.

La literatura de mercadotecnia ha favorecido típicamente modelos paramétricos estructurados, como logit, probit o modelos de riesgo. Estos modelos se basan en la teoría económica y son fáciles de interpretar. En contraste, la vasta literatura de informática en minería de datos, aprendizaje automático y estadísticas no paramétricas ha generado muchos enfoques que enfatizan la capacidad predictiva. Estos incluyen modelos de proyección-búsqueda; modelos de redes neuronales; modelos de árbol de decisión; modelos basados en splines, tales como modelos aditivos generalizados (GAM), splines de regresión adaptativa multivariable (MARS), árboles de clasificación y regresión (CART); y soporte de máquinas vectoriales (SVM). Muchos de estos enfoques pueden ser más adecuados para el estudio de la rotación de clientes, donde generalmente tenemos un gran número de variables, lo que comúnmente se conoce como la "maldición de la dimensionalidad". La escasez de datos en estas situaciones infla la varianza de las estimaciones, haciendo que los modelos paramétricos y no paramétricos tradicionales sean menos útiles. Para superar estas dificultades, Hastie y Tibshirani (1990) propusieron modelos aditivos generalizados en los que la media de la variable dependiente depende de un predictor aditivo a través de una función de enlace no lineal, no paramétrica. Otro enfoque para superar la maldición de la dimensionalidad es MARS. Este es un procedimiento de regresión no paramétrica que funciona como una regresión lineal por partes con puntos de interrupción estimados a partir de los datos (Friedman 1991).

Más recientemente, se ha utilizado SVM para propósitos de clasificación. En lugar de suponer que una función o plano lineal puede separar las dos (o más) clases, este enfoque puede manejar situaciones en las que se necesita una función curvilínea o un hiperplano para una mejor clasificación. Efectivamente, el método transforma los datos sin procesar en un "espacio destacado" utilizando un núcleo matemático de tal manera que este espacio puede clasificar objetos usando planos lineales. En un estudio reciente, Cui y Curry (2005) realizaron simulaciones extensas de Monte Carlo para comparar predicciones basadas en el modelo logit multinomial y SVM. En todos los casos, SVM superó el modelo logit.

Las predicciones también se pueden mejorar combinando modelos. La literatura de aprendizaje automático sobre *bagging*, la literatura econométrica sobre la combinación de pronósticos y la literatura estadística sobre promedios de modelos sugieren que ponderar las predicciones de muchos modelos diferentes puede producir mejoras en la capacidad predictiva. Recientemente, Lemmens y Croux utilizaron técnicas de *bagging* y *boosting* para predecir la rotación de una base de datos de clientes inalámbricos de EE.UU. El *bagging* (Bootstrap AGGregatING) consiste en estimar secuencialmente un modelo de elección binaria, llamado clasificador de base en aprendizaje

automático, a partir de versiones remuestreadas de una muestra de calibración. Los clasificadores obtenidos forman un grupo del cual se deriva un modelo de elección final por agregación o ensamble. El refuerzo (*boosting*) esencialmente consiste en estimar secuencialmente un clasificador para versiones ponderadas adaptativamente de la muestra de calibración inicial. El esquema de ponderación proporciona a los clientes clasificados erróneamente un mayor peso en la siguiente iteración. Esto obliga al método de clasificación a concentrarse en clientes difíciles de clasificar. Lemmens y Croux compararon los resultados de estos métodos con el modelo logit binario y encontraron una ganancia relativa en la predicción de más del 16% para el coeficiente de Gini y del 26% para el levantamiento del decil superior.

3. Definición del problema, justificación y alcance

3.1 Definición del problema

El presente trabajo pretende abordar el CLTV en una organización bancaria específica que carece de un modelo predictivo para conocer el valor futuro de sus clientes. Si bien la organización posee sistematizada la rentabilidad mensual histórica de cada cliente, como así también un algoritmo estadístico que estima la rentabilidad potencial de los clientes, entre un vasto repertorio de herramental analítico-predictivo, no cuenta con un modelo predictivo que permita clasificar en qué grupo de valor futuro se encontrará el cliente a partir de su relación actual con su banco. Es decir, no trabaja con proyecciones estadísticas sobre el flujo de rentabilidad que aportaría cada cliente a lo largo del tiempo en su relación con la firma, si se tiene en cuenta el comportamiento transaccional y la evolución histórica de la contratación y uso de los productos y servicios.

3.2 Justificación del estudio

El desarrollo e implementación de un modelo de valor de ciclo de vida del cliente le permitirá a la empresa, por una parte, conocer las variables y su combinación que tienen un nivel de significatividad o explicación relevante en el valor futuro del cliente (por ejemplo, combinación de productos, transaccionalidad, relación con el banco, etc.) para optimizar la estrategia de segmentación y marketing, y – al mismo tiempo – contribuir a perfeccionar el algoritmo de próxima mejor acción hacia el cliente (*Next Best Action* en inglés) con el objeto de aumentar la tasa de efectividad de la gestión integral del cliente (comercial, crediticia, de canales y de experiencia) y así poder incrementar el *Customer Equity* de la firma.

3.3 Alcances del trabajo y limitaciones

El presente trabajo se circunscribe a predecir el componente del “valor” o rentabilidad del CLTV de un cliente bancario. Por motivos de extensión no se profundizará en el componente de “churn” que completa la definición de CLTV y requiere de otro modelado predictivo adicional. Esto implica que, desde el punto de vista de clasificación de las variables a estudiar, el modelo se centra principalmente en transacciones de tipo continuas donde la relación con el cliente es contractual (ej: uso de la tarjeta de crédito), como así también variables sociodemográficas propias del cliente.

Por otra parte, no se consideran dentro de las variables explicativas de los modelos necesarios para calcular el CLTV la evolución de las variables macroeconómicas que afectan el comportamiento general de las firmas y consumidores ni la evolución de los fundamentos microeconómicos específicos del sector financiero. Si bien esto podría aumentar el nivel de confianza de las estimaciones de CLTV, requeriría desarrollar modelos de mayor complejidad y reunir variables externas a la firma, que están fuera del alcance de esta investigación.

4. Hipótesis y objetivos de investigación

4.1 Hipótesis

Pretendo demostrar que se puede predecir, con cierto grado de exactitud y precisión, el valor futuro de los clientes utilizando modelos de clustering y clasificación a partir de la información histórica transaccional y sociodemográfica de los clientes, mediante los algoritmos más modernos de ciencia de datos.

Siendo que el valor del ciclo de vida del cliente se compone, en su formulación más sencilla, de la proyección de la anualidad del margen de rentabilidad futura del cliente que acumulará para la firma, corregido por la tasa de retención (o 1 menos la tasa de abandono del cliente) y ajustado por el costo de capital de la empresa, en este trabajo, se busca encontrar las variables que pronostiquen el flujo futuro de ingresos provenientes del margen de rentabilidad del cliente.

Se espera que exista una relación multivariada entre el flujo futuro de ingresos y variables explicativas tales como: antigüedad del cliente, cantidad de contratos o productos del cliente con el banco, cantidad de transacciones en cuenta, frecuencia de transacciones, nivel de ingresos, región geográfica, edad, cantidad de bancos con los que opera, principalidad respecto del resto de los bancos, volumen de depósitos y créditos, canales por los que opera y su frecuencia, margen de rentabilidad mensual histórico, entre otras.

Algunas variables podrán tener un tipo de relación de paralelismo, por ejemplo:

- A mayor transaccionalidad pasada y mayor margen de rentabilidad histórica, mayor transaccionalidad y margen futuro.

De causa – efecto, tal como:

- Si el cliente contrató un producto de largo plazo como un préstamo hipotecario, tendrá un mayor ciclo de vida para la firma.

O bien, de oposición:

- A mayor cantidad de bancos con los que opera, y mayor principalidad con respecto al resto de los bancos, menor probabilidad de abandono.

Sin embargo, se espera que existan combinaciones múltiples entre las distintas variables que puedan implicar relaciones no previstas por el sentido común. Esto también estará influido por la operacionalización de las mismas ya que se utilizarán las variables mencionadas (operacionalizadas mediante variables ordinales y cardinales, tales como cantidades, frecuencias, montos, etc.) y sus derivadas de *feature engineering.*

4.2 Objetivos de la investigación

La presente investigación tiene como objetivo general el desarrollo e implementación de modelos predictivos de CLTV para pronosticar, manteniendo constantes los efectos micro y macroeconómicos, el valor futuro devengado por los clientes actuales de la firma en los diferentes segmentos del negocio.

Los objetivos específicos del trabajo consisten en:

1) Reunir los datos sociodemográficos e históricos de tenencia y uso de productos, interacción y comportamiento con los canales de atención, riesgo crediticio, rentabilidad, y segmentaciones analíticas existentes de los clientes de banca minorista de la entidad bancaria.
2) Realizar un análisis exploratorio de datos sobre las principales variables planteadas en la hipótesis de la investigación.
3) Entrenar y validar un modelo predictivo de rentabilidad anual (*net profit*) por cliente para los siguientes 12 meses.

5. Metodología

Para el análisis exploratorio de datos se trabajará con las medidas estándar de estadística descriptiva aplicadas a las principales variables del conjunto de datos, incluyendo correlaciones de a pares y diagramas de box-plot, entre otros.

El modelo predictivo de rentabilidad por cliente es un problema de regresión, mientras que el modelo de churn es un problema de clasificación (este último se encuentra fuera del alcance del presente trabajo). Sin embargo, ambos pueden ser abordados mediante algoritmos de ensamble de árboles como *random forests*. En particular, se intentará una mejor predicción del valor del cliente con técnicas de gradient boosting, tales como XGBoost, LightGBM, etc.

Los pasos a seguir para el desarrollo de los modelos son: procesamiento de datos, generación de feature engineering, entrenamiento, calibración y presentación.

Las medidas de performance a emplear para evaluar los modelos, según recomienda la literatura, es el coeficiente de correlación de rangos de Spearman para un problema de regresión y el área bajo la curva (AUC) para la clasificación. En este caso profundizaremos en particular sobre las métricas de Accuracy, Precision y Recall.

6. Desarrollo

6.1 Extracción y Manipulación de datos

Los problemas de aprendizaje supervisado requieren de un dataset con cierta estructura. La misma consiste en un conjunto de n-tuplas o registros que identifican al objeto, suceso o evento bajo análisis y a las variables correspondientes a las características del mismo, como así también la etiqueta, clase o valor a predecir.

Para confeccionar el dataset de este trabajo, se procedió a identificar las características o información de los clientes en un determinado mes del año 2018. En la literatura del marketing analítico esto se suele denominar la "visión 360 del cliente", es decir poder estructurar y consolidar un conjunto de variables que resuman las principales características del cliente en su relación con la empresa, además de las propias personales. En términos de base de datos, se busca crear una *Analytical Base Table* o ABT del cliente, esto es, una tabla plana que contiene información sobre quién es el sujeto (que describe las características del sujeto relacionadas con la empresa, como datos sociodemográficos y geográficos, eventos, etc.) y qué hace el sujeto (que describe las características del comportamiento del sujeto, compra de productos, uso de productos, comportamiento de pago, instancias de relación, etc.).

El procedimiento para crear el ABT del cliente se realizó un conjunto de queries en SQL sobre el datamart de clientes de la empresa, consolidando información de los distintos modelos estrella y aplanados de las siguientes entidades o dominios de información:

- **Clientes:** métricas sociodemográficas del cliente (edad, sexo, antigüedad, segmento, sucursal, etc.).
- **Sucursales:** métricas específicas de tipo de sucursal del cliente y ubicación.
- **Calificación Crediticia:** métricas que definen la aprobación o no de una precalificación crediticia a los clientes, el monto y sus sublíneas.
- **Saldos en Cuentas:** métricas que describen la situación en el tiempo de los saldos deudores y acreedores en los distintos tipos de cuentas del cliente.
- **Operaciones y Movimientos:** métricas que detallan la operatoria típica del cliente entre sus principales productos (pagos, depósitos, transferencias, débitos, etc.).
- **Deuda y Mora:** métricas que indican la deuda que tiene el cliente con la institución bancaria, con el sistema financiero y su nivel de mora respectivo.
- **Plazos Fijos:** métricas específicas sobre los clientes que tuvieron plazos fijos, montos, plazos, etc.
- **Préstamos:** métricas detalladas sobre los clientes que tuvieron préstamos respecto del tipo de préstamo, tasa, monto, plazo, frecuencia, etc.
- **Tenencia de Productos:** métricas que describen la tenencia de todo el abanico de productos bancarios en la forma de *flags* o "banderas" (tiene o no tiene) sobre: cuentas, tarjetas, seguros, etc.
- **Valoración:** métricas que reflejan el cálculo de ingresos y egresos por cliente, mediante costeo por absorción, con un *lag* o retraso de 3 a 6 meses.

