Predicting user satisfaction from customer service chats

Alejandro Romanisio[1], Agustín Gravano[1,2]

[1] Master in Management + Analytics, Escuela de Negocios, Universidad Torcuato Di Tella, Buenos Aires, Argentina
[2] Consejo Nacional de Investigaciones Científicas y Tecnológicas (CONICET), Argentina
aromanisio@gmail.com, agravano@utdt.edu

Abstract. Customer service is a determining factor in the user experience of Fintech companies. This work seeks to understand, using machine learning techniques, what factors lead the clients of a specific Fintech company to positively evaluate their experience. Two data sources were used to achieve this: user records from their sign up and the log of conversations with customer service via WhatsApp. We experimented with predictive models based on XGBoost, trained with features of the user context, the characteristics of the conversations and the semantics of the words used in the conversations. The results were lower than expected (AUC = 0.5152), but they leave valuable lessons for those who face similar problems in the future, related to the challenges of the following critical aspects: i. avoid data leakage, ii. evaluate models and scoring metrics thoroughly, iii. carry out intermediate checkpoints, iv. do not underestimate the time required for data transformation, v. perform a unit testing process and vi. know the domain. This paper describes the different stages of the methodology: data extraction and transformation, feature generation, predictive model training, optimal model selection and test data evaluation.

Keywords: customer service, satisfaction surveys, predictive models, XGBoost, natural language processing.

Received January 2024; Accepted March 2024; Published May 2024

https://doi.org/10.24215/15146774e037

ISSN 1514-6774

Predicción de la satisfacción del usuario a partir de chats de atención al cliente

Alejandro Romanisio[1], Agustín Gravano[1,2]

[1] Master in Management + Analytics, Escuela de Negocios, Universidad Torcuato Di Tella, Buenos Aires, Argentina
[2] Consejo Nacional de Investigaciones Científicas y Tecnológicas (CONICET), Argentina
aromanisio@gmail.com, agravano@utdt.edu

Resumen. Los servicios de atención al cliente son determinantes de la experiencia de usuario de las empresas Fintech. Este trabajo busca entender, empleando técnicas de machine learning, qué factores llevan a los clientes de una Fintech a evaluar de forma positiva su experiencia. Esto se hizo a partir de dos fuentes de datos: los registros de los usuarios y las conversaciones del servicio de atención al cliente vía WhatsApp. Experimentamos con modelos predictivos basados en XGBoost, entrenados con features del contexto del usuario, las características de las conversaciones y la semántica de las palabras utilizadas en las conversaciones. Los resultados fueron menores a lo esperado (AUC = 0.5152), pero dejan aprendizajes valiosos para quienes encaren problemas semejantes en el futuro, relacionados a los desafíos de los siguientes aspectos críticos: i. evitar el data leakage, ii. evaluar modelos y scoring metrics exhaustivamente, iii. realizar chequeos intermedios, iv. no subestimar el tiempo necesario para la transformación de datos, v. realizar un proceso de unit testing y vi. conocer el dominio. Este trabajo describe las distintas etapas de la metodología: extracción y transformación de los datos, generación de features, entrenamiento de modelos predictivos, selección del modelo óptimo y evaluación en datos de *test*.

Keywords: atención al cliente, encuestas de satisfacción, modelos predictivos, XGBoost, procesamiento de lenguaje natural.

1 Introducción

La experiencia de los clientes en el mundo Fintech se encuentra fuertemente determinada por los niveles de servicio. La experiencia con atención al cliente es parte fundamental para determinar estos niveles de servicio y resulta uno de los touchpoints clave para poder tornar clientes detractores en clientes promotores[1].

[1] "Detractor" es el término utilizado para aquellos clientes que ponen los puntajes más bajos en la escala de puntuación del NPS (Net Promoter Score), uno de los sistemas de medición de la experiencia de los usuarios más relevantes para las empresas en los últimos 15 años. Del mismo modo, un "Promotor" es un cliente que pone los puntajes más altos.

En este contexto, resulta relevante poder detectar con velocidad y precisión si un usuario que se contactó con los canales de atención al cliente tiene potencial de convertirse en un detractor luego del contacto y tomar una acción de mejora inmediata para evitar que el usuario termine siendo un detractor.

Existe una correlación positiva entre la proporción de promotores y detractores y la rentabilidad y crecimiento de una empresa, en particular en la industria financiera. Por ejemplo, los clientes promotores de bancos tienen saldos casi 45% superiores a los detractores; asimismo, la tasa de abandono de los detractores resulta ser el triple comparada a los promotores (Reicheld 2011). De esta forma, se encuentra una conexión directa entre experiencia de los clientes y rentabilidad y crecimiento del negocio.

El **problema** que intentará resolverse en este trabajo es el de predecir el puntaje que pondrán los clientes al finalizar un chat de atención con un agente vía WhatsApp para la empresa estudiada, ante el envío de la encuesta de satisfacción. Ésta tiene la siguiente forma: "En una escala del 1 (muy mala 👎) al 5 (muy buena 👍), ¿cómo fue tu experiencia con el asesor que te atendió?".

Vale la pena mencionar que no todas las conversaciones entre clientes y agentes tienen una respuesta a la encuesta por dos motivos: primero, no siempre se envía una encuesta al final de una conversación, sino que se parametrizan reglas para el envío de forma tal de no "saturar" al usuario con encuestas demasiado frecuentes. Segundo, no siempre los usuarios responden a las encuestas.[2]

El objetivo de este trabajo consiste en formular un modelo predictivo que permita predecir el puntaje con el que un cliente puntuará la atención recibida por un agente a partir de distintas características del cliente y la conversación. Este problema es relativamente reciente y ha sido abordado con distintos enfoques en la literatura. Por ejemplo, Park et al. (2015) experimentan con modelos basados en support vector machines (SVM) entrenados en chats en inglés de soporte técnico online de Samsung, empleando principalmente features de sentimiento y meta-información contextual. Auguste et al. (2019) hace lo propio sobre chats, también en inglés, de soporte técnico online de Orange, en este caso experimentando con SVM, redes neuronales convolucionales y redes neuronales recurrentes, empleando sólo features léxicas.

En el presente trabajo, seguiremos procedimientos similares para lograr este objetivo específicamente sobre chats en español de una Fintech argentina. La metodología de trabajo planteada buscará encontrar features que caractericen la interacción entre agente y cliente para intentar predecir el puntaje final que pondrá el cliente. La Sección 2 describe las fuentes de datos disponibles; la Sección 3, los atributos y algoritmos de machine learning elegidos; la Sección 4, los resultados; y la Sección 5, las conclusiones del trabajo.

[2] Según Reichheld (2011), quienes no responden las encuestas hubiesen respondido la encuesta con una nota peor que quienes sí lo hicieron. En la empresa estudiada, sólo el 36% de las encuestas enviadas tienen una respuesta que permita asignar un puntaje. De esta forma, podría haber una sobreestimación del puntaje del servicio de atención al cliente al no contar con la información de la percepción del servicio de aquellos usuarios que deciden no responder la encuesta.

2 Datos

Se cuenta con dos grandes fuentes de datos. La primera es la transcripción de los chats vía WhatsApp del servicio de Atención al Cliente de la empresa. La segunda es la información del legajo digital de los clientes de la empresa que proviene del registro de los clientes en la aplicación de la empresa.

