Cajamar Salesforce Predictive Modelling

Data Ninjas

Claudia Lucio Sarsa, Antonio Martínez Payá, Juan Carlos Pereira Kohatsu

14 de marzo de 2017

1 Introducción.

El objetivo de este reto es dar con un modelo que sea capaz de predecir el poder adquisitivo de los clientes de la entidad financiera a partir de los datos anónimos proporcionados por ésta misma. Saber el poder adquisitivo de un cliente es fundamental, pues permite a la entidad financiera crear segmentaciones para ofrecer productos más adecuados. A parte de realizar predicciones, se pretende que mediante el uso del análisis exploratorio de los datos y del modelo, seamos capaces de entender el modelo de negocio.

Para cumplir nuestro objetivo, primeramente realizamos un análisis exploratorio de los datos tratando de comprenderlos y realizando suposiciones lógicas sobre qué representan. Gracias a este análisis podemos obtener una visión general del negocio y sus clientes, así como ser capaces de filtrar y transformar las variables de forma que sean más representativas para nuestro problema. Posteriormente, se justificará la elección de la métrica implicando esto la selección del modelo y de parámetros. Finalmente, estudiaremos cómo el modelo puede ayudarnos a comprender el negocio además de su viabilidad de puesta en producción.

2 Análisis exploratorio de los datos.

Nuestro conjunto de datos de entrenamiento tiene 363.834 instancias cada una evaluada sobre 89 variables. Además, aproximadamente dos tercios de la matriz son ceros, algo que obviamente debe tenerse muy en cuenta. A este tipo de datasets se les denomina con el termino de *sparse* dataset. La entidad financiera ha agrupado las variables del cliente según estos criterios: identificación, variables sociodemográficas, importe de consumos habituales, importe de los saldos de los productos financieros, tenencia de los productos financieros, número de operaciones realizadas en los productos financieros y poder adquisitivo.

2.1 Identificación (ID_Customer).

Representa el identificador del cliente dentro del conjunto de datos. Esta variable no aporta ninguna información a la hora de predecir el poder adquisitivo, por lo tanto queda automáticamente eliminada de nuestro conjunto de datos.

2.2 Variables sociodemográficas.

Disponemos de cinco variables sociodemográficas que aparentemente son muy diferentes entre sí. Vamos a estudiar cada una de ellas.

2.2.1 Socio_Demo_01.

Es una variable cualitativa con 509 categorías. Dada la codificación que tenía, intuímos que podía representar la ocupación del cliente. Al ser tantas categorías, sospechamos que una primera categorización agrupando por los dos primeros dígitos podría ser adecuada. Por tanto, para sustentarnos en algo más sólido, realizamos un boxplot con respecto al logaritmo del poder adquisitivo y otro con respecto a la variable Socio_Demo_03, la cual pensamos que puede representar la edad.

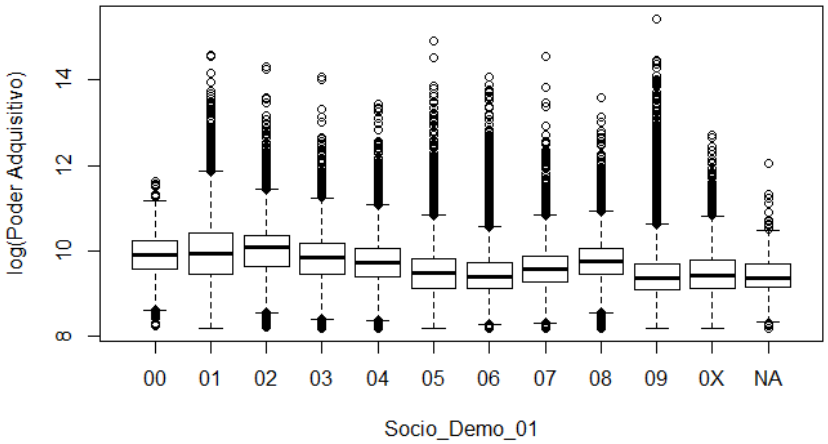

Fig. 1: Logaritmo del poder adquisitivo frente a Socio_Demo_01 categorizada.

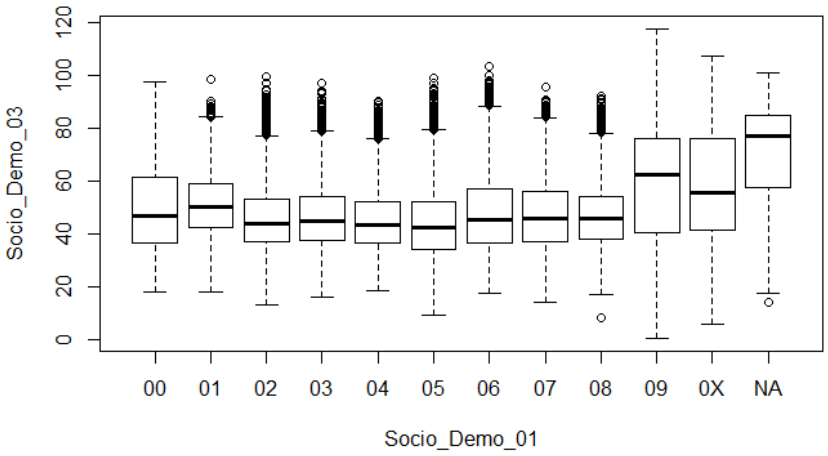

Fig. 2: Socio_Demo_03 frente a Socio_Demo_01 categorizada.

Los resultados obtenidos, especialmente al compararla con la supuesta edad, sugerían que estas variables guardaban una relación ya que la mediana de edad era distinta en cada grupo. Otra interpretación que creímos posible fue que se tratase de una variable de tipo geográfica, pero esto quedó descartado debido a que si fuera así, existirían poblaciones (categorías 03 y 09) donde la

mediana de edad tendría una diferencia de unos 20 años, lo cual no tiene sentido.

