Master in Management + Analytics

Tesis

Modelo predictivo de detractores en casos de c*ustomer service*

por

Lucas Novo

Tutora: **Josefina Dalla Via Monti**

Junio 2021

Resumen

La industria del *e-commerce* viene creciendo a un ritmo exponencial tras cambios en los hábitos del consumo. En entornos cada vez más competitivos, aquellas empresas que adquieren políticas enfocadas en el usuario logran captar la mayor parte del crecimiento del negocio y así subsistir en el tiempo.

En la presente tesis se argumenta que, mediante técnicas de aprendizaje automático, se puede obtener información valiosa de los grandes volúmenes de datos que maneja una de las empresas latinoamericanas más importantes de *e-commerce* y así tomar decisiones de negocio que ayuden a mejorar la experiencia de los usuarios.

Se trabajaron diferentes modelos de aprendizaje automático con los datos (estructurados y no estructurados) disponibles de todos los casos abiertos por los usuarios en los canales *offline* del centro de *customer service*, y que tuvieron una respuesta en la encuesta de NPS (*Net Promoter Score*). El foco del trabajo estuvo puesto en la construcción de un modelo de aprendizaje automático que pudiera predecir la probabilidad de que un caso entrante termine con una mala experiencia, en función de la información conocida (nivel del usuario, país, proceso, etc.) y del texto que escribe explicando su motivo de contacto.

Mediante el uso de técnicas de ingeniería de atributos, procesamiento del lenguaje natural y experimentación con diferentes algoritmos, se logró sacar provecho de una base de datos en un contexto de un problema de clasificación.

Como resultado, se obtuvo un modelo con un rendimiento que alcanzó un puntaje de 0.70 de área bajo la curva ROC. Dicho modelo, tiene como objetivo ser la base para construir una mejora al sistema de asignación actual de la empresa (el cual se basa en asignar los casos de manera aleatoria) y así proponer una asignación de los casos con mayor probabilidad de detracción a los mejores representantes, de forma que podría servir para mitigar una mala experiencia del usuario, asignando el mejor representante al caso abierto. Tomando una serie de supuestos, se midió el impacto económico de implementar este nuevo sistema de asignación y se obtuvo que la ganancia anual sería de 4,2 millones de dólares.

Abstract

The e-commerce industry has been growing up at an exponential rate after changes in consumer habits. In increasingly competitive environments, those companies that acquire policies with special focus on customers, manage to capture most of the business growth and survive over time.

In this thesis it is argued that, with Machine Learning techniques, it is possible to get valuable information of the large volumes of data provided by one of the most important Latin American e-commerce companies, and then make business decisions that allow to improve customer experience.

We have worked on different Machine Learning models with the data (structured and unstructured) available from all cases opened by users in offline channels of customer service center, and which had a response in the NPS (Net Promoter Score) survey. The focus of this work was on the construction of a Machine Learning model that could predict the probability that an incoming case ends with a bad experience, based on the known information (user level, country, process, etc.) and the text explaining the reason of contact.

Using feature engineering techniques, natural language processing and experimentation with different algorithms, it was possible to take advantage of a database in the context of a classification problem.

As a result, a model was obtained with a performance that reached a score of 0.70 area under the ROC curve. This model aims to be the basis for an improvement of the current allocation system of the company (which is based on assigning cases randomly) and thus propose an assignment of the cases with the highest probability of detraction to the best representative, in a way that could serve to mitigate a bad experience, assigning the best representative to the open case. Taking a series of assumptions, the economic impact of implementing this new allocation system was measured and it was obtained that the annual profit would be 4.2 million dollars.

Índice de contenidos

Capítulo 1
Introducción al problema

1.1. Contexto

La atención al cliente en la industria del *e-commerce* abarca un conjunto de acciones y protocolos dispuestos por una tienda online para prevenir y solventar incidencias o necesidades concretas de manera individualizada [1].