En cuanto al origen de las bases, es importante destacar que vienen de dos proveedores distintos: la base de clientes viene directamente del sistema CRM (Customer Relationship Management) de la empresa, mientras que la base de mensajes viene de los servicios de un proveedor de la empresa, el cual provee la plataforma para la comunicación entre agentes y clientes.

Como la parte central de los datos de la conversación proviene de la base de datos de mensajes, vale la pena hacer un breve apartado comentando cómo llega a darse la interacción que genera la información de esta base. La empresa en cuestión ofrece un servicio de atención al cliente vía WhatsApp. Cuando un cliente se contacta al WhatsApp, la primera respuesta es siempre de un chatbot de atención. Este bot interactúa con los clientes ofreciéndoles menús con opciones o también interpretando palabras libres que escribe el cliente. Ante distintas situaciones que pueden darse por las elecciones que hacen los clientes en el menú, o por las palabras que le dicen al bot, éste puede derivar a los clientes con un agente para que pueda atender su consulta, pedido o reclamo.

2.1 Definiciones

A los fines de simplificar los conceptos que se utilizan a lo largo de este trabajo, se definen algunos de los principales conceptos que se utilizarán:

- Emisor: es una variable categórica que define qué parte envió un mensaje, pudiendo ser ésta: agente, cliente o bot.
- Agente: parte interviniente en un chat de atención al cliente que brinda atención por parte de la empresa ante consultas de clientes por medio de WhatsApp.
- Cliente: parte interviniente en un chat de atención al cliente que busca atención por parte de la empresa por medio de WhatsApp.
- Bot: parte interviniente en un chat de atención al cliente que envía mensajes en formato de texto (respuestas) a los clientes ante la recepción de un estímulo (disparador), que puede ser un mensaje enviado por el cliente, o bien una opción seleccionada (botón) por el cliente desde un menú de opciones en la conversación de WhatsApp.
- Mensaje: cada uno de los textos enviados en un chat de WhatsApp, equivalente al contenido de cada “globo” en la aplicación. Los clientes pueden enviar mensajes de audio, que son traducidos automáticamente a texto para la lectura de los agentes y del bot. Los agentes y el bot no pueden enviar mensajes de audio.
- Bloque: grupo de mensajes enviados consecutivamente por un mismo emisor.
- Conversación: grupo de bloques con interacción de agentes (humanos) y clientes, delimitado por eventos de inicio y eventos de fin.

- Evento de inicio: mensajes que marcan el inicio a una conversación, marcados por un evento de derivación a un agente (e.g..: cuando a partir de un mensaje del cliente, el bot responde con un mensaje que indica "Te estoy derivando con un asesor").
- Evento de fin: mensajes o sucesos que marcan el final de una conversación, definidos por las siguientes 3 condiciones:
  - "No more wait": cuando un cliente es derivado a un asesor y no hay uno disponible inmediatamente, el bot ofrece al cliente dos opciones: continuar esperando o salirse de la cola de espera. Si el cliente elige salirse de la cola de espera, se considera un evento de fin del tipo *no_more_wait*.
  - "Day end": el servicio de atención al cliente de la empresa termina a la medianoche, por lo que cada cambio de día calendario se considera un evento de fin del tipo *day_end*[3].
  - "Agent close": cuando los agentes desde su consola de atención hacen clic en el botón de cerrar conversación, dejan de tener la posibilidad de intervenir en la conversación, y vuelve a intervenir el bot enviando un mensaje con la encuesta para evaluación del agente. Esto se considera un evento de fin del tipo *agent_close*.
- Encuesta: cuando un agente hace clic en el botón de cerrar conversación, se envía un mensaje al cliente que contiene la encuesta de satisfacción. La respuesta del cliente a la encuesta es lo que define la variable dependiente y se llama "Puntaje".
- Puntaje: el puntaje considerado para las encuestas toma en cuenta los números del 1 al 5, siendo el resto de los valores considerados como no válidos. Se considera Promotor a todo puntaje de 5. Se considera Detractor a todo puntaje entre 1 y 4 inclusive. Este puntaje, será la variable a predecir del modelo.

2.2 Estructura de un chat

Para facilitar el entendimiento de la dinámica de las conversaciones, a continuación se muestra un ejemplo de un chat, indicando los distintos elementos que lo componen:

[3] Es importante mencionar que al final del turno laboral a la medianoche, los agentes cierran necesariamente todas las conversaciones en curso. Eso implica que, si se cumplían algunos parámetros definidos por la empresa, como por ejemplo que el cliente no haya recibido encuestas en los últimos días (este parámetro varió a lo largo del período de estudio), el cliente con una conversación en curso recibiría una encuesta luego del final de turno a las 12 de la noche. De esta forma, el puntaje de la encuesta sería completado por el cliente posterior a las 12 de la noche, pero necesariamente antes del inicio de una nueva conversación.

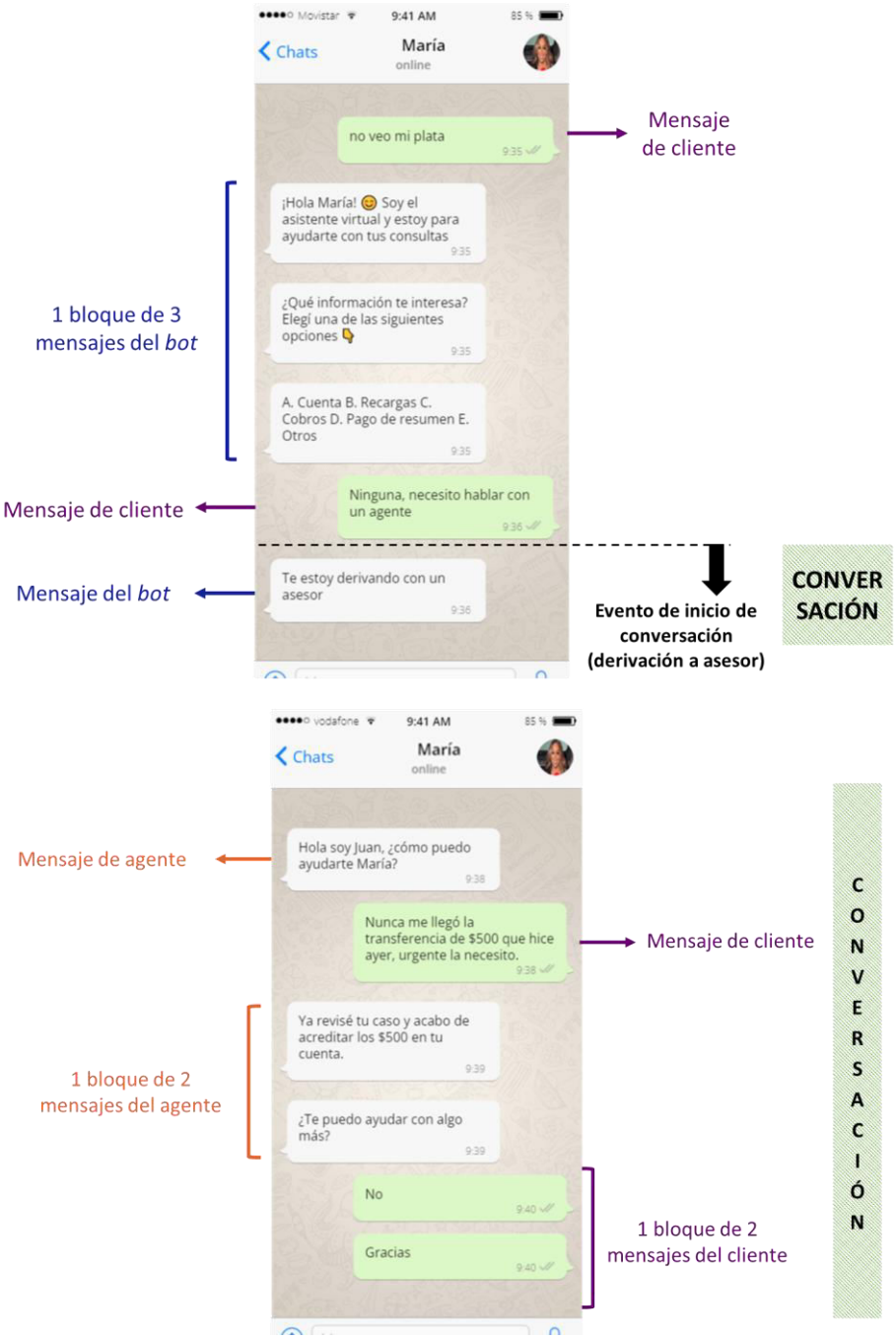
Movistar 9:41 AM 85 %
Chats
María
online
no veo mi plata
9:35
Mensaje de cliente
¡Hola María! Soy el asistente virtual y estoy para ayudarte con tus consultas
9:35
¿Qué información te interesa? Elegí una de las siguientes opciones
9:35
A. Cuenta B. Recargas C. Cobros D. Pago de resumen E. Otros
9:35
1 bloque de 3 mensajes del bot
Ninguna, necesito hablar con un agente
9:36
Mensaje de cliente
Te estoy derivando con un asesor
9:36
Mensaje del bot
Evento de inicio de conversación (derivación a asesor)
CONVERSACIÓN
vodafone 9:41 AM 85 %
Chats
María
online
Hola soy Juan, ¿cómo puedo ayudarte María?
9:38
Mensaje de agente
Nunca me llegó la transferencia de $500 que hice ayer, urgente la necesito.
9:38
Mensaje de cliente
Ya revisé tu caso y acabo de acreditar los $500 en tu cuenta.
9:39
¿Te puedo ayudar con algo más?
9:39
1 bloque de 2 mensajes del agente
No
9:40
Gracias
9:40
1 bloque de 2 mensajes del cliente
CONVERSACIÓN

