UNIVERSIDAD TORCUATO DI TELLA

Trabajo de Tesis

MASTER IN MANAGEMENT + ANALYTICS

PREDICCIÓN DE UPSELLING CON TÉCNICAS DE MACHINE LEARNING PARA EMPRESA DE TELECOMUNICACIONES

Andrés Darmograj

Tutor: Ph.D. Guido de Caso

Julio, 2020

Contenido

Resumen

Predicción de upselling con técnicas de Machine Learning para empresa de Telecomunicaciones

Este trabajo trata sobre la implementación y automatización de un modelo de Machine Learning con capacidad de predecir operaciones de cambio de plan ascendente para el segmento B2B de una empresa de Telecomunicaciones. Es relevante, ya que es el primer modelo de inteligencia artificial que se utilizará dentro de dicho segmento en la compañía.

El mismo se entrenó con información anonimizada de cada cliente, que incluye patrones de consumo y pago, uso del móvil, entre otros. Esta fue volcada en una tabla que tiene mensualmente, para cada cliente, más de 350 variables que lo caracterizan.

Para elaborarlo, se utilizaron diversas técnicas y algoritmos aprendidos durante la maestría. Entre las más destacadas se encuentran las técnicas de limpieza y transformación de datos, regularización, uso de modelos ensamblados como ser *Random Forest* o *Gradient-boosted tree* con su respectiva evaluación de performance.

Fue probado en febrero de 2020 en la campaña de cambio de plan, lográndose un incremento en la efectividad de +139% (equivalente a +4,09 pp.) respecto al método tradicional utilizado anteriormente.

Esta tesis demuestra el gran impacto que puede generar la implementación de un algoritmo de Machine Learning en la operación de una gran corporación, y lo superior que resulta frente a técnicas básicas de análisis de datos.

Abstract

Upselling prediction with Machine Learning techniques for Telecommunications company

This work is about the implementation and automation of a Machine Learning algorithm with the ability to predict upselling operations for the B2B segment of a Telecommunications company. It is relevant, since it is the first artificial intelligence algorithm to be used within this segment in the company.

It was trained with anonymized information of each client, which includes consumption and payment patterns, mobile usage, among others. This information was embedded into a table that has, for each client, more than 350 variables, with monthly updates.

To design it, various techniques and algorithms learned during the master's degree were used. Among the most prominent are the techniques of data cleaning and transformation, regularization, use of assembled models such as *Random Forest* or *Gradient-boosted tree*, with their respective performance evaluation.

It was tested in February 2020 during the upselling campaign, achieving an increase in effectiveness of + 139% (equivalent to +4.09 pp.) Compared to the traditional method previously used.

This thesis demonstrates the great impact that the implementation of a Machine Learning algorithm can generate in the operation of a large corporation, and the superior results compared to basic data analysis techniques.

1. Introducción

1.1. El dominio

En una sociedad cada vez más hiperconectada, los individuos buscan tener más acceso a los datos e información. Las personas tienen la necesidad de estar conectados constantemente, de tener una vida/huella digital que hace unos años no existía. Sumado a esto, el uso de internet se hizo imprescindible para un gran número de tareas cotidianas. En este contexto es dónde surgen y se desarrollan las empresas de telecomunicaciones, que brindan servicios de conectividad a cualquier individuo, empresa o gobierno.

Las compañías de telecomunicaciones brindan servicios de conectividad tanto móvil como fija. Para ello, tienen montada una infraestructura de red, que permite asegurar el enlace entre los usuarios. Cuentan con ingresos mensuales operativos debido a la contratación de estos servicios, y también por el uso de su red por parte de terceros. Los egresos provienen principalmente del mantenimiento operativo de esta infraestructura, salarios y gastos de actividad comercial. Es importante destacar que para brindar estos servicios a nivel país se requiere de una gran inversión, que únicamente se puede sustentar con una gran masa crítica de clientes.

En el caso particular referido a este trabajo, la empresa de telecomunicaciones en cuestión es una de las 3 más grandes de la Argentina, acumulando aproximadamente un 33% del marketshare nacional.

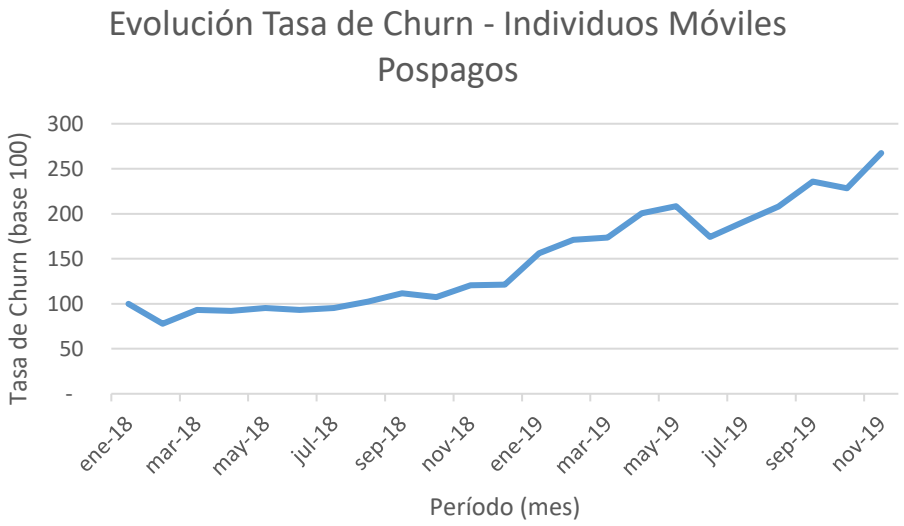
1.2. El problema

A partir de la sanción del Decreto 1340/2017, y con el fin de promover la competencia y la "convergencia tecnológica", se habilitó a las telefónicas a ofrecer servicios audiovisuales por cable. Esto inició la era del cuádruple play (4P) en Argentina, con las empresas ofreciendo descuentos por la contratación de ofertas multi-productos. Estas ofertas 4P se promueven con el fin de blindar al cliente desde la parte fija, dificultando a la competencia ingresar al hogar.

Desde enero 2019, fecha de plena puesta en vigencia de la reglamentación, el mercado se recalienta, con ofertas de captación agresivas y multi-producto para determinadas regiones. A continuación, se puede apreciar cómo evolucionó la tasa de churn[1] debido a este fenómeno:

[1] Se define tasa de churn como el porcentaje de clientes activos que deja de utilizar un producto o servicio en un determinado período. También se conoce como la tasa de abandono de clientes.

Figura 1.1: Evolución de tasa de churn de individuos móviles pospagos, uno de los segmentos más afectados por la convergencia



En este contexto, con una competencia cada vez más agresiva y con un mercado móvil saturado, se tornó relevante la adquisición de ingresos alternativos vía desarrollo de clientes. Para lo cual se diseñaron inicialmente distintos modelos predictivos enfocados en el segmento *Business-to-Customer* (B2C), ya que este segmento acumula la mayor cantidad de ingresos operativos. Con los mismos se logró, por ejemplo, incrementar los ingresos unitarios por contacto de campañas en un 10%, sin aumentar la cantidad de representantes destinados a la tarea.

En esta segunda etapa, se propuso comenzar a trabajar sobre el segmento *Business-to-Business* (B2B), con el fin de optimizar la base mensual enviada a campañas para ser gestionada en canales proactivos, y mejorar la efectividad en los canales reactivos[2]. La meta es lograr este aumento de efectividad sin aumentar los recursos destinados a la venta.

1.3. Universo de clientes alcanzados

Dentro del segmento B2B, existe una sub-clasificación dependiendo de la envergadura o tipo de cliente. A continuación, se presentan los diferentes sub-segmentos existentes:

[2] Entiéndase por canales proactivos aquellos en dónde nuestros representantes se ocupan de contactar al cliente (ejemplo: Televentas-OUT). Los canales reactivos son aquellos canales que esperan el contacto del cliente para su atención o venta (ejemplos: Agentes, Centros de Experiencia, Televentas-IN)

Tabla 1.1: Sub-segmentos existentes dentro de B2B. Se puede identificar a cada cliente por su CUIT, y cada CUIT tiene asociado una cantidad de líneas telefónicas (ANIs - *Automatic number identification*)

Segmento	Sub-Segmento	% CUITs	% ANIs
B2B	Emprendedor	69%	46%
B2B	Premium	28%	45%
B2B	Gobierno	2%	1%
B2B	TOP	1%	6%
B2B	Resto	≈0%	1%

Para la construcción de los modelos, nos focalizaremos únicamente en las líneas B2B móviles pospagas[3] pertenecientes al sub-segmentos 'Premium' y 'Emprendedor'. Corresponden en su mayoría a PyMEs y emprendedores. Estos sub-segmentos suelen tener contratos similares a los de B2C, y no se les proporciona atención personalizada. Como se aprecia en la tabla anterior, alcanzaríamos al 97% de los CUITs y 92% de los ANIs registrados en el *Customer Relationship Management* (CRM) bajo el segmento B2B.

El resto de los sub-segmentos corresponden a empresas grandes o entes gubernamentales. Todos estos clientes tienen asociado un ejecutivo de cuentas. Generalmente sus contratos son más complejos, ya que incluyen requerimientos específicos según cada licitación. Debido a esto, solo una pequeña proporción de los mismos ha migrado al nuevo CRM.

1.4. Modelos de Machine Learning en empresas de telecomunicaciones

Las empresas de telecomunicaciones se caracterizan por tener una gran cantidad de datos sobre cualquier persona que se mueva bajo su pisada o zona de influencia, siempre y cuando esta lleve consigo un dispositivo móvil. Estos datos van desde cuestiones básicas como la facturación, hasta la ubicación (vive-trabaja) y consumo específico de la persona. Somos capaces de identificar cambios de hábitos de los usuarios casi de manera inmediata, *trackeando* sus patrones de consumo.

En este contexto, existe una necesidad imperiosa de transformar estos datos en información, para así tomar decisiones de negocios que nos permitan incrementar los ingresos y a la vez mejorar la experiencia de nuestros clientes.

Existe una amplia bibliografía de casos de éxito en modelos de Machine Learning aplicados en empresas de telecomunicaciones. Entre los múltiples papers escritos al respecto, se pueden destacar los siguientes, citados correctamente en la bibliografía:

- *Artificial Intelligence (AI) and Big Data for Telecom.*
- *Customer churn prediction in telecom using machine learning in big data platform.*

[3] Las líneas pospagas son aquellas que tienen un contrato con un monto fijo mensual más excedentes. Las líneas prepagas no tienen contrato asociado, y el cliente solo gasta lo que recarga.

- *Customer Churn Prediction and Upselling using MRF (Modified Random Forest) technique.*

De ellos se desprende que las empresas de telecomunicaciones suelen enfocar sus modelos de Machine Learning en los siguientes tres puntos:

- *Churn*: utilizado para predecir la tasa de baja de los clientes.
- *Upselling*: utilizado para predecir el cambio de plan ascendente.
- *Port-OUT / Port-IN*: modelos con foco en predecir entrada/salida de clientes desde y hacia la competencia.

1.4.1. Modelos de Upselling en empresas de telecomunicaciones

Actualmente existen 3 formas de que un cliente realice una operación de cambio de plan ascendente, y esencialmente se diferencian según el canal y forma que se realice dicha operación:

- Canales proactivos
    - Televentas-OUT: Esencialmente consiste en representantes dispuestos en distintos *call-centers*, que llaman a los clientes y les ofrecen un *upgrade* de plan con el fin de mejorar su experiencia de navegación. Estas ofertas pueden venir con algún descuento asociado.
- Canales reactivos
    - Televentas-IN y canales presenciales: en este caso, el cliente se contacta con uno de nuestros representantes, solicitándole alguna recomendación para un cambio de plan.
    - Canales digitales: estas operaciones se canalizan a través de la APP. A nivel mercado, hay un fuerte foco en incrementar la participación porcentual de este tipo de operaciones, ya que las mismas tienen un costo asociado muy bajo. Lamentablemente, todavía en B2B la APP no permite realizar este tipo de operaciones.

Lo recomendable es tener un modelo para cada canal, entrenándolo con aquellas personas que convirtieron en cada uno de ellos y luego evaluándolo en todo el parque. Esto, para optimizar el flujo de clientes hacia los distintos canales y así potenciar la conversión.

Es frecuente, debido a la falta de tiempo/recursos, armar un solo modelo, priorizando aquel canal que genera el mayor volumen de ventas. En el caso de B2C, por ejemplo, el modelo se entrena con los clientes que fueron enviados a campaña en el canal Televentas-OUT, ya que el mismo genera el 80% de las operaciones de cambio de plan de este segmento.

1.5. Cómo se suele abarcar

Generalmente en la industria este problema se abarca de dos maneras diferentes:

- El método tradicional consiste en darle seguimiento a distintas variables que el negocio considera relevantes a la hora de impulsar una campaña de upselling, y gestionar bases confeccionadas según reglas duras de negocio. Con este método, es

imposible captar sutilezas de cada cliente, por lo cual las efectividades que se logran son bajas.

- Actualmente se pasaron a utilizar modelos de Machine Learning, que teniendo un histórico de variables de todos los clientes, permite detectar patrones que apalanquen las operaciones de cambio de plan. Se da vía libre a que un algoritmo prediga la posibilidad de upselling, intentando reducir al mínimo las reglas duras de negocio (exceptuando asuntos estratégicos).

1.6. Propuesta de trabajo

El propósito de este trabajo es encontrar un método basado en análisis de datos que prediga de la mejor manera quienes serán los clientes que realizarán una operación de cambio de plan ascendente en el futuro.

El modelo arrojará una probabilidad de upselling para cada uno de nuestros clientes. Para lograrlo, se experimentará con distintos algoritmos de Machine Learning disponibles en el paquete PySpark, quedándonos con aquel que tenga mayor precisión. Se deberá evaluar si se debe descomponer el problema entre las distintas modalidades para realizar la operación, o si se analizará todo junto.

Se analizará a cuántos meses es necesario predecir el upselling, calculando la performance de los distintos modelos en diferentes escalas temporales. Adicionalmente, se analizarán limitantes en cuanto a la disponibilización de distintas fuentes de información.

2. Materiales y Métodos

2.1. Datos

Los datos provienen principalmente de dos fuentes: un tablón ABT *(Analytical Base Table*) con información variada de cada uno de nuestros clientes, y bases generadas por el área que se ocupa por enviar a los clientes a cada una de las campañas. A continuación, se explicará más en detalle cómo están conformadas estas tablas.

ABT Suscripción Móvil

Es la fuente principal desde la cual se sacarán los datos históricos de los clientes. Es una tabla plana alojada en *Hadoop*, que un área específica de la Dirección de Big Data se ocupa de mantener y actualizar. Se utiliza para construir modelos analíticos y predecir comportamientos futuros de los clientes. Básicamente incluye dos categorías de datos:

- Quién es el sujeto: describe las características propias de cada cliente como ser documento, cuenta financiera, número telefónico, región, etc.
- Qué hace el sujeto: describen el comportamiento de cada cliente, la adquisición de nuevos productos o servicios, el uso que le da el cliente a cada uno de nuestros servicios, su comportamiento de pago, etc.

Se procesa todos los meses (n), con información cerrada del mes n-1, para utilizarse en campañas a realizarse en n+1. Está disponible aproximadamente el 10 de cada mes.

Cada registro de esta base de datos corresponde a una línea telefónica, para un determinado mes. Es decir, se define como clave primaria de la tabla el número telefónico (ANI o 'valor_recurso_primario_cd') y el período. Debido a que en el segmento B2B es normal que un cliente tenga contratadas múltiples líneas, asociamos todas ellas a través de su CUIT.

Para el segmento B2B, por causa de migraciones de sistemas no se tiene información anterior a Mayo de 2019.