En entornos cada vez más competitivos, se hace vital para la subsistencia de empresas de *e-commerce* aplicar políticas enfocadas al usuario, con especial foco en la experiencia de los usuarios.

Así como cumplir con las promesas de entrega de los paquetes y ofrecer un sinfín de funcionalidades en las aplicaciones es importante para el negocio, disponibilizar canales de atención al cliente con soluciones eficaces y buen atendimiento, también lo es.

1.1.1. Customer service

Para tener éxito en el mercado online de hoy en día, las tiendas de *e-commerce* deben centrarse en ofrecer una experiencia completa al cliente. Esto se está convirtiendo en el elemento diferenciador en las decisiones de los clientes [2].

Las tiendas de *e-commerce* ya están compitiendo principalmente en el servicio al cliente (*customer service*). Además de ofrecer un buen producto a un precio competitivo, *customer service* es claramente el nuevo campo de batalla.

La manera de ganar esta batalla es proporcionar a los clientes una experiencia que sea lo más fluida y sin esfuerzo como sea posible.

Según un estudio realizado por la firma Forrester [3], el 45% de los compradores de *e-commerce* de Estados Unidos abandona una transacción online si sus preguntas o inquietudes no se resuelven rápidamente. Con lo cual, brindar una experiencia completa en la atención al usuario y responder sus consultas correctamente, es de vital importancia para poder competir en la industria.

Como resultado de un mal servicio, los tiempos de interacción son más largos, los costos son más altos, y las empresas pierden oportunidades valiosas.

Las empresas que compiten en el mercado actual necesitan una estrategia de acción y un diferenciador real. Hoy, más que nunca, los asuntos de *customer service* son necesarios.

1.1.2. *Net Promoter Score* (NPS)

El NPS es un sistema y un indicador para medir la lealtad y la satisfacción del cliente. Su primera referencia apareció en 2003, en un artículo escrito por *Reichheld* y publicado en *Harvard Business Review* [4]. Es la métrica adoptada por la mayoría de las industrias, incluyendo las de *e-commerce*, para medir el grado de satisfacción del cliente.

Para lograrlo, el NPS se basa en la realización de una simple pregunta al cliente: 'En base a tu última experiencia con nosotros, ¿qué tan probable es que nos recomiendes a otras personas?' Donde 0 es 'Nada probable' y 10 'Muy probable'.

En función de su respuesta, se categoriza al cliente en 3 categorías:

- **Promotores:** clientes que respondieron con un 9 o un 10. Están satisfechos y, por lo tanto, leales a la marca. Con disposición a re-utilizar los productos/servicios de la empresa, y a recomendarla.
- **Pasivos:** clientes que respondieron con un 7 o un 8. Están satisfechos pero no son leales, por lo que son susceptibles de irse con la competencia.
- **Detractores:** clientes que respondieron desde el 0 hasta el 6. Están insatisfechos y pueden ser partícipes de un boca en boca negativo.

Tras reunir las respuestas de la población muestreada de clientes, el índice NPS se obtiene finalmente al restar el porcentaje de promotores con el de detractores, sin tomar en cuenta a los pasivos.

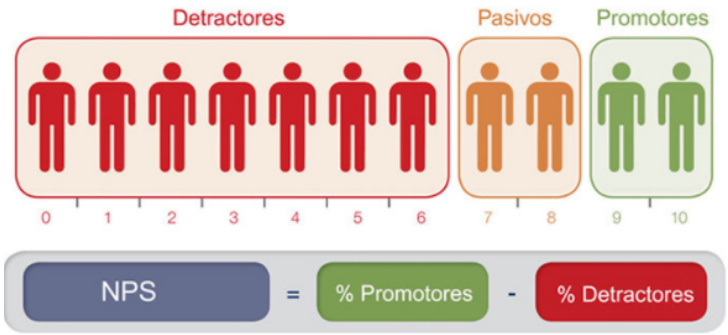


Figura 1.1: cálculo del Net Promoter Score

En línea con la política de enfoque en el usuario, todas las decisiones de negocio tienen foco en disminuir el porcentaje de usuarios que sean detractores, y convertir a todos aquellos usuarios pasivos en promotores.