Fig. 1. Pantallas mostrando la estructura de un chat y sus componentes.

2.3 Estructura de datos

A continuación, se presenta un esquema resumido de la representación de los datos de una conversación, en el que cada línea se corresponde a un mensaje.

Para mayor claridad, se destacan en celeste los mensajes enviados por el bot, en violeta los mensajes enviados por el cliente y en naranja los mensajes enviados por el agente:

Tabla 1. Estructura resumida de datos de una conversación.

mensaje	*conv_event*	*conv_id*	*conv_seq*	*block_id*
no veo mi plata				223
¿Qué información te interesa? Elegí una de las siguientes opciones ??				224
¡Hola María! ?? Soy el asistente virtual y estoy para ayudarte con tus consultas				224
A. Cuenta B. Recargas C. Cobros D. Pago de resumen E. Otros				224
Ninguna, necesito hablar con un agente				225
Te estoy derivando con un asesor	derivation	77	1	226
Hola soy Juan, ¿cómo puedo ayudarte María?		77	2	227
Nunca me llegó la transferencia de $500 que hice ayer, urgente la necesito		77	3	228

mensaje	*conv_event*	*conv_id*	*conv_seq*	*block_id*
Ya revisé tu caso y acabo de acreditar los $500 en tu cuenta		77	4	229
¿Te puedo ayudar con algo más?		77	5	229
No		77	6	230
Gracias		77	7	230
Te dejo una breve encuesta:	agent_close	77	8	231
En una escala del 1 (muy mala ??) al 5 (muy buena ??), ¿cómo fue tu experiencia con el asesor que te atendió?				231
5				232
¿Nos querés contar los motivos de tu evaluación				233
Resolvió rápido				234
Gracias por escribirnos, sigo acá para lo que necesites. ¡Un abrazo! ??				235

Un punto a destacar de los datos, que dará como necesidad la manipulación para su secuenciación en orden correcto es la siguiente: el orden de los bloques refleja lo sucedido en la conversación real de WhatsApp. Por ejemplo, el bloque 225 ocurrió inmediatamente después que el bloque 224. Sin embargo, no sucede lo mismo con el orden de los mensajes dentro de los bloques, como puede observarse por ejemplo en el bloque 224. En este caso, el orden de los mensajes es correcto sólo para mensajes enviados por agentes o clientes. Pero cuando se trata de mensajes enviados por el bot, esto no es necesariamente así. Esto tiene que ver con que el timestamp en la tabla de datos se encuentra truncado al segundo más cercano, resultando en un desordenamiento en algunos mensajes que se envían dentro del mismo segundo (i.e.: mensajes consecutivos enviados por el bot).

2.4 Distribución del puntaje

A continuación, se muestra la distribución de la variable a predecir, incluyendo la información de la cantidad de encuestas completas e incompletas:

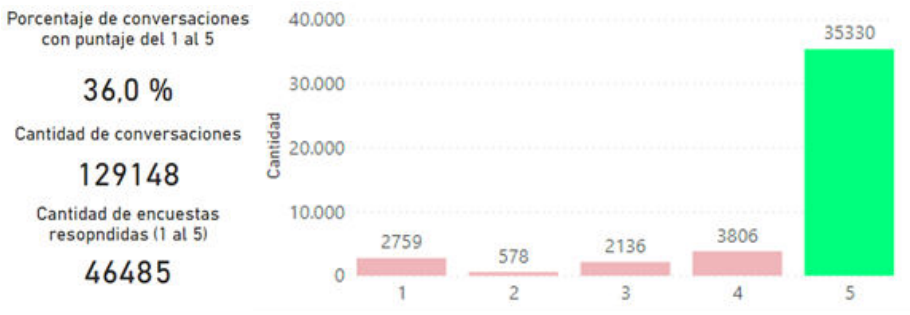

Fig. 2. distribución del puntaje otorgado por los clientes a los agentes.

Como puede observarse, existe cierto desbalance de clases en las encuestas respondidas con un puntaje válido, con un 79% de las encuestas respondidas con el valor 5. Asimismo, vemos que se trabajará con el 36% de las encuestas totales, ya que esta es la proporción de encuestas que contiene un puntaje válido del 1 al 5 como respuesta de los clientes (ver nota al pie 2).

3 Metodología

El esquema de trabajo cuenta con 5 grandes pasos que se describen a continuación, y se profundizarán en las secciones siguientes: i. Extracción y transformación de los datos; ii. Generación de features; iii. Entrenamiento de modelos predictivos; iv. Selección del modelo óptimo; v. Evaluación en datos de *test*.

3.1 Origen de datos

Como ya se mencionó, se utilizaron 2 bases de datos de una empresa Fintech argentina: la primera *bm_mensajes*, que contiene la información de los mensajes enviados y recibidos y la segunda es la base de datos *nx_clientes*, que contiene la información correspondiente a los datos filiatorios de los clientes de la empresa al momento de registro en la aplicación.

3.2 Extracción y transformación de los datos

Este proceso se realizó en 5 etapas que se explicarán a continuación:

Etapa 1 – Preparación y carga de datasets. En esta etapa se realiza la carga de los archivos de la base de datos *bm_mensajes* y de la base *nx_clientes* desde el directorio de trabajo.