Anticipamos que los algoritmos utilizados dan mucha importancia a esta variable a la hora de predecir. Posteriormente se explicará el proceso de agrupar en nuevas categorías las 509 originales.

2.2.2 Socio_Demo_02.

Es una variable categórica que toma dos valores. Creemos que podría tratarse del sexo del cliente.

2.2.3 Socio_Demo_03.

Es una variable continua. Creemos que representa la edad del cliente. Hay instancias que tienen números pequeños como 10'0 o 0'35 que pueden ser justificados como perfiles creados a menores de edad o bebés recién nacidos. Por otro lado, también encontramos el caso contrario, cifras mayores de 110, lo cual puede ser debido a personas ya fallecidas cuyo perfil aún no ha sido eliminado. Por ejemplo, en el caso de los bancos, éstos están obligados a mantener una cuenta bancaria abierta hasta que pasen 20 años de la inactividad de la misma.

2.2.4 Socio_Demo_04.

Es una variable numérica discreta, según la información proporcionada por los organizadores del concurso. Creemos que su comportamiento es muy extraño, veamos como se distribuyen los siete valores más grandes de la variable dentro del dataset.

Valor	31	33	46	70	151	414	7097
Frecuencia	1	20	20	42	6	14	308

Tabla 1: Valores más grandes de Socio_Demo_04 con sus frecuencias.

Creemos que no es lógico que una variable numérica pase por ejemplo, de tomar el valor 414 a directamente tomar el valor 7097. Por su raro comportamiento hemos decidido eliminarla de los datos.

2.2.5 Socio_Demo_05.

Es una variable numérica discreta. Por su rango pensamos que puede ser la antigüedad del perfil del cliente, a pesar de que ésta se mida de forma discreta y lo que creemos que es la edad (Socio_Demo_03) se mida de manera continua. Suponiendo que esto sea cierto, hemos comprobado que sólo el 0'75% de los clientes tienen una supuesta antigüedad mayor que su supuesta edad, lo cual podría ser explicado con que estas cuentas hayan sido heredadas.

2.3 Imp_Cons_01 - Imp_Cons_17.

Estas variables continuas representan los importes de consumos habituales del cliente en base a sus operaciones con tarjetas y domiciliaciones más comunes. Al tratarse de importes y ser datos del

territorio español, creemos que están medidos en euros. Observando la variable suma de todas ellas por cliente, la mediana es 742'5€ y la media es 995'4€, lo que induce a pensar que sean medias trimestrales.

Hemos observado que cada uno de estos importes en media son valores bajos, a excepción de Imp_Cons_03, Imp_Cons_11, Imp_Cons_12 y Imp_Cons_15, que superan los 100€. En la siguiente sección se verá que algunas de estas variables son importantes a la hora de predecir el poder adquisitivo.

Como se observa en el corrplot, no hemos encontrado correlaciones entre ellas, lo cual nos dificulta la labor de reducción de variables. Por ejemplo, para llevar acabo una reducción de variables vía Principal Component Analysis (PCA), se necesitarían 13 componentes de 17 para obtener una varianza explicada acumulada del 75%.

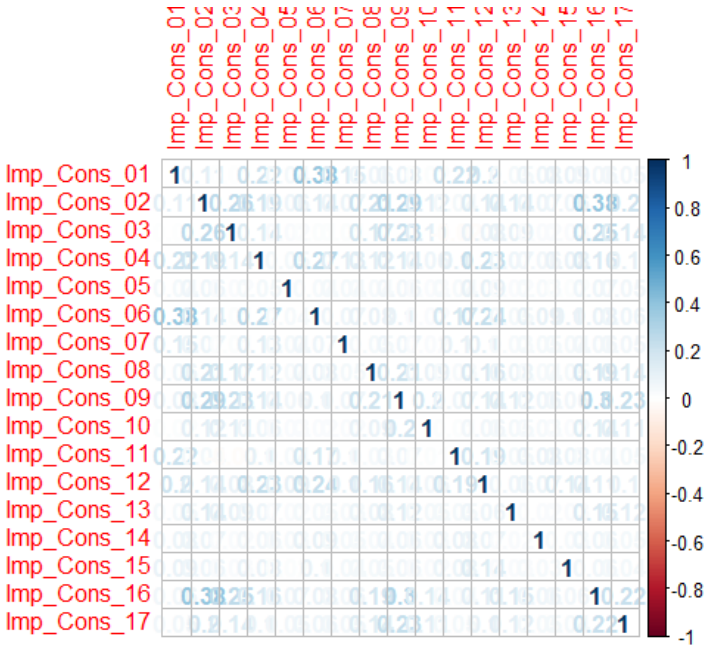

Fig. 3: Correlaciones entre las variables Imp_Cons_01 a Imp_Cons_17.

2.4 Imp_Sal_01 - Imp_Sal_21.

Estas variables continuas representan los importes de los saldos de 21 productos financieros. Al tratarse de importes y ser datos del territorio español, creemos que están medidos en euros. Primeramente, la variable Imp_Sal_14 fue eliminada ya que contenía únicamente ceros. Un primer análisis fue buscar correlaciones en los saldos.