El procesamiento de las consultas SQL se realizó con la consola *Microsoft SQL Studio Management* apuntando al Datamart de Visión 360 del cliente alojado en un servidor *MySQL*. A continuación se detalla como ejemplo una de las queries principales para armar el ABT de clientes necesario para comenzar el análisis.

```
SET NOCOUNT ON
DECLARE @VIG INT, @PER INT, @PER_TAR INT; DECLARA VARIABLES

SET @PER_TAR = N'$(per_tar)'; ---SELECCIONAMOS PERIODO DEL TARGET

SET @PER = (SELECT CASE WHEN (@PER_TAR % 100) - 3 > 0 THEN @PER_TAR  - 3 ELSE @PER_TAR - 91 END); ---
CALCULAMOS PERIODO DE LOS FEATURES


SET @VIG = (SELECT
      MAX(CC.CC)
    FROM
      DATAMARTV360_CC_VIGENCIA AS CC ---Buscamos la calificación crediticia de los clientes correspondiente al período
de los features


    WHERE CAST(CONCAT(YEAR(CC.FECHA_CALIFICACION), FORMAT(CC.FECHA_CALIFICACION,'MM')) AS INT) <=
CAST(@PER AS INT));

SELECT
            CLI.[PERIODO],
            CLI.[CODIGO_CLIENTE],
            CLI.[CLI_INDICE_CROSS_SELL],
            CLI.[CLI_ANTIG_CLIENTE_PROD_VIGENTES],
            CLI.[CLI_POSICION_PASIVA],
            CLI.[CLI_POSICION_ACTIVA],
            CLI.[CLI_SEXO],
            DATEDIFF(YEAR,CLI.[CLI_FECHA_NACIMIENTO],GETDATE()) AS EDAD,
            CLI.[CLI_CATEGORIA],
            CLI.[CLI_SITUACION_BCRA],
```

```
CLI.[CLI_SUCURSAL_SOURCE_RADICACION],
CLI.[CLI_CODIGO_IVA],
CLI.[CLI_SEGMENTO],
CLI.[CLI_SUB_SEGMENTO],
CLI.[CLI_SEGMENTO_COMERCIAL],
CLI.[CLI_SUB_SEGMENTO_COMERCIAL],
CLI.[CLI_SCORE_DATOS_ADICIONALES],
CLI.[CLI_CANTIDAD_EMPLEADORES],
CASE WHEN CLI.[CLI_CUIT_EMPLEADOR] > '0' THEN 1 ELSE 0 END AS FLAG_CUIT_EMPLEADOR,
CLI.[CLI_INGRESO_CONSIDERADO],
CLI.[CLI_INGRESO],
CLI.[ACRED_CUPONES_ULT_3],
CLI.[ANTERIOR_CC_VIGENTE],
CLI.[FLAG_TIENE_ANTERIOR_CC_VIG],
CLI.[ULTIMA_CC_VIGENTE],
CLI.[FLAG_TIENE_ULTIMA_CC_VIG],
CASE WHEN CC.CODIGO_CLIENTE IS NULL THEN 0 ELSE 1 END AS FLAG_CALIF_CTE,
CC.CC,
SAL.DV_ANT_SALDO_NEGATIVO_3,
SAL.DV_ANT_SALDO_NEGATIVO_6,
SAL.DV_ANT_SALDO_NEGATIVO_12,
SAL.DV_ANT_SALDO_POSITIVO_3,
SAL.DV_ANT_SALDO_POSITIVO_6,
SAL.DV_ANT_SALDO_POSITIVO_12,
SAL.DV_PROM_SALDO,
SAL.DV_PROM_SALDO_3,
SAL.DV_PROM_SALDO_6,
SAL.DV_PROM_SALDO_12,
OP.DV_CA_ACRED,
OP.DV_CA_EXTRAC,
OP.DV_CA_MONTO_TRANS,
OP.DV_CA_ACRED_3,
OP.DV_CA_EXTRAC_3,
OP.DV_CA_MONTO_TRANS_3,
CASE WHEN OP.DV_CA_MONTO_TARDEB_3 > 0 THEN 1 ELSE 0 END AS FLAG_COMPRA_TD,
DE.CLI_DIAS_MORA,
DE.CLI_SALDO_DEUDA,
DE.CLI_DIAS_SOBREGIRO,
DE.CLI_DIAS_ATRASO_BCRA,
DE.CLI_FLAG_SITUACION_BANCO_1_3,
DE.CLI_FLAG_SITUACION_BANCO_2_3,
DE.CLI_FLAG_SITUACION_BANCO_3_3,
CAN.CAN_MONTO_ATM,
CAN.CAN_MONTO_TAS,
CAN.CAN_MONTO_HB,
CAN.CAN_MONTO_IVR,
CAN.CAN_MONTO_CAT,
CAN.CAN_MONTO_ATM_3,
CAN.CAN_MONTO_TAS_3,
CAN.CAN_MONTO_HB_3,
CAN.CAN_MONTO_IVR_3,
CAN.CAN_MONTO_CAT_3,
CASE WHEN CAN.CAN_CANT_NO_MO_HB_3 > 0 THEN 1,
WHEN CAN.CAN_CANT_NO_MO_MOB_3> 0 THEN 1 ELSE 0 END AS FLAG_CLI_DIGITAL,
PF.PF_SALDO,
PF.PF_CANT,
PF.PF_SALDO_3,
PF.PF_CANT_PROM_3,
PMO.PR_HIPOTECARIOS,
PMO.PR_PERSONALES,
PMO.PR_ATRASO_30,
PMO.PR_ATRASO_60,
PMO.PR_MONTO_ORIGINAL,
PMO.PR_MESES_CON_PRESTAMOS,
PMO.PR_ANTIGUEDAD_PRESTAMOS,
PMO.PR_MESES_ENTRE_PRESTAMOS,
PMO.PR_MESES_CON_PERSONALES,
PMO.PR_PORC_RENOVADOS_RECIENTE,
PMO.PR_Q_RENOVADOS_RECIENTE,
TEN.FLAG_CUENTA_TC,
TEN.FLAG_CA,
TEN.FLAG_CA_USD,
TEN.FLAG_CC,
TEN.FLAG_TD,
```

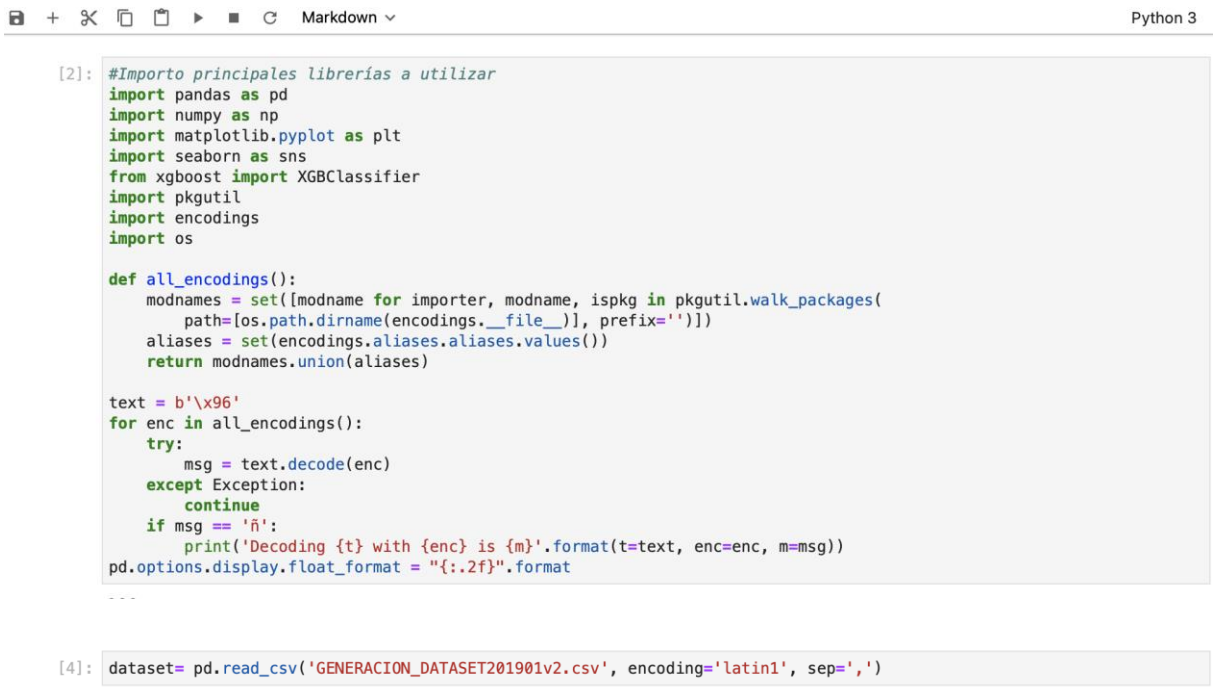
```
        TEN.FLAG_SEGURO,
        TEN.FLAG_PAQUETE_BI,
        VAL.VALOR_PROPIO_PROM_1,
        VAL.VALOR_PROPIO_PROM_3,
        VAL.VALOR_PROPIO_PROM_6,
        VAL.VALOR_PROPIO_PROM_12,
        VAL_T.VALOR_PROPIO_PROM_1,
        VAL_T.VALOR_PROPIO_PROM_3,
        SUC.DIVISION_SUCURSAL

 FROM
        [KPP].[dbo].[DATAMARTV360_CLIENTES] AS CLI
        INNER JOIN DATAMARTV360_SUCURSALES_DATOS AS SUC
        ON CLI.CLI_SUCURSAL_SOURCE_RADICACION = SUC.SUCURSAL_SOURCE
        INNER JOIN DATAMARTV360_CLIENTES_DIARIO AS DIA ON CLI.CODIGO_CLIENTE = DIA.CODIGO_CLIENTE
LEFT    JOIN DATAMARTV360_CC_APROBADOS AS CC
        ON CLI.CODIGO_CLIENTE = CC.CODIGO_CLIENTE AND CC.CC = @VIG
        LEFT JOIN DATAMARTV360_CA_CC_SALDOS_DATASET AS SAL
        ON CLI.CODIGO_CLIENTE = SAL.CODIGO_CLIENTE  AND CLI.PERIODO = SAL.PERIODO
        LEFT JOIN DATAMARTV360_CA_CC_OPERACIONES_DATASET AS OP
        ON CLI.CODIGO_CLIENTE = OP.CODIGO_CLIENTE  AND CLI.PERIODO   = OP.PERIODO
        LEFT JOIN DATAMARTV360_DEUDA_Y_MORA_DATASET AS DE
        ON CLI.CODIGO_CLIENTE = DE.CODIGO_CLIENTE  AND CLI.PERIODO   = DE.PERIODO
        LEFT JOIN DATAMARTV360_CANALES_DATASET AS CAN
        ON CLI.CODIGO_CLIENTE = CAN.CODIGO_CLIENTE  AND CLI.PERIODO   = CAN.PERIODO
        LEFT JOIN DATAMARTV360_PLAZO_FIJO_DATASET AS PF
        ON CLI.CODIGO_CLIENTE = PF.CODIGO_CLIENTE   AND CLI.PERIODO   = PF.PERIODO
        LEFT JOIN DATAMARTV360_PRESTAMOS_DATASET AS PMO
        ON CLI.CODIGO_CLIENTE = PMO.CODIGO_CLIENTE   AND CLI.PERIODO   = PMO.PERIODO
        LEFT JOIN DATAMARTV360_TENENCIAS_PRODUCTOS_DATASET AS TEN
        ON CLI.CODIGO_CLIENTE = TEN.CODIGO_CLIENTE   AND CLI.PERIODO   = TEN.PERIODO
        INNER JOIN DATAMARTV360_VALORACION_DATASET AS VAL
        ON CLI.CODIGO_CLIENTE = VAL.CODIGO_CLIENTE AND CLI.PERIODO   = VAL.PERIODO
        INNER JOIN DATAMARTV360_VALORACION_DATASET AS VAL_T
        ON CLI.CODIGO_CLIENTE = VAL_T.CODIGO_CLIENTE     AND VAL_T.PERIODO   = @PER_TAR


WHERE
        CLI.CLI_INDICE_CROSS_SELL > 0
        AND
        CLI.PERIODO = @PER
        AND
        CLI.CODIGO_CLIENTE % 100 > 90
        AND
        (DIA.CLIENTE_VINCULADO_GRUPO_MACRO+
        DIA.EMPLEADO+
        DIA.FUNCIONARIO+
        DIA.DIRECTOR+
        DIA.RESTRINGIDO+
        DIA.PEP+
        DIA.FALLECIDO+
        DIA.NO_CONTACTAR+
        DIA.NO_CONTACTAR_POR_CARTA+
        DIA.NO_CONTACTAR_POR_MAIL+
        DIA.NO_CONTACTAR_POR_TELEFONO+
        DIA.NO_PRE_CALIFICAR+
        DIA.SMSV_CTA+
        DIA.SMSV_TC+
        DIA.NO_VALIDADO_MERLIN+
        DIA.COTOLENGO+
        DIA.PROGRAMA_HOGARES_GARRAFA+
        DIA.ALUMBRA+
        DIA.EMPLEADO_FUN_INCLUIR) = 0
```

Dado que la empresa cuenta con varios millones de clientes y considerando la cantidad de variables a analizar, se decidió trabajar con una muestra del universo de datos levemente superior al 20%, utilizando un muestreo mediante la selección de todos los clientes cuyo número interno de identificación finaliza en 90, con el objeto de poder estructurar un dataset que pueda manipularse en un entorno de ciencia de datos local como así también en un entorno de nube gratuito, y para poder procesar la información en no más de 2 horas.