A los fines de minimizar el riesgo de data leakage, se realiza la separación de lo que llamaremos los datos de *train* (70% de la base *bm_mensajes*) y los datos de *test* (30%). En el resto del trabajo no se utilizarán más los datos de *test* y sólo se volverán a utilizar en la etapa de evaluación. La separación aleatoria en los conjuntos de *train* y *test* se hace a partir del teléfono de contacto, principalmente para evitar desordenar los mensajes y para evitar que aparezcan conversaciones de una misma persona tanto en *train* como en *test*, lo que podría resultar en un data leakage si una conversación previa influyera en una posterior.

Etapa 2 – Ordenamiento y etiquetado. En esta etapa se realiza el ordenamiento de los mensajes resultando en una secuencia de mensajes que ordena a los clientes por número telefónico; además, dentro de cada grupo de mensajes asociados a ese cliente (sean mensajes con emisor bot, agente o cliente), los ordena por la fecha en que fueron enviados. Así, tenemos la historia de interacciones de cada cliente ordenada cronológicamente.

En lo referente al etiquetado, en esta etapa se comienzan a identificar algunos elementos de las conversaciones que serán los ladrillos que construirán la macroestructura de conversación (delimitada por los eventos de inicio y eventos de fin de conversación) y algunos otros eventos que permitirán construir algunos features más adelante.

A continuación, se describen algunas de las etiquetas generadas para las conversaciones:

Table 2. Listado de etiquetas generadas para cada conversación y su significado.

Etiqueta	Descripción
tag_button_no	El cliente elige la opción "No" del menú de opciones ofrecido por el bot.
tag_button_si	El cliente elige la opción "Sí" del menú de opciones ofrecido por el bot.
tag_wait_offer	El bot envía un mensaje con las palabras "¿Te gustaría seguir esperando?" a un cliente que está esperando ser atendido por un agente. Ofrece las opciones de "continuar esperando" o "salirse de la cola de espera".
tag_no_more_wait	El cliente que recibió la oferta de "continuar esperando" o "salirse de la cola de espera" elige la segunda opción.
tag_derivation	El bot deriva al cliente con un agente.
tag_day_end	Etiqueta que señala el último mensaje del día.
tag_agent_close	El agente finaliza una conversación y se envía una encuesta al cliente.
image_sent_tag	Etiqueta que señala que el cliente envió una imagen durante la conversación.
wait_offer_count	Recuento de veces que se aplicó la etiqueta *tag_wait_offer* en cada conversación.
msg_per_conv_agente	Recuento de mensajes enviados por el agente en una conversación.
msg_per_conv_cliente	Recuento de mensajes enviados por el cliente en una conversación.
tag_weekday	Etiqueta que señala el día de la semana en que se completó una encuesta.
agent_delay	Demora máxima y promedio de respuesta de los agentes ante cada bloque de mensajes enviado por el cliente.

Etapa 3 – Eventos de inicio y de fin de conversación. En esta etapa se delimita en el dataset la estructura de conversación definida. De esta forma, se marcan y anexan como columnas el evento de inicio (derivación a asesor) y los eventos de fin de una

conversación (final del día, respuesta del cliente de no querer seguir en espera y cierre de conversación por parte del agente).

Etapa 4 – Secuenciación. Una vez definido el inicio y final de las conversaciones, cobra relevancia entender la secuencia en la que se dan los mensajes dentro de la conversación. Para eso se realizan tres funciones que ordenan los datos según lo realmente ocurrido en la conversación, consiguiendo una noción de orden secuencial a tres niveles: conversación, bloque y mensaje.

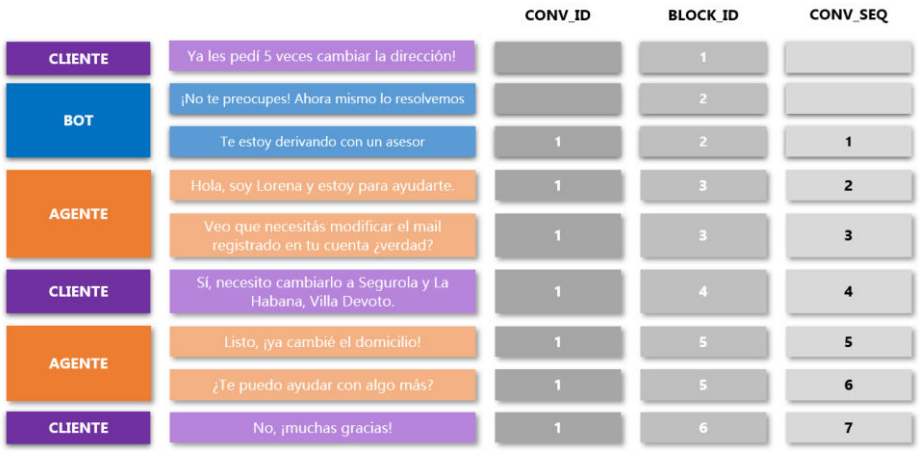

Fig. 3. Representación de una conversación de ejemplo con su etiquetado jerárquico de *conv_id*, *block_id* y *conv_seq*.

Sin entrar en detalles de la implementación de las funciones que generan la secuenciación en estas tres jerarquías, a continuación se explica de forma resumida qué hacen estas funciones que dan origen a los números de *conv_id*, *block_id* y *conv_seq* para cada mensaje.

conv_id. Esta función lee los mensajes en orden y asigna un número de conversación igual (*conv_id*) a aquellos mensajes que son parte de una misma conversación.

La función lee los mensajes desde el primero y cuando detecta un evento de inicio de conversación (por ejemplo, un mensaje de "Te estoy derivando con un asesor") comienza a asignar un número de conversación a ese mensaje y los siguientes, por ejemplo, el número 1.

La función continuará asignando el número 1 a cada mensaje de la conversación, hasta toparse con un evento de fin de conversación (por ejemplo, un cierre de conversación por parte del agente).

Luego de este evento de fin de conversación, la función continuará leyendo los mensajes en orden, sin asignarles número de conversación hasta que encuentre un nuevo evento de inicio. En ese momento, comenzará nuevamente a asignar un número de conversación a los mensajes, en este caso, el número 2, y así sucesivamente.

block_id. Esta función lee los mensajes en orden y asigna un número de bloque (*block_id*) a cada secuencia de mensajes enviada consecutivamente por un mismo emisor, sin importar si los mensajes están dentro o fuera de una conversación.

La función lee el primer mensaje de todos y le asigna el número 1 para *block_id*. Luego continúa al segundo mensaje. Si detecta que el emisor del segundo mensaje es igual al del mensaje anterior, continúa asignando el mismo número para *block_id*, es decir, 1. Luego, continúa leyendo el siguiente mensaje. La función continúa con este funcionamiento hasta detectar que hay un cambio en el emisor (i.e.: el emisor del mensaje que la función está leyendo es distinto al emisor del mensaje inmediatamente anterior). En este caso, la función asigna a este nuevo mensaje el siguiente número como *block_id*, en este ejemplo, el número 2 y luego continúa con la misma lógica con el resto de los mensajes.

conv_seq. Esta función lee los mensajes en orden y asigna un número de secuencia de conversación (*conv_seq*) a cada mensaje dentro de una conversación en orden ascendente. Cada número asignado es único dentro de cada conversación.