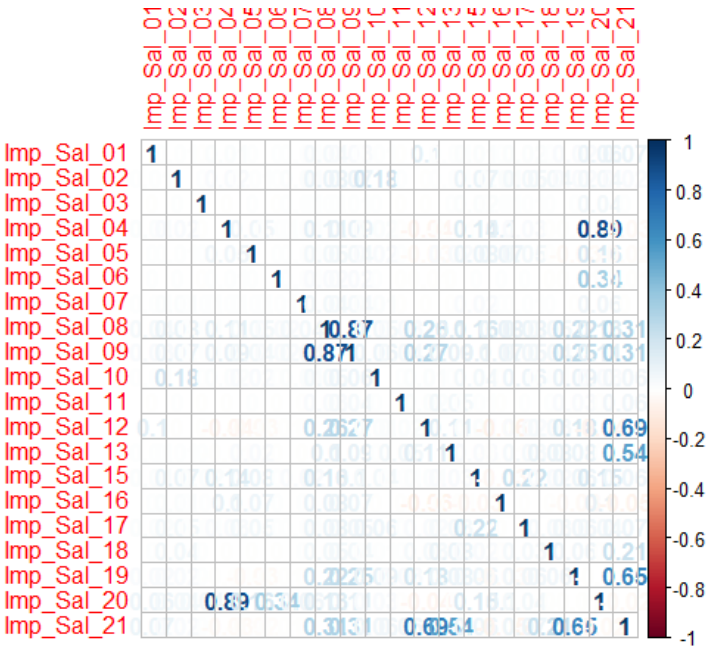

Fig. 4: Correlaciones entre las variables Imp_Sal_01 a Imp_Sal_21.

Como encontramos correlación entre algunas variables, intentamos aplicar Principal Component Analysis para reducir el número de variables, pero para conseguir un 75% de varianza explicada acumulada necesitábamos 15 componentes y más aún, aplicando PCA únicamente a las variables para las cuales existen correlaciones altas (Imp_Sal_04, Imp_Sal_08, Imp_Sal_09, Imp_Sal_12, Imp_Sal_19, Imp_Sal_20 e Imp_Sal_21), obtenemos que para conseguir un 75% de varianza explicada acumulada necesitaríamos cinco componentes de seis.

Estudiando más en profundidad las variables correladas, nos dimos cuenta de que por un lado las variables Imp_Sal_08 e Imp_Sal_09 tenían un 40% de valores "repetidos", entendiendo esto como que el 40% de los valores de Imp_Sal_09 estaban entre $0'9 * \text{Imp\_Sal\_08}$ y $1'1 * \text{Imp\_Sal\_08}$. Esta misma situación ocurría con el 26% de las variables Imp_Sal_19 e Imp_Sal_21. Comparando las funciones de densidad de cada par se observan que tienen un comportamiento muy similar.

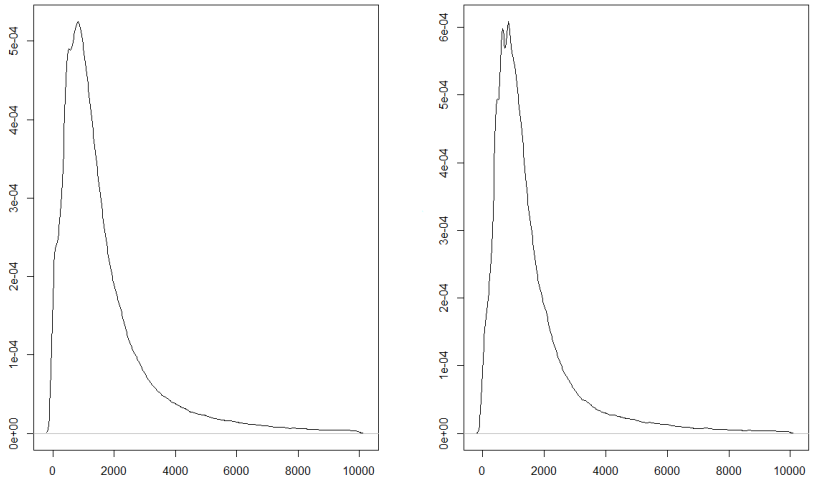

Fig. 5: Funciones de densidad de las variables Imp_Sal_08 e Imp_Sal_09.

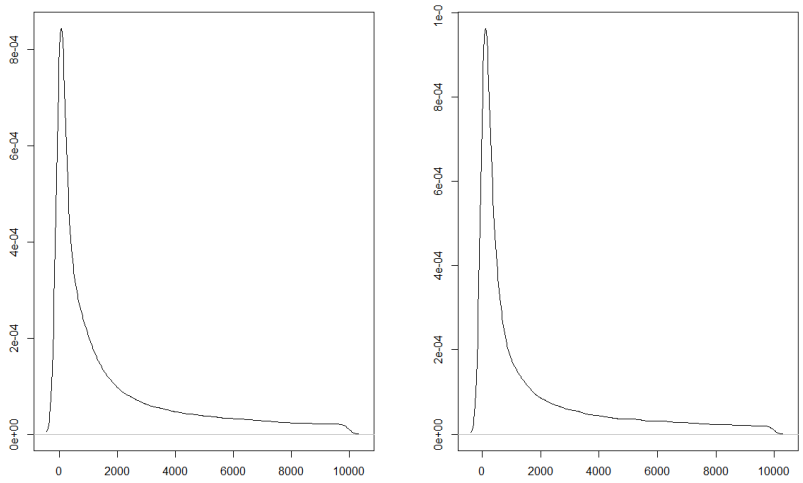

Fig. 6: Funciones de densidad de las variables Imp_Sal_19 e Imp_Sal_21.

Comentar por último que las variables Imp_Sal_04, Imp_Sal_05 e Imp_Sal_19 tienen valores negativos, no superando éstos en ningún caso el 3% de los datos.

2.5 Ind_Prod_01 - Ind_Prod_24.

Estas variables categóricas representan la tenencia de los distintos productos financieros. Toman tres valores: 1 significa que tiene el producto, 0 que no lo tiene y 2 que lo ha tenido en los últimos 3 meses. Hemos decidido transformar los 2 en ceros ya que cada una de las variables en media sólo posee un 3% de 2. Además, de esta forma estamos expresando realmente si el cliente tiene el producto o no.