Con respecto a la privacidad de datos, se procedió a la anonimización de los mismos mediante una función de hash determinista sobre el código de cliente interno, esto es un proceso de transformación del código interno de identificación del cliente que impide reconocer quién es el cliente de cada tupla o registro de información.

Luego de realizar la iteración de la query a lo largo de varios meses del año 2019, se buscó elegir un mes para establecer como dataset principal que estuviera lo menos influenciado posible de ciertos ciclos estacionales como ser: paritarias, aguinaldo o eventos externos relevantes. Esto nos llevó a optar por el mes de Enero de 2019 ya que estaba lo suficientemente lejos de los eventos mencionados y posteriormente, debido a las elecciones PASO, existieron eventos micro y macroeconómicos coyunturales relevantes que podrían afectar el análisis del dataset en cuestión.

A continuación se importó el dataset en un entorno *IDE (Integrated Development Environment)* local de Python en *Jupyter Lab* de *Anaconda-Navigator*.



El notebook que se adjunta al presente trabajo se estructuró en 4 ejes:

- Importación de Dataset de Features de Clientes y su Valoración
- Análisis Descriptivo Exploratorio
- Desarrollo de Agrupamiento de Clientes por Similitud
- Desarrollo de Modelo de Predicción de Cluster de Valor del Cliente

6.2. Análisis Exploratorio de Datos

Luego de realizar las principales validaciones del dataset importado con los comandos básicos de Python, con librerías como Pandas, tales como dataset.head(), dataset.info(), dataset.dtypes(), dataset.isnull().sum(), dataset.dropna(), dataset.fillna(0), con el objeto de limpiar o rellenar campos nulos según corresponda, verificar tipos de datos y eliminar duplicados de existir, comenzamos con un primer análisis descriptivo de ciertas variables elegidas del dataset, utilizando el comando dataset.describe para analizar distintos ejes del dataset:

1- Datos Básicos

```
[15]: # Calculo las medidas de estadística descriptiva de las variables del dataset: DATOS BÁSICOS
dataset[['CLI_INDICE_CROSS_SELL' , 'CLI_ANTIG_CLIENTE_PROD_VIGENTES', 'CLI_POSICION_PASIVA', 'CLI_POSICION_ACTIVA',
```

[15]:

	CLI_INDICE_CROSS_SELL	CLI_ANTIG_CLIENTE_PROD_VIGENTES	CLI_POSICION_PASIVA	CLI_POSICION_ACTIVA	EDAD	CLI_CANTID
count	43180.00	43180.00	43180.00	43180.00	43180.00	
mean	2.18	84.99	35839.61	28546.99	48.66	
std	1.34	69.71	404244.95	88087.28	17.57	
min	1.00	0.00	0.00	-31405.87	4.00	
25%	1.00	29.00	0.82	0.00	35.00	
50%	2.00	67.00	43.42	0.00	46.00	
75%	3.00	128.00	3094.73	25236.08	61.00	
max	9.00	585.00	65154060.00	6607512.37	108.00	

Esto nos permite conocer la distribución de las principales variables del dataset. Por ejemplo, observamos en la muestra aleatorizada del dataset que tenemos bajo análisis 43.180 clientes con un nivel de cross sell (venta cruzada) de 2.18 productos, una antigüedad mínima de 0 meses y máxima de 585 meses, un saldo en cuentas bancarias vista y a plazo (posición pasiva) promedio de $35.839 con un desvío estándar de $404.245 y un saldo de deuda (posición activa) promedio de $ 28.547. La edad promedio de los clientes es de alrededor de 49 años.

2- Movimientos en Cuenta

```
[16]: # Calculo las medidas de estadística descriptiva de las variables del dataset: MOVIMIENTOS EN CUENTA
dataset[['DV_PROM_SALDO', 'DV_CA_ACRED', 'DV_CA_EXTRAC', 'DV_CA_MONTO_TRANS', 'PF_CANT', 'PF_SALDO', 'FLAG_COMPRA_TD
```

[16]:

	DV_PROM_SALDO	DV_CA_ACRED	DV_CA_EXTRAC	DV_CA_MONTO_TRANS	PF_CANT	PF_SALDO	FLAG_COMPRA_TD
count	43180.00	43180.00	43180.00	43180.00	43180.00	43180.00	43180.00
mean	12220.16	0.73	2.74	3492.41	0.10	25488.30	0.41
std	127852.75	1.36	4.21	72028.63	0.52	385063.60	0.49
min	-96774.19	0.00	0.00	0.00	0.00	0.00	0.00
25%	10.04	0.00	0.00	0.00	0.00	0.00	0.00
50%	512.59	0.00	1.00	0.00	0.00	0.00	0.00
75%	3217.88	1.00	4.00	0.00	0.00	0.00	1.00
max	19014888.56	25.00	117.00	11314558.20	46.00	65154060.00	1.00

En este conjunto de variables podemos observar que el saldo promedio vista de los clientes es de $ 12.330, pero que la mitad de los clientes no tienen más de $ 513 pesos en promedio en la cuenta por mes; la cantidad de acreditaciones en cuenta por cliente promedio es de 0,73 mientras que hay un máximo de clientes con 25 acreditaciones en el mes; por otra parte el promedio de extracciones en cuenta es de 2,74 transacciones por mes. El monto de las transferencias promedio mensual es de $ 3.492 y en promedio sólo un 10% tiene plazo fijo, con un saldo promedio de $ 25.844. El 41% realizó una compra con tarjeta de débito en el mes.

3- Deuda

```
[17]: # Calculo las medidas de estadística descriptiva de las variables del dataset: DEUDA
dataset[['CLI_DIAS_MORA', 'CLI_SITUACION_BCRA','CLI_SALDO_DEUDA', 'CLI_DIAS_SOBREGIRO', 'CLI_DIAS_ATRASO_BCRA', 'PR_
```

[17]:

	CLI_DIAS_MORA	CLI_SALDO_DEUDA	CLI_DIAS_SOBREGIRO	CLI_DIAS_ATRASO_BCRA	PR_HIPOTECARIOS	PR_PERSONALES	PR_MES
count	43180.00	43180.00	43180.00	43180.00	43180.00	43180.00	
mean	8.34	28814.42	37.93	37.93	0.00	0.36	
std	49.44	88238.27	358.33	358.33	0.04	0.81	
min	0.00	0.00	0.00	0.00	0.00	0.00	
25%	0.00	0.00	0.00	0.00	0.00	0.00	
50%	0.00	0.00	0.00	0.00	0.00	0.00	
75%	0.00	25965.53	0.00	0.00	0.00	0.00	
max	361.00	6607512.37	9017.00	9017.00	1.00	12.00	

Los clientes con deuda en mora tienen un promedio de 8,3 días de mora en el banco, lo que podría tratarse de lo que se conoce como mora operativa, es decir que se resuelve dentro del mes, con un máximo de 361 días y un mínimo de 0. Con respecto al resto de los bancos, registran un promedio de 38 días de atraso y un máximo de 9.000 días. La deuda promedio es de $ 28.814, pero el desvío estándar es muy alto. Esto se explica porque menos del 1% de los clientes posee un préstamo hipotecario, que aumenta el saldo de deuda muy por encima del promedio a los clientes que poseen uno. Un 36% en promedio tienen préstamos personales, llegando a extremos de clientes con 12 préstamos.

4- Tenencia de Otros Productos

```
[18]: # Calculo las medidas de estadística descriptiva de las variables del dataset: TENENCIA PRODUCTOS
dataset[['FLAG_CA', 'FLAG_CA_USD', 'FLAG_CC', 'FLAG_TD', 'FLAG_SEGURO', 'FLAG_PAQUETE_BI']].describe()
```

[18]:

	FLAG_CA	FLAG_CA_USD	FLAG_CC	FLAG_TD	FLAG_SEGURO	FLAG_PAQUETE_BI
count	43180.00	43180.00	43180.00	43180.00	43180.00	43180.00
mean	0.90	0.11	0.29	0.81	0.36	0.31
std	0.31	0.31	0.46	0.39	0.48	0.46
min	0.00	0.00	0.00	0.00	0.00	0.00
25%	1.00	0.00	0.00	1.00	0.00	0.00
50%	1.00	0.00	0.00	1.00	0.00	0.00
75%	1.00	0.00	1.00	1.00	1.00	1.00
max	1.00	1.00	1.00	1.00	1.00	1.00

El 90% de los clientes posee una caja de ahorros en pesos, un 11% caja de ahorro en dólares y un 29% posee una cuenta corriente; el 36% posee algún seguro y el 31% posee un paquete.

5- Otros cortes de información

Al abrir la información por región donde se encuentran radicados los productos del cliente o por segmento al que pertenecen, encontramos composiciones distintas de las variables anteriormente mencionadas. Por ejemplo el índice de cross sell promedio es de 2.38 para un cliente de NEA, mientras que es de 1.75 para un cliente de Tucumán; la antigüedad promedio de los productos de un cliente Plan Sueldo Público (es decir, aquellos que trabajan en la administración pública y cobran su sueldo en

una cuenta del banco) es de 132 meses, mientras que un cliente perteneciente a Otros Segmentos Especiales (esto es, aquellos que cobran una prestación del Estado) es de 33 meses.

DIVISION_SUCURSAL	CLI_INDICE_CROSS_SELL
A DETERMINAR	4.33
AMBA	2.02
CASA MATRIZ	1.30
CORDOBA	2.08
JUJUY	2.30
NEA	2.38
PROVINCIA BS. AS.	2.11
SALTA	2.37
SANTA FE	2.14
SUROESTE	1.96
TUCUMAN	1.75

CLI_SEGMENTO_COMERCIAL	CLI_INDICE_CROSS_SELL	CLI_ANTIG_CLIENTE_PROD_VIGENTES
JUBILADOS	2.70	100.98
MERCADO ABIERTO	1.54	69.69
OTROS SEGMENTOS ESPECIALES	1.21	58.51
PLAN SUELDO PRIVADO	2.69	78.22
PLAN SUELDO PUBLICO	3.40	132.85
PROFESIONALES Y NEGOCIOS	2.04	90.40

6- Valoración del Cliente

Como indicamos al comienzo del trabajo, la valoración del cliente es el indicador que surge del sistema de cálculo de rentabilidad por cliente de la empresa basado en costeo por absorción y reglas de negocio específicas acordadas por el *management* para asignar ingresos y egresos del negocio. Debido a que el sistema tiene un *delay* relevante en el cálculo de este indicador, hemos utilizado en el dataset un promedio mensual de los últimos 12 meses disponibles de la valoración del cliente para evitar distorsiones y suavizar cualquier impacto producto de un cambio estacional en la composición de productos y servicios del cliente.