La función lee cada mensaje en orden y detecta, utilizando el resultado de la función que generó los *conv_id*, si está leyendo un mensaje que se encuentra dentro o fuera de una conversación. En el caso de que el mensaje no pertenezca a una conversación (i.e.: *conv_id* está vacío), continúa al siguiente mensaje.

Cuando finalmente se encuentra con un mensaje que pertenece a una conversación (i.e.: *conv_id* tiene un número), le asigna el número 1 para *conv_seq* y pasa al siguiente mensaje. Si el siguiente mensaje pertenece a la misma conversación, entonces le asigna el número 2 para *conv_seq*, y así sucesivamente para los mensajes que siguen, siempre en incrementos de 1.

Esto sucede hasta que se encuentra con un mensaje que no pertenece a una conversación, o bien pertenece a una conversación con un *conv_id* distinto. Si el caso es el primero, no asignará ningún número para *conv_seq* y continuará hacia el siguiente mensaje. Si el caso es el segundo, "reiniciará" el número de *conv_seq*, asignándole el número 1 a este nuevo mensaje. Luego, continuará con la misma lógica descripta anteriormente.

Etapa 5 – Variable objetivo. Para capturar la variable a predecir, se implementó una función que pueda detectar en qué línea el cliente da la respuesta a la encuesta de satisfacción, y en caso de que tenga una repuesta inválida, volver a intentar capturar la respuesta en el siguiente envío de encuesta de satisfacción del bot. Además, la función captura únicamente las respuestas que contienen los números del 1 al 5[4].

[4] Podría considerarse que existe cierto valor en las respuestas del tipo "0" (cliente para nada satisfecho) o "10" (cliente muy satisfecho que responde utilizando una escala más tradicional en la vida cotidiana de los clientes), pero, además de que el porcentaje de estas respuestas es despreciable, se decidió apegarse estrictamente a aquellos valores que son exactamente 1, 2, 3, 4 o 5 para no agregar ninguna interpretación subjetiva a lo escrito por el cliente. Por ejemplo, el "0" podría ser un cliente muy poco satisfecho, pero también podría ser un cliente que escribió mal y quiso poner un "10".

3.3 Generación de features

Este proceso se realizó en 4 etapas para generar features esperando que tengan poder predictivo sobre la nota de evaluación a los asesores de atención.

Se definen, entonces, 3 categorías de features:

Etapa 6 – Features contextuales. Para la generación de features contextuales que refieren a características del usuario, se incorporó la información de la base de datos *nx_clientes*. A partir de esta base y los datos preprocesados de la base de datos *bm_mensajes* se obtuvieron features contextuales de los clientes como, por ejemplo: la fecha de alta en la aplicación, la nacionalidad, el género, el estado civil, el código postal de su domicilio y su condición tributaria ante el fisco.

Etapa 7 – Features conversacionales. La generación de features conversacionales que refieren a características propias de la dinámica de la conversación. En específico, lo que se busca es capturar características de la dinámica que puedan contener algún indicio de la molestia (o satisfacción) señalizada por indicadores de tiempo de la conversación y de forma de conversar de ambas partes: agente y cliente.

Algunos de los features conversacionales extraídos son: duración de la conversación, presencia de imágenes enviadas en la conversación, cantidad total de mensajes enviados por cliente/agente, y demora promedio de respuesta de los agentes.

Etapa 8 – Features semánticos. Para la generación de los features de carácter semántico se decidió incorporar features que puedan señalar y diferenciar la frecuencia con que se encuentran presentes algunas palabras dentro de las conversaciones, entendiendo que la connotación o la semántica de esas palabras, en el contexto de un contacto con un servicio de atención al cliente, pueden ser indicativas de la emocionalidad de la conversación e incluso del nivel de percepción del cliente respecto del agente. Por este motivo, se espera que puedan brindar poder explicativo respecto a la evaluación que hará el cliente al responder la encuesta de satisfacción.

Para ello, luego de manipular la información de los mensajes (definiendo stopwords, concatenando los mensajes, tokenizando las palabras), se llegó a un dataframe con el formato matriz documento-término (document-term matrix), que indica la frecuencia de aparición de cada “término” dentro de cada “documento”, donde los “documentos” corresponden al id de conversación y los “términos” corresponden a las palabras dichas por agentes y clientes (se distingue con un prefijo “cl_” o “ag_” a los términos dichos por los clientes o por los agentes, respectivamente).

Etapa 9 – Dataframe final. En esta etapa se busca hacer los arreglos finales a los dataframes generados y se los une en un único dataframe que contenga a todos los features generados. Asimismo, en esta etapa se realiza la normalización tf-idf de los features semánticos. A continuación, una breve descripción de las acciones realizadas:

- **Remoción de filas redundantes.** En el dataframe que contiene toda la información de los mensajes y los features conversacionales y contextuales, hay muchas filas para cada conversación (recordar que hay 1 fila por cada mensaje enviado), pero sólo se necesita 1 fila por conversación, que tenga las columnas correspondientes a los features asociados a ésta. Se deja sólo 1 fila por conversación con todos sus features asociados en las columnas.
- **Creación de matriz semántica.** Se realiza el join de las document-term matrix de clientes y agentes, usando como clave primaria el id de conversación. Los conteos nulos, se reemplazan por el número 0 para preparar la matriz para la normalización tf-idf.
- **Normalización tf-idf.** Mediante la función TfidfTransformer de la librería de feature extraction de Sci-Kit Learn (scikit-learn.org) se realiza la normalización de la matriz semántica con la normalización tf-idf
- **Binarización variable dependiente**. Se ejecuta la binarización de las notas de la encuesta respondida por los usuarios, es decir, la variable dependiente, asignando un 0 (detractor) a los puntajes del 1 al 4 y un 1 (promotor) al puntaje 5. En los modelos predictivos se aprenderá a predecir promotores (1's).
- **Unión de datasets**. Como paso final, se hace la unión de los datasets, anexando la matriz semántica normalizada y obteniendo un dataframe final de dimensión 45.003 x 8.221 (conversaciones x features) que se utilizará para el entrenamiento de los modelos predictivos.

3.4 Entrenamiento de modelos predictivos

Para entrenar los modelos predictivos, se trabajará sobre distintos subconjuntos de los datos, cada uno con distintos conjuntos de features (conversacionales, contextuales y semánticos) "encendidos". De esta forma se evaluarán por validación cruzada en datos de entrenamiento todas las combinaciones de features con los 2 algoritmos seleccionados: Random Forest y XGBoost (Cutler et al. 2012; Chen and Guestrin 2016; Mohammad & Hemmatian 2019). A continuación, se describe cómo se realizó este entrenamiento.

Etapa 10 – Preparación de datasets para algoritmos. En esta etapa se generan todos los subconjuntos de datos que se utilizarán como datos de entrenamiento en los 2 algoritmos nombrados. Como resultado se obtienen 14 dataframes con los 14 modelos posibles[5].

Antes de avanzar a la siguiente etapa, vale la pena aclarar cuál será la estrategia a seguir en lo referido al entrenamiento y refinamiento de hiperparámetros de los modelos.

[5] La cantidad de modelos posibles es igual a $(2^f - 1) \times a$, donde f es la cantidad de categorías de features (3) y a es la cantidad de algoritmos a utilizar (2).