Al tratarse de una variable cualitativa, decidimos estudiar la dependencia entre pares de variables. Dicha dependencia fue estudiada de dos formas. La primera fue usando la prueba chi cuadrado de independencia entre variables cualitativas. Se creó un grafo que representaba la dependencia entre las variables y se comparó con el grafo obtenido de la segunda forma: una red bayesiana que usa el algoritmo *hill climbing*. Los resultados obtenidos fueron muy parecidos.

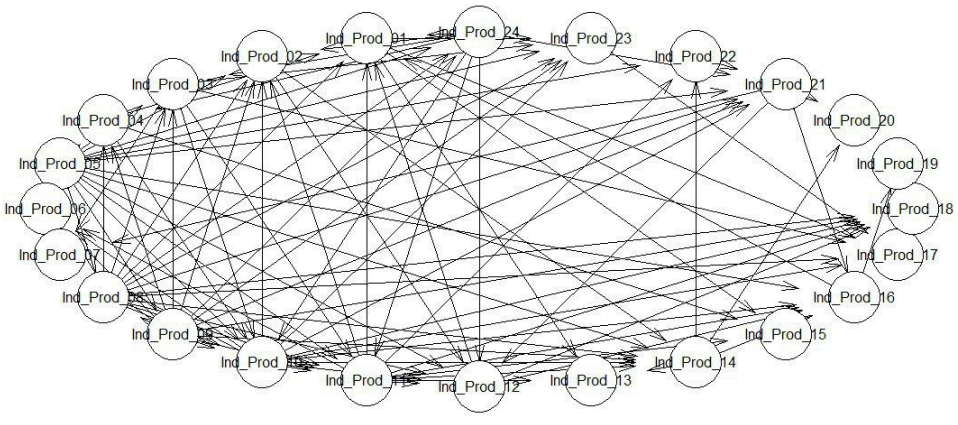

Fig. 7: Red bayesiana con *hill climbing*.

Es de destacar que existen muchas dependencias entre las variables y que, por ejemplo, desde Ind_Prod_17 hasta Ind_Prod_20 no dependen de ninguna otra. Posteriormente, combinado con el uso de los algoritmos utilizados para predecir se realizará una reducción de estas variables.

2.6 Num_Oper_01 - Num_Oper_20.

Estas variables numéricas discretas representan el número de operaciones a través de los distintos productos financieros. Pensamos que estas operaciones han sido realizadas por el propio cliente y no por la entidad financiera. Primeramente buscamos correlaciones entre las variables con un corrplot.

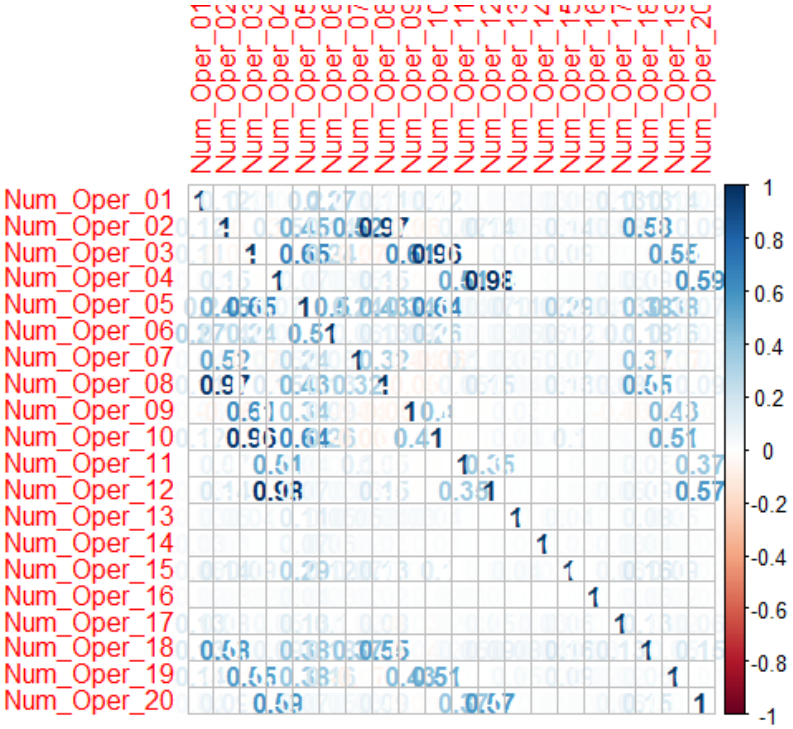

Fig. 8: Correlaciones entre las variables de Num_Oper.

Observemos que existen ciertas correlaciones altas entre las variables Num_Oper_02, Num_Oper_03, Num_Oper_04, Num_Oper_05, Num_Oper_08, Num_Oper_09 y Num_Oper_10.

Intentamos aplicar Principal Component Analysis (teniendo en cuenta que pasamos a tener variables continuas en vez de discretas) para reducir el número de variables, pero para conseguir un 75% de varianza explicada acumulada necesitábamos 15 componentes y más aún, aplicando PCA únicamente a las variables para las cuales existen correlaciones altas (Num_Oper_02, Num_Oper_03, Num_Oper_04, Num_Oper_05, Num_Oper_08, Num_Oper_09 y Num_Oper_10), obtenemos que para conseguir un 75% de varianza explicada acumulada necesitaríamos cinco componentes de seis. Por tanto, decidimos que PCA no es una técnica útil a la hora de reducir dimensiones.

2.7 Poder adquisitivo.

Esta es la variable que ha de ser predicha, se trata de una variable continua que define el poder adquisitivo del cliente. Al ser datos del territorio español, suponemos que está medida en euros.