Al analizar la variable con los indicadores de estadística descriptiva básica encontramos una fuerte dispersión, que se observa en los siguientes valores de media, desvío, mínimos y máximos:

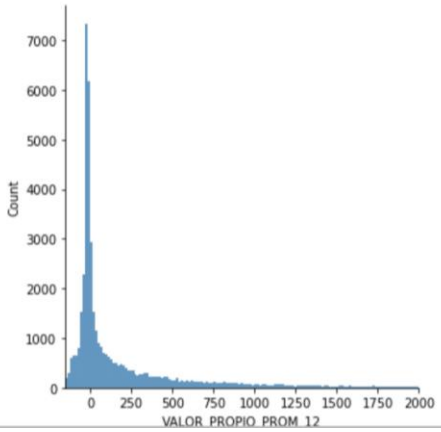
```
[10]: # Realizo un análisis específico de la métrica Valoración
      dataset['VALOR_PROPIO_PROM_12'].describe()

[10]: count    47750.00
      mean       261.03
      std       1234.85
      min     -46825.16
      25%        -22.19
      50%          2.77
      75%        249.38
      max      57599.34
      Name: VALOR_PROPIO_PROM_12, dtype: float64
```

Por lo anterior, graficamos un histograma recortando al dataset los cuantiles del 5% de las colas de la distribución:

```
[25]: dataset['VALOR_PROPIO_PROM_12'].quantile(0.95)

[25]: 1635.1169999999956

[26]: dataset['VALOR_PROPIO_PROM_12'].quantile(0.05)

[26]: -134.5
```

```
[12]: #Grafico un histograma de la Valoración Promedio Mensual de los últimos 12 meses recortando el 5% de las colas
      sns.displot(dataset.VALOR_PROPIO_PROM_12)
      plt.xlim(-150, 2000)
```

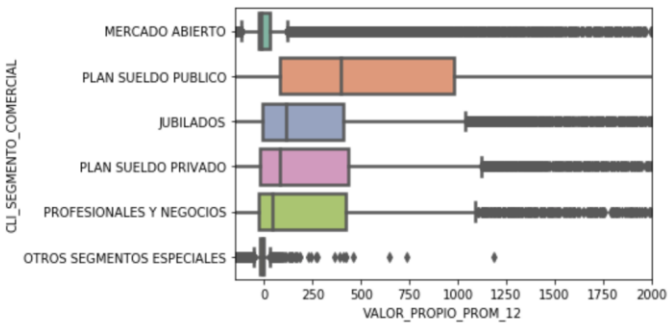
```
[12]: (-150, 2000)
```



Sin embargo, seguimos encontrando una fuerte distribución con mediana cercana a cero. Para poder diferenciar la valoración por cliente, realizamos un boxplot, o diagrama de cajas y bigotes, con los mismos límites pero abierto por segmento del cliente, y observamos ciertas diferencias: el cliente del segmento Plan Sueldo Público tiene una valoración mensual promedio cercana a los $ 500, mientras que en Otros Segmentos Especiales está muy cercana a 0.

```
[19]: # Grafico el boxplot por Segmento del Cliente
      sns.boxplot(y='CLI_SEGMENTO_COMERCIAL', x='VALOR_PROPIO_PROM_12',
                  data=dataset, palette='Set2', linewidth=2.5)
      plt.xlim(-150, 2000)
```

```
[19]: (-150, 2000)
```

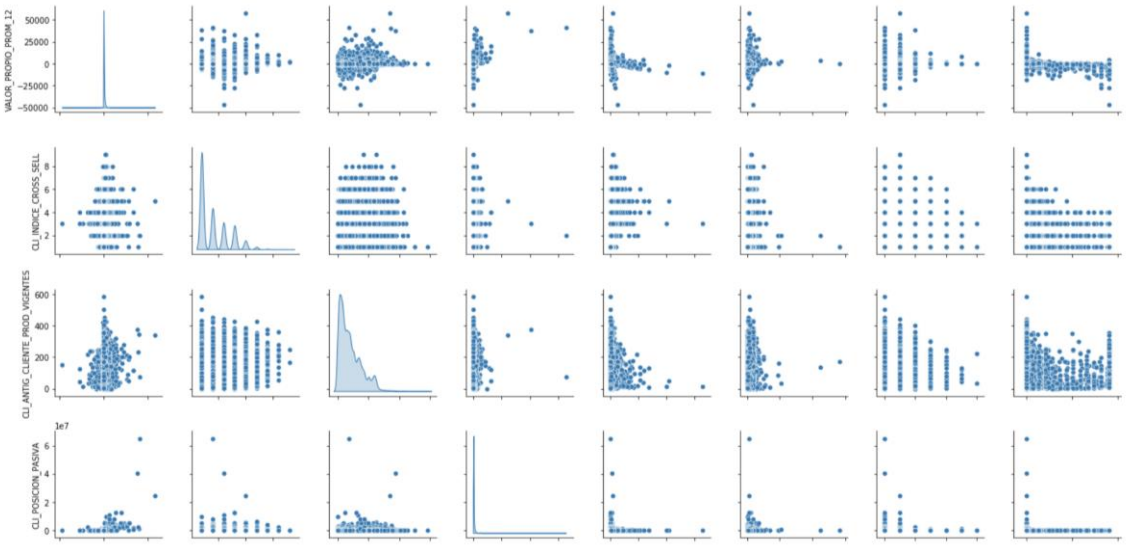


7- Análisis de correlación

A continuación, se realiza un análisis de correlación de a pares de variables para entender si podemos detectar visualmente ciertas relaciones claras entre las variables del dataset respecto de la Valoración del cliente:

```
[20]: # Grafico la correlación de a pares entre principales variables del dataset
      sns.pairplot(dataset, vars=["VALOR_PROPIO_PROM_12", "CLI_INDICE_CROSS_SELL", "CLI_ANTIG_CLIENTE_PROD_VIGENTES", "CLI
```

```
[20]: <seaborn.axisgrid.PairGrid at 0x7fe81e7bcdd8>
```



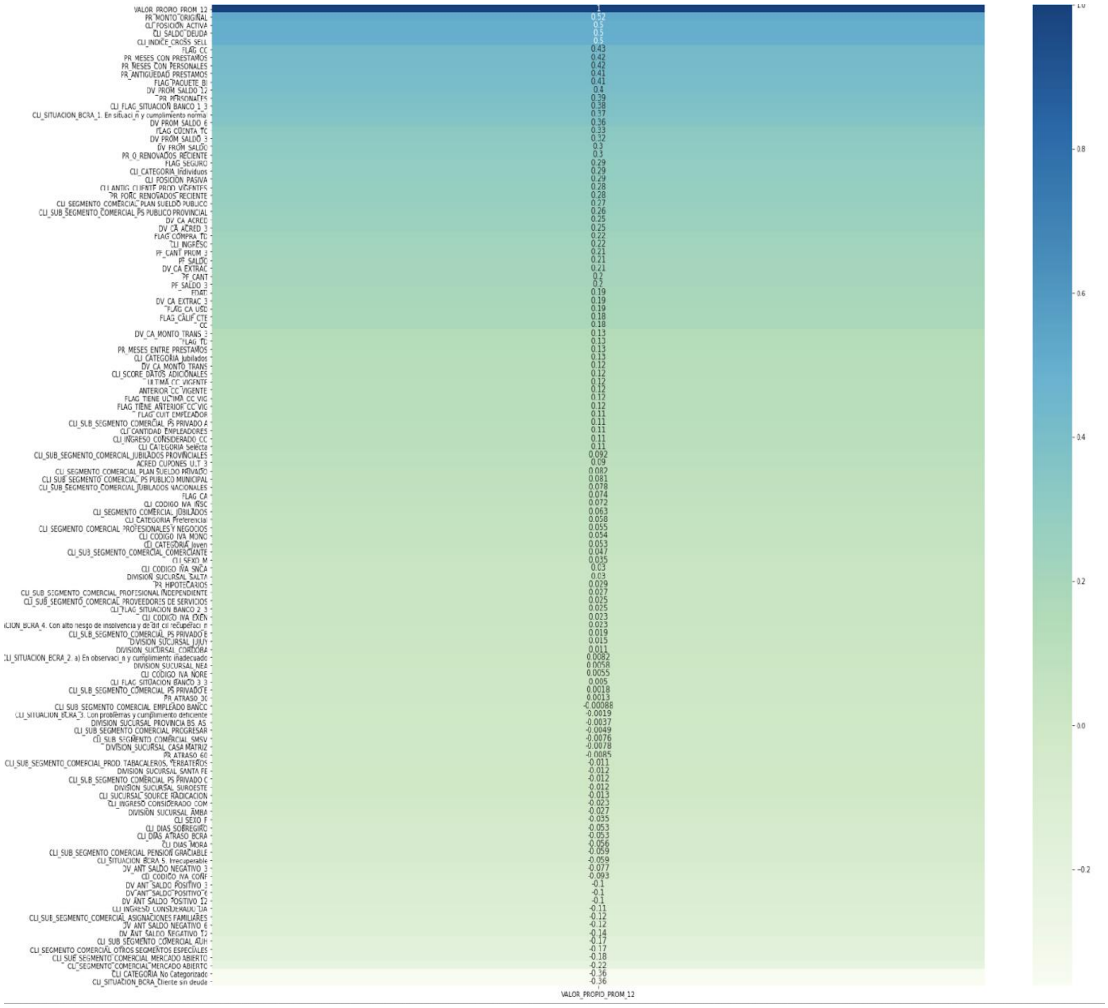
En la primera fila de gráficos se visualiza la correlación de la valoración promedio mensual de los clientes con los distintos features o variables del dataset. A priori, no es fácil detectar relaciones fuertes y claras positivas o negativas. Por eso, realizamos el cálculo de correlación en forma numérica, y entonces encontramos, por ejemplo, que la Valoración (Valor_Propio_Prom_12) tiene una correlación de 0.5 con la cantidad de productos vigentes que tiene un cliente (Cli_Indice_Cross_Sell):

```
[32]: cil encontrar correlaciones muy claras
TES", "CLI_POSICION_PASIVA","CLI_POSICION_ACTIVA","CLI_INGRESO","CLI_CANTIDAD_EMPLEADORES", 'CLI_DIAS_MORA']].corr()
```

[32]:

	VALOR_PROPIO_PROM_12	CLI_INDICE_CROSS_SELL	CLI_ANTIG_CLIENTE_PROD_VIGENTES	CLI_POSICI
VALOR_PROPIO_PROM_12	1.00	0.50	0.28	
CLI_INDICE_CROSS_SELL	0.50	1.00	0.32	
CLI_ANTIG_CLIENTE_PROD_VIGENTES	0.28	0.32	1.00	
CLI_POSICION_PASIVA	0.29	0.10	0.05	
CLI_POSICION_ACTIVA	0.50	0.53	0.22	
CLI_INGRESO	0.22	0.16	0.21	
CLI_CANTIDAD_EMPLEADORES	0.11	0.12	0.04	
CLI_DIAS_MORA	-0.06	-0.02	0.06	

Para poder realizar un análisis más claro y exhaustivo de correlaciones, realizamos transformaciones adicionales al dataset, tales como generar *dummies* para las variables con valores nominales o categóricas, y luego realizamos un análisis visual de correlaciones de todas las variables respecto de la Valoración del Cliente ordenadas de mayor a menor con un mapa de calor:



Ahora sí podemos observar una correlación con algún grado de relevancia entre la Valoración del Cliente (Valor_Propio_12) y: a) la tenencia de **préstamo personal** (PR_Monto_Original), saldos de deudas en general y variables relacionadas, b) el **índice de cross sell** del cliente (CLI_Indice_Cross_Sell) y variables relacionadas como Tenencia de un Paquete, Tenencia de Tarjeta de Crédito, Seguro, etc., c) el saldo de **depósitos vista** (DV_PROM_SALDO_12) y variables relacionadas como Plazo Fijo, Acreditaciones y Extracciones, d) la **Situación 1 en Banco Central**, es decir, sin mora (CLI_SITUACION_BCRA_1…) y relacionadas, e) su ingreso personal (CLI_Ingreso), f) la antigüedad de los productos vigentes (CLI_Antig_Cliente_Prod_Vigentes), etc.

6.3. Desarrollo de Agrupamiento de Clientes por Similitud

Habiendo encontrado correlaciones relevantes entre variables que describen el comportamiento del cliente con respecto a su Valoración, avanzamos en una segunda etapa para agrupar a los clientes en conjuntos con comportamiento similar en las principales variables mencionadas anteriormente. El objetivo de esta etapa es validar si los grupos de clientes con valoración similar entre sí tienen un nivel de correlación relevante con los grupos de clientes similares entre sí respecto de las principales variables de comportamiento y relación con el banco identificadas. Esto nos permitirá, en última instancia, avanzar en un modelo predictivo que nos permita clasificar a los clientes en los distintos grupos de valor.

1- Valor del Cliente

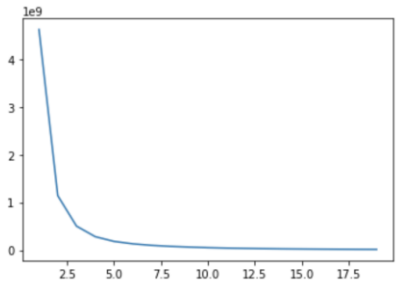
Mediante la técnica de clustering K-means, comenzamos por entrenar un modelo para agrupar los clientes con Valoración Promedio 12 meses similar:

In [40]:
```
# Entreno cluster de Valoración Promedio 12 meses
from sklearn.cluster import KMeans
km = KMeans(n_clusters=5)
y=dataset['VALOR_PROPIO_PROM_12']
y=y.values.reshape(-1,1)
km.fit(y)
```

Out[40]:
```
KMeans(n_clusters=5)
```

Con la técnica del codo ajustamos la cantidad de clusters a la óptima:

In [41]:
```
# Chequeo con la técnica del codo la cantidad óptima de clusters
n = range(1,20)
KMScore = []
for i in n:

    km = KMeans(n_clusters=i)
    km.fit(y)
    KMScore.append(km.inertia_)
import matplotlib.pyplot as plt
plt.plot(n,KMScore);
```



Ajustando la cantidad óptima de clusters a 4, incoporamos al dataset el valor del cluster de Valoración (VALOR12Cluster) correspondiente a cada cliente.