La estrategia elegida se resume en el siguiente esquema:

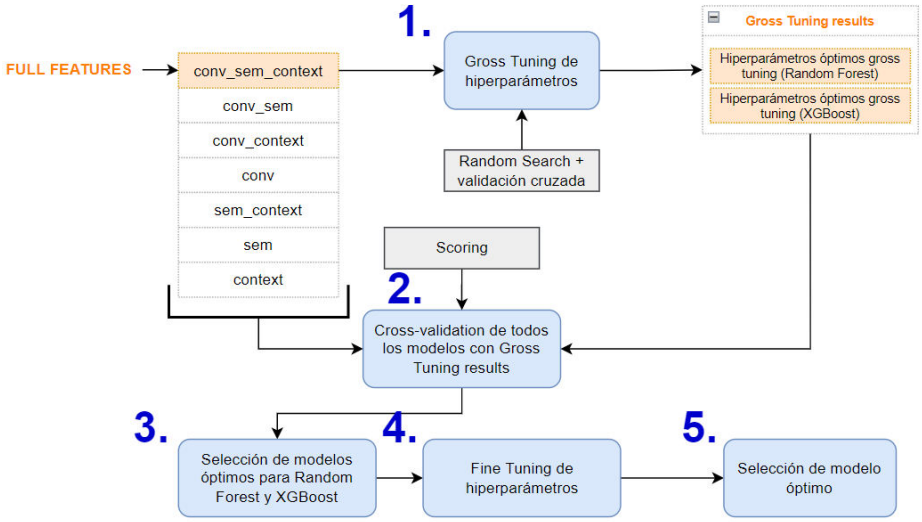

Fig. 4. Esquema con la estrategia para el refinamiento de hiperparámetros y selección de modelo óptimo.

1. Sobre el modelo que contiene los 3 grupos de features (de ahora en más, modelo full features), se realiza un gross tuning de los hiperparámetros para los algoritmos de XGBoost y Random Forest. Se utiliza un random search con validación cruzada para la selección de los mejores hiperparámetros.
2. Con los resultados de 1., se evalúan en los datos de *train* los restantes 6 modelos, en ambos algoritmos, y se obtiene el score para cada uno de estos modelos.
3. En bases al criterio de score seleccionado, se elige 1 modelo óptimo para Random Forest y 1 modelo óptimo para XGBoost
4. Sobre los 2 modelos óptimos elegidos en 3., se realiza un fine tuning de los hiperparámetros
5. Se evalúa el score de los modelos y se elige el modelo óptimo.

Para el refinamiento de los hiperparámetros, se crea una grilla de hiperparámetros para Random Forest y para XGBoost, con un rango más acotado (±10%) alrededor de los parámetros óptimos del Gross Tuning. Para la elección de los hiperparámetros óptimos se utiliza validación cruzada de 5 splits. La elección del algoritmo óptimo se realiza con el F1-score óptimo de esta validación cruzada. Algunos de los factores, se dejan fijos según los resultados del Gross Tuning dado el crecimiento exponencial de combinaciones que implica y la demora excesiva según el equipo utilizado para este trabajo.

Luego de revisar los resultados por validación cruzada, se selecciona finalmente el algoritmo óptimo a utilizar con sus hiperparámetros ya optimizados. El criterio de evaluación para seleccionar el modelo y los resultados, se detallarán a continuación.

4 Resultados

4.1 Criterio de evaluación

Para decidir el criterio de evaluación, es necesario tener en cuenta que el problema a resolver está relacionado con la predicción de promotores en las interacciones con atención al cliente.

A los fines de poder decidir una métrica que capture de la mejor forma los objetivos relevantes de este modelo predictivo, se plantean dos casos de uso relevantes para la empresa en términos de la aplicación de este modelo: 1. aprender cuáles son las buenas prácticas conversacionales de forma correcta, para replicarlas; y 2. capturar todas las buenas prácticas conversacionales para no perderse de ninguna. Este es un enfoque basado en potenciar los promotores. Para ello se decide utilizar la métrica del F1 score, que captura ambos objetivos.

4.2 Selección de modelo óptimo

Luego de realizar el Gross Tuning, se realiza el Fine Tuning. A continuación, se muestran los resultados en la validación cruzada de todos los modelos, cuando se les aplica los parámetros óptimos del Random Search obtenidos sobre el modelo full feature (modelo ganador del Gross Tuning que contiene features conversacionales, semánticos y contextuales):

Tabla 3. Performance de modelos con parámetros optimizados por Grid Search.

Algoritmo	Features	Accuracy	Precision	Recall	F1 score
Random Forest	conversational + semantic + context	0.8076	0.8076	0.9993	0.8933
XGBoost	conversational + semantic + context	0.8116	0.8147	0.9919	0.8946

Como se observa en la tabla anterior, con los hiperparámetros optimizados por fine tuning, el modelo que mejor performance tiene por el criterio de F1 score, termina siendo el modelo con algoritmo XGBoost que incluye todos los features[6]. Por lo tanto, este será el modelo elegido para la evaluación en datos de *test*.

Un punto muy relevante para destacar y ya anticipar los resultados que podríamos ver en la evaluación en datos de *test*: el accuracy que muestra el modelo es de 81,2%. Cabe recordar que la proporción de promotores en los datos de entrenamiento es del 80,0%. Con este número puede verse fácilmente que el modelo tiene un accuracy sólo 1,2 puntos porcentuales mayor que un modelo que prediga a todos las observaciones como 1's (promotores).

[6] Se realizaron las validaciones con las 14 combinaciones de features y algoritmos posibles. Por simplicidad se resumen sólo los 2 resultados de los modelos con *full features*.

4.3 Evaluación en datos de *test*

Para realizar la evaluación del modelo elegido y optimizado sobre los datos de *test* se utiliza el modelo XGBoost con todos los features, ya que es el modelo que mostró la mejor performance en cuanto al F1 Score. En la siguiente figura se muestra la curva ROC del modelo:

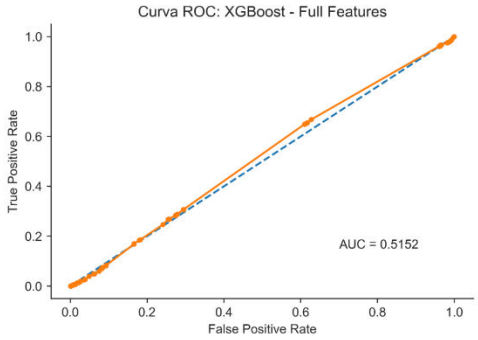

Fig. 5. Curva ROC del modelo Full Features con algoritmo XGBoost (AUC = 0.5152).

La curva ROC muestra un área bajo la curva (AUC) muy baja, cercana al 0.5. En los umbrales de los extremos se dan valores de False Positive Rate y True Positive Rate que incluso se encuentran por debajo de la curva.

5 Conclusiones

5.1 Objetivos propuestos

Con respecto al objetivo de predecir el puntaje con el que un cliente puntuará la atención recibida por un agente a partir de distintas características del cliente, el agente y la conversación, luego de atravesar el proceso de aprendizaje de conocimiento de los datos y de optimización de un modelo predictivo, el objetivo no pareciera estar cumplido al nivel esperado. Lamentablemente, el modelo predictivo optimizado no terminó aportando un gran valor predictivo ya que en el óptimo predijo igual que un modelo sin "skills" que predice a todos los clientes como promotores. Debido al bajo valor predictivo encontrado en los grupos de features elegidos, tampoco se pudo avanzar hacia el objetivo de entender cuáles son aquellos factores determinantes que llevan a los clientes a poner un puntaje promotor o detractor.