El ABT está conformado en la actualidad por 372 variables, que se agrupan en las siguientes categorías:

Tabla 2.1: Categorías disponibles en *ABT Suscripción Móvil* con su descripción

Categorías	# Variables	Descripción
Parque	98	Variables referidas a características propias del cliente: identificación del cliente, qué servicios tiene contratado, donde vive y trabaja, qué equipo tiene, si tiene la APP o no y status de su chip SIM.
Tráfico	88	Variables referidas al tráfico: tráfico de voz y datos, agotamiento de datos del plan, offloading.

Facturacion	51	Agrupa principalmente datos de facturación y cobranzas.
Recargas Bonos y Saldos	42	Informa el tipo de recargas que hizo el cliente, el monto y el uso de las mismas, si usó o no bonos e información referida a las bonificaciones.
Operaciones Comerciales	28	Datos sobre operaciones, financiamiento y portabilidad numérica
Red	24	Variables relacionadas a la experiencia de red y tráfico de datos
Gestion de Campo	13	Tickets que se generaron por inconvenientes del cliente
Analytics	10	Información sobre distintos modelos predictivos
Otros	10	Agrupa principalmente filtros de campañas
Gestion de Clientes	6	Datos del call-center
Periodo	2	Período mensual cerrado al cual corresponden los datos.
Total general	**372**	

Como se puede apreciar, este tablón agrupa cualquier tipo de información que podamos llegar a requerir de nuestros clientes. Esto, creado para ser una fuente única y validada de información, facilita enormemente el armado de distintos modelos predictivos.

A continuación, se presenta un dataset real anonimizado, incluyendo 13 variables relevantes para el negocio.

Tabla 2.2: Dataset real anonimizado, con el evolutivo desde Mayo a Diciembre 2019 de 13 variables relevantes al negocio:

- Parque
  - ANI: número telefónico
  - Antigüedad: antigüedad de la línea
  - Método Pago: método de pago que utilizó el cliente para el abono de su factura mensual en dicho mes
  - MB Plan: Megabytes que incluía el plan de la línea en dicho mes
- Tráfico
  - MB Consumo: consumo mensual de datos en dicho mes
  - Llamadas caídas: cantidad de llamadas caídas en dicho mes
  - Red: red sobre la cual la línea tuvo más del 80% del tráfico mensual de datos
- Facturación
  - Importe cobrado: importe facturado al total del CUIT
  - Recarga total: recarga efectuada sobre la línea en dicho mes
- Operaciones Comerciales

    - Cambio de Oferta: *flag* que toma valor 1 si durante dicho mes la línea hizo un cambio de plan
- Red
    - CEI: indicador que oscila entre 0-100, representando la experiencia que tuvo el usuario con la red. Valores mayores a 60 indican experiencia buena o muy buena
- Gestión de Clientes
    - Tickets atención: indica la cantidad de veces que la línea recibió atención de un representante en dicho mes
- Período
    - Periodo: indica el período al cual corresponden los datos

UNIVERSIDAD TORCUATO DI TELLA

ANI	Antiguedad	Metodo Pago	MB Plan	MB Consumo	Llamadas caidas	Red	Importe cobrado	Recarga total	Cambio de Oferta	CEI	Tickets Atención	Periodo
6006997721	44	Cash	1024	289	0	2G	4289.57	232.1	0	78	0	201905
2562942158	23	Cash	6144	11351	0	3G	9333.88	0	0	66	0	201905
6694125720	154	Cash	9216	754	0	4G	1040	0	0	82	0	201905
5815418812	4	Cash	9216	0	0	No Inf	1590.64	0	0	0	0	201906
6085459993	122	Debit	9216	8917	0	4G	6999.17	0	0	71	0	201906
9587660065	231	Cash	9216	1148	4	4G	4735.81	0	0	41	0	201906
3597012143	15	Cash	6144	637	2	4G	4250	0	0	56	1	201907
7873527846	97	Cash	9216	4519	0	4G	2975.54	0	0	82	0	201907
3355578659	7	Credit Card	4096	5986	0	4G	1553.36	0	0	67	0	201907
5673648746	1	Cash	4096	307	0	4G	2341.5	1006.87	0	75	0	201908
7189925041	218	Debit	9216	451	1	4G	1038.2	0	0	71	0	201908
4684909719	36	Cash	4096	556	1	4G	1558.63	0	1	74	0	201908
5946654646	2	Credit Card	2048	0	1	2G	1600.4	0	0	72	0	201909
9982134668	16	Cash	1024	0	0	2G	3092	0	0	71	0	201909
7418529772	123	Cash	6144	1474	0	4G	3992.89	0	0	78	0	201909
6194507471	76	Debit	6144	7406	2	4G	2437.11	0	0	76	1	201910
7529531409	126	Cash	4096	1069	0	4G	1612.28	0	0	68	0	201910
8546138324	11	Cash	6144	0	0	No Inf	5703.2	0	0	0	0	201910
9944894341	76	Cash	6144	1317	0	4G	19008.19	0	0	85	0	201911
5315745606	129	Cash	9216	442	1	4G	2965.37	0	0	63	0	201911
1288504652	2	Credit Card	3072	0	1	2G	2156.88	596.84	0	51	0	201911
2214974166	136	Cash	9216	9033	0	4G	1300.6	0	0	77	0	201912
4563403652	29	Cash	4096	275	1	4G	1822.81	0	1	73	0	201912
1745668741	84	Cash	9216	450	0	4G	963.6	0	0	83	0	201912

Base de Campañas

El único dato que no tiene el tablón y podría llegar a resultar relevante para el modelo corresponde a la base de datos de clientes enviados a campañas. Esta información se encuentra alojada en el datawarehouse móvil utilizado por el área de Campañas, y se debe migrar vía Sqoop a Hadoop. Una vez transformada, la base quedaría de la siguiente manera:

Tabla 2.3: Ejemplo de *dataset* de Campañas. Incluye las siguientes variables:

- ANI: número telefónico
- Campana: puede tomar los siguientes valores:
  - 'ALTAS': corresponde a campaña de alta de nueva línea
  - 'UPSELLING': corresponde a campaña de cambio de plan ascendente
  - 'CATER': corresponde a campaña de cambio de terminal / venta de equipo
- Periodo: indica el período en el cual se gestionó la campaña

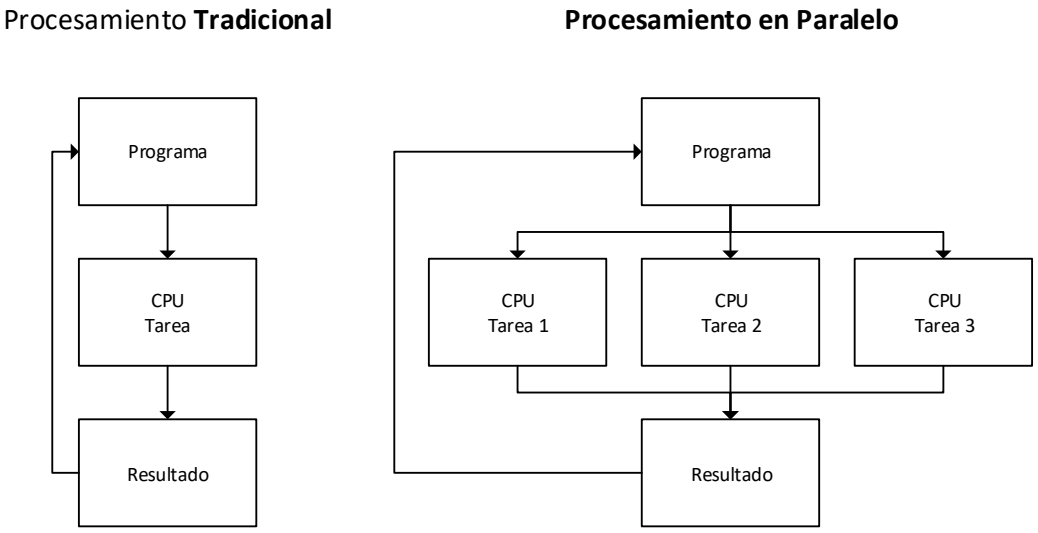
ANI	Campana	Periodo
3729724280	ALTAS	201905
3886871712	UPSELLING	201908
5171699260	UPSELLING	201907
6682741655	CATER	201906
9267120787	UPSELLING	201912
9327553501	UPSELLING	201907
5840336513	ALTAS	201907
2003560705	CATER	202001
2867009095	ALTAS	201906

2.2. Plataforma analítica

Debido al gran volumen de datos que manejamos y a políticas internas de la compañía, se nos requiere que los modelos corran en el *cluster*. Según la web Solingest, un *cluster* es un grupo de múltiples ordenadores (o máquinas virtuales) unidos mediante una red de alta velocidad tal que el conjunto es visto como un único ordenador. Se caracteriza por tener alto rendimiento, alta disponibilidad, equilibrio de carga y escalabilidad.

El *cluster* permite procesamiento en paralelo, dividiendo un problema grande, en caso que sea posible, en otros más pequeños que luego se ejecutan y resuelven simultáneamente en distintos nodos.

Figura 2.1: Diferencia entre procesamiento tradicional y paralelo



Un nodo es cada uno de los ordenadores (o máquinas virtuales) que forman parte de un *cluster*. Hay distintos tipos de nodos, entre los que se hayan los *masters* y los *workers*. Los masters supervisan el almacenamiento de los datos y se asegura de que los cómputos de los datos se lleven a cabo en paralelo. Los *workers*, en cambio, reciben las instrucciones del *master*, almacenan los datos y lanzan los cálculos necesarios.

Este procesamiento en paralelo y de forma distribuida se sustenta en un modelo de programación que se denomina *MapReduce*. Este modelo consta de dos etapas principales:

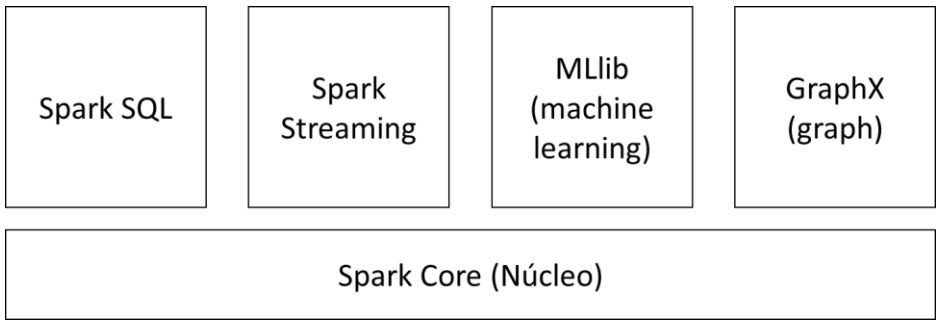
- *Map*: se toma un conjunto de datos y se convierte en otro donde cada elemento se convierte en una tupla (llave/valor)
- *Reduce*: se toma el resultado del *Map* y se combinan las tuplas en conjuntos más pequeños de tuplas.

El *cluster* en el cual se estarán procesando los datos consta de 120 nodos. Dependiendo de la necesidad de procesamiento, se utilizarán más o menos de ellos.

2.2.1. Spark

Para el procesamiento de datos en el cluster en paralelo utilizaremos un framework de propósito general denominado Spark. En la documentación del framework citada en la bibliografía se menciona que el mismo se compone de 5 unidades principales, que se representan en la figura a continuación.

Figura 2.2: Esquema de framework Spark con sus 5 unidades principales: Spark Core, Spark SQL, Spark Streaming, MLlib y GraphX.



Se describirá con mayor detalle las siguientes tres, ya que resultan relevantes a este trabajo:

- Spark Core: contiene las funcionalidades básicas de Spark. Esto incluye tareas de planificación, manejo de memoria, recuperación de fallos, interacción con sistemas de almacenamiento, entre otras.
- Spark SQL: es una extensión de Spark para el procesamiento de datos estructurados. Provee una abstracción de programación llamada *DataFrames* y además puede actuar como un motor de consultas distribuidas SQL.
- Spark ML: es la biblioteca de *Machine Learning* de Spark. Incluye algoritmos de clasificación, regresión, clustering, recomendación, entre otras utilidades.

2.3. Modelos de Machine Learning disponibles en Spark

Se pueden dividir los algoritmos de Machine Learning de aprendizaje supervisado en dos tipos: de regresión o clasificación. Principalmente se diferencian en el tipo de variable a predecir.

Los algoritmos o problemas de regresión predicen variables cuantitativas, es decir, valores numéricos. En cambio, los algoritmos de clasificación predicen variables cualitativas, que serían las K clases o categorías que puede tomar una variable. Como ejemplos de variables cualitativas se pueden mencionar el género de una persona (masculino o femenino), la marca de un producto (marca A, B o C), o el estado de cuenta de un cliente (activo o moroso). (James, Witten, Hastie & Tibshirani, 2013, pág. 28)

En este trabajo, la variable a predecir es cualitativa binaria, pudiendo tomar la misma dos valores: 0 o 1. El valor 1 equivaldría a un cliente que convirtió, y el valor 0 lo opuesto. Por lo cual, será necesario utilizar modelos predictivos de clasificación.

Según la documentación de Spark, los modelos de clasificación que tiene disponible son los siguientes:

- Regresión logística
- Árboles de decisión (*decision tree*)
- Bosques aleatorios (*random forest*)
- Potenciación del gradiente (*gradient-boosted tree, GBTs*)

- Redes neuronales (*multilayer perceptron*)
- Máquina de Vector Soporte (*linear support vector machine*)
- Uno contra el todos (*one-vs-rest*)
- Bayes ingenuo (*naive bayes*)

Los modelos que combinan las predicciones de modelos más pequeños (o modelos base) se los conoce como modelos de ensamble. *Random Forest* y *Gradient-boosted tree* son ejemplos de este tipo de modelos. Los mismos tiene una buena performance para el tipo de problema a tratar en este trabajo, por lo cual son los que utilizaremos a la hora de experimentar.

2.3.1. Modelo base: Árboles de decisión

Es un modelo de aprendizaje supervisado, que sirve para atacar problemas de clasificación como de regresión.

Para entender su funcionamiento, resulta más sencillo comenzar con una explicación sobre su aplicación a problemas de regresión. Para lo cual primero es necesario definir una medida de calidad de la predicción: el error cuadrático medio (ECM).

$$ECM = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{f}(x_i)\right)^2$$

Siendo:

- y_i: verdadero valor de la i-ésima observación
- $\hat{f}(x_i)$: predicción que la función $\hat{f}(x)$ hace respecto a la i-ésima observación
- n: cantidad de observaciones

El objetivo es minimizar el ECM en datos desconocidos, pero usaremos los datos de entrenamiento para guiar la búsqueda de patrones con poder predictivo que se espera generalicen bien. En árboles de decisión, la construcción de los mismos estará guiada por minimizar el error cuadrático medio de entrenamiento, aun cuando no sea el que interese.

Se propone un modelo que particione el espacio de atributos en regiones. Las regiones serás definidas como aquellas que minimicen el ECM.

Figura 2.3: Ejemplo de un árbol de decisión, optimizado tal que se minimice el ECM. La división en lo alto del árbol genera dos grandes ramas: la izquierda corresponde a aquellos registros en los cuales Years < 4,5, y en la derecha se acumula el resto. El árbol tiene dos nodos internos y tres terminales, u hojas. El número que se ve en cada una de las hojas corresponde al promedio de las observaciones que caen en la misma (valor predecido).