In [42]:
```
# Creo el cluster y lo agrego al dataset
km = KMeans(n_clusters=4)
km.fit(y)
dataset['VALOR12Cluster'] = km.predict(y)
```

Luego analizamos la distribución de los valores de Valor que toman los clientes en cada cluster tanto numérica como gráficamente:

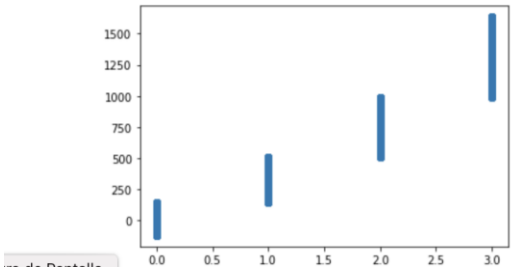
In [43]:
```
# Creo un nuevo dataset para analizar cómo agrupó los valores el cluster
dataset_cluster = dataset.copy()
dataset_cluster = order_cluster('VALOR12Cluster', 'VALOR_PROPIO_PROM_12',dataset_cluster ,True)
dataset_cluster.groupby('VALOR12Cluster')['VALOR_PROPIO_PROM_12'].describe()
```

Out[43]:

VALOR12Cluster	count	mean	std	min	25%	50%	75%	max
0	26117.00	-2.86	53.85	-122.60	-29.49	-12.48	19.31	146.54
1	6960.00	294.80	103.25	146.57	203.01	277.56	378.32	507.53
2	3536.00	717.59	136.91	507.66	597.62	705.87	829.90	991.35
3	2006.00	1262.67	179.85	991.64	1111.42	1233.81	1404.19	1635.07

Se puede observar que los grupos de Valor están correctamente diferenciados, pero desbalanceados como era previsible, dada la alta cantidad de clientes con bajo valor medio.

```
In [44]: # Evaluo gráficamente la distribución de valores del Cluster
         plt.plot(dataset_cluster['VALOR12Cluster'], dataset_cluster['VALOR_PROPIO_PROM_12'],'o')
Out[44]: [<matplotlib.lines.Line2D at 0x7f97bce0a390>]
```

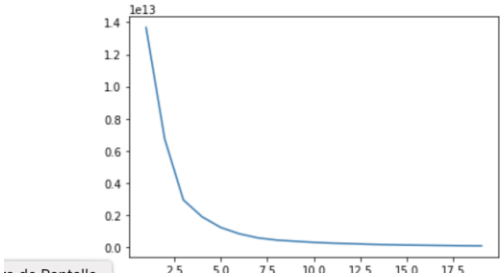


A continuación realizamos el mismo procedimiento con las variables principales con correlación relevante detectada con respecto al Valor del cliente.

2- Saldo de Depósitos a la Vista

Entrenamos el modelo de K-means con 4 clusters pero luego la técnica del codo nos indica que es mejor un agrupamiento de 3 clases:

```
In [45]: # Entreno cluster de Posición Pasiva
         from sklearn.cluster import KMeans
         km = KMeans(n_clusters=4)
         a=dataset['DV_PROM_SALDO_12']
         a=a.values.reshape(-1,1)
         km.fit(a)
Out[45]: KMeans(n_clusters=4)

In [46]: # Chequeo con la técnica del codo la cantidad óptima de clusters
         n = range(1,20)
         KMScore = []
         for i in n:

             km = KMeans(n_clusters=i)
             km.fit(a)
             KMScore.append(km.inertia_)
         import matplotlib.pyplot as plt
         plt.plot(n,KMScore);
```



Esto lo ratificamos en la distribución de los indicadores estadísticos por cluster y también visualmente, notando una distribución bastante desequilibrada:

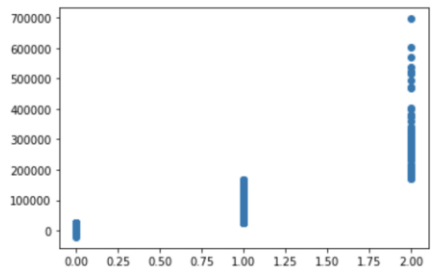
In [48]:
```
# Creo un nuevo dataset para analizar cómo agrupó los valores el cluster
dataset_cluster = dataset.copy()
dataset_cluster = order_cluster('SALDOVISTACluster', 'DV_PROM_SALDO_12',dataset_cluster ,True)
dataset_cluster.groupby('SALDOVISTACluster')['DV_PROM_SALDO_12'].describe()
```

Out[48]:

SALDOVISTACluster	count	mean	std	min	25%	50%	75%	max
0	36459.00	2579.69	4570.55	-20655.58	66.62	644.47	2652.32	25224.41
1	2078.00	47861.44	23634.24	25240.14	30908.16	40566.79	56722.95	168547.92
2	82.00	290976.92	112404.09	170424.20	206904.05	264548.09	325538.31	696124.48

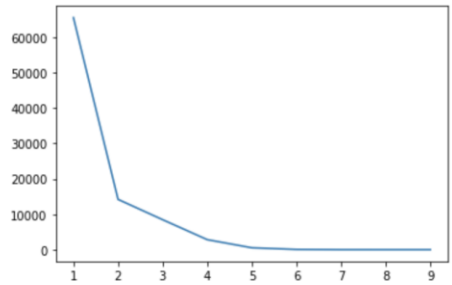
In [49]:
```
# Evaluo gráficamente la distribución de valores del Cluster
plt.plot(dataset_cluster['SALDOVISTACluster'], dataset_cluster['DV_PROM_SALDO_12'],'o')
```

Out[49]:
```
[<matplotlib.lines.Line2D at 0x7f97bd92f668>]
```



3- Índice de Cross Sell

Del mismo modo, realizamos el procedimiento de K-means con el índice de cross de productos y determinamos 3 clusters:



[52]:
```
# Creo el cluster y lo agrego al dataset
km = KMeans(n_clusters=3)
km.fit(b)
dataset['CROSSELLCluster'] = km.predict(b)
dataset = order_cluster('CROSSELLCluster', 'CLI_INDICE_CROSS_SELL',dataset,True)
```

[53]:
```
# Creo un nuevo dataset para analizar cómo agrupó los valores el cluster
dataset_cluster = dataset.copy()
dataset_cluster = order_cluster('CROSSELLCluster', 'CLI_INDICE_CROSS_SELL',dataset_cluster ,True)
#see details of the clusters
dataset_cluster.groupby('CROSSELLCluster')['CLI_INDICE_CROSS_SELL'].describe()
```

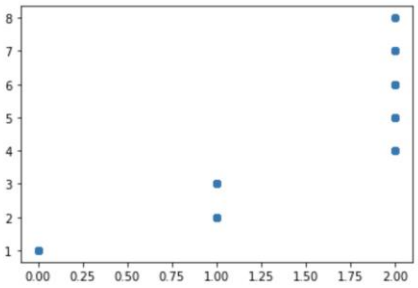
La distribución es un poco más equilibrada que en el caso anterior:

[53]:

CROSSELLCluster	count	mean	std	min	25%	50%	75%	max
0	18155.00	1.00	0.00	1.00	1.00	1.00	1.00	1.00
1	13470.00	2.39	0.49	2.00	2.00	2.00	3.00	3.00
2	6994.00	4.38	0.63	4.00	4.00	4.00	5.00	8.00

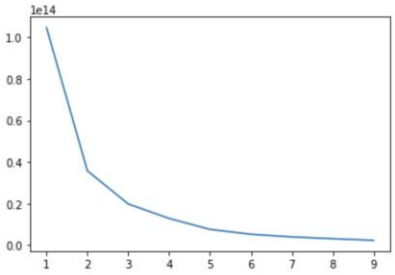
[54]:
```
# Evaluo gráficamente la distribución de valores del Cluster
plt.plot(dataset_cluster['CROSSELLCluster'], dataset_cluster['CLI_INDICE_CROSS_SELL'],'o')
```

[54]:
```
[<matplotlib.lines.Line2D at 0x7f97bd27f9e8>]
```



4- Préstamos

Nuevamente hacemos el ejercicio para el cluster de clientes con préstamos y obtenemos 3 grupos:



In [57]:
```
# Creo el cluster y lo agrego al dataset
km = KMeans(n_clusters=3)
km.fit(d)
dataset['PRESTAMOCluster'] = km.predict(d)
dataset = order_cluster('PRESTAMOCluster', 'PR_MONTO_ORIGINAL',dataset,True)
```

In [58]:
```
#creatinga new cluster dataframe
dataset_cluster = dataset.copy()
dataset_cluster = order_cluster('PRESTAMOCluster', 'PR_MONTO_ORIGINAL',dataset_cluster ,True)
#see details of the clusters
dataset_cluster.groupby('PRESTAMOCluster')['PR_MONTO_ORIGINAL'].describe()
```

Out[58]:

PRESTAMOCluster	count	mean	std	min	25%	50%	75%	max
0	33236.00	2755.54	8853.55	0.00	0.00	0.00	0.00	47000.00
1	4422.00	91552.78	34244.09	47200.00	61400.00	83000.00	115000.00	172500.00
2	961.00	253659.84	111993.79	173000.00	196000.00	226000.00	275800.00	1700000.00

5- Score General

Habiendo realizado los agrupamientos de la variable objetivo (Valoración del Cliente) y de un conjunto de variables que encontramos relevantes en términos de correlación positiva con la anterior, intentamos generar una suerte de “score general” de los clientes, que consiste en sumar todas las posiciones que ocupa el cliente en cada cluster de variables “explicativas”. A mayor valor de ese score general, teniendo en cuenta que sumamos clusters que van de 0 a 3 en su gran mayoría, esperamos encontrar una relación positiva más fuerte con el valor absoluto de la Valoración del cliente:

```
[60]: # Calculo Score General y utilizo mean() para ver detalles
dataset['ScoreGeneral'] = dataset['SALDOVISTACluster'] + dataset['CROSSELLCluster'] + dataset['PRESTAMOCluster']
dataset.groupby('ScoreGeneral')['DV_PROM_SALDO_12','CLI_INDICE_CROSS_SELL','PR_MONTO_ORIGINAL', 'VALOR_PROPIO_PROM_12'].mea
```

[60]:

ScoreGeneral	DV_PROM_SALDO_12	CLI_INDICE_CROSS_SELL	PR_MONTO_ORIGINAL	VALOR_PROPIO_PROM_12
0	1462.72	1.00	178.01	23.82
1	4898.65	2.28	3867.52	135.35
2	12155.72	3.48	33289.91	391.81
3	13547.48	4.31	89862.21	601.42
4	25088.16	4.63	215391.65	1012.11
5	68480.96	4.90	312127.11	1006.84

Efectivamente encontramos que la media del Valor_Promo_Prom_12 aumenta a medida que el “score general” de cliente también lo hace.

A continuación decidimos agrupar a los clientes por grupos de score, para luego visualizar gráficamente la relación descripta:

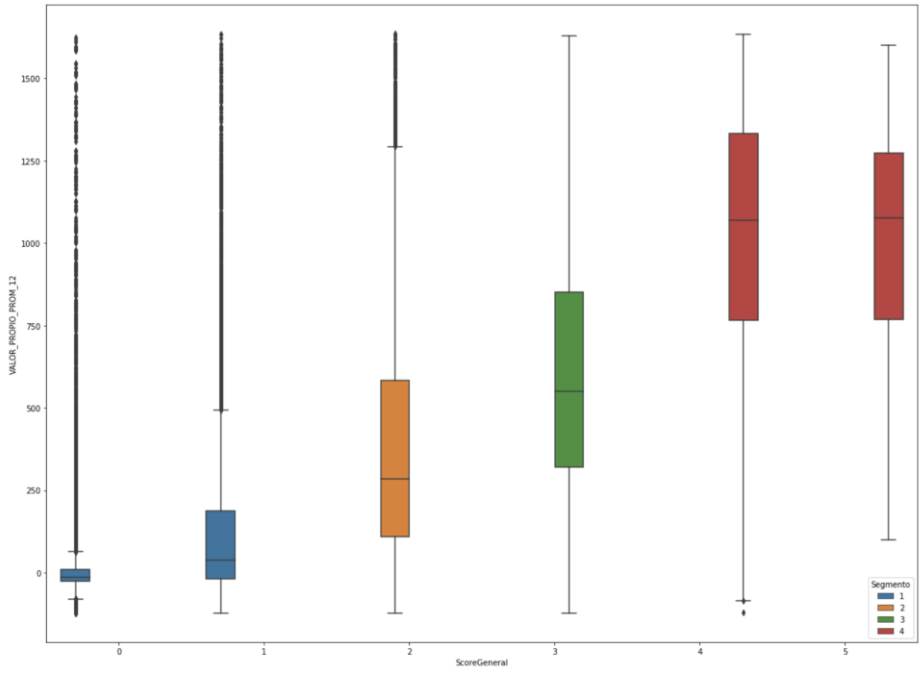
```
[62]: # Agrupo los clientes en conjuntos de score general de 0 a 5, siendo 1 el grupo menos rentable y 4 el más rentable:
dataset['Segmento'] = 1
dataset.loc[dataset['ScoreGeneral']>=2,'Segmento'] = 2
dataset.loc[dataset['ScoreGeneral']>=3,'Segmento'] = 3
dataset.loc[dataset['ScoreGeneral']>=4,'Segmento'] = 4
```

```
[63]: dataset.groupby('Segmento')['VALOR_PROPIO_PROM_12'].mean()
```

```
[63]: Segmento
1     67.50
2    391.81
3    601.42
4   1011.88
Name: VALOR_PROPIO_PROM_12, dtype: float64
```

La correlación positiva es visualmente clara. Cuanto mayor es el grupo de score general (eje X), mayor es la valoración promedio del cliente (eje Y):

```
Out[64]: <matplotlib.axes._subplots.AxesSubplot at 0x7f97accfb828>
```



6.4. Desarrollo de Modelo de Predicción de Clusters de Valor del Cliente

Antes de construir el modelo de aprendizaje automático, necesitamos identificar cuál es el tipo de problema de predicción a resolver. LTV en sí es un problema de regresión. Un modelo de aprendizaje automático puede predecir el valor monetario en pesos del LTV del cliente. Pero aquí, identificamos segmentos de LTV. Porque hace que sea más práctico y fácil identificar estos segmentos para luego accionar sobre los mismos. Al aplicar la agrupación de K-means previamente, pudimos identificar nuestros grupos de LTV existentes y construir segmentos encima de ellos.