Sin embargo, resulta importante comprender dos cuestiones para poder pensar en futuras investigaciones o desarrollos que puedan resultar en modelos más alentadores en términos de su poder predictivo. La primera es asegurarse de que se haya aplicado una metodología de trabajo que minimice errores y pérdida de poder predictivo en los modelos. La segunda es entender qué modificaciones o novedades deberían

introducirse a los modelos predictivos de este trabajo para lograr una mejora en los resultados.

Respecto a la primera cuestión, vale la pena enumerar algunas de las técnicas utilizadas en este trabajo para minimizar los errores y la pérdida de poder predictivo de los modelos:

- En las etapas del proceso en que existió pérdida de información (e.g.: al normalizar los números de teléfono de las bases *nx_clientes* y *bm_mensajes* para su unión[7]) se aseguró que la pérdida no fuese significativa relativa al total de información, y además que la pérdida fuese aleatoria. Por ejemplo: no existe razón para afirmar que los números de teléfono "perdidos" durante el *join* entre las dos bases de datos, perteneciesen a algún segmento de usuarios con una probabilidad marcadamente mayor/menor de ser promotores que el promedio de usuarios.
- El *data leakage* se evitó tanto para los datos como para la metodología del investigador. Al hacer el *train-test split* al principio del código, se dejó apartado el conjunto de *test* hasta el final. Incluso las transformaciones básicas de los datos se realizaron sobre *train* para evitar cualquier vistazo de los datos de *test* que pudiesen influenciar al investigador. Asimismo, las funciones de *fit*, *encoding* y *scaling* utilizadas durante este trabajo, se realizaron siempre sobre *train* y se implementaron funciones que guardaban los parámetros aprendidos en *train* para luego utilizarlos en *test*, sin realizar un *refit*[8].
- Se realizaron distintas versiones de la metodología para comparar resultados con la metodología actual y asegurar que no existan diferencias de resultados muy grandes (lo que podría haber sido signo de una implementación o una elección erróneas de las funciones para desarrollar el modelo predictivo). Por ejemplo:
  - La validación cruzada se probó con distintos métodos (métodos que vienen implementados por default, función *K-Fold* y función *Stratified K-Fold*).
  - Para descartar problemas de desbalance de clases, se intentó balancear el modelo *XGBoost* mediante el factor *scale_pos_weight* = 8, obteniendo resultados muy similares a los resultados sin el escalamiento.
  - Si bien luego del *fine tuning* ya se obtuvo un modelo ganador, se evaluaron los 14 modelos en los datos de *test* para descartar un error de implementación en el

[7] En esta normalización se quitaron números de teléfonos duplicados y se normalizó la longitud de los números de teléfono tomando los últimos 9 caracteres del número telefónico. De esta forma, se logró unir los números de teléfono de ambas bases sin importar el formato en el que se encontraba el código de país (e.g.: un número de celular registrado como 005491155443322 en una base, resultará aparejado con el número de teléfono +541155443322 en otra base si se normalizan ambos números extrayendo los últimos 9 dígitos).

[8] Los únicos casos en los que se hizo un *refit* fueron aquellos en los que los datos categóricos de *test* contaban con nuevos valores, lo que no permitía codificarlos sin el aprendizaje de estos nuevos valores. Por ejemplo: cuando se encontraron palabras nuevas en las conversaciones de *test* que no aparecían en las conversaciones de *train*, hubo que incorporarlas a los features semánticos y hacer el *fit*.

código particular del modelo ganador, y se obtuvieron resultados consistentes con lo que se observó en datos de *train*.

- Se probaron métricas de *scoring* alternativas como *G-Means*, obteniendo resultados similares.
- A lo largo del proceso se aplicaron constantemente "verificaciones vs. la realidad". Siempre se verificó la coherencia entre los resultados parciales obtenidos y el conocimiento del dominio, por ser parte de la práctica diaria de uno de los investigadores, de forma de no confiar únicamente en los datos resultantes de la "caja negra" del modelo. Se buscó, en cambio, que todos los pasos tuvieran una interpretabilidad lógica y sentido. Cada objeto nuevo que se creó en el entorno fue visto varias veces por el ojo del investigador, *a mano*, sin utilizar ninguna función de agregación o resumen de los datos, valiéndose de exportaciones de los datos crudos con Microsoft Excel para su visualización y verificación. Así se descubrieron y corrigieron algunos errores, como la falta de espacios entre la última palabra de un mensaje y la primera del mensaje siguiente en una función de concatenado de mensajes, por dar un ejemplo.

Respecto a la segunda cuestión, es decir, el entendimiento sobre qué modificaciones o novedades deberían introducirse a los modelos predictivos de este trabajo para lograr una mejora en los resultados, vale la pena compartir algunos indicios sobre cómo podría reformularse el modelo predictivo en uno más potente.

Pareciera haber información que valdría la pena incorporar a las bases que nutren al modelo y así mejorar la predicción de la valoración de la experiencia de atención por parte de los clientes. Observando los reportes de la empresa, se observa que hay distintos factores externos que parecen afectar fuertemente la nota de la atención, que podríamos atribuir a *factores de contingencia*.

A modo de ejemplo de lo recién mencionado, se presenta el análisis de evolución de la experiencia de atención al cliente, presentado por el sector de calidad de la empresa, para los meses de abril y mayo del período estudiado[9]. En el eje Y se observa la misma variable dependiente que se estudió en este trabajo, pero normalizada a una escala del 0% al 100%:

[9] Algunos datos se encuentran ofuscados por cuestiones de confidencialidad.

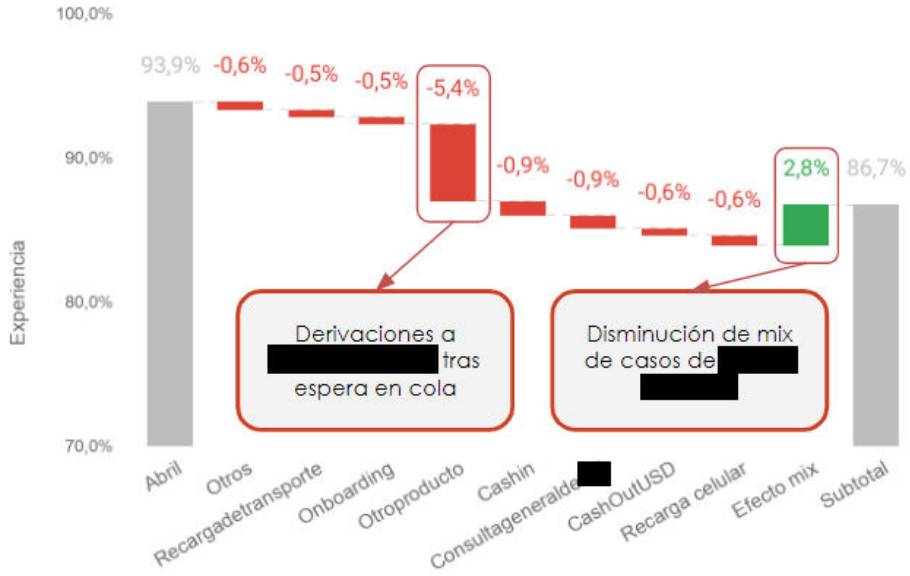

Fig. 6. Gráfico de cascada explicando los componentes de la caída en la variable dependiente de abril a mayo.[10]

En el eje X del gráfico se observan lo que se llaman las *tipologías de los chats*: cada vez que un agente concluye un chat, asigna una tipología que hace referencia a los motivos de la consulta o reclamo. Además, se muestra un efecto *mix*, que hace referencia a las variaciones en la métrica de experiencia vs. el mes anterior a causa de variaciones en el *mix* de tipologías.