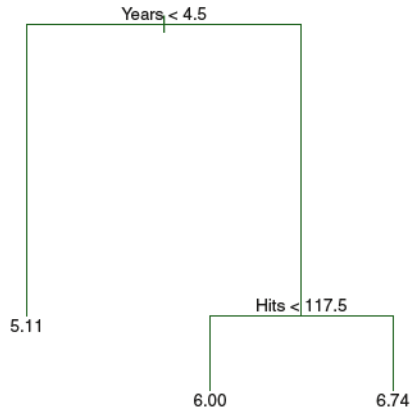
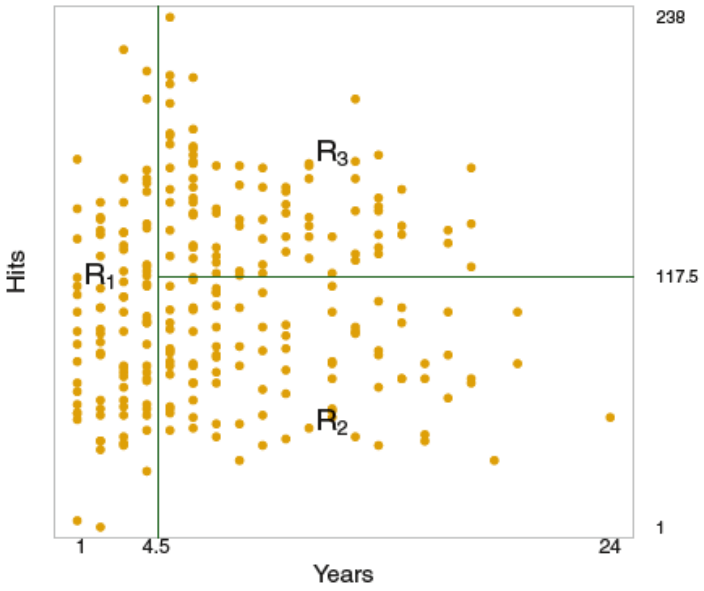


Figura 2.4: muestra la partición del plano en tres regiones para el árbol de la Figura 2.3.

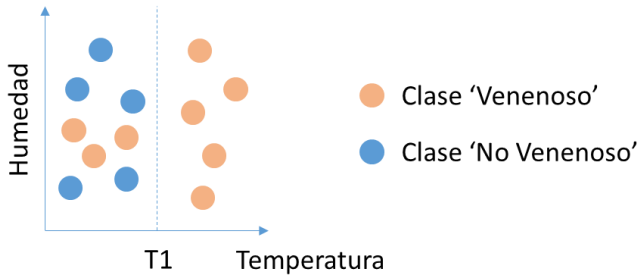


Considerar todas las posibles particiones es un problema intratable computacionalmente, por lo cual se aplica el paradigma de *divide & conquer*, que consiste en dividir recursivamente un problema en subproblemas más pequeños que conservan la misma estructura.

En cada paso, se elige una partición tal que minimice la suma de los ECM de las particiones resultantes. Luego, esto se repite para cada subregión recursivamente, hasta cumplir con un criterio de parada (por ejemplo, que en la región quede una única observación).

Si adaptamos la idea a clasificación, en vez de predecir con el promedio de las observaciones que caen en una región, se predice como probabilidad condicional de una clase k a la proporción de observaciones de la clase k que cae en la región correspondiente a x_i. En clasificación, en vez de guiar la construcción del árbol utilizando ECM, se utilizan lo que se denominan medidas de impureza, como ser el índice de Gini o la entropía (James, Witten, Hastie & Tibshirani, 2013, Cap. 8).

Figura 2.5: ejemplo de regiones para el caso de clasificación. Para una nueva observación cuya covariable Temperatura sea mayor a T1, se le asignará un 100% de probabilidad de que dicha observación corresponda a la clase 'Venenoso'. Si su covariable Temperatura fuera menor o igual a T1, tendría 62,5% (5/8) de probabilidad de ser 'Venenoso', y 37,5% (3/8) de ser 'No Venenoso'.



2.3.2. Random Forest

Random Forest propone un remuestreo doble en cada uno de sus árboles (observaciones y covariables), rompiendo las estructuras de correlación entre cada uno de ellos. Luego, al realizar el ensamble, termina impactando en la reducción de la variabilidad del modelo.

2.3.3. Gradient-boosted tree

Mientras que *Random Forest* ensambla modelos complejos para reducir la variabilidad de los mismos, *Boosting* ensambla una secuencia de modelos simples (con poca variabilidad) y siguiendo una estrategia de re-ponderaciones para obtener un estimador complejo. Es uno de los algoritmos más poderosos para dominios con tipo de dato mixtos.

Funciona de la siguiente manera:

- Se establece una muestra de entrenamiento de tamaño n: $(\boldsymbol{x_1}, y_1), \ldots, (\boldsymbol{x_n}, y_n)$
- Se entrena un primer clasificador débil $G_1(x)$, siendo la tasa de error levemente superior al *random-guessing* (escoger al azar)
- Cada árbol de manera secuencial produce una predicción para cada dato de entrenamiento. Mientras el algoritmo avanza, le va prestando más atención a aquellas observaciones que los árboles anteriores clasificaron mal.
- Luego, al ensamblar los árboles, el algoritmo le asigna más peso a los modelos más acertados.

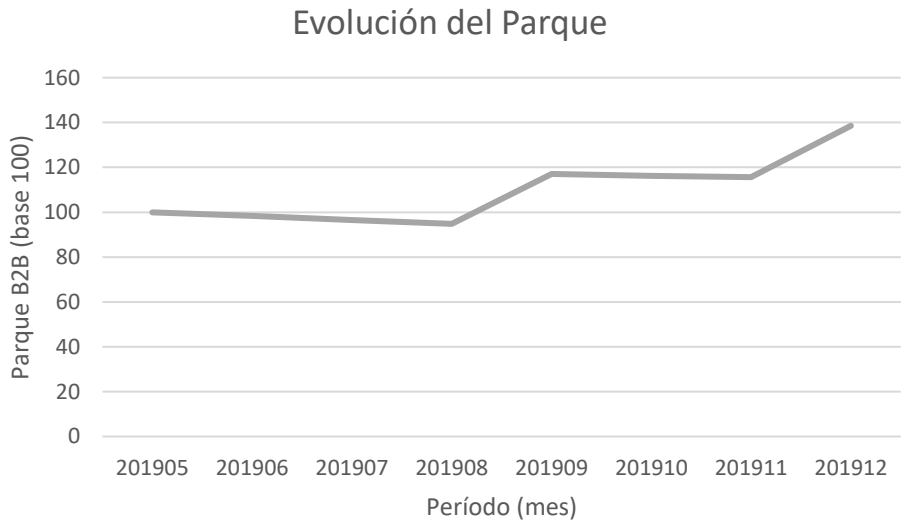
3. Exploración y armado de modelo

3.1. Exploración de datos

3.1.1. Parque y Operaciones

En Enero de 2020, se contaba con datos desde mayo a diciembre de 2019, y todos los meses se van agregando al *ABT Suscripción Móvil* datos correspondientes al cierre del último mes. En la Figura 3.1 se aprecian las variaciones del parque que se encuentra alojado en el nuevo CRM.

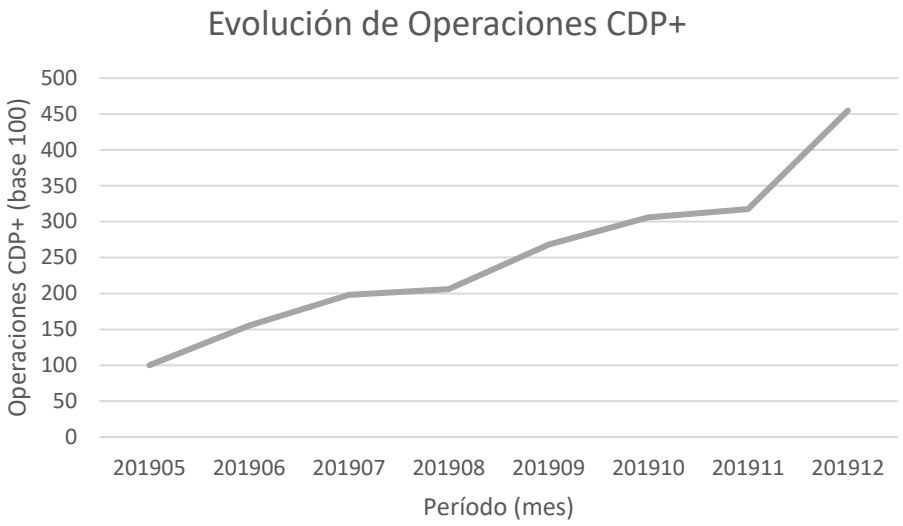
Figura 3.1: Evolución de cantidad de clientes del segmento B2B alojados en el nuevo CRM.



Se pueden apreciar dos picos ascendentes en septiembre y diciembre de 2019. Esto no se debe a altas puras, sino a la segunda y tercera migración de clientes al nuevo CRM.

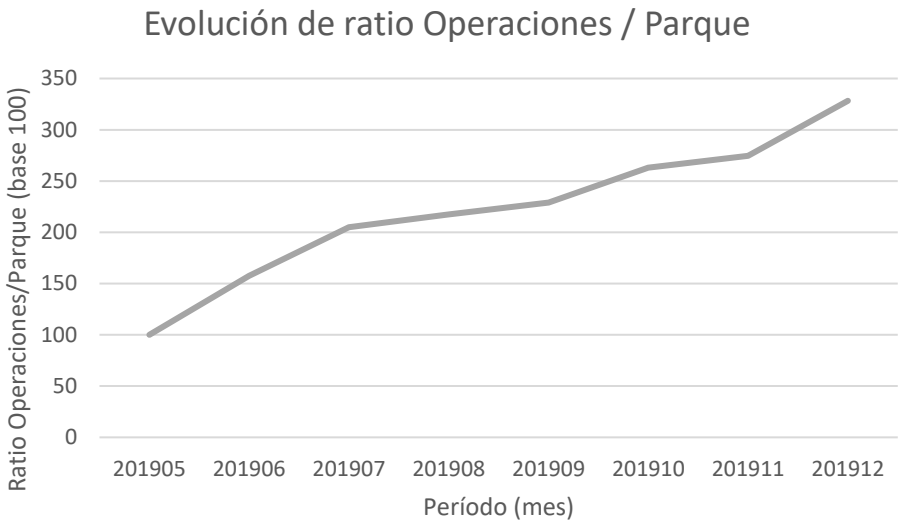
A continuación, se detallan la evolución de las operaciones de cambio de plan ascendente en el mismo período.

Figura 3.2: Evolución de operaciones de cambio de plan ascendente (CDP+) del segmento B2B alojados en el nuevo CRM.



Para evitar confundir el crecimiento de las operaciones por causa de las migraciones al nuevo CRM, es conveniente analizarlo según el ratio Operaciones / Parque.

Figura 3.3: Evolución del ratio Operaciones de cambio de plan ascendente (CDP+) sobre Parque, del segmento B2B alojados en el nuevo CRM.

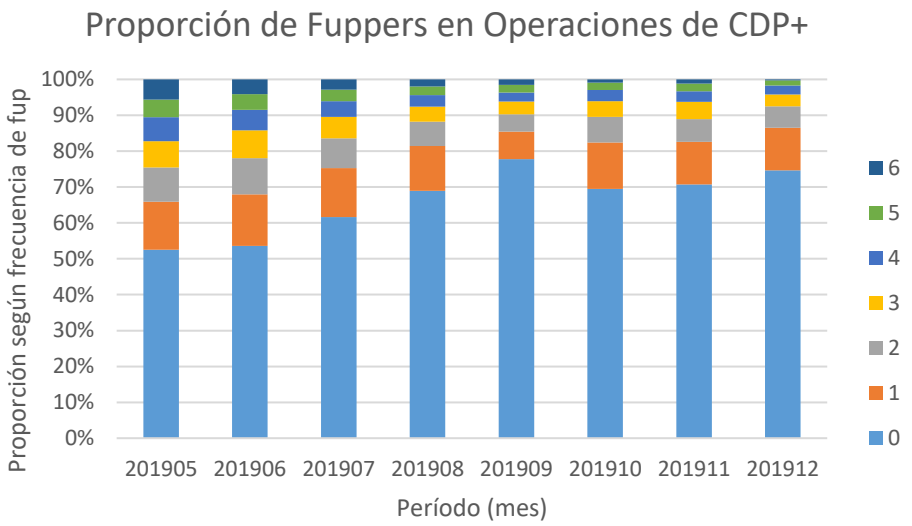


3.1.2. Agotamiento de datos

El termino FUP (*Fair Usage Policy*) se utiliza frecuentemente en las compañías de telecomunicaciones para referirse al máximo de MB que dispone un cliente durante su ciclo mensual. Se denomina *fupper* a un cliente que agotó sus datos mensuales. Para evitar trazar conclusiones basándonos en un solo mes, generalmente se almacena la historia de fupeo de los últimos 6 meses, para entender la regularidad que tuvo el cliente a la hora de agotar sus datos.

La marca de fupeo es un filtro duro que se utiliza actualmente en la campaña, ya que se supone que los clientes que agotan sus datos estarían más dispuestos a realizar un cambio de plan ascendente. Sin embargo, cuando vemos la distribución mensual de clientes que realizaron la operación de cambio de plan ascendente según su grado de FUP, concluimos que esta marca no debe ser excluyente a la hora de enviar a alguien a campañas. Se aprecia por ejemplo que el 75% de las operaciones realizadas en el mes de diciembre de 2019 corresponde a clientes que en ninguno de los 6 meses anteriores agotó los datos de su plan.

Figura 3.4: Muestra la proporción del fuppers sobre el total de operaciones de cambio de plan ascendente (CDP+) realizadas en el mes. Las categorías 0, 1, 2, 3, 4, 5, 6 indica la cantidad de meses que cada cliente agotó los datos de su plan en los últimos 6 meses.

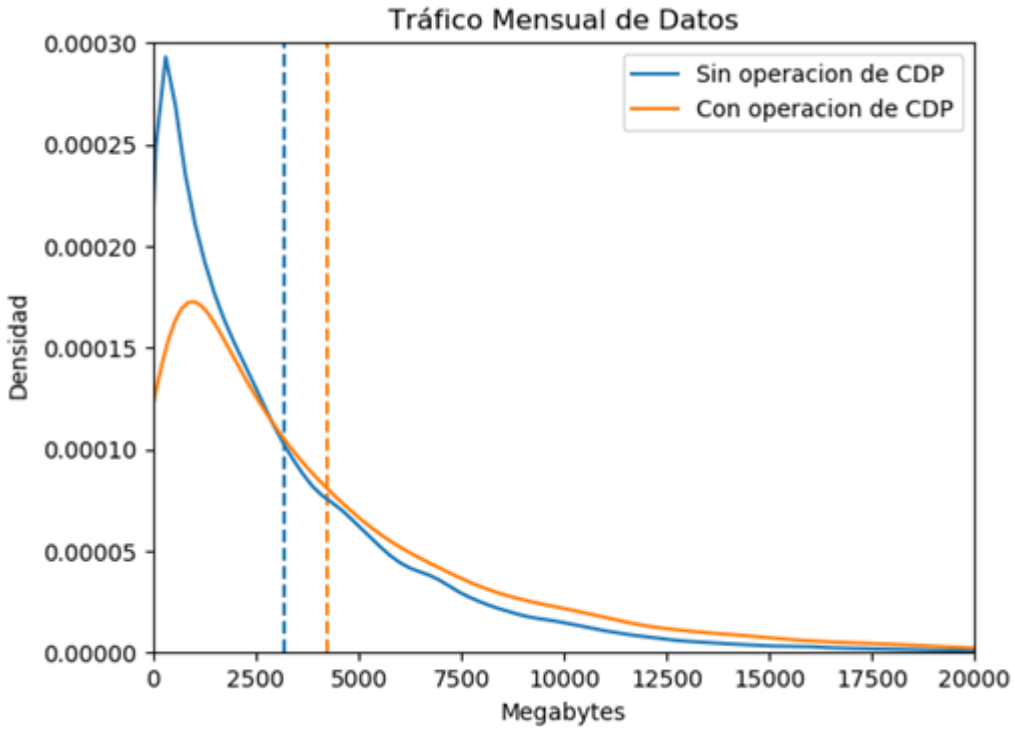


3.1.3. Tráfico de Datos

Otra variable interesante para analizar es el consumo mensual de datos de los clientes. El negocio supone que mientras más datos consuma el cliente, más tendencia tendrá a realizar una operación de cambio de plan ascendente. Pero esto no necesariamente es así todas las veces.