1.2. Problema

De acuerdo a lo conversado con el equipo de *customer service* de una importante empresa latinoamericana de *e-commerce*, las decisiones de negocio que toman los *managers* de *customer service* son basadas en las tasas de contacto, y en resultados de NPS, donde el foco que hacen está en reducir todos aquellos usuarios que tuvieron una mala experiencia luego de contactarse con *customer service*. No toman un enfoque

proactivo, sino más bien reactivo: posterior a que el usuario complete la encuesta de NPS.

Los principales *drivers* de experiencia que tiene un usuario al contactarse con *customer service* son: buen atendimiento y eficacia en la respuesta.

Los canales de ingreso de contacto que facilita esta importante empresa de *e-commerce* son dos:

- **Canales online:** donde el usuario puede comunicarse con un representante de *customer service* a través de *chat* (tanto en la plataforma de la tienda como por *Whatsapp*) como por llamada telefónica. En estos canales, como la interacción es de manera inmediata, el usuario suele tener una buena experiencia, ya que facilita la comunicación y entendimiento del problema.
- **Canales offline:** donde el usuario puede comunicarse con un representante de *customer service* ingresando su consulta a través de un formulario, y detallar su problema a través de texto. Estos canales tienen la ventaja de ser más económicos que los canales online, pero la experiencia suele ser inferior ya que la respuesta se le da vía mail, y los tiempos terminan siendo más largos.

Dado el crecimiento de la demanda, y por cuestiones de capacidad restringida de presupuesto, los canales offline representan el 52% de los contactos, brindando un NPS de 14% (30pp inferior a la de los canales online, cuyo NPS es de 44%).

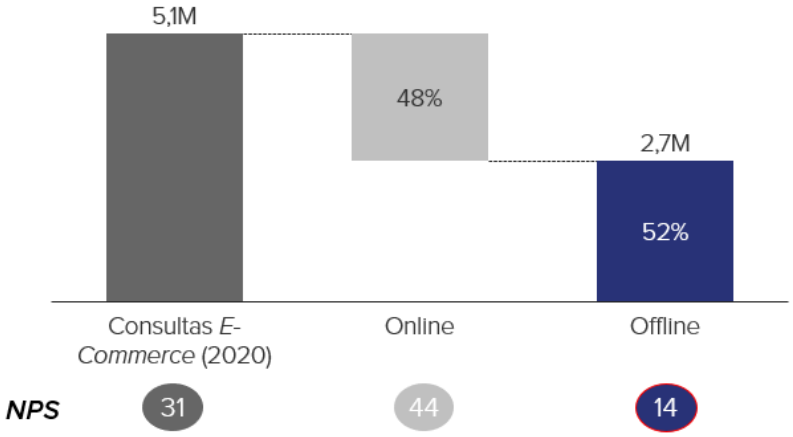


Figura 1.2: número de casos abiertos por usuario según origen de contacto

Actualmente, todos los contactos que ingresan a *customer service* en cada uno de los canales de atención, son asignados de manera aleatoria entre todos los representantes que atienden en dichos canales, independientemente de su grado de *expertise*; es decir, de cuán eficaces son dando respuestas al usuario, de forma tal de resolverles sus problemas, o bien, de darles respuestas de contención.

La razón que dan los *managers* de *customer service* de esta empresa se debe principalmente a que utilizan los modelos clásicos de asignación aleatoria dado que son fáciles de implementar y modificar, sin necesidad de incorporar complejidad a la asignación. A su vez, nunca se han planteado un enfoque de *Machine Learning* en las consultas que ingresan de los usuarios.