Intuimos que esta variable posee una alta relación con el salario del cliente, por ejemplo, sumar o restar una cierta cantidad a su sueldo. Esta afirmación la apoyamos en que el sueldo medio español es de 23.000 euros (INE), mientras que la media del poder adquisitivo dentro del dataset es de aproximadamente 16.000 euros. Algo parecido ocurre con la mediana, en España es de 16.000 euros (INE) y en el dataset es aproximadamente de 13.000 euros.

Al ver cómo se distribuye la variable poder adquisitivo, nos indica que ésta se distribuye de forma irregular. Además, la naturaleza de esta variable definirá para nosotros la métrica utilizada.

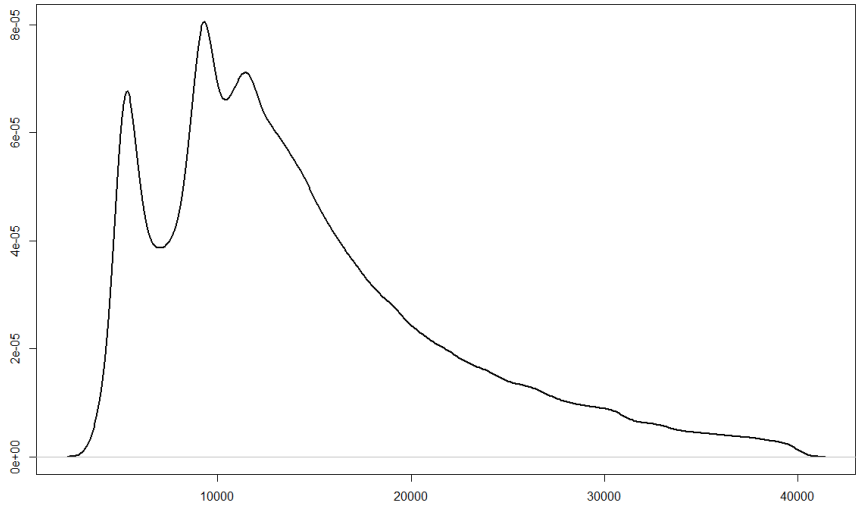

Fig. 9: Función de densidad del poder adquisitivo hasta los 40.000 euros.

3 Métrica elegida.

A pesar de que una de las métricas más utilizadas a la hora de hacer predicciones de variables continuas es el Mean Absolute Error (MAE), ésta no va a ser utilizada para resolver este problema. A continuación, se muestra un ejemplo práctico de por qué no. Recordemos cómo se mide el MAE:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|\hat{Y}_i - Y_i|$$

Y_i es el poder adquisitivo real del cliente e $\hat{Y}_i$ es el poder adquisitivo que predice el algoritmo. Usando esta fórmula hacemos una media de los errores absolutos de cada uno de los pares de valores. De esta forma, el valor absoluto del error en cada predicción va a tener la misma importancia. Este concepto va a quedar claro en el siguiente ejemplo:

Supongamos que el poder adquisitivo de un cliente es de 4.000€ y que nuestro algoritmo realiza una predicción de 10.000€. Esto implicaría que el MAE para este cliente sería de $|10.000 - 4.000| =$ 6.000€. Si otro cliente tiene un poder adquisitivo de 60.000€ y el algoritmo realiza una predicción

de 66.000€, para este cliente el MAE sería igualmente de $|66.000 - 60.000| = 6.000$€. Por lo tanto, MAE va a ponderar estas dos predicciones de la misma manera. Esto no tiene sentido desde un punto de vista de una entidad financiera que pretende establecer segmentaciones estratégicas más eficientes que les ayuden en la toma de decisiones a la hora de ofrecer el producto más adecuado.

Un cliente con poder adquisitivo de 4.000€ no tiene nada que ver con un cliente con poder adquisitivo de 10.000€, mientras que un cliente con poder adquisitivo de 60.000€, probablemente sea similar a un cliente con poder adquisitivo de 66.000€. La solución es medir el error de cada predicción de forma proporcional. Esta forma de medir el error se llama Mean Absolute Proporcional Error:

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\frac{|\hat{Y}_i - Y_i|}{|Y_i|}$$

MAPE va a medir, en media, cada uno de los errores proporcionales que se cometen a la hora de predecir Y_i, poder adquisitivo real del cliente, con respecto a $\hat{Y}_i$, poder adquisitivo que predice el algoritmo. Veamos el siguiente ejemplo:

Supongamos que el poder adquisitivo de un cliente es de 60.000€ y que nuestro algoritmo predice 66.000€. Para otro cliente, supongamos que su poder adquisitivo es de 4.000€. ¿Cúal debería ser el poder adquisitivo predicho para que se mantuviera el mismo MAPE?

$$\frac{|66.000 - 60.000|}{|60.000|} = \frac{|4.000 - Valor\ Predicho|}{|4.000|} \Longrightarrow Valor\ Predicho = 3.636€$$

Tal y como se ve, predecir el error de forma proporcional es mucho más lógico, pues ahora podemos afirmar que un cliente con poder adquisitivo de 4.000€ predicho como 3.636€ va a ser segmentado de forma correcta a la hora de ofrecer productos adecuados.

4 ¿Qué algoritmo de predicción utilizamos?

Las principales características de nuestro dataset son:

- Es *sparse* (gran cantidad de ceros).
- Tiene un gran número de variables cualitativas.
- Se trata de una matriz de datos con grandes dimensiones (363.834 filas y 89 columnas).

Un primer enfoque fue usar los algoritmos clásicos de la librería de scikit-learn de Python, tales como SVM, Random Forest y AdaBoost, pero encontramos diversas dificultades. El tiempo para entrenar los algoritmos era muy elevado, por lo que intentamos tomar una muestra más pequeña de los datos (10%) para entrenar, pero el error obtenido era muy alto.