Se puede apreciar en este caso que aquellas personas que convirtieron en diciembre de 2019 consumieron en promedio 32,1% más megabytes que aquellos que no convirtieron.

Figura 3.5: Muestra la densidad del Tráfico Mensual de datos comparando los clientes que hicieron operación de cambio de plan y los que no, durante el mes de diciembre de 2019.

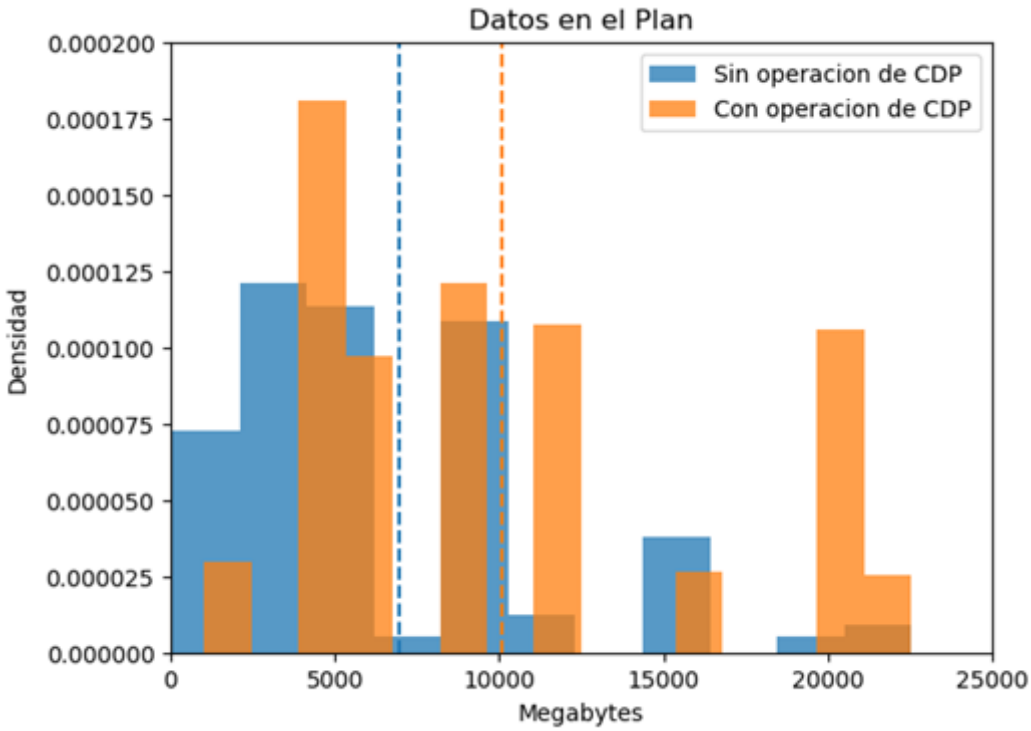


3.1.4. Datos en el Plan

Comportamiento similar se ve para la variable que muestra la cantidad de datos en el plan que tiene cada cliente. Al igual que en el apartado anterior, distinguimos entre aquellos que hicieron operación de cambio de plan de los que no.

Se puede ver que aquellos que convierten tienen en promedio un 44,2% más de datos en el plan que aquellos que no convierten (10.063 Mb vs 6.976 Mb).

Figura 3.6: Muestra histograma de Datos en el Plan comparando los clientes que hicieron operación de cambio de plan y los que no, durante el mes de diciembre de 2019.



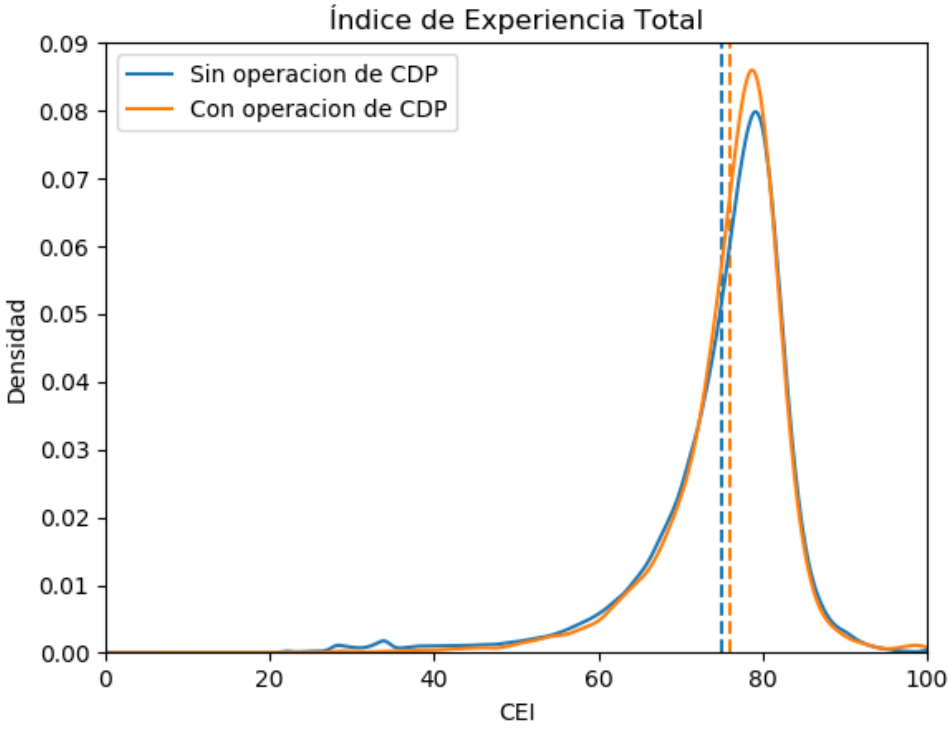
3.1.5. Índice de Experiencia Cliente

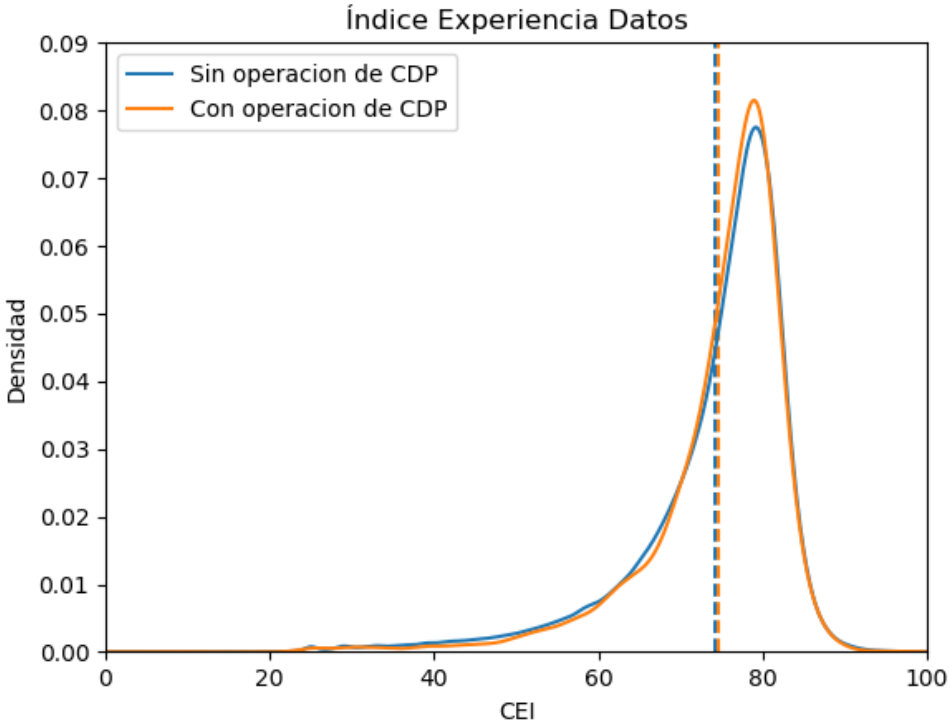
Otro indicador que resulta interesante analizar es el *Customer Experience Index* (CEI). Es un índice propiedad de un proveedor de tecnología chino, que mide en escala 0-100 la satisfacción del cliente en cuanto a los siguientes puntos relativos a la red:

- *Quality of Experience* (QoE) Web: Indicador que refleja la experiencia en el uso de páginas web
- QoE Streaming: Indicador que refleja la experiencia en el uso de streaming
- QoE Mensajería Instantánea: Indicador que refleja la experiencia en el uso de mensajería instantánea
- QoE VOIP: Indicador que refleja la experiencia en el uso de Voz IP (Voz Digital)
- QoE Archivos: Indicador que refleja la experiencia en el uso de descarga de archivos
- QoE SMS: Indicador que refleja la experiencia en el uso de SMS
- QoE Voz: Indicador que refleja la experiencia en el uso de la Voz
- Índice Experiencia Datos: Indicador que refleja la experiencia en el uso de la red, en servicios relacionados con Datos
- Índice de Experiencia Total: Indicador que refleja la experiencia en el uso de la red, en servicios relacionados con Datos, Voz y SMS

En los gráficos que se exponen a continuación se aprecia que la densidad de CEI Total y de Datos es similar en los clientes que hicieron operación de cambio de plan y los que no.

Figura 3.7: Muestra la densidad del CEI comparando los clientes que hicieron operación de cambio de plan y los que no, durante el mes de diciembre de 2019.





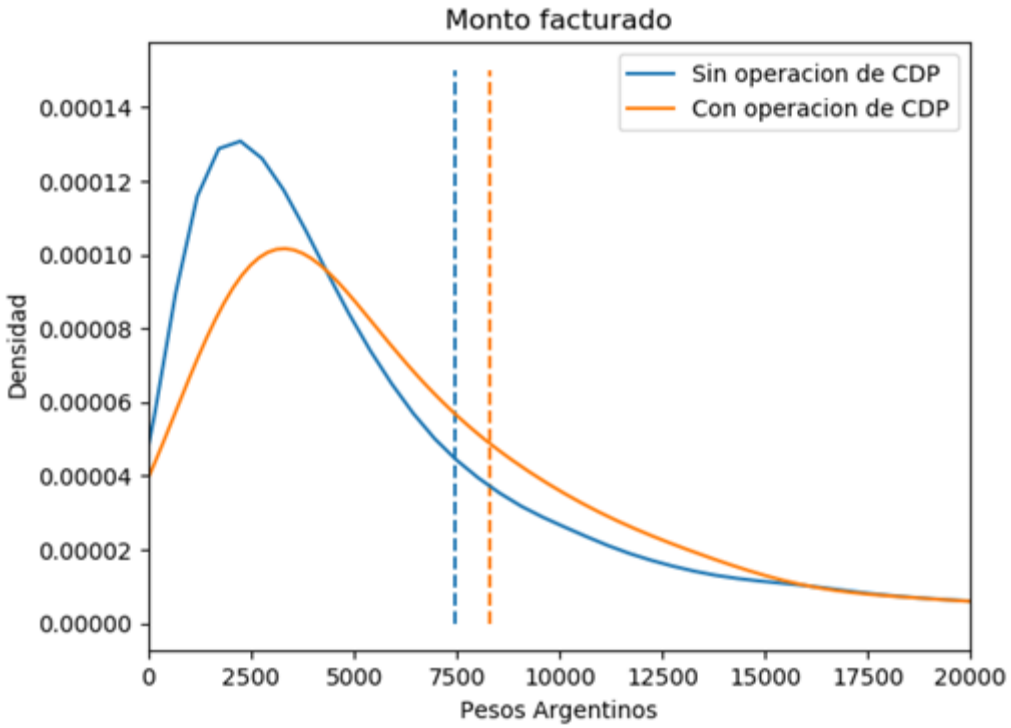
3.1.6. Monto Pagado

Resulta interesante analizar si aquellos clientes que convierten tienen un monto facturado mayor o menor a los que no convirtieron. El monto facturado al CUIT se compone del precio del plan y el agregado de cualquier excedente que consuman las líneas.

Como ya expusimos anteriormente, los clientes que convierten tienen en promedio más megabytes en su plan. Por lo cual resulta lógico que aquellos que conviertan tengan en promedio un monto facturado mayor, ya que a mayor megabytes en el plan mayor es su precio.

Para esta variable, la diferencia entre los promedios no es tan importante como la que había en los Megabytes traficados (32,1%): aquellos que convierten tienen un monto facturado 11,9% mayor a aquellos que no convierten.

Figura 3.8: Muestra la densidad del monto facturado a cada cliente en pesos argentinos, comparando los clientes que hicieron operación de cambio de plan y los que no, durante el mes de diciembre de 2019.



3.2. Limpieza del dataset (Data Wrangling)

Debido a que el ABT Móvil se encuentra mantenido y actualizado por un área específica de la dirección de Big Data, los datos se encuentran limpios de origen. Sin embargo, se optó como precaución generar una función '*ConvertNulls*', que convierte los strings "null" del dataframe a nulls nativos del lenguaje Python.

Adicionalmente, para evitar problemas de formato, se generó la función '*convert_month_dt*', con el objetivo de transformar la variable del período desde un formato 'yyyyMM' a formato

fecha en Python. Esto se logra transformando en primer lugar la fecha 'yyyyMM' a *unix_timestamp*[4], para luego transformarla en formato fecha de Python.

3.3. Selección de variables relevantes

En datasets de altas dimensiones como el que estamos utilizando, que incluye gran cantidad de covariables, la colinealidad es un problema habitual. Si entrenáramos los modelos con todas las covariables que incluye el dataset inicial del cual partimos, el algoritmo debería calcular más parámetros (problemas lineales) o realizar más cortes de los necesarios (árboles de decisión), perjudicando la capacidad predictiva del modelo (*overfitting*).

Se llevan a cabo mecanismos de regularización como ser la selección de variables para evitar los siguientes inconvenientes:

- A mayor cantidad de covariables, mayor será la variabilidad de la predicción, siendo este un efecto no deseado a la hora de entrenar un algoritmo de Machine Learning.
- Cuando se utilizan muchas covariables se ocultan correlaciones importantes, dificultando la interpretación de los modelos.

Existen varias formas de llevar a cabo esta selección, entre las que se incluyen:

- *Subset-selection*: realiza una búsqueda heurística. Es un método bruto, que entrena modelos con todas las combinatorias posibles de variables para después decidir cuál funciona mejor, de acuerdo a la métrica definida de validación. Es computacionalmente difícil al tener más de 30 covariables, por lo cual no se utilizará en este trabajo.
- *Shrinkage methods*: estos métodos realizan una regularización, penalizando la complejidad del modelo. Dicha penalización reduce las covariables incluidas en el modelo final. Dentro de estos métodos podemos encontrar a Ridge, LASSO o Elastic-Nets.

Tanto Ridge como LASSO son modelos lineales. La idea detrás de estos métodos es encontrar una recta de regresión tal que no se ajuste a los datos de entrenamiento a la perfección. Para lo cual se introduce entonces un pequeño desvio para el cálculo de la recta de regresión que se ajusta a los datos. Como resultado de agregar este pequeño desvío, se consigue una significativa reducción de la varianza del modelo. En síntesis, comenzando con un peor ajuste a los datos, se logra una mejor predicción a largo plazo.