Teniendo en cuenta el negocio potencial subyacente dentro de cada grupo de clientes, debemos tratar a los clientes de manera diferente según su LTV estimado. En este caso, aplicamos la agrupación en clústeres y tendremos 4 segmentos (el número de segmentos siempre depende de la dinámica y los objetivos del negocio). En este caso, buscamos predecir para luego accionar diferentemente sobre 4 grupos de clientes: LTV Muy Bajo, Bajo, Medio y Alto.

Habiendo definido la variable a predecir cómo el Grupo o Segmento de Valoración del Cliente (Valor12Cluster) y desarrollado un feature nuevo (Score General) presentado previamente, volvemos a calcular las correlaciones entre las variables del dataset pero ahora respecto del cluster de valor, y encontramos naturalmente una alta correlación contra su variable continua (Valor_Propio_Prom_12, Valor_Propio_Prom_6, Valor_Propio_Prom_3), que serán descartadas del dataset antes de modelar el algoritmo de predicción, y también encontramos un nivel importante de correlación con las nuevas variables creadas (Score General y Segmento), que entendemos serán útiles para nuestro futuro modelo:

```
[63]: # Convierto columnas categóricas a numéricas
      dataset = pd.get_dummies(dataset)
      # Calculo y muestro correlaciones
      corr_matrix = dataset.corr()
      corr_matrix['VALOR12Cluster'].sort_values(ascending=False)
```

```
[63]: VALOR12Cluster                                          1.00
      VALOR_PROPIO_PROM_12                                  0.78
      VALOR_PROPIO_PROM_6                                   0.65
      ScoreGeneral                                          0.57
      Segmento                                              0.54
      VALOR_PROPIO_PROM_3                                   0.50
      CLI_INDICE_CROSS_SELL                                 0.49
      CROSSELLCluster                                       0.47
      FLAG_CC                                               0.43
      PRESTAMOCluster                                       0.43
      FLAG_PAQUETE_BI                                       0.42
      PR_MONTO_ORIGINAL                                     0.41
      CLI_FLAG_SITUACION_BANCO_1_3                          0.41
      CLI_POSICION_ACTIVA                                   0.40
      CLI_SALDO_DEUDA                                       0.40
      PR_MESES_CON_PRESTAMOS                                0.40
      PR_MESES_CON_PERSONALES                               0.40
      CLI_SITUACION_BCRA_1. En situaci_n y cumplimiento normal  0.40
      PR_ANTIGUEDAD_PRESTAMOS                               0.39
      PR_PERSONALES                                         0.37
      FLAG_CUENTA_TC                                        0.35
      DV_PROM_SALDO_12                                      0.32
      FLAG_SEGURO                                           0.31
      SALDOVISTACluster                                     0.31
      VALOR_PROPIO_PROM_1                                   0.28
      DV_PROM_SALDO_6                                       0.28
      CLI_CATEGORIA_Individuos                              0.28
```

1- Modelo Base

Partimos de uno de los modelos supervisados de ensamble de árboles más utilizados en la actualidad que se basa en el principio de boosting: XGBoost. El mismo consiste en generar múltiples modelos de predicción “débiles” en forma secuencial, tomando cada uno el resultado del modelo anterior para generar un modelo más “fuerte” en términos de capacidad de predicción.

Para desarrollar el modelo partimos del dataset, definiendo la variable explicada Y como Valor12Cluster y las variables explicativas o features como el resto de las variables del dataset, excluyendo aquellas que generan autocorrelación, es decir las variables continuas de Valoración del cliente. Dividimos el dataset resultado en 95% para train o entrenamiento y 5% para test o validación:

```
[64]: # Creo X e y, X será el conjunto de features e y es la etiqueta Cluster LTV
X = dataset.drop(['VALOR12Cluster','VALOR_PROPIO_PROM_12', 'VALOR_PROPIO_PROM_3','VALOR_PROPIO_PROM_6', 'VALOR_PROPIO_PROM_
y = dataset['VALOR12Cluster']

# Hago split en training y test sets
import xgboost as xgb
from sklearn.model_selection import KFold, cross_val_score, train_test_split
from sklearn.metrics import classification_report,confusion_matrix
import xgboost as xgb
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.05, random_state=56)
```

Luego entrenamos el modelo y calculamos las métricas de evaluación del mismo:

```
[65]: #XGBoost Multiclassification Model
ltv_xgb_model = xgb.XGBClassifier(max_depth=2, learning_rate=0.1,objective= 'multi:softprob',n_jobs=-1).fit(X_train, y_trai

print('Accuracy of XGB classifier on training set: {:.2f}'
      .format(ltv_xgb_model.score(X_train, y_train)))
print('Accuracy of XGB classifier on test set: {:.2f}'
      .format(ltv_xgb_model.score(X_test[X_train.columns], y_test)))

y_pred = ltv_xgb_model.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
Accuracy of XGB classifier on training set: 0.82
Accuracy of XGB classifier on test set: 0.81
              precision    recall  f1-score   support

           0       0.90      0.94      0.92      1291
           1       0.54      0.53      0.54       172
           2       0.64      0.53      0.58       371
           3       0.57      0.62      0.59        97

    accuracy                           0.81      1931
   macro avg       0.66      0.66      0.66      1931
weighted avg       0.80      0.81      0.81      1931
```

Para evaluar correctamente un modelo de predicción multiclase con clases desbalanceadas como es el de este caso (segmentos de valor del cliente), nos interesa no sólo obtener un buen resultado en el Accuracy (exactitud), es decir el porcentaje de casos en que el modelo ha acertado (verdaderos positivos + verdaderos negativos / (verdaderos positivos + verdaderos negativos + falsos positivos + falsos negativos)), sino fundamentalmente obtener un buen rendimiento respecto de las métricas Precision (precisión), definida como verdaderos positivos / (verdaderos positivos + falsos positivos), y Recall (exhaustividad), definida como verdaderos positivos / (verdaderos positivos + falsos negativos). La métrica Precision nos informará cuán bueno es el modelo en identificar la clase correcta, mientras que la métrica Recall nos ayudará a entender cuánto identifica de la clase correcta.

En este primer modelo encontramos que el Accuracy del modelo es de 0.82 para el dataset de entrenamiento y de 0.81 para el de validación. La precisión y el recall son aceptables para el Cluster 0. Como ejemplo, para el grupo 0 (LTV muy bajo), si el modelo nos dice que este cliente pertenece al grupo 0, 90 de 100 serán correctos (precisión). Y el modelo identifica con éxito el 94% de los clientes reales del clúster 0 (recall). Sin embargo, necesitamos mejorar el modelo para otros grupos. Por ejemplo, apenas detectamos el 52% de los clientes de Cluster 1 (LTV medio-bajo).

Para mejorar el rendimiento del modelo, encontramos oportunidades en los siguientes cursos de acción:

a) Reducir el problema de clasificación a 3 clusters.

b) Probar otras combinaciones de split para train y test.
c) Aplicar ajuste de hiperparámetros al modelo actual.
d) Utilizar una técnica para mejorar la predicción de clases desbalanceadas.
e) Probar algún modelo adicional al XGBoost.

2- Modelo para 3 clases

Realizamos una simulación con 3 clusters de Valoración en lugar de 4 para entender si mejoran sustancialmente las métricas de Precision y Recall:

[71]:
```
# Realizo una prueba con 3 Clusters de  Valoración (Alto, Medio y  Bajo)
km = KMeans(n_clusters=3)
y=dataset['VALOR_PROPIO_PROM_12']
y=y.values.reshape(-1,1)
km.fit(y)
dataset['VALOR12Cluster3'] = km.predict(y)
```

[73]:
```
# Creo un nuevo dataset para analizar cómo agrupó los valores el cluster
dataset_cluster = dataset.copy()
dataset_cluster = order_cluster('VALOR12Cluster3', 'VALOR_PROPIO_PROM_12',dataset_cluster ,True)
dataset_cluster.groupby('VALOR12Cluster3')['VALOR_PROPIO_PROM_12'].describe()
```

[73]:

VALOR12Cluster3	count	mean	std	min	25%	50%	75%	max
0	28716.00	14.52	75.71	-122.60	-26.94	-8.82	43.72	236.93
1	6762.00	457.58	156.18	236.99	324.34	429.95	579.67	793.23
2	3141.00	1125.84	234.34	793.38	918.57	1091.41	1306.36	1635.07

[75]:
```
y2 = dataset['VALOR12Cluster3']
```

[76]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y2, test_size=0.05, random_state=56)
```

Entrenamos nuevamente el modelo para predecir las 3 clases de Valor del cliente y evaluamos el rendimiento del mismo:

[77]:
```
#XGBoost Multiclassification Model
ltv_xgb_model = xgb.XGBClassifier(max_depth=2, learning_rate=0.1,objective= 'multi:softprob',n_jobs=-1).fit(X_train, y_trai

print('Accuracy of XGB classifier on training set: {:.2f}'
      .format(ltv_xgb_model.score(X_train, y_train)))
print('Accuracy of XGB classifier on test set: {:.2f}'
      .format(ltv_xgb_model.score(X_test[X_train.columns], y_test)))

y_pred = ltv_xgb_model.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
Accuracy of XGB classifier on training set: 0.87
Accuracy of XGB classifier on test set: 0.87
              precision    recall  f1-score   support

           0       0.93      0.94      0.94      1440
           1       0.65      0.70      0.68       155
           2       0.66      0.60      0.63       336

    accuracy                           0.87      1931
   macro avg       0.75      0.75      0.75      1931
weighted avg       0.86      0.87      0.86      1931
```

Observamos que al reducir a 3 clusters el problema de predicción multiclase eleva de base el Accuracy de 0.81 a 0.87, como así también los Precision y Recall de cada clase con respecto al modelo anterior. Sin embargo, intentaremos mejorar el modelo manteniendo la misma cantidad de clases a predecir que la cantidad original de clusters.

3- Combinación diferente de dataset de entrenamiento y validación

Volvemos a entrenar el modelo base pero ahora con una distribución de 70% de dataset para entrenamiento y 30% para validación, con la expectativa de reducir el overfitting y mejorar las métricas en evaluación.

[441]:
```
# Pruebo otras combinaciones de split y train
import xgboost as xgb
from sklearn.model_selection import KFold, cross_val_score, train_test_split
from sklearn.metrics import classification_report,confusion_matrix
import xgboost as xgb
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=56)
```

[442]:
```
#XGBoost Multiclassification Model
ltv_xgb_model = xgb.XGBClassifier(max_depth=2, learning_rate=0.1,objective= 'multi:softprob',n_jobs=-1).fit(X_train, y_trai
```

[443]:
```
# Calculo los indicadores provenientes de la matriz de confusión de cada clase
print('Accuracy of XGB classifier on training set: {:.2f}'
      .format(ltv_xgb_model.score(X_train, y_train)))
print('Accuracy of XGB classifier on test set: {:.2f}'
      .format(ltv_xgb_model.score(X_test[X_train.columns], y_test)))

y_pred = ltv_xgb_model.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
Accuracy of XGB classifier on training set: 0.82
Accuracy of XGB classifier on test set: 0.81
              precision    recall  f1-score   support