La contracara de esta asignación aleatoria, es que uno de los drivers de mayor peso en la experiencia de los usuarios previamente mencionado es la eficacia en la respuesta. Y este fenómeno se lo puede visualizar aperturando el NPS según el nivel del *seniority* del representante de *customer service*.

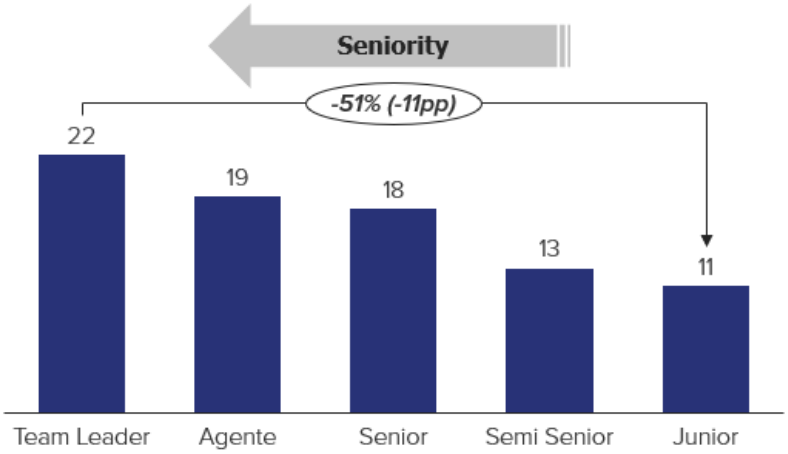


Figura 1.3: NPS por nivel de seniority del representante de customer service (offline)

Como se puede ver en la ***Figura 1.3***, a medida que crece el nivel de *seniority* del representante de *customer service*, mejor es la experiencia del usuario, llegando a existir una diferencia de 11pp de NPS según si el representante tiene poca experiencia (*junior*) o mucha experiencia (*team leader*).

1.3. Background

Para analizar la mejor solución que se adecúe al problema descripto, se procedió a investigar las soluciones que existen actualmente en el mercado. Es decir, si ya existen resoluciones al problema de asignación de representantes a casos de consulta y qué herramientas tecnológicas se han empleado para esto, entre otras cosas.

Se mencionan a continuación tres *papers* cuyos objetivos fueron estudiar enfoques tecnológicos con el fin de mejorar la atención al cliente.

- Oliver Müller [5] plantea el uso de *text analytics* para un mejor *management* de *customer service*, y de cómo extraer información de data no estructurada. Además, plantea el concepto de que una vez que las empresas logran comprender el análisis de los textos de las consultas de los usuarios, pueden extender el análisis a un monitoreo en tiempo real y tomar acciones que favorezcan a la efectividad y experiencia de los usuarios.

- Otro enfoque parecido lo plantea Hui [6] quien investiga sobre cómo aplicar técnicas de *data mining* para extraer conocimiento de la base de datos y así lograr dos actividades en *customer service*: (i) soporte para la toma de decisiones y (ii) diagnóstico de errores en tiempo real. A su vez plantea el concepto de *resource management* aplicado en *customer service*, donde el manipuleo de los

datos no estructurados permite asignar las tareas con la lógica de la performance de sus ingenieros (quienes atienden los casos abiertos por los usuarios)

- Por último, Auguste [7] plantea el uso de herramientas de *Machine Learning* para poder predecir el grado de satisfacción del usuario a través de su interacción con *customer service*. Para este objetivo, muestra como conclusión que usando modelos de aprendizaje supervisado de clasificación se logra medir el grado de satisfacción del usuario frente a un primer contacto.

1.4. *Machine Learning* y NLP

Una vez identificado el problema y estudiado las posibles soluciones planteadas para el mismo, se decidieron optar dos conceptos fundamentales en la presente tesis: *Machine Learning* y NLP. A continuación, se describe cada uno de ellos.