En Ridge, técnicamente lo que hacen los modelos es minimizar la siguiente función de pérdida

$$F = \sum cuadrados\_residuales + \lambda . \left(pendiente_{recta\_regresion}\right)^2$$

Lambda puede tomar un valor entre 0 e infinito. Mientras más se acerque a cero, más parecido será al cálculo por cuadrados mínimos (recta de regresión tradicional, minimizando por el

[4] En programación, se utiliza *unix_timestamp* para representar una fecha como los segundos transcurridos desde el inicio del año 1970. Esto facilita operaciones aritméticas como la suma o diferencia entre fechas.

cuadrado de los residuos). Para lambda infinito, la pendiente de la recta de regresión tenderá asintóticamente a cero. ¿Cómo se sabe entonces qué valor de lambda usar? Se usa aquel que por el método de validación cruzada arroje la menor varianza.

Extrapolando este mismo concepto a datasets de altas dimensiones, la función sería la siguiente:

$$F = \sum cuadrados_{residuales} + \lambda . \left(pendiente_1{}^2 + pendiente_2{}^2 + \cdots + pendiente_n{}^2\right)$$

Siendo las pendientes 1 a 'n' aquellas que ajustan cada una de las 'n' covariables a los datos.

LASSO es muy similar en cuánto a su función a minimizar, pero diferente en cuanto a los resultados que se obtiene. La función de pérdida a minimizar tiene la siguiente estructura:

$$F = \sum cuadrados\_residuales + \lambda . \left|pendiente_{recta\_regresion}\right|$$

La diferencia entre Ridge y LASSO es que, al aproximarse el valor Lambda a infinito, en LASSO la pendiente tomará valor 0, cuando en Ridge solo se acerca asintóticamente a 0. Esta característica permite en datasets de altas dimensiones descartar variables irrelevantes, que son aquellas que suman muy poca información a aquello que se quiere predecir.

El método LASSO se usa para reducir la varianza en modelos que contienen muchas variables irrelevantes, mientras que Ridge suele funcionar mejor en modelos cuyas covariables son en su mayoría relevantes para la predicción. Como en este trabajo el dataset se acerca más a lo primero, se usará LASSO como método para selección de variables.

La librería SkLearn de Python tiene implementado este modelo. Como input requiere de un dataset escalado, con sus variables numéricas variando entre 0-1 y sus variables categóricas binarizadas en tantas columnas como categorías tengan dichas variables.

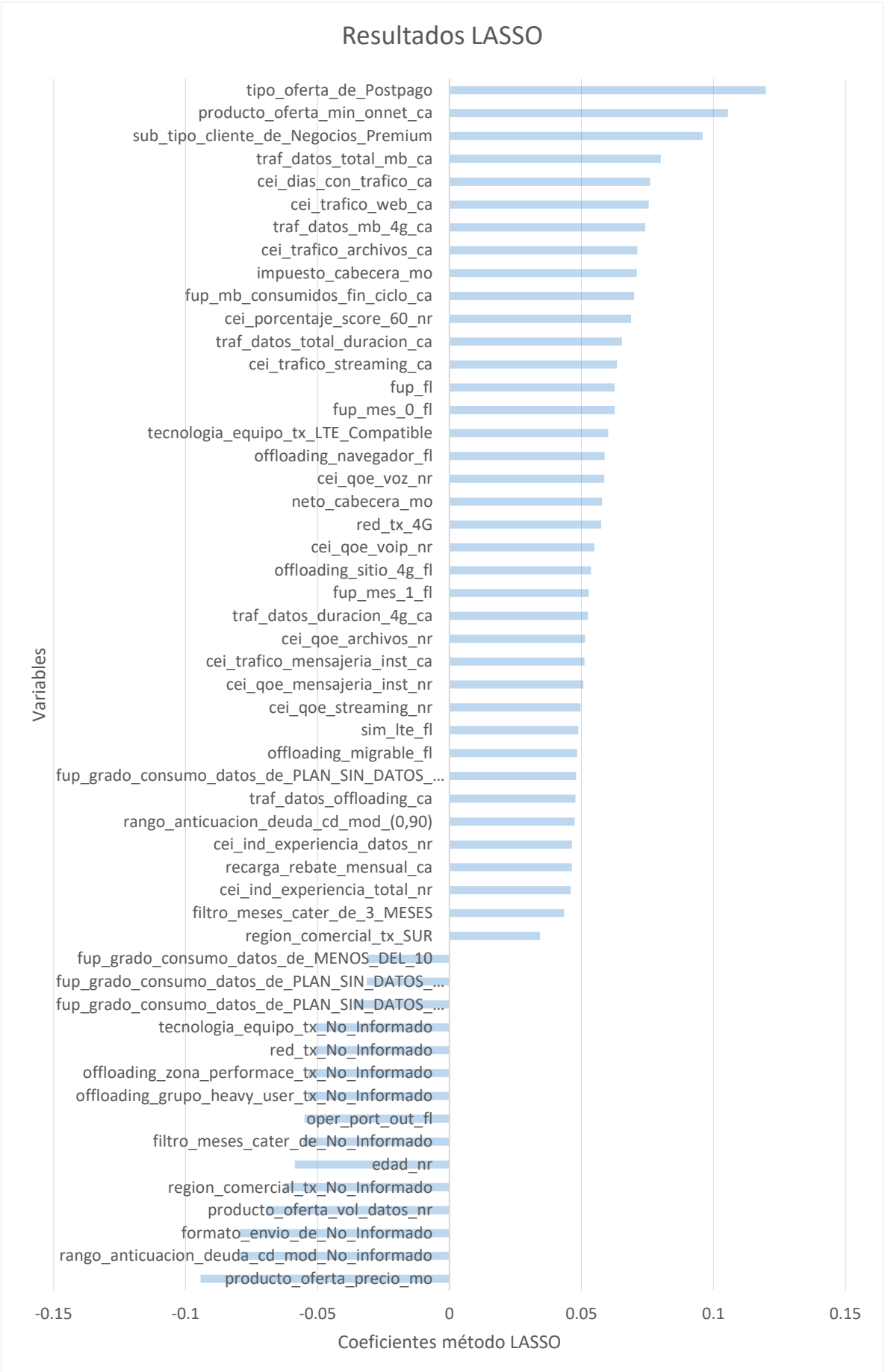
Figura 3.9: Ejemplo de cómo binariza Python las variables categóricas, para que luego puedan ingresar como input en el modelo LASSO.

Registro	Producto
1	Prepago
2	Prepago
3	Pospago
4	Prepago
5	Pospago
6	Pospago

Registro	Producto_prepago	Producto_pospago
1	1	0
2	1	0
3	0	1
4	1	0
5	0	1
6	0	1

Una vez implementado, se obtiene el siguiente resultado:

Figura 3.10: Coeficientes distintos de 0 que arroja el método Lasso para cada una de las variables relevantes. Se aprecia que, en las variables categóricas (9 seleccionadas), el método arroja la relevancia de sus respectivas binarizaciones.


Resultados LASSO
tipo_oferta_de_Postpago
producto_oferta_min_onnet_ca
sub_tipo_cliente_de_Negocios_Premium
traf_datos_total_mb_ca
cei_dias_con_trafico_ca
cei_trafico_web_ca
traf_datos_mb_4g_ca
cei_trafico_archivos_ca
impuesto_cabecera_mo
fup_mb_consumidos_fin_ciclo_ca
cei_porcentaje_score_60_nr
traf_datos_total_duracion_ca
cei_trafico_streaming_ca
fup_fl
fup_mes_0_fl
tecnologia_equipo_tx_LTE_Compatible
offloading_navegador_fl
cei_qoe_voz_nr
neto_cabecera_mo
red_tx_4G
cei_qoe_voip_nr
offloading_sitio_4g_fl
fup_mes_1_fl
traf_datos_duracion_4g_ca
cei_qoe_archivos_nr
cei_trafico_mensajeria_inst_ca
cei_qoe_mensajeria_inst_nr
cei_qoe_streaming_nr
sim_lte_fl
offloading_migrable_fl
fup_grado_consumo_datos_de_PLAN_SIN_DATOS_...
traf_datos_offloading_ca
rango_anticuacion_deuda_cd_mod_(0,90)
cei_ind_experiencia_datos_nr
recarga_rebate_mensual_ca
cei_ind_experiencia_total_nr
filtro_meses_cater_de_3_MESES
region_comercial_tx_SUR
fup_grado_consumo_datos_de_MENOS_DEL_10
fup_grado_consumo_datos_de_PLAN_SIN_DATOS_...
fup_grado_consumo_datos_de_PLAN_SIN_DATOS_...
tecnologia_equipo_tx_No_Informado
red_tx_No_Informado
offloading_zona_performace_tx_No_Informado
offloading_grupo_heavy_user_tx_No_Informado
oper_port_out_fl
filtro_meses_cater_de_No_Informado
edad_nr
region_comercial_tx_No_Informado
producto_oferta_vol_datos_nr
formato_envio_de_No_Informado
rango_anticuacion_deuda_cd_mod_No_informado
producto_oferta_precio_mo
Variables
-0.15
-0.1
-0.05
0
0.05
0.1
0.15
Coeficientes método LASSO

3.4. Creación y transformación de variables

Con el fin de mejorar la asertividad del modelo, se generaron las siguientes variables:

- Flag LTE Compatible: variable que toma valor 1 si la tecnología del equipo del cliente es LTE/4G. Para el resto de los casos toma valor 0. Corresponde a la tecnología de equipo que más efectividad tiene en cambio de plan.
- Flag Red 4G: variable que toma valor 1 si la tecnología del equipo del cliente es LTE/4G y el cliente tiene un chip 4G. Para el resto de los casos toma valor 0. Es similar al flag anterior, adicionándole el chip 4G. El objetivo es experimentar cuál de los dos flags resulta más relevante.
- Flag Meses CATER: variable que toma valor 1 si el cliente cambió su terminal hace menos de 3 meses. Para el resto de los casos toma valor 0. Se agrega esta variable ya que luego de un cambio de terminal es habitual que suceda un aumento en el consumo de datos, factor que impulsaría la operación de cambio de plan ascendente.
- Flag Región Comercial: variable que toma valor 1 si el cliente pertenece a la región Sur. Para el resto de los casos toma valor 0. Se agrega esta variable ya que desde la privatización de ENTel, en la cual la prestación de servicios quedó asignada por región a dos empresas, se priorizó comercial y tecnológicamente el desarrollo de esta región.
- Potencia: se generan las variables detalladas debajo elevadas a la potencia de 2. Esto con el fin de separar los valores y que el modelo pueda captar más información de las mismas.
    - MB Excedidos
    - Tráfico de datos total
    - Índice CEI de Experiencia Total
    - Días con CEI menor a 60
    - Días con tráfico
- Derivadas: se generan variables de tendencia. El resultado son cuatro nuevas variables por cada uno de los siguientes puntos: variable del período anterior, crecimiento del período en relación al período anterior, promedio de los últimos 3 meses, y crecimiento del período en relación al promedio de los últimos 3 meses.
    - Monto facturado
    - Impuesto cobrado
    - Precio del plan
- Derivadas a nivel CUIT: igual al procedimiento anterior, pero a nivel CUIT. Esto debido a que es habitual que el apoderado haga cambios de plan para múltiples líneas del mismo CUIT.
    - Monto facturado
    - Impuesto cobrado
    - Precio del plan

3.5. Modelos

Previo a la experimentación, es necesario planificar los modelos a evaluar (detallados en el apartado 2.3), definir la métrica con la cual compararemos los desempeños de los mismos,

evaluar división de datos en entrenamiento (*train*) y validación (*test*), ventanas de tiempo a considerar, meses a utilizar, entre otros.

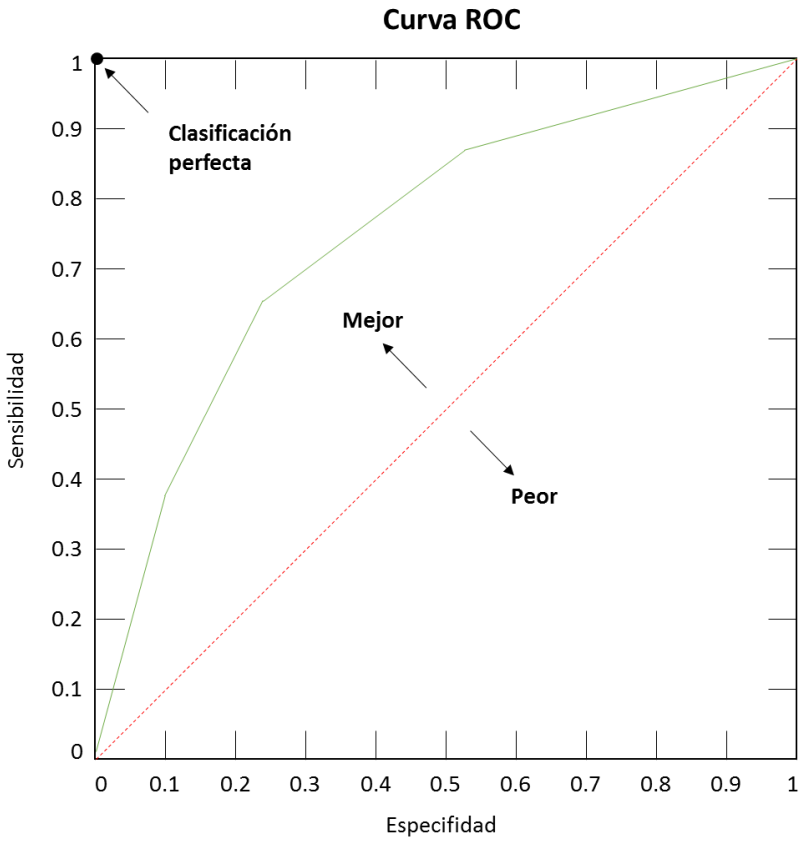
Como ya se mencionó anteriormente, los dos modelos que utilizaremos a la hora de experimentar serán el Random Forest y GBM, ambos disponibles en el framework Spark. Se entrenará un modelo de clasificación binaria, siendo la variable a predecir si el cliente hará la operación de *upselling* o no. Se denomina binario, ya que la variable a predecir puede tomar dos valores: 0 o 1.

Tabla 3.1: Ejemplo de dataset input para el modelo, en dónde se pueden ver las variables dependientes (FL_UPSELLING) e independientes (VAR_1, VAR_2, …, VAR_N).

VAR_1	VAR_2	VAR_3	(…)	VAR_N	FL_UPSELLING
					0
					0
					1
					0
					1
					0

Para medir el desempeño de los modelos, resulta conveniente usar la métrica del área bajo la curva ROC (AUC). En ella, se mide la probabilidad de que el modelo puntúe una instancia positiva aleatoria más que una negativa (Fawcett, T., 2006). La curva ROC se define graficando la tasa de verdaderos positivos en el eje y, y la tasa de falsos positivos en el eje y.

Figura 3.11: ejemplo de curva ROC. La línea roja a 45° representa lo que se denomina *random guessing*, que significa elegir de manera aleatoria. Mientras más la curva se acerque a lo que se denomina clasificación perfecta, mejor. Si la curva cae por debajo del random guessing, significa que el modelo predice peor que el azar.