Como se observa en el gráfico, hay marcadas variaciones en los niveles de la variable de experiencia causados por contingencias puntuales asociadas a alguna tipología en particular, que no tienen tanto que ver con los 3 grupos de features utilizados en este trabajo, sino más bien con el tema que se está consultando (que a veces no es capturado del todo por las variables semánticas planteadas en este trabajo). Existe, entonces, un gran peso de factores "externos" o "de contingencia" que hacen a las variaciones en el puntaje que los clientes otorgan. Para ilustrar de qué se tratan estas contingencias, y siguiendo con el ejemplo de la caída en la experiencia de la Figura 6, vemos que hubo una caída de 5,4pp en la métrica normalizada de experiencia de abril a mayo a causa de las consultas clasificadas como "Otro producto". En concreto, lo sucedido ese mes en la empresa estudiada fue que se enviaron comunicaciones a clientes por correo electrónico, que al final del correo ofrecían un enlace para contactarse al canal de atención al cliente de WhatsApp de la empresa. El enlace dirigía al cliente al canal de atención atendido por agentes con *expertise* en otro producto, distinto al que se mencionaba en la comunicación del correo electrónico. Esto, además de generar una saturación en la cola de atención,

[10] Informe de calidad del mes de mayo del área de *Customer Experience* de la empresa

hizo que la experiencia de estos clientes cayera mucho ya que, después de un largo tiempo de espera, se encontraban con un asesor que no podía resolver su consulta y debía derivarlos nuevamente a otro asesor que sí tuviese *expertise* en el tema, lo que empeoró la experiencia de los clientes y por lo tanto el puntaje que ponían finalmente en la encuesta al final de la conversación.

5.2 Aprendizajes y aporte de valor

A pesar de no haber conseguido un valor predictivo significativo, la metodología aplicada nos deja varios aprendizajes que pueden aportar valor a quienes se enfrenten a problemas similares. A continuación, se detallan estos aprendizajes.

- **Asegurar una metodología de trabajo y arquitectura del código que evite el data leakage tanto de los datos como del investigador**, siendo riguroso con el momento en el que se hace el *train-test split* y realizando todas las transformaciones, fittings, encondings y scalings de los datos sin observar el conjunto de test.
- **Observar la exhaustividad de las combinaciones de modelos y algoritmos testeados, así como la exploración de métricas de scoring alternativas**, de esta forma se asegura la coherencia de los resultados obtenidos.
- A lo largo de todo el trabajo, resulta importante realizar "**reality checks" mirando resultados parciales y en formato crudo de los datos manejados para asegurarse que toda transformación, secuenciación y etiquetado realizados tengan sentido** (lo cual muchas veces exigirá al investigador alto conocimiento del dominio).
- **No subestimar el tiempo necesario para la transformación de datos cuando el origen de datos venga de fuentes no pensadas para su explotación**. La relevancia de este punto es mayor cuando el orden o secuencia de los datos es de vital importancia y se cuentan con pocos elementos para la corrección o recuperación de esta secuenciación.
- Es muy importante hacer lo posible por descartar errores de programación. Para eso, en este trabajo (y es la recomendación sugerida) se vio conveniente realizar el unit testing para la mayoría de las funciones. En este unit testing **resulta particularmente relevante asegurar el correcto tamaño de los dataframes manejados y la completitud de las funciones de etiquetado de eventos**, principalmente en casos como este en el que hay una extracción de eventos y variables a partir de un único campo de texto que contiene mensajes y metadata en forma de string.
- Finalmente, el bajo poder predictivo del modelo pareciera estar relacionado con diversos factores ajenos a los features contextuales, conversacionales y semánticos que, según informes internos de la empresa, son los que más fuertemente afectaron los valores del puntaje a predecir. Por ejemplo: se realizaron envíos masivos de email en ciertos días que ofrecían el canal de WhatsApp como punto de contacto, generando picos de demanda que hicieron que las conversaciones de esos días tuviesen mayor espera en cola y, consecuentemente, un menor puntaje. También,

se comunicaron nuevas funcionalidades del producto a los clientes sin haber realizado la capacitación sobre la nueva funcionalidad a los agentes, lo cual genera un menor puntaje en los días en que se realizó la comunicación de la nueva funcionalidad (previo a la capacitación de los agentes). Como estos dos, existen muchos ejemplos; el aprendizaje principal en este caso es **la importancia de conocer el dominio y la operatoria real que genera los datos con los que se trabajará**.

5.3 Trabajos futuros

Como parte de los aprendizajes de este trabajo, se esbozan a continuación algunos posibles trabajos futuros para profundizar en algunos aspectos que exceden al alcance de la investigación aquí presentada.

- **Análisis en ventanas temporales menores**. Como se mencionó en la sección previa, existen eventos externos puntuales, no reflejados por completo en los features estudiados, que afectan el valor de la variable a predecir. Por ello, resultaría de interés replicar el análisis realizado en ventanas temporales menores (e.g.: de entre 1 y 4 semanas), para poder analizar el valor predictivo y la curva ROC tanto de ventanas temporales que contengan a estos eventos externos, como ventanas temporales que no los contengan.
- **Control del sobreajuste (*overfitting*)**. Si bien se realizaron algunas técnicas particulares para evitar el sobreajuste (e.g.: uso de un umbral de frecuencia mínima de palabras al crear la matriz documento-término y uso del parámetro *scale_pos_weight* en el modelo XGBoost), sería bueno explorar algunas técnicas adicionales para corroborar si efectivamente hay o no sobreajuste y para controlarlo, como por ejemplo:
  - Verificar si existe una caída significativa en la curva ROC al pasar de datos de *train* a datos de test,
  - Realizar técnicas de subsampling de la clase mayoritaria (promotores),
  - Generar instancias sintéticas a través de técnicas de oversampling.
- **Exploración de nuevos features**. Resulta de interés explorar e incorporar al modelo predictivo nuevos features que reflejen algunas características como: *sentiment analysis*, parametrización de factores de "contingencia" de la empresa, características de la conversación previa con el bot previo a la derivación con agente.

Referencias

1. Auguste, J., Charlet, D., Damnati, G., Béchet, F., & Favre, B. (2019). Can we predict self-reported customer satisfaction from interactions? In IEEE International conference on acoustics, speech and signal processing (ICASSP), pp. 7385-7389.
2. Cutler, A., Cutler, D. R., & Stevens, J. R. (2012). Random forests. Ensemble machine learning: Methods and applications, 157-175.
3. Chen, T., and Guestrin, C. (2016). "Xgboost: A scalable tree boosting system." Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining.
4. Mohammad, K. S., & Hemmatian, F. (2019). An efficient preprocessing method for supervised sentiment analysis by converting sentences to numerical vectors: A twitter case study. Multimedia Tools and Applications, 78(17), 24863-24882
5. Park, K., Kim, J., Park, J., Cha, M., Nam, J., Yoon, S., & Rhim, E. (2015). Mining the minds of customers from online chat logs. In Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, pp. 1879-1882.
6. Reichheld, Fred (2011). La Pregunta Decisiva 2.0, LID Editorial, 1° edición.