Machine Learning es una disciplina científica que se refiere a una categoría de algoritmos que utilizan estadísticas para encontrar patrones en cantidades masivas de datos. La presente tesis tiene como foco esta disciplina, a través de la cual se procesará el *dataset* y se extraerán *insights* valiosos y predicciones acorde al problema descripto.

El procesamiento del lenguaje natural [8], o NLP (*Natural Language Processing*) es un campo de las ciencias de la computación, inteligencia artificial y lingüística que estudia las interacciones entre las computadoras y el lenguaje humano.

El lenguaje humano es increíblemente complejo y diverso. Nos expresamos de maneras infinitas, verbalmente y por escrito. No sólo existen cientos de lenguajes y dialectos, sino que, además, existe un conjunto único de reglas gramáticas y de sintaxis, términos y palabras coloquiales. Cuando escribimos, a menudo cometemos errores ortográficos o abreviamos palabras, o bien omitimos signos de puntuación. Cuando hablamos, tenemos acentos regionales, y mascullamos o tomamos palabras prestadas de otros idiomas. [8]

Aunque el aprendizaje supervisado y no supervisado se utilizan ahora ampliamente para modelar el lenguaje humano, se necesitan también entendimiento sintáctico y semántico, y conocimientos de dominio que no están necesariamente presentes en estos métodos de *Machine Learning*. NLP es importante porque ayuda a resolver la ambigüedad del lenguaje y agrega estructura numérica útil a los datos para muchas aplicaciones industriales, como el reconocimiento del habla o la analítica de texto.

En la presente tesis, se emplearon técnicas de NLP, en las etapas de ingeniería de atributos, análisis descriptivo y entrenamiento de los modelos de *Machine Learning.*

1.5. Objetivo

La presente tesis tuvo como objetivo obtener un modelo de *Machine Learning* que permita predecir, con el mayor grado de precisión posible, la probabilidad de detracción de un usuario que abre una consulta con *customer service* en los canales *offline*.

Se decidió considerar a la categoría de detractor como categoría a predecir, ya que son estos tipos de usuarios los más afectados por una mala experiencia, y terminan generando un boca en boca negativo que afecta al negocio.

A su vez, la razón por la cual se hizo foco en los canales *offline* se debió a estos dos factores:

- Se cuenta con el problema del usuario detallado en el texto del formulario, previo a que el caso se asigne a un representante.
- El presupuesto de la empresa es limitado, y disminuir el mix de canales *offline* no es una opción viable.

Por lo mencionado anteriormente, se hizo especial foco en el análisis predictivo, desarrollando modelos que maximicen la performance.

Capítulo 2
Técnicas utilizadas

A continuación, luego de haber comprendido el problema a resolver, se detallan todas las técnicas que se utilizaron para construir el modelo predictor de detractores en casos de *customer service*, para cumplir con el objetivo descripto en el **Capítulo 1**.

2.1. Pre-procesamiento de datos

2.1.1. Evaluación de duplicados y valores faltantes

Dado que el *dataset* fue descargado mediante SQL, se descargó la base de forma tal que no existieran valores duplicados con el ID del caso único. Como *sanity check*, se verificó posteriormente los valores únicos de los ID de los casos.

Se realizó un análisis para chequear la existencia de valores nulos. Tras este análisis, se concluyó que los *features* a analizar no contienen valores nulos.

2.1.2. Eliminación de texto de las preguntas predeterminadas

Como el campo que contiene el texto del usuario incluye la pregunta predeterminada de la empresa de *e-commerce*, se hizo un listado de todas las preguntas predeterminadas y se filtraron las mismas, de forma tal de quedarse únicamente con el texto escrito por el usuario en el formulario.

2.2. Ingeniería de atributos

Alice Zheng [9] define a la ingeniería de atributos como el acto de extraer atributos de la data cruda y transformar los mismos en formatos que sean aptos para el modelo de *Machine Learning*.