Adicionalmente, calcularemos otras 3 métricas que nos servirán para saber con mayor exactitud cómo está prediciendo el modelo:

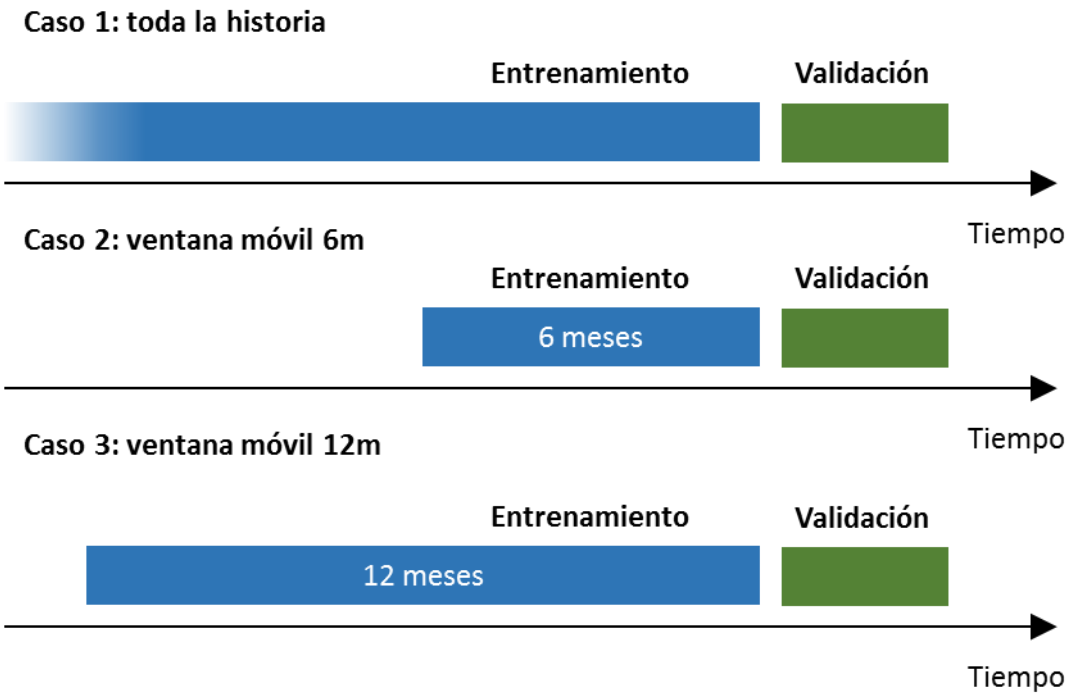
- Precision: métrica que calcula cuántos de los que el modelo predijo que iban a convertir, realmente convirtieron.
- Recall: métrica que calcula del total de los clientes que convirtieron, cuántos detectó el modelo que iban a convertir.
- F1: métrica que trata de ponderar de manera conjunta el *precision* y el *recall*. Se calcula de la siguiente manera: $F_1 = (2.prec.rec)/(prec + rec)$. El valor de F1 sube, sólo si suben las dos métricas al mismo tiempo. En el caso que solo una de ellas mejore, F1 tiende a acercarse al valor más bajo.

Para entrenar un modelo, será necesario separar el dataset disponible en entrenamiento y validación. Debido a las particularidades y dinamismo del negocio, los datos de validación deben ser lo más recientes posibles. Esto implica que usemos como datos de validación el último mes con resultados disponibles, y los anteriores como datos de entrenamiento.

Con respecto a la cantidad de meses a utilizar en el entrenamiento del modelo, debido a que actualmente solo se disponen de 8 meses de historia (siendo el último utilizado para

validación), aún no se puede trazar una conclusión válida y justificada. En un futuro, será necesario analizar si resulta mejor tomar toda la historia, o ventanas móviles de 6 o 12 meses.

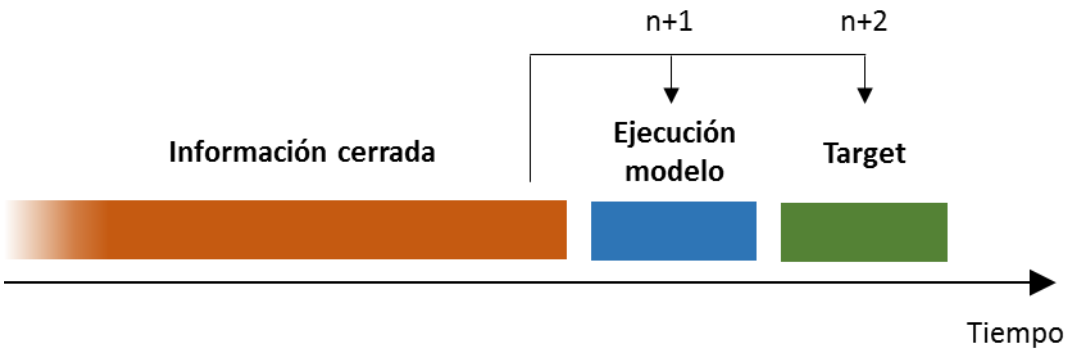
Figura 3.12: ejemplos de meses a utilizar a la hora de entrenar el modelo. Para el primer caso, cada mes que pasa se suma un mes de historia a los datos de entrenamiento. En los otros dos casos, no se suman meses, pero la ventana de tiempo se va desplazando.



Por último, es necesario definir a qué ventana de tiempo se va a predecir. A la hora de entrenar el modelo, lo ideal sería tener la información actualizada hasta el último día. Pero esto es muy costoso, tanto computacionalmente como a nivel recursos humanos, ya que las bases que utilizamos como input siguen teniendo un componente manual.

Actualmente el tablón ABT Suscripción Móvil se actualiza aproximadamente el día 10 de cada mes, agregándose al mismo la información cerrada del mes anterior. Esto implica que, a la hora de correr el modelo, uno está utilizando información cerrada del mes anterior para predecir el comportamiento de los clientes durante el próximo mes.

Figura 3.13: muestra la diferencia temporal entre los datos provistos del tablón ABT Suscripción Móvil y la campaña propiamente dicha (*target*).



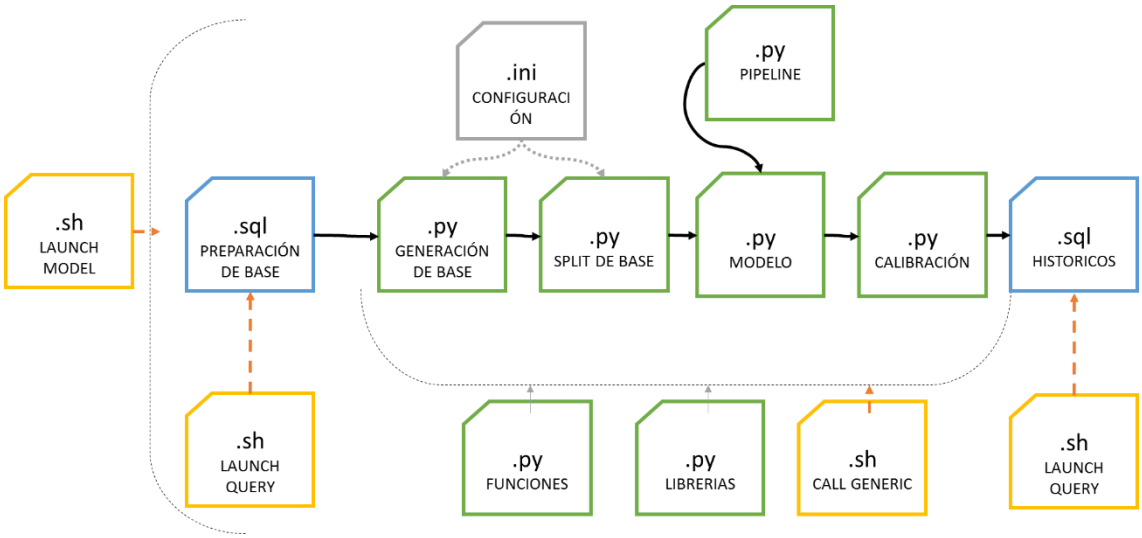
De esto se desprende que al generar nuestro flag a predecir, se deberá considerar una predicción a dos meses vista. Debido al desbalanceo de clases (variable a predecir con muchos ceros y pocos unos), resulta conveniente predecir a n+1 y n+2. Con esto logramos sumar casos positivos, balanceando un poco más las clases y aumentando el poder de predicción del modelo.

En el segmento B2C, el 82% de las operaciones de cambio de plan ascendente concretadas se realizaron en el canal Televentas-OUT. Por esto, a la hora de entrenar el modelo, se utiliza como input la base enviada a campañas y luego se extrapola al resto del parque. En el caso de B2B, únicamente el 6,5% de las operaciones mensuales de cambio de plan se concretan en el canal Televentas-OUT. El resto se realizan en lo que anteriormente se mencionó como canales reactivos. Por este motivo, se entrenará el modelo sin filtrar por aquellos enviados a campañas. Es decir, se entrenará con información de todo el parque, para predecir la operación de cambio de plan ascendente independientemente del canal por el cual se realiza.

3.6. Automatización

Como se mencionó anteriormente, para productivizar el modelo es necesario automatizarlo. Para ello, se plantea el siguiente esquema, con el fin de poder ejecutarlo en un servidor de forma remota sin intervención humana.

Figura 3.14: esquema de script a ejecutar en el servidor. En amarillo figuran aquellos archivos escritos en bash para ser ejecutados en la consola del servidor. En azul, archivos con extensión SQL. En verde, archivos con extensión py, programados con la librería PySpark. En gris, archivo para configurar la separación del dataset en Train y Test.



A continuación, detallaremos qué realiza cada uno de estos pasos:

1. Launch Model (.sh): es un archivo programado en lenguaje bash, en dónde se establecen todas las instrucciones o pasos que debe seguir el modelo para ejecutarse satisfactoriamente. Únicamente ordena los pasos, y luego accede a Launch Query o a Call Generic para ejecutar los correspondientes archivos. Adicionalmente, genera un registro que se puede leer por consola, con el fin de saber en qué paso está y si hubo algún error en la ejecución.
2. Launch Query (.sh): archivo programado en lenguaje bash. Tiene 3 funciones principales: generar un registro de ejecución, conectarse con Hive para ejecutar las consultas SQL y detener el proceso en caso que haya habido algún error.
3. Call Generic (.sh): archivo programado en lenguaje bash. Tiene 3 funciones principales: generar un registro de ejecución, conectarse con Spark y definirle los parámetros que se van a utilizar en la ejecución (memoria y núcleos del contenedor a crear, cantidad de nodos a utilizar), y detener el proceso en caso que haya habido algún error.
4. Funciones y Librerías (.py): son dos archivos programados en lenguaje Python, generados con el objetivo de agrupar todas las funciones y librerías utilizadas en los modelos. Esto, para evitar errores a la hora de importar las librerías necesarias, y no duplicar código de funciones en varios archivos.
5. Preparación de base (.sql): archivo programado en lenguaje SQL. Genera el dataset que luego ingresará como input al modelo, identificando las variables a predecir. El objetivo es que este dataset también se pueda utilizar para realizar otros modelos de B2B, por lo cual debe ser lo más genérico posible.
6. Generación de base (.py): archivo programado en Python. Realiza los siguientes pasos:
    a. Selección de variables, en base a lo analizado anteriormente
    b. Data Wrangling: limpieza de datos

   c. Agrega variables detalladas anteriormente
   d. Filtra el dataset en caso de ser necesario. Para este caso, debido a que se va a utilizar la totalidad el parque para entrenar, no se va a hacer uso de este apartado.
   e. Guarda la base generada en Hadoop para que el siguiente paso la pueda importar.
7. Split de base (.py): archivo programado en Python. Separa el dataset generado anteriormente en tres: target, test y train.
8. Configuración (.ini): archivo de configuración en dónde se define qué período se considera para entrenar, cuál para testear y cuál es el target. Se genera este archivo para facilitar testeos con diferentes meses y facilitar futuras corridas cuando el modelo se encuentre productivizado.
9. Pipeline (.py): archivo programado en Python. Define la variable objetivo, las covariables a considerar y transforma las variables categóricas en binarias, como se explicó anteriormente.
10. Modelo (.py): archivo programado en Python. Ejecuta el modelo predictivo y lo guarda en Hadoop. Arroja por consola indicadores relevantes para la evaluación de los modelos: AUC, Recall, Precision, F1.
11. Calibración (.py): archivo programado en Python. Realiza una regresión isotónica, para ajustar las probabilidades que arroja el modelo a probabilidades reales del mes anterior. Esto, con el objetivo de incorporar el modelo al motor de selección de ofertas.
12. Históricos (.sql): archivo programado en Python. Genera un histórico de los valores que calcula el modelo a lo largo de los meses, para poder evaluar el impacto del modelo a lo largo de los meses.

4. Evaluación de modelos e impacto

4.1. Ejecución de modelos y principales indicadores

Habiendo introducido previamente los modelos predictivos a utilizar, y adicionalmente todos los parámetros a contemplar a la hora de entrenar el mismo, se comienza con la experimentación con el fin de evaluar el que tiene mejor performance.

Tabla 4.1: Detalle de los intentos de corrida del modelo, con sus respectivas métricas de salida y comentarios.

Intento	Configuración	AUC	Comentarios
1	Modelo: Random Forest Target: 2019.09 *numTrees=20* *maxDepth=5* *impurity='entropy'* *minInstancesPerNode=1* *seed=42*	Train: 0.6819 Test: 0.6613	Primer modelo ejecutado. Es la base para luego ir mejorando la predicción.
2	Modelo: Random Forest Target: 2019.09 *numTrees=20* *maxDepth=5* *impurity='entropy'* *minInstancesPerNode=1* *seed=42*	Train: 0.6793 Test: 0.6565	Se remueven del modelo algunas variables con poco peso en el intento anterior[5].
3	Modelo: Random Forest Target: 2019.09 *numTrees=20* *maxDepth=5* *impurity='entropy'* *minInstancesPerNode=1* *seed=42*	Train: 0.6737 Test:0.6527	Se remueven del modelo otras variables poco importantes[6].
4	Modelo: GBT Target: 2019.09 *numTrees=20* *maxDepth=5* *minInstancesPerNode=1* *seed=42*	Train: 0.7075 Test: 0.6792	Primera prueba con modelo Gradient-boosted tree. Se utilizan todas las variables, excepto las creadas (detalle en apartado 3.4.)

[5] Se remueven las siguientes variables: *recarga_rebate_mensual_ca, fl_region_comercial_tx, offloading_navegador_fl, sim_lte_fl, offloading_migrable_fl, offloading_sitio_4g_fl, edad_nr, fl_filtro_meses_cater, oper_port_out_fl, impuesto_cabecera_mo*
[6] Se remueve solo: *edad_nr, fl_filtro_meses_cater, oper_port_out_fl, impuesto_cabecera_mo*

5	Modelo: GBT Target: 2019.09 *maxIter=30* *maxDepth=5* *minInstancesPerNode=1* *seed=42*	Train: 0.7096 Test: 0.6809	Se agrega el parámetro maxIter = 30 y se remueve numTrees = 20
6	Modelo: GBT Target: 2019.09 *maxIter=30* *maxDepth=5* *minInstancesPerNode=1* *seed=42*	Train: 0.7095 Test: 0.6790	Se agrega a lo anterior las variables creadas a nivel ANI (detalle en apartado 3.4.)
7	Modelo: GBT Target: 2019.09 *maxIter=30* *maxDepth=5* *minInstancesPerNode=1* *seed=42*	Train: 0.7095 Test: 0.6790	Se remueven las variables de la iteración anterior cuya importancia luego de corrido el modelo es 0
8	Modelo: GBT Target: 2019.09 *maxIter=30* *maxDepth=5* *minInstancesPerNode=1* *seed=42*	Train: 0.9145 Test: 0.6660	Se entrena únicamente con aquellos que fueron a campañas de Upselling para evaluar qué arroja el modelo.
9	Modelo: GBT Target: 2019.09 *maxIter=30* *maxDepth=5* *minInstancesPerNode=1* *seed=42*	Train: 0.6963 Test: 0.6576	Se agrega a lo ejecutado en la iteración 7 las variables creadas a nivel CUIT (detalle en apartado 3.4.)
10	Modelo: GBT Target: 2019.09 *maxIter=30* *maxDepth=5* *minInstancesPerNode=1* *seed=42*	Train: 0.7033 Test: 0.6749	Se corrige un pequeño bug en las fechas de train, test y target.
11	Modelo: GBT Target: 2019.09 *maxIter=70* *maxDepth=5* *minInstancesPerNode=14* *seed=42*	Train: 0.7088 Test: 0.6743	Se cambian los hiperparámetros del modelo.