En segundo lugar, para solucionar el problema anterior, probamos la estrategia de ensemble usando bagging con diferentes regresores y entrenando cada uno de estos con diferentes muestras del dataset, para posteriormente realizar una predicción única en base a las predicciones de los distintos regresores. Pero en este caso tampoco obtuvimos buenos resultados.

Por otro lado, un algoritmo que está muy de moda son las redes neuronales, especialmente los perceptrones multicapa (Deep Learning), pero dado este problema pensamos que la interpretabilidad

es importante ya que estamos tratando de establecer segmentaciones entre los clientes a partir de su poder adquisitivo. Por tanto, conocer las variables más importantes es una información de gran interés para la entidad financiera, algo que las redes neuronales no pueden proporcionar facilmente ya que al final se usan como cajas negras.

Otro enfoque que se utilizó fue realizar un clustering previo sobre los clientes. Los algoritmos utilizados fueron K-means, K-medoids y CLARA, variando sobre éstos el número de clusters deseados, pero no llegamos a obtener grupos en los que hubiera una diferencia significativa en cuanto al poder adquisitivo. Intuimos además que la realización de clustering con resultados útiles usando estos algoritmos sería difícil, ya que el dataset posee un gran número de variables cualitativas que han de ser transformadas a dummies y ser normalizadas, lo cual implica que estas variables van a tener mucho peso en la métrica.

Para afrontar los tres problemas que tiene el dataset: es sparse, tiene grandes dimensiones y hay un gran número de variables cualitativas, decidimos utilizar otros regresores que pensábamos que podían ser más adecuados para este problema. Éstos son LigthGBM y CatBoost.

LigthGBM incluye dos técnicas novedosas: Gradient-based One-Sided Sampling (GOSS) y Exclusive Feature Bundling (EFB). Con GOSS se excluyen grandes proporciones de información de instancias con un bajo gradiente, y a las que tienen gradientes más altos se les da más importancia ya que proporcionan más información. Con EFB se da un enfoque capaz de afrontar el problema de tener tantos ceros en nuestra matriz, ya que reduce el número de atributos sin influir demasiado en la precisión del regresor.

Para afrontar el problema del gran número de variables cualitativas, hemos decidido usar **CatBoost**, es una librería que permite el manejo de las variables categóricas con mucha facilidad. Además es robusto porque no es necesario una búsqueda extensiva de parámetros y reduce las posibilidades de sobreajustarse al modelo, por lo que el regresor generaliza mejor. Es de destacar la transformación por logaritmos hecha en todas las variables de Imp_Cons e Imp_Sal, hecha para reducir el rango de éstas. Esta tranformación realizada por $X_{new} = Log(X + |min(X)| + 1)$ y ha proporcionado mejores resultados.

En concreto, nos hemos basado en el ranking de CatBoost para seleccionar variables puesto que una reducción previa no ha sido posible. Estas variables seleccionadas han sido las utilizadas por el algoritmo que hemos considerado como regresor, LigthGBM, a pesar de que esta combinación pueda parecer extraña, pues normalmente se suele utilizar las variables seleccionadas con el mismo algoritmo. Este ha sido el método que ha conseguido reducir más el MAPE.

Se ha dividido el dataset en particiones de entrenamiento (81%), validación (9%) y test (10%). En este caso se ha considerado suficiente utilizar validación simple dadas las dimensiones de nuestro conjunto de datos. Con las particiones de entrenamiento y validación se ha creado el ranking de atributos y la selección de hiperparámetros. Con la partición de test se ha obtenido la estimación futura del error. Además, se ha utilizado la función RobustScaler de scikit-learn, la cual resta la mediana y divide por el rango intercuartílico, para tener los datos estandarizados de esta forma.

Hemos seleccionado las 29 variables que han sido proporcionadas por CatBoost, pues eran las únicas que el algoritmo consideraba útiles, ya que las demás no las usaba. Previamente hemos realizado estas observaciones que nos ayudan a reducir las dimensiones del dataset, así como reducir el dominio de la variable Socio_Demo_01.

- Ninguna variable Ind_Prod aparecía en el ranking de las más utilizadas, por lo que decidimos compactarlas en una única variable que fuera la suma de las mismas. Esta variable se nombró

como Ind_Prod_01, tiene una media de 4'13 productos y una mediana de 4 productos por cliente. Se verá que posteriormente se le da importancia en la predicción.

- Socio_Demo_01 es una variable cualitativa que toma 509 valores y aparecía en el ranking pero, debido a la gran cantidad de categorías que tiene y a las pocas instancias que hay en algunas de ellas, se optó por agrupar en nuevas categorías de forma que dentro de cada una de ellas contuviera categorías cuyas medianas (debido a que es más representativa que la media) de poder adquisitivo "fueran similares". Entendiendo esto como que no se rechaza el test Mood, en el cual la hipótesis nula es que las medianas entre categorías son iguales. Se agruparon categorías cuyo p-valor al realizar el test fuera mayor que 0'99, con esto obtuvimos 120 nuevas categorías de las 509 originales. Este método no asegura que las categorías se agrupen por medianas similares pero, se comprobó gráficamente que las medianas agrupadas no se distanciaban mucho, y más aún, obtuvimos mejor MAPE.

Las variables proporcionadas por CatBoost son: 'Imp_Cons_01', 'Imp_Cons_02', 'Imp_Cons_03', 'Imp_Cons_04', 'Imp_Cons_06', 'Imp_Cons_08', 'Imp_Cons_09', 'Imp_Cons_11', 'Imp_Cons_12', 'Imp_Cons_15', 'Imp_Cons_16', 'Imp_Cons_17', 'Imp_Sal_08', 'Imp_Sal_09', 'Imp_Sal_10', 'Imp_Sal_12', 'Imp_Sal_15', 'Imp_Sal_19', 'Imp_Sal_20', 'Imp_Sal_21', 'Ind_Prod_01', 'Num_Oper_05', 'Num_Oper_06', 'Num_Oper_09', 'Num_Oper_17', 'Num_Oper_18', 'Socio_Demo_01', 'Socio_Demo_02' y 'Socio_Demo_03'.