           0       0.90      0.95      0.92      7797
           1       0.52      0.48      0.50      1063
           2       0.64      0.55      0.59      2108
           3       0.60      0.61      0.61       618

    accuracy                           0.81     11586
   macro avg       0.67      0.65      0.66     11586
weighted avg       0.80      0.81      0.81     11586
```

Nuevamente nos encontramos en niveles similares de Accuracy que el modelo base, con una leve mejoría en el F1-score (métrica que resume el Precision y el Recall) para las clases más desbalanceadas.

4- Ajuste de hiperparámetros

La literatura y experiencia en ciencia de datos recomienda ajustar o “tunear” los hiperparámetros del algoritmo de predicción. Un mismo tipo de modelo de aprendizaje automático puede requerir diferentes restricciones, pesos o tasas de aprendizaje para generalizar diferentes patrones de datos. Estas medidas se denominan hiperparámetros y deben ajustarse para que el modelo pueda resolver de manera óptima el problema del aprendizaje automático. La optimización de hiperparámetros encuentra una tupla de hiperparámetros que produce un modelo óptimo que minimiza una función de pérdida predefinida en datos independientes dados.

En este caso de aplicación utilizamos la metodología de Random Search, un algoritmo menos exigente en términos computacionales que el Grid Search, para la búsqueda de los mejores hiperparámetros para el modelo en un entorno gratuito en la nube de Google Colab, que permite utilizar máquinas virtuales más potentes que una computadora local.

Debido a que tuvimos que realizar distintas pruebas e iteraciones a lo largo del trabajo, por un tema de tiempos de procesamiento, optamos por presentar la versión final del notebook en Python que se adjunta como documentación anexa, con una muestra más pequeña del dataset en entorno local.

[446]:
```
# Implemento Random Search sobre XGBoost Multiclassification Model para encontrar hiperparámetros que mejoren el modelo
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
param_grid = {
        'min_child_weight': [1, 5, 10],
        'gamma': [0.5, 1, 1.5, 2, 5],
        'subsample': [0.6, 0.8, 1.0],
        'colsample_bytree': [0.6, 0.8, 1.0],
        'max_depth': [3, 4, 5]
        }
# En Google Colab se simuló con el dataset completo
random_search = RandomizedSearchCV(ltv_xgb_model, param_distributions=param_grid, scoring='f1_macro', n_jobs=4, cv=5, verbo
start_time = timer(None) # variable para calcular tiempo de inicio y fin
random_search.fit(X, y)
timer(start_time)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits

 Time taken: 0 hours 13 minutes and 18.03 seconds.
```

Luego re-entrenamos el modelo con los hiperparámetros optimizados y evaluamos su rendimiento:

[448]:
```
# Entreno nuevamente el modelo con la optimización de hiperparámetros
from numpy import nan
ltv_xgb_model = XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=0.8, gamma=2, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.1, max_delta_step=0, max_depth=3,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=100, n_jobs=-1, num_parallel_tree=1,
              objective='multi:softprob', random_state=0, reg_alpha=0,
              reg_lambda=1, scale_pos_weight=None, subsample=0.8,
              tree_method='exact', validate_parameters=1, verbosity=None).fit(X_train, y_train)
```

[449]:
```
# Calculo los indicadores de precisión de cada clase
print('Accuracy of XGB classifier on training set: {:.2f}'
       .format(ltv_xgb_model.score(X_train, y_train)))
print('Accuracy of XGB classifier on test set: {:.2f}'
       .format(ltv_xgb_model.score(X_test[X_train.columns], y_test)))

y_pred = ltv_xgb_model.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
Accuracy of XGB classifier on training set: 0.83
Accuracy of XGB classifier on test set: 0.82
              precision    recall  f1-score   support

           0       0.91      0.94      0.93      7797
           1       0.54      0.50      0.52      1063
           2       0.65      0.57      0.61      2108
           3       0.60      0.64      0.62       618

    accuracy                           0.82     11586
   macro avg       0.67      0.66      0.67     11586
weighted avg       0.81      0.82      0.82     11586
```

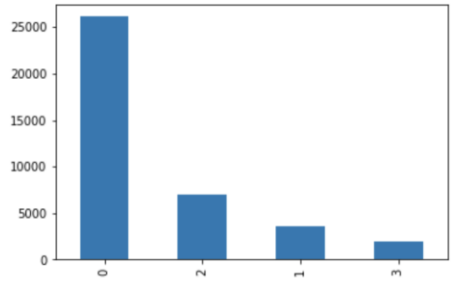
Logramos una pequeña mejora del Accuracy en training y testing y una leve mejoría en el Precision y Recall para las clases más pequeñas.

5- Técnica para clases desbalanceadas

Como observamos previamente, estamos intentando predecir segmentos de valor que se corresponden con un problema típico de se da en llamar “clases desbalanceadas”:

[451]:
```
# Resumidamente, la distribución de las clases de los clusters es la siguiente:
dataset["VALOR12Cluster"].value_counts().plot.bar()
```

[451]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcab5ae7c88>
```



Una manera de contrarrestar las clases desbalanceadas es entrenar el modelo sobre un dataset "sintético", esto es, una variación del dataset original que intenta "agrandar" la cantidad de casos correspondientes a las clases más pequeñas, para que pueda "aprender" más de estos grupos y luego identificarlos mejor.

Para ello utilizamos la librería imblearn y dentro de ella aplicamos la técnica de oversampling al dataset en estudio.

[452]:
```
#Importo librerías para oversamplear el dataset que se utilizan cuando tengo variables categóricas y continuas
from collections import Counter
from imblearn.pipeline import Pipeline
from sklearn.datasets import make_classification
from imblearn.over_sampling import RandomOverSampler
from imblearn.pipeline import Pipeline
# summarize class distribution
print(Counter(y))
```

```
Counter({0: 26124, 2: 6953, 1: 3536, 3: 2006})
```

[465]:
```
oversample = RandomOverSampler(sampling_strategy='minority')
# fit and apply the transform
X_over, y_over = oversample.fit_resample(X, y)
# summarize class distribution
print(Counter(y_over))
```

```
Counter({0: 26124, 3: 26124, 2: 6953, 1: 3536})
```

A continuación, creamos un pipeline para entrenar el modelo sobre el dataset "oversampleado" en su clase más pequeña:

[466]:
```
from numpy import mean
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedStratifiedKFold
# define pipeline
steps = [('over', RandomOverSampler()), ('model', ltv_xgb_model)]
ltv_xgb_modelover = Pipeline(steps=steps)
```

[467]:
```
ltv_xgb_modelover.fit(X_train, y_train)
```

[467]:
```
Pipeline(steps=[('over', RandomOverSampler()),
                ('model',
                 XGBClassifier(base_score=0.5, booster='gbtree',
                               colsample_bylevel=1, colsample_bynode=1,
                               colsample_bytree=0.8, gamma=2, gpu_id=-1,
                               importance_type='gain',
                               interaction_constraints='', learning_rate=0.1,
                               max_delta_step=0, max_depth=3,
                               min_child_weight=1, missing=nan,
                               monotone_constraints='()', n_estimators=100,
                               n_jobs=-1, num_parallel_tree=1,
                               objective='multi:softprob', random_state=0,
                               reg_alpha=0, reg_lambda=1, scale_pos_weight=None,
                               subsample=0.8, tree_method='exact',
                               validate_parameters=1, verbosity=None))])
```

Finalmente, analizamos el rendimiento del modelo resultante:

[468]:
```
# Calculo los indicadores de precisión de cada clase
print('Accuracy of XGB classifier on training set: {:.2f}'
      .format(ltv_xgb_modelover.score(X_train, y_train)))
print('Accuracy of XGB classifier on test set: {:.2f}'
      .format(ltv_xgb_modelover.score(X_test[X_train.columns], y_test)))

y_pred = ltv_xgb_modelover.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
Accuracy of XGB classifier on training set: 0.81
Accuracy of XGB classifier on test set: 0.79
              precision    recall  f1-score   support

           0       0.95      0.87      0.91      7797
           1       0.50      0.50      0.50      1063
           2       0.55      0.67      0.60      2108
           3       0.54      0.75      0.63       618

    accuracy                           0.79     11586
   macro avg       0.64      0.70      0.66     11586
weighted avg       0.82      0.79      0.80     11586
```

No observamos que el balanceo de clases haya logrado un F1-score más balanceado para los 4 clusters de Valoración ni mejoró el F1 global. Intentaremos una optimización de hiperparámetros que incluya pesos para las clases desde el XGBoost (hiperparámetro scale_pos_weight):

[478]:
```
# Implemento Random Search sobre XGBoost Multiclassification Model
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
param_grid = {
        'min_child_weight': [1, 5, 10],
        'gamma': [0.5, 1, 1.5, 2, 5],
        'subsample': [0.6, 0.8, 1.0],
        'colsample_bytree': [0.6, 0.8, 1.0],
        'max_depth': [3, 4, 5],
        'scale_pos_weight': [1, 10, 25, 50, 75, 100, 200, 300, 1000, 5000, 10000, 15000]
        }
random_search = RandomizedSearchCV(ltv_xgb_model, param_distributions=param_grid, scoring='f1_macro', n_jobs=4, cv=5, verbos

start_time = timer(None) # timing starts from this point for "start_time" variable
random_search.fit(X, y)
timer(start_time) # timing ends here for "start_time" variable
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[16:11:53] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:480:
Parameters: { scale_pos_weight } might not be used.

  This may not be accurate due to some parameters are only used in language bindings but
  passed down to XGBoost core.  Or some parameters are not used but slip through this
  verification. Please open an issue if you find above cases.


 Time taken: 1 hours 2 minutes and 27.39 seconds.
```

Luego de la optimización, re-entrenamos el modelo con los hiperparámetros indicados:

[480]:
```
random_search.best_estimator_
```

[480]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=0.8, gamma=1, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.1, max_delta_step=0, max_depth=3,
              min_child_weight=5, missing=nan, monotone_constraints='()',
              n_estimators=100, n_jobs=-1, num_parallel_tree=1,
              objective='multi:softprob', random_state=0, reg_alpha=0,
              reg_lambda=1, scale_pos_weight=100, subsample=0.8,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

[481]:
```
# Entreno nuevamente el modelo con la optimización de hiperparámetros considerando los pesos de los clusters
ltv_xgb_model = XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=0.8, gamma=1, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.1, max_delta_step=0, max_depth=3,
              min_child_weight=5, missing=nan, monotone_constraints='()',
              n_estimators=100, n_jobs=-1, num_parallel_tree=1,
              objective='multi:softprob', random_state=0, reg_alpha=0,
              reg_lambda=1, scale_pos_weight=100, subsample=0.8,
              tree_method='exact', validate_parameters=1, verbosity=None).fit(X_train, y_train)
```

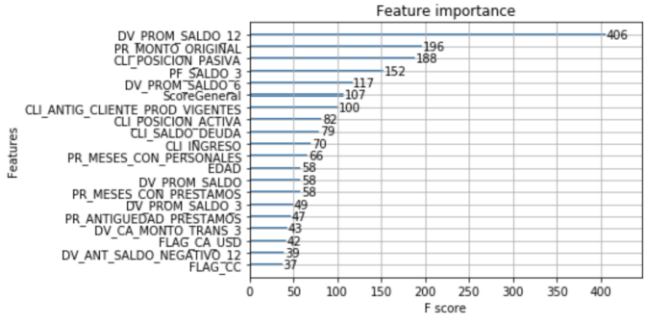
Como resultado, alcanzamos un modelo optimizado para clases desbalanceadas con un F1-score razonable:

[482]:
```
# Calculo los indicadores de precisión de cada clase
print('Accuracy of XGB classifier on training set: {:.2f}'
      .format(ltv_xgb_model.score(X_train, y_train)))
print('Accuracy of XGB classifier on test set: {:.2f}'
      .format(ltv_xgb_model.score(X_test[X_train.columns], y_test)))

y_pred = ltv_xgb_model.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
Accuracy of XGB classifier on training set: 0.83
Accuracy of XGB classifier on test set: 0.82
              precision    recall  f1-score   support

           0       0.91      0.94      0.93      7797
           1       0.54      0.49      0.52      1063
           2       0.64      0.57      0.60      2108
           3       0.60      0.64      0.62       618

    accuracy                           0.82     11586
   macro avg       0.67      0.66      0.67     11586
weighted avg       0.81      0.82      0.81     11586
```

Finalmente, analizamos las variables o features con mayor importancia para predecir el Segmento de Valor del Cliente:

[484]:
```
from xgboost import plot_importance
plot_importance(ltv_xgb_model, max_num_features=20)
plt.show()
```



Encontramos que el modelo XGBoost aprendió de varias variables que identificamos al comienzo del trabajo que tenían alta correlación con la Valoración, como así también del feature creado como Score General.

6- Aplicación de modelo alternativo

Intentamos una mejora más utilizando el algoritmo LightGBM, desarrollado por Microsoft, que incluye un método de muestreo de árboles más eficiente que el XGBoost.

[583]:
```
import lightgbm as lightgbm
# Creo los LightGBM data containers
# categorical_features = [c for c, col in enumerate(data.columns) if 'cat' in col]
import re
X_train = X_train.rename(columns = lambda x:re.sub('[^A-Za-z0-9_]+', '', x))
y_train = y_train.rename(columns = lambda x:re.sub('[^A-Za-z0-9_]+', '', x))
X_test = X_test.rename(columns = lambda x:re.sub('[^A-Za-z0-9_]+', '', x))
y_test = y_test.rename(columns = lambda x:re.sub('[^A-Za-z0-9_]+', '', x))
train_data = lightgbm.Dataset(X_train, label= y_train)
test_data = lightgbm.Dataset(X_test, label=y_test)

# Entreno el modelo
parameters = {
    'objective': 'multiclass',
    'metric': 'multi_logloss',
    'is_unbalance': 'true',
    'boosting': 'gbdt',
    'num_class': 4,
    'num_leaves': 31,
    'feature_fraction': 0.5,
    'bagging_fraction': 0.5,
    'bagging_freq': 20,
    'learning_rate': 0.05,
    'verbose': 0
}

lgbm = lightgbm.train(parameters,
                      train_data,
                      valid_sets=test_data,
                      num_boost_round=5000,
                      early_stopping_rounds=100)
```

Luego del entrenamiento, evaluamos la performance del modelo con los mismos criterios que los modelos anteriores:

[527]:
```
# Calculo los indicadores de precisión de cada clase
print(classification_report(y_test, y_pred, target_names=['0', '1', '2', '3']))
```

```
              precision    recall  f1-score   support

           0       0.92      0.94      0.93      7797
           1       0.53      0.50      0.51      1063
           2       0.64      0.63      0.63      2108
           3       0.62      0.61      0.61       618

    accuracy                           0.82     11586
   macro avg       0.68      0.67      0.67     11586
weighted avg       0.82      0.82      0.82     11586
```

El modelo base de LightGBM pareciera tener un F1-score en las clases desbalanceadas levemente mejor respecto al modelo base de XGBoost.