12	Modelo: GBT Target: 2019.09 *maxIter=100* *maxDepth=7* *minInstancesPerNode=14* *seed=42*	Train: 0.7555 Test: 0.6758	Se vuelven a modificar los hiperparámetros y se hace downsampling[7].
13	Modelo: GBT Target: 2019.09 *maxIter=100* *maxDepth=7* *minInstancesPerNode=14* *seed=42*	Train: 0.7322 Test: 0.6773	Se mantienen los hiperparámetros pero no se aplica downsampling.
14	Modelo: GBT Target: 2019.10 *maxIter=30* *maxDepth=5* *minInstancesPerNode=1* *seed=42*	Train: 0.6959 Test: 0.6591	Iteración igual a la 10, pero se cambia el mes target a 2019.10
15	Modelo: GBT Target: 2019.10 *maxIter=30* *maxDepth=5* *minInstancesPerNode=1* *seed=42*	Train: 0.6959 Test: 0.6591	Se remueve la variable Producto_Oferta_Precio_MO, que viene incompleta de origen.
16	Modelo: GBT Target: 2019.12 *maxIter=30* *maxDepth=5* *minInstancesPerNode=1* *seed=42*	Train: 0.7085 Test: 0.6796	Se entrena modelo para hacer prueba piloto durante el mes de febrero-2020.

Aprendizajes

Luego de realizar las iteraciones, se puede concluir lo siguiente:

- No siempre remover variables mejora la performance del modelo. En las iteraciones 2 y 3 se ve que al remover las variables empeora el modelo. Esto se debe a que por más que sean poco relevantes para el modelo, el mismo logra captar algo de información de las mismas.

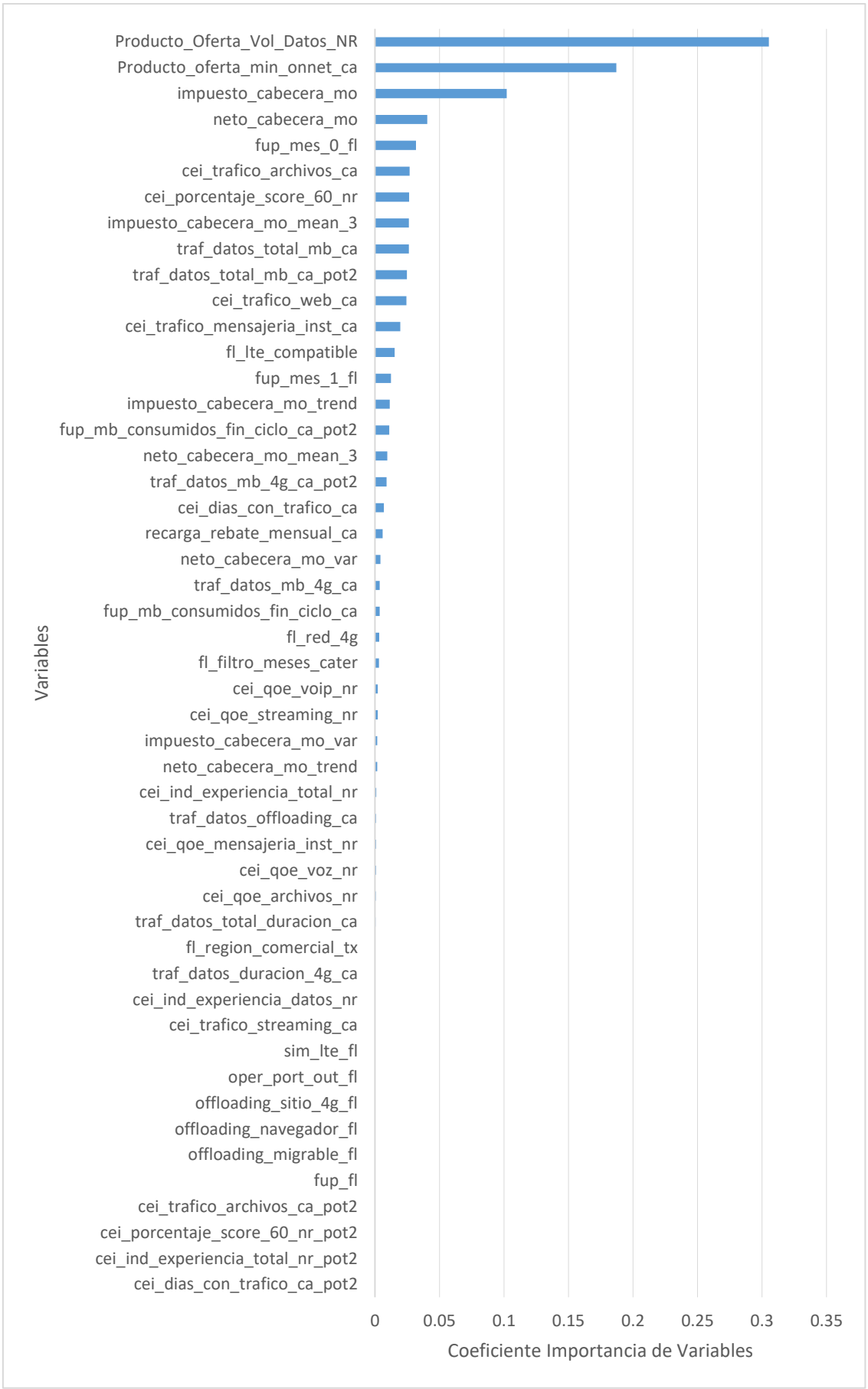
[7] El objetivo de hacer *downsampling* es aumentar la proporción de casos positivos en nuestro dataset. Existen varias técnicas, entre ellas la de duplicar casos positivos o eliminar casos negativos redundantes. Es utilizado en algunos modelos específicos para aumentar su poder predictivo.

- Para los datos a predecir, es mejor el modelo *Gradient-boosted tree* que *Random Forest* (intento 4 mejora +0.0179 el AUC de Test con respecto al intento 1). Uno de los motivos que puede explicar este fenómeno es que para este problema puntual los datos de test / target son similares a los de entrenamiento, y el modelo de *gradient-boosted tree* performa mejor en este caso. Si las características de los consumidores que acceden a realizar una operación de cambio de plan ascendente variara sustancialmente de mes a mes, probablemente el modelo de *Random Forest* tendría una mejor performance.
- Las variables creadas (detalle en apartado 3.4.) no mejoran la performance del modelo. Este resultado llamó la atención, ya que choca con una hipótesis que nos había transmitido el negocio al comenzar el armado del mismo: que al analizar los comportamientos de una cuenta de manera colectiva (es decir, teniendo en cuenta todas sus líneas asociadas), puede llevar a que líneas que individualmente son poco propensas estén mejor ponderadas por el modelo, y luego atraerlos con una oferta multilínea. Se cree que el motivo por el cual estas variables creadas no funcionaron es que existe un '*modelo de titularidad*', que a cada ANI le asigna una línea telefónica con poder de decisión sobre todos los productos asociados a la cuenta. Y, cuando un cliente es enviado a campañas, se contacta a este ANI con poder de decisión, con lo cual esta *visión global* de la cuenta estaría saldada de este modo, estando los datos del dataset sesgados por nuestra forma de gestionar a estos clientes.
- Al aplicar *downsampling*, se mejora sustancialmente el AUC de Train pero no el de Test. Técnicamente este fenómeno se denomina *overfitting*, y aparece cuando el modelo entrenado se ajusta demasiado a los datos de entrenamiento. Es importante reducir este efecto, ya que cuando sucede afecta la capacidad de realizar predicciones futuras del modelo.
- Realizando una óptima selección de hiperparámetros se puede generar un gran impacto en la predicción. Este es uno de los requisitos que exige el modelo *gradient-boosted tree* para superar en performance a *Random Forest*.

4.2. Importancia de variables

Al entrenar un modelo, es importante entender cuál es la importancia que el mismo le está asignando a cada una de las variables. A continuación, se expone el detalle de la importancia de variables para la última corrida.

Figura 4.1: Detalle de importancia de variables para la corrida con target 2019.12. Mientras más se acerque el valor a 0, menor importancia tendrá la variable en el modelo entrenado.


Producto_Oferta_Vol_Datos_NR
Producto_oferta_min_onnet_ca
impuesto_cabecera_mo
neto_cabecera_mo
fup_mes_0_fl
cei_trafico_archivos_ca
cei_porcentaje_score_60_nr
impuesto_cabecera_mo_mean_3
traf_datos_total_mb_ca
traf_datos_total_mb_ca_pot2
cei_trafico_web_ca
cei_trafico_mensajeria_inst_ca
fl_lte_compatible
fup_mes_1_fl
impuesto_cabecera_mo_trend
fup_mb_consumidos_fin_ciclo_ca_pot2
neto_cabecera_mo_mean_3
traf_datos_mb_4g_ca_pot2
cei_dias_con_trafico_ca
recarga_rebate_mensual_ca
neto_cabecera_mo_var
traf_datos_mb_4g_ca
fup_mb_consumidos_fin_ciclo_ca
fl_red_4g
fl_filtro_meses_cater
cei_qoe_voip_nr
cei_qoe_streaming_nr
impuesto_cabecera_mo_var
neto_cabecera_mo_trend
cei_ind_experiencia_total_nr
traf_datos_offloading_ca
cei_qoe_mensajeria_inst_nr
cei_qoe_voz_nr
cei_qoe_archivos_nr
traf_datos_total_duracion_ca
fl_region_comercial_tx
traf_datos_duracion_4g_ca
cei_ind_experiencia_datos_nr
cei_trafico_streaming_ca
sim_lte_fl
oper_port_out_fl
offloading_sitio_4g_fl
offloading_navegador_fl
offloading_migrable_fl
fup_fl
cei_trafico_archivos_ca_pot2
cei_porcentaje_score_60_nr_pot2
cei_ind_experiencia_total_nr_pot2
cei_dias_con_trafico_ca_pot2
Variables
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
Coeficiente Importancia de Variables

Si se agrupan las 10 variables de corte más importantes en facturación, tráfico y experiencia se obtiene lo siguiente.

Tabla 4.2: Variables más importantes para el modelo según su peso agrupado por las categorías Facturación, Tráfico y Experiencia.

Categoría	Variables	Peso
Facturación	Valor Neto Facturado	6%
Tráfico	- Megas del plan - Megas que llevaba consumidos el último día del ciclo - Minutos on net incluidos en el plan - Flag si fupea o no en el mes actual - Flag si fupea o no en el mes n-1 - Tráfico de datos total	83%
Experiencia	- CEI Total traficado en megas en un mes descarga de archivos - CEI Porcentaje de días que el cliente tuvo una experiencia mala en la red, con score menor o igual a 60 puntos - CEI Total traficado en megas en un mes en páginas web	11%

4.3. Evaluación teórica de Impacto

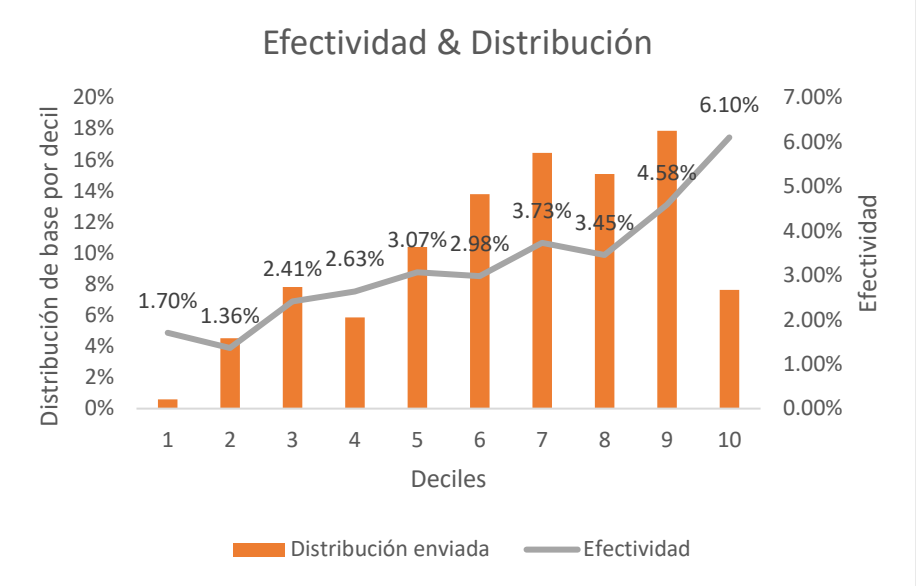
Con el fin de empezar a utilizar el modelo para las campañas mensuales, el negocio solicitó realizar un análisis teórico de impacto. A continuación, se expone parte de la información entregada.

Este análisis se realizó en el mes de Enero de 2020, evaluando resultados de la campaña del mes anterior (Diciembre de 2019). Para lo cual se corrió el modelo prediciendo las operaciones que iban a suceder durante dicho mes. Se normalizarán los resultados para asegurar la confidencialidad de los datos de parque, operaciones y facturación.

El primer paso es ordenar la base total en deciles, siendo el decil 10 el más propenso a realizar la operación de cambio de plan ascendente, y el decil 1 el menos propenso. Esto se hace a fin de entender si el modelo ordena acorde a lo esperado.

Si observamos la base enviada, se aprecia un ordenamiento claro de las efectividades, logrando en el decil 10 una efectividad de 6,10% contra un 1,70% del decil 1. Teniendo en cuenta que el promedio de la efectividad de la campaña fue de 3,56%, se podría llegar a casi duplicar la efectividad si pudiéramos enviar únicamente base del decil 10.

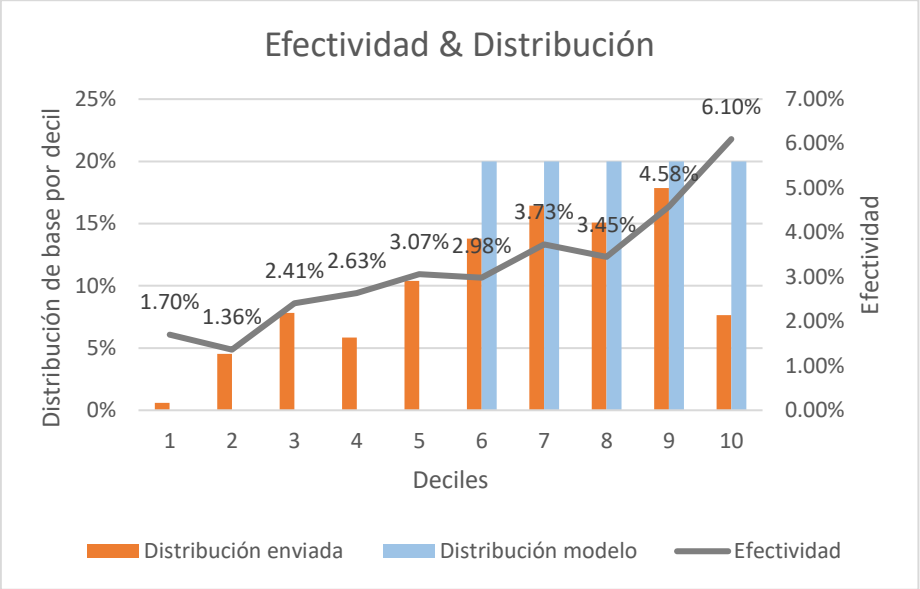
Figura 4.2: Muestra la distribución por decil de la totalidad de la base enviada a campañas, con la efectividad que tuvo cada uno de ellos



Se envía recurrentemente a campañas el 6,6% del parque B2B. Si enviáramos únicamente clientes del decil 10, estaríamos enviando todos los meses el 66% de los datos pertenecientes a dicho universo. Esto sería inviable si se realizara de manera recurrente, ya que existen políticas de Contactabilidad y otros filtros (por ejemplo el Registro No Llame, clientes con mora, etc.) que terminan reduciendo el universo disponible para enviar a campañas.