Podemos observar las siguientes relaciones con el análisis exploratorio de las variables realizado al principio de este informe.

- Observemos que la variable Imp_Cons_03, la cual en media tiene el consumo más alto de los clientes, aparece en el ranking.
- Dentro de las variables Imp_Sal, encontramos que las dos parejas cuyas densidades eran muy parecidas y que "repetían" muchos valores, Imp_Sal_08 con Imp_Sal_09 e Imp_Sal_19 con Imp_Sal_21, aparecen en el ranking.
- La variable suma de tenencia de productos financieros, Ind_Prod_01, aparece también en el ranking.

5 Selección del modelo y ajuste de hiperparámetros.

Una vez que se han seleccionado las 29 variables vía CatBoost, nos hemos apoyado en el conjunto de validación para hacer la selección de hiperparámetros. En el caso de LightGBM, los parámetros más relevantes a ajustar han sido:

- num_leaves: es el número máximo de hojas del árbol, en nuestro caso 511.
- objetive: función objetivo que se desea minimizar, en nuestro caso MAPE.
- max_depth: profundidad máxima que puede alcanzar un árbol, en nuestro caso -1 ya que no establecemos una profundidad determinada.
- learning_rate: tasa de aprendizaje del algoritmo, en nuestro caso tiene el valor de 0'05.
- max_bin: número de máximo de valores únicos por cada atributo, en nuestro caso son 255.

- min_child_samples: número de instancias en la hoja de un árbol, en nuestro caso tenemos 100.
- min_child_weight: cantidad de hessiano mínima para realizar la poda del árbol, en nuestro caso 0'0001.
- boosting: método de boosting utilizado, en nuestro caso GBDT.
- top_k: valor utilizado para el algoritmo Voting Parallel, en nuestro caso hemos fijado el valor a 100.
- bagging_frec: frecuencia de iteraciones en las que se realiza bagging, en nuestro caso 1.
- bagging_fraction: porción de elementos a seleccionar cada vez que se realiza bagging, en nuestro caso 1.
- feature_fraction: porción de atributos proporcionados al árbol para poder realizar la partición, en nuestro caso 0'66.
- bin_construct_sample_cnt: cantidad de elementos a tener en cuenta para la construcción de los bins, en nuestro caso 60.000.

Todos estos parámetros han sido explorados en rangos que han sido basados en la opinión de expertos. No se ha podido hacer un GridSearch debido al alto coste computacional que supone, aún así, por la exploración que se ha llevado acabo se puede decir que los parámetros son bastante adecuados. Tampoco se ha fijado un número de iteraciones a la hora de buscar los mejores parámetros, sino que se han realizado evaluaciones en cada iteración contra el conjunto de validación, deteniendo el entrenamiento en caso de que no hubiera reducción de MAPE en 25 iteraciones.

6 Bondad del modelo.

Para medir la bondad del modelo, tal y como se ha dicho antes, se ha utilizado la métrica MAPE. Con los parámetros elegidos anteriormente, se construye un nuevo modelo usando como partición de entrenamiento la unión de los conjuntos anteriores de entrenamiento y validación (90% del dataset). Una vez se ha entrenado con este porcentaje, usando la partición de test (10% de los datos) se ha obtenido un valor de MAPE = 0'211. Esto quiere decir que en media la predicción se desvía un 21'1% con respecto al poder adquisitivo original. La siguiente gráfica muestra como se distribuyen los errores.

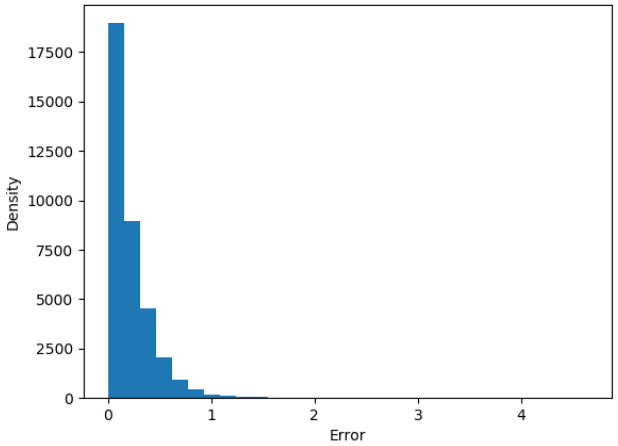

Fig. 10: Histograma de los errores.

Se observa que la gran mayoría están acumulados entre 0 y 0'4. Si ordenamos los valores del poder adquisitivo en el eje X y en el eje Y representamos el MAPE cometido para cada uno de ellos, obtenemos los siguientes gráficos.

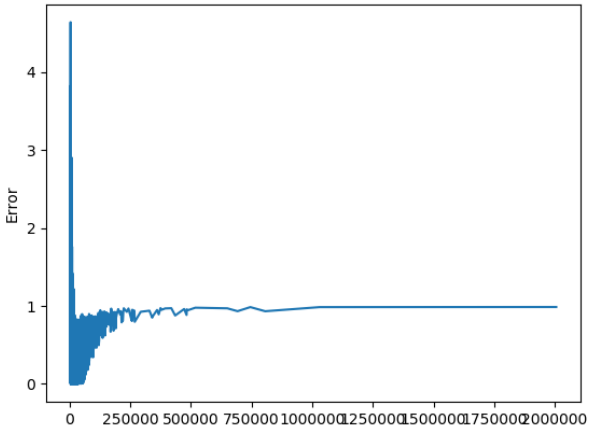

Fig. 11: Gráfico de los errores generales.