Procedemos, entonces, a realizar el ajuste de hiperparámetros al nuevo modelo (a los efectos del uso computacional, se prueban 10 puntos de mejora en lugar de 100 que es lo habitual):

[592]:
```
# Realizo la optimización de hiperparámetros para el modelo en cuestión
from scipy.stats import randint as sp_randint
from scipy.stats import uniform as sp_uniform
param_test ={'num_leaves': sp_randint(6, 50),
             'min_child_samples': sp_randint(100, 500),
             'min_child_weight': [1e-5, 1e-3, 1e-2, 1e-1, 1, 1e1, 1e2, 1e3, 1e4],
             'subsample': sp_uniform(loc=0.2, scale=0.8),
             'colsample_bytree': sp_uniform(loc=0.4, scale=0.6),
             'reg_alpha': [0, 1e-1, 1, 2, 5, 7, 10, 50, 100],
             'reg_lambda': [0, 1e-1, 1, 5, 10, 20, 50, 100]}

#Este parámetro define la cantidad de puntos de hiperparámetros a testear (suele probarse en 100 pero se bajó a 10 por perf
n_HP_points_to_test = 10

clf = lgb.LGBMClassifier(max_depth=-1, random_state=314, silent=True, metric='multi_logloss', objective='multi_class', n_jo
gs = RandomizedSearchCV(
    estimator=clf, param_distributions=param_test,
    n_iter=n_HP_points_to_test,
    scoring='f1_macro',
    cv=3,
    refit=True,
    random_state=314,
    verbose=True)
```

[593]:
```
start_time = timer(None) # comienza a contar el tiempo desde este punto
gs.fit(X_train, y_train)
timer(start_time) # termina de contar el tiempo aquí
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits

 Time taken: 0 hours 1 minutes and 25.69 seconds.
```

A continuación ejecutamos el modelo optimizado y evaluamos su performance:

[615]:
```
#Alternativa no utilizada: configuro desde la optimización de HP
#clf_final = lgb.LGBMClassifier(**gs.best_estimator_.get_params())

#Configuro localmente desde valores hardcodeados
clf_final = lgb.LGBMClassifier(**clf.get_params())

#Seteo parámetros óptimos
clf_final.set_params(**opt_parameters)

#Entreno el modelo optimizado
clf_final.fit(X_train, y_train)
```

[627]:
```
# Realizo la predicción con el modelo óptimo entrenado
y_pred = clf_final.predict_proba(X_test)
y_pred = pd.DataFrame(y_pred).apply(lambda x: np.argmax(x), axis=1)
```

[628]:
```
# Calculo los indicadores de precisión de cada clase
print(classification_report(y_test, y_pred, target_names=['0', '1', '2', '3']))
```

```
              precision    recall  f1-score   support

           0       0.92      0.93      0.93      7797
           1       0.54      0.51      0.52      1063
           2       0.63      0.61      0.62      2108
           3       0.60      0.64      0.62       618

    accuracy                           0.82     11586
   macro avg       0.68      0.67      0.67     11586
weighted avg       0.82      0.82      0.82     11586
```

Este modelo tiene un F1-score levemente mejor para la clase 2 que el XGBoost optimizado, con un Weighted Average de 0.82. Al igual que el anterior es un modelo satisfactorio para predecir el segmento del valor del cliente.

7. Posibles extensiones y próximos pasos del trabajo

El trabajo realizado hasta aquí establece un punto de partida para profundizar en, al menos, dos frentes principales el análisis y aplicación del valor del ciclo de vida del cliente en una entidad bancaria. Por una parte, se puede y es recomendable continuar el trabajo de extracción de nuevas variables disponibles en el datamart de la visión 360 del cliente y realizar su respectivo feature engineering con

el objeto de encontrar mejoras en la capacidad de predicción de las variables, que sumadas a la optimización de hiperparámetros, permitan mejorar aún más el *recall* en las clases más desbalanceadas. Asimismo, sería razonable evaluar la inclusión en el dataset a nivel cliente de la evolución de algunas variables macroeconómicas con una desagregación regional y/o sectorial, tales como Producto Bruto Geográfico, Nivel de Actividad, etc. que puedan tener incidencia en la capacidad económica y comportamiento de los clientes en relación con el uso de productos y su transaccionalidad. Además se debería establecer una frecuencia de recalibración de los modelos cada 3 o 6 meses, debido al cambio de las condiciones financieras y *pricing* de los productos bancarios, que podrían afectar el nivel de significatividad de predicción de las variables actuales sobre el valor del cliente en el tiempo.

Por otra parte, entendemos que es fundamental la capacidad de automatizar el proceso de clasificación de clientes en forma mensual, debido a la variabilidad de tenencias de productos, saldos y operaciones que los mismos realizan a lo largo del año y porque ello puede implicar detectar cambios discretos en las clases de valor predichas para un cliente y, así, ayudar por ejemplo a prevenir posibles fugas o *attrition* de productos de los mismos. Para ello sería recomendable "productivizar" la solución por alguno de los siguientes caminos:

1) *Entorno híbrido:* si se desea correr el proceso de predicción en la nube, se podría crear un ETL desde el datamart local que genere el dataset mensual de clientes con los features finales utilizados por el modelo, disponibilizando el mismo mediante una carga a un Object Storage del tipo AWS S3 en un directorio que indique el número de mes de los para el procesamiento del modelo predictivo. El output de este proceso sería un nuevo archivo con la clase de valor predicha para cada cliente con una secuencia numérica determinada para cada mes para que otro ETL del datamart de clientes lo tome como input y alimente una tabla en el mismo datamart local con el valor predicho de cliente para el período correspondiente, y se pueda así aprovechar su uso posteriormente en las segmentaciones y análisis del quehacer diario de la entidad.
2) *Entorno local:* se puede embeber el modelo desarrollado en Python mediante un pickle que haya almacenado el mismo para que pueda ser invocado en el ETL correspondiente mediante un procedimiento almacenado y así correr la predicción para cada cliente mensualmente en la misma base de datos, actualizando una tabla final en el datamart que acumule mensualmente la clasificación del valor de cada cliente.

Por último, otro eje de acción recomendado es complementar el resultado de la clasificación anterior con el desarrollo de un modelo de predicción de churn para ajustar el valor del cliente en el tiempo y poder realizar el cálculo completo del CLTV como se describió en el enfoque teórico-conceptual al comienzo del presente trabajo.

8. Conclusiones

A lo largo del presente trabajo hemos podido revisar la literatura más reciente sobre la teoría y aplicación del concepto y de las metodologías asociadas al CLTV. Desde el punto de vista conceptual, encontramos que el enfoque adecuado para calcular el valor del ciclo de vida del cliente para un banco implica darle un tratamiento de tipo de relación contractual con el cliente, y respecto de la oportunidad de sus transacciones, para algunos productos es discreta (ej. póliza de seguros) mientras que para otros es continua (ej: tarjeta de crédito). Desde el punto de vista de aplicación, esto último nos orientó a optar por modelos de agrupamiento y clasificación, en lugar de los modelos estadísticos del tipo BTYD (*Buy Till You Die*), más específicos de los clientes no contractuales.

Una vez abordado el problema conceptual y definido el enfoque de las herramientas de aprendizaje automático a utilizar, desarrollamos el proceso de adquisición, ingesta y manipulación de datos a través de consultas en SQL a una base de datos de tipo relacional de la empresa, que luego de la anonimización y muestreo del universo correspondiente, nos permitió estructurar un "tablón" o *ABT* de clientes con gran parte de la información deseada para realizar el análisis y modelización posterior.

A través del análisis exploratorio de datos realizado en lenguaje Python, desde un equipo local y también en un servidor de nube gratuito bajo un entorno de desarrollo integrado (IDE) de Jupyter Notebooks, logramos describir los valores estadísticos básicos de las distintas variables de comportamiento y sociodemográficas asociadas a cada cliente, como así también su correlación con la Valoración histórica del cliente calculada por el sistema interno de rentabilidad del banco.

El análisis realizado nos permitió encontrar correlaciones relevantes entre variables que preveíamos (tenencias de productos, saldo de depósitos, etc.) como así también entre otras que no eran tan claras a partir de los datos (ingresos, edad, etc.) y construir un *feature* que denominamos "Score General" que tuvo relevancia a la hora de construir el modelo predictivo. Con respecto a esto último, realizamos un trabajo en dos etapas. Primero agrupamos a los clientes en grupos de Valoración histórica y validamos que existía una correlación relevante entre dichos grupos y las principales variables de interés. Luego, definimos la etiqueta o *label* a predecir como el grupo de Valoración o LTV al que pertenecía el cliente en función de su rentabilidad histórica (Muy Bajo, Bajo, Medio y Alto).

A los efectos de probar la hipótesis del trabajo, desarrollamos iteraciones de modelos supervisados de clasificación con el algoritmo XGBoost y luego con el LightGBM. En las distintas pruebas nos encontramos con el típico caso de clases desbalanceadas que debían ser predichas por el algoritmo de predicción multiclase. Luego de sucesivas mejoras mediante técnicas de optimización de hiperparámetros y balanceo de clases, logramos desarrollar 2 modelos, uno con cada algoritmo mencionado, que nos permitió clasificar con un accuracy de entre 0.82. y 0.83 y un F1-score razonable los grupos de valor del cliente.

De esta forma concluimos el trabajo, aceptando la hipótesis de que es posible desarrollar un modelo predictivo que clasifique con un grado de confianza razonable el grupo de valor al que pertenecerá un cliente en los próximos 12 meses, a partir de sus variables demográficas y de comportamiento con el banco.

9. Referencias bibliográficas

1. Blattberg, R., Deighton, J. 1996. Manage Marketing by the Customer Equity Test. En Harvard Business Review – July-August 1996 Issue [en línea]. https://hbr.org/1996/07/manage-marketing-by-the-customer-equity-test

2. Borle, S., Sing, S., Jain, D. 2008. Customer Lifetime Value Measurement. En Management Science, Vol. 54, No. 1, pp. 100-112.

3. Chamberlain, B., Cardoso, A., Liu, B., Pagliari, R., Deisenroth, M. 2017. Customer Lifetime Value Prediction Using Embeddings. En Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '17). Nueva York: DOI: https://doi.org/10.1145/3097983.3098123. Pag: 1753-1762.

4. Colombo, R., Jiang, W. 1999. A stochastic RFM model. En Journal of Interactive Marketing, Vol. 13, Issue 3, 1999, pp. 2-12. https://doi.org/10.1002/(SICI)1520-6653(199922)13:3%3C2::AID-DIR1%3E3.0.CO;2-H

5. Cui, D., Curry, D. 2005. Prediction in marketing using Support Vector Machine. En Marketing Science, Vol. 24, No. 4. https://doi.org/10.1287/mksc.1050.0123

6. Fader, P., Hardie, B., Lok, K. 2005. RFM and CLV: Using Iso-Value Curves for Customer Base Analysis. En Journal of Marketing Research, 42(4), pp. 415–430. https://doi.org/10.1509/jmkr.2005.42.4.415

7. Fader, P., Hardie, B., Lok, K. 2006. More than meet the eye. En Marketing Research – Summer 2006. Wharton, University of Pennsylvania [en línea]. https://faculty.wharton.upenn.edu/wp-content/uploads/2013/08/fader_et_al_mr_06.pdf

8. Friedman, J. 1991. Multivariate Adaptive Regression Splines. Ann. Statist. 19 (1991), no. 1, 1--67. https://doi:10.1214/aos/1176347963

9. Glady, N., Baesens, B., Croux, C. 2006. Modeling Churn Using Customer Lifetime Value. K.U. Leuven KBI Working Paper No. 0618.

10. Gupta, S., Hanssens, D., Hardie, B., Kahn, W., Kumar, V., Lin, N., Ravishanker, N., Sriram, S. 2006. Modeling Customer Lifetime Value. En Journal of Service Research 2006; 9; 139. DOI: 10.1177/1094670506293810

11. Harman, D. 2016. Stochastic process customer lifetime value models with time-varying covariates. University of Iowa - Fall 2016.