Por lo expuesto anteriormente, se opta por un análisis conservador, simulando un envio de base con deciles mayores o iguales a 6. Para este caso, se asume que la efectividad por decil se mantiene constante.

Figura 4.3: Se aprecia en naranja la distribución por decil de la totalidad de la base enviada a campañas, y en azul una simulación de lo que se hubiera mandado por modelo. Como se menciona anteriormente, se asume que la efectividad se mantiene constante



Para hacer un análisis de impacto será necesario calcular el diferencial de ingresos generados, que se calcula como:

$$\Delta Ingresos = \Delta(Precio.Cantidad) = (P_{simul}.Q_{simul}) - (P_{real}.Q_{real})$$

Se verifica un aumento de 34% en los ingresos mensuales de campaña, debido a los siguientes dos factores:

- Incremento en la efectividad de la campaña de un +17% (equivalente a +0,61 pp.) por envio de deciles más propensos.
- Adicionalmente, debido a que estos clientes incorporados traen asociado un mayor diferencial de ingreso por campaña, cada operación adicional traería un +14% más de ingresos.

Puesto que la campaña se gestiona en un *call-center* en dónde se paga por horas posición y no por comisión de venta, se considera este ingreso adicional como ganancia neta, sin necesidad de restar ningún costo adicional.

Habiendo presentado estos resultados al negocio, se acuerda para la campaña de Febrero de 2020:

- Envio de 50% de la base a modo testigo, cortando según métodos tradicionales.
- Envio del 50% restante priorizando por el modelo, enviando únicamente a los deciles más propensos. Debido a que enero fue un mes con bajas ventas, se nos solicita enviar únicamente decil 10 para recuperar parte de lo perdido. Teniendo en cuenta que existe cantidad suficiente para este envio, se accede a realizarlo de manera

excepcional[8]. Debido a esto, veremos un impacto todavía mayor cuando analicemos los resultados de la campaña.

4.4. Evaluación real de Impacto

En el mes de Marzo de 2020 se procede a evaluar el resultado de la campaña de Febrero, en el cuál se envió como se mencionó anteriormente dos bases: una cortando de la manera tradicional y la otra cortando por el modelo predictivo.

Luego de realizar el corte, la distribución de base por decil quedó de la siguiente manera:

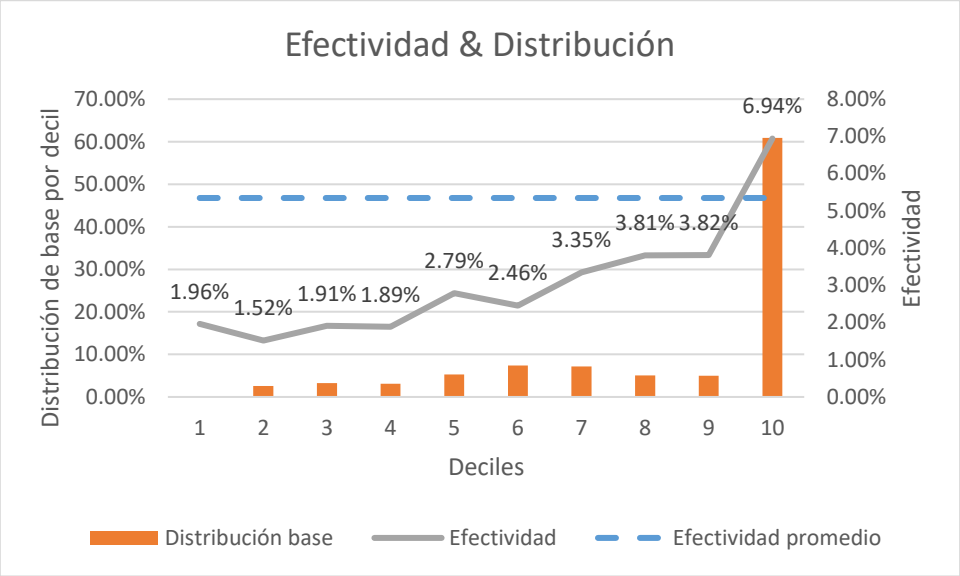
Tabla 4.3: Distribución de cada una de las bases por decil

Decil	Base Tradicional	Base modelo	Base total
1	0.41%	0,00%	0.17%
2	6.31%	0,00%	2.61%
3	7.93%	0,00%	3.28%
4	7.60%	0,00%	3.14%
5	12.83%	0,00%	5.30%
6	17.83%	0,00%	7.37%
7	17.35%	0,00%	7.17%
8	12.34%	0,00%	5.10%
9	12.02%	0,00%	5.00%
10	5.38%	100,00%	60.88%
% de base total	**41.31%**	**58.69%**	

Analizando primero la totalidad de la base, se aprecia la siguiente distribución por deciles:

[8] Como se expuso anteriormente, no es conveniente enviar recurrentemente únicamente clientes del decil 10, ya este envio no se puede mantener constante a lo largo del tiempo. Se recomienda enviar un gran porcentaje de base buena y otro poco de base no tan efectiva, para mantener una constancia en la efectividad mensual.

Figura 4.4: Se aprecia en naranja la distribución por decil de la totalidad de la base enviada a campañas, y en gris la efectividad que se logró en cada uno de los deciles.



La efectividad promedio de la campaña fue de 5,34%, impulsada fuertemente por la gran cantidad de base enviada del decil 10. Si lo comparamos contra la efectividad de diciembre de 2019, que fue de 3,56%, se aprecia un gran incremento de la efectividad a causa del modelo (+50%).

Figura 4.5: Muestra la base aperturada por el corte tradicional y el corte por el modelo.

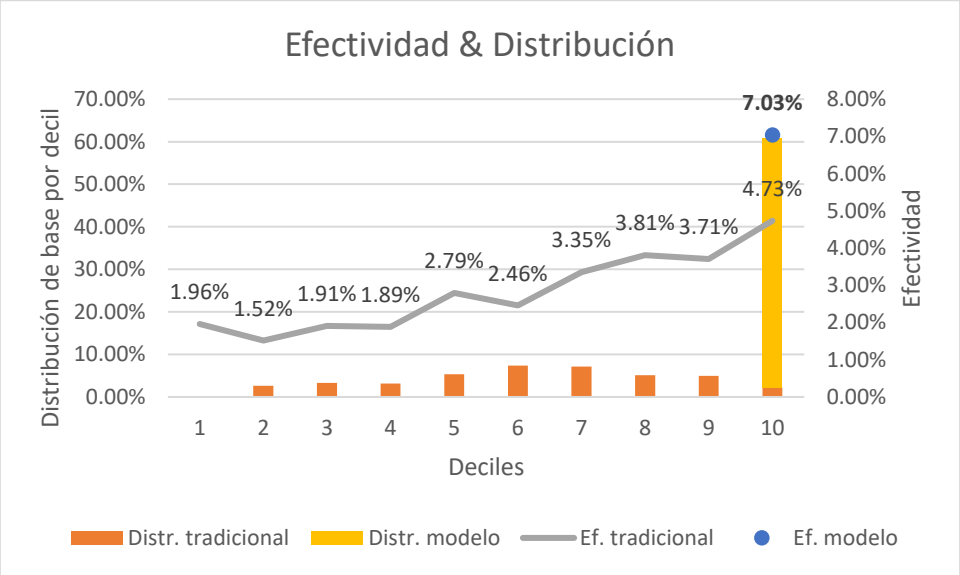


Tabla 4.4: Efectividades de cada una de las bases por decil

Decil	Base Tradicional	Base modelo	Base total
1	1.96%	-	1.96%
2	1.52%	-	1.52%
3	1.91%	-	1.91%
4	1.89%	-	1.89%
5	2.79%	-	2.79%
6	2.46%	-	2.46%
7	3.35%	-	3.35%
8	3.81%	-	3.81%
9	3.71%	-	3.82%
10	4.73%	7,03%	6.94%
Total	**2.95%**	**7,03%**	**5,34%**

Para hacer un análisis de impacto será necesario calcular el diferencial de ingresos generados, que se calcula como:

$$\Delta Ingresos = \Delta(Precio.Cantidad) = (P_{simul}.Q_{simul}) - (P_{real}.Q_{real})$$

En un entorno real, se verifica un aumento de 60% en los ingresos mensuales de campaña, debido a los siguientes dos factores:

- Incremento en la efectividad de la campaña de un +139% (equivalente a +4,09 pp.) por envio de deciles más propensos.
- Disminución de un -33% de ingresos unitarios percibidos por cliente que realizó operación de cambio de plan ascendente.

Se aprecia que el aumento de efectividad viene en parte acompañado por una disminución del delta ARPU unitario. Esto se debe a que el modelo únicamente está buscando conseguir más operaciones, sin importar el diferencial de ingreso que generan las mismas. A pesar de esto, el aumento en la efectividad más que compensa la caída en el diferencial de ingresos, por lo cual consideramos la implementación del modelo como un caso de éxito.

En futuras recalibraciones del mismo, se deberá analizar si es conveniente priorizar la maximización de las ganancias (contemplando tanto las operaciones como el diferencial de ingresos), o continuar prediciendo tal cual se diseñó en este trabajo. Si se decidiera ir por la maximización de las ganancias, se deberá incorporar al dataset la oferta a proponer a cada uno de los clientes, y adicionalmente se deberá armar el flag de forma tal que contemple las operaciones que generen un alto diferencial de ingreso. Esto desbalancearía aún más el dataset, empeorando la performance del modelo.

5. Conclusiones

En este trabajo se muestra cómo puede impactar positivamente la implementación de un modelo de *Machine Learning* para predecir una operación de cambio de plan ascendente en una empresa de telecomunicaciones. Se observa cómo, identificando a los clientes más propensos, es posible lograr aumentos notables en los ingresos mensuales de campañas. Adicionalmente, generando la confianza suficiente en el modelo, será posible productivizar el mismo en otros canales de venta, potenciando aún más su uso. En un mercado saturado de líneas móviles, es una excelente alternativa para captar ingresos erosionados por la inflación.

El modelo ayuda a la compañía a concentrar su fuerza de ventas en los clientes más propensos, permitiendo una mayor eficiencia en la operación y aumento de las ganancias. Debido al modelo de gestión que tiene B2B, en dónde se le abona al representante según las horas hombres trabajadas y no por comisión, todo aumento de la efectividad impacta directa y linealmente en las ganancias de la operación.

En cuanto al uso que se le va a dar al modelo en el futuro, será necesario tener los siguientes puntos en cuenta. En primer lugar, que no es conveniente el envio recurrente de la totalidad de la base propensa, debido a que esos clientes se sentirán hostigados y a su vez porque esto no es sustentable en el tiempo. En caso que el negocio promueva este comportamiento, se podrá agregar como input al modelo el *feedback* del cliente y también la cantidad de contactos que tuvimos con el mismo en los últimos 12 meses (esto ya se hace en B2C, debido al elevado volumen que se envía mensualmente a las campañas). En segundo lugar, que debido a que el flag está construido teniendo en cuenta la totalidad de las operaciones de cambio de plan ascendente, será importante estimular su uso en el resto de los canales (recordando que únicamente el 6,5% de las operaciones mensuales se concretan en el canal Televentas-OUT).

Cuando en uno de los primeros apartados se habló de la motivación de este trabajo, el punto que más peso tuvo fue la convergencia tecnológica. El principal desafio para el año 2020 es comenzar a ofrecer paquetes multiproducto, por lo cual resulta necesario analizar cómo combinar este modelo con el resto para entregarle al cliente una oferta lo más adaptada a sus necesidades.

En este trabajo se evaluaron los modelos de *Random Forest* y *Gradient-boosted tree*, pero en un futuro, con la totalidad de los datos migrados, se podrá profundizar en redes neuronales o un ensamble entre un modelo lineal y uno de árboles. La única dificultad para hacer esto es la necesidad de realizar el trabajo en pySpark, que hasta el momento no tiene una documentación tan extensa para consultar. Esto hace que el desarrollo se demore más, y será necesario analizar si el tiempo destinado a esto compensa las posibles ganancias a obtener (considerando también el costo de oportunidad de dejar de hacer otras tareas).

Será necesario hacer un seguimiento mensual del modelo por dos motivos. En primer lugar, para verificar que siga ordenando correctamente, manteniendo su efectividad. Teniendo en cuenta que siempre se va a optar por el envio de los deciles más propensos, será necesario

enviar recurrentemente un 10% de base testigo, para evaluar cómo performa el modelo en aquellos clientes que predice con baja efectividad. En segundo lugar, ya que debido a la constante evolución que está teniendo el área, algunas de las variables que consideramos como input para el modelo pueden mutar o desaparecer.

Como se mencionó en el apartado anterior, el modelo únicamente va en búsqueda de obtener la mayor cantidad de operaciones, sin considerar la ganancia total obtenida. Será parte de otro análisis entender si fuera conveniente cambiar el foco del modelo (flag a predecir).

Para posteriores trabajos, en dónde se cuente con más historia acumulada, se podrá analizar cuál es la ventana de tiempo óptima a utilizar para el entrenamiento del modelo y a qué clientes ofrecerles un descuento como forma de impulsar la conversión, que resulta de gran interés para el negocio.

6. Bibliografía

- Decreto 1340/2016. Boletin Oficial de la República Argentina. Buenos Aires, 30 de diciembre de 2016. Fuente: http://servicios.infoleg.gob.ar/infolegInternet/anexos/270000-274999/270115/norma.htm
- Ahmad, A.K., Jafar, A. & Aljoumaa, K. (2019). Customer churn prediction in telecom using machine learning in big data platform. *J Big Data* **6**, 28. https://doi.org/10.1186/s40537-019-0191-6
- Swetha P., Dayananda R. B. (2020). Customer Churn Prediction and Upselling using MRF (Modified Random Forest) technique. *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*. http://www.ijitee.org/wp-content/uploads/papers/v9i3/C8392019320.pdf
- Bhawan, K. L. (2019). Artificial Intelligence (AI) and Big Data for Telecom. *FN Division, TEC*. http://tec.gov.in/pdf/Studypaper/Study%20Paper%20AI%20and%20Big%20Data%20for%20Telecom.pdf
- James, G., Witten, D., Hastie, T. & Tibshirani, R. (2013). An introduction to statistical learning: with applications in R. *New York: Springer*.
- Friedman, J., Hastie, T. & Tibshirani, R. The elements of statistical learning. *Springer series in statistics New York*. Fuente: https://web.stanford.edu/~hastie/Papers/ESLII.pdf
- Bramer, Max (2007). Principles of data mining. Vol. 180. *London: Springer*.
- Documentación de Spark. Fuente: https://spark.apache.org/docs/latest
- Solingest, desarrollo web para profesionales. Cluster de servidores, ¿qué es y cómo funciona? Fuente: https://www.solingest.com/cluster-de-servidores-que-es-y-como-funciona
- Fawcett, T. (2006). An introduction to ROC analysis. *Pattern recognition letters*, 27(8), 861-874.
- Nathan Marz with James Warren (2015). Big Data: Principles and best practices of scalable realtime data systems. *Manning Publications*.
- Karau, H., Konwinski, A., Wendell, P. & Zaharia M. (2015). Learning Spark. *O'Reilly Media, Inc.*