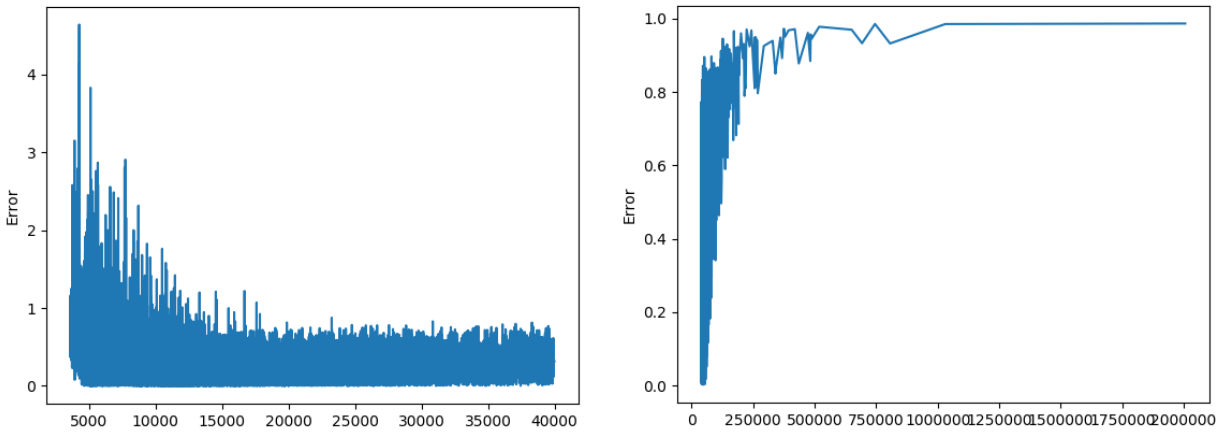

Fig. 12: Gráficos de los errores para poder adquisitivo de menos de 40.000€ y más de 40.000€.

Debido a la gran cantidad de valores que se tienen es difícil apreciar un comportamiento claro del error mediante los gráficos proporcionados por Python. Lo que sí que se puede afirmar es que para valores bajos del poder adquisitivo se cometen varios errores grandes.

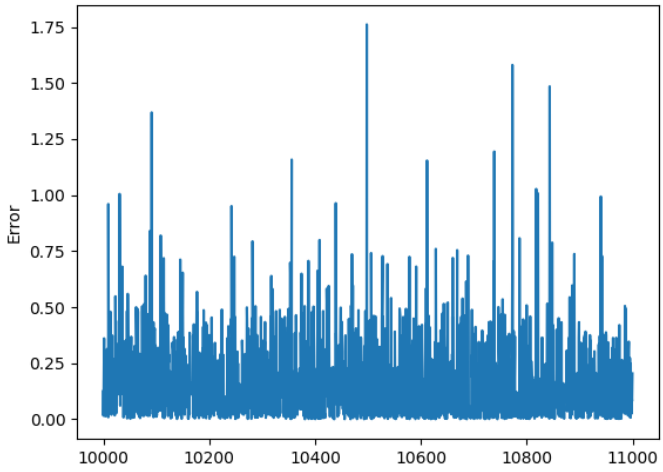

Fig. 13: Gráfico de los errores para poder adquisitivo de entre 10.000€ y 11.000€.

Tal y como se aprecia en la imagen previa entre un poder adquisitivo de 10.000€ y uno de 11.000€, el MAPE de cada individuo sigue un comportamiento cercano a 0'211. La siguiente tabla refleja que podemos considerar que el algoritmo realiza una buena predicción.

	MAPE = 0'2	MAPE = 0'3	MAPE = 0'4	MAPE = 0'5
Número de predicciones > MAPE	14204	8868	5380	3307
Número de predicciones <= MAPE	22180	27516	31004	33077

Tabla 2: Umbral de predicciones con respecto a un MAPE fijo.

Por ejemplo, si fijamos un MAPE superior a la media, como puede ser 0'3, se observa que 8868 predicciones están por encima de este umbral, lo que supone que únicamente en un 24% de las predicciones, se comete un error superior al 30% con respecto al poder adquisitivo original.

7 Interpretabilidad y viabilidad de puesta en producción.

Una de las razones por las que decidimos usar este algoritmo es porque permite establecer un ranking de variables y por lo tanto esto puede ayudar a la entidad financiera a distinguir cuáles son las variables críticas a la hora de predecir el poder adquisitivo de un cliente. A continuación, se comentan cuáles son estas variables importantes:

- Los importes en consumos habituales tienen una gran relevancia, destacando Imp_Cons_03 que es el que posee la media más alta. Además, es de destacar que aparecen 12 de los 17 importes.
- Socio_Demo_01, que suponemos que es la ocupación del cliente.
- Socio_Demo_03 y Socio_Demo_05, las supuestas edad del cliente y antigüedad de la cuenta respectivamente.
- Los saldos en ciertos productos financieros.
- La cantidad total de productos financieros que posee el cliente.
- El número de operaciones a través de ciertos productos financieros.

7.1 Viabilidad.

A continuación, se exponen las características del ordenador donde se han realizado el entrenamiento y el test del modelo.

- Procesador Intel i7 6700HQ, 2.60GHz.
- 16GB de RAM.
- 8 cores lógicos.
- Arquitectura de 64 bits con sistema operativo Windows.

El tiempo medio de CPU para entrenar con el 90% de los datos es 109 segundos.

El tiempo medio de CPU para predecir el 10% de los datos es 12 segundos.

Por tanto, dados estos tiempos y los resultados explicados sobre los errores, consideramos que su puesta en producción es más que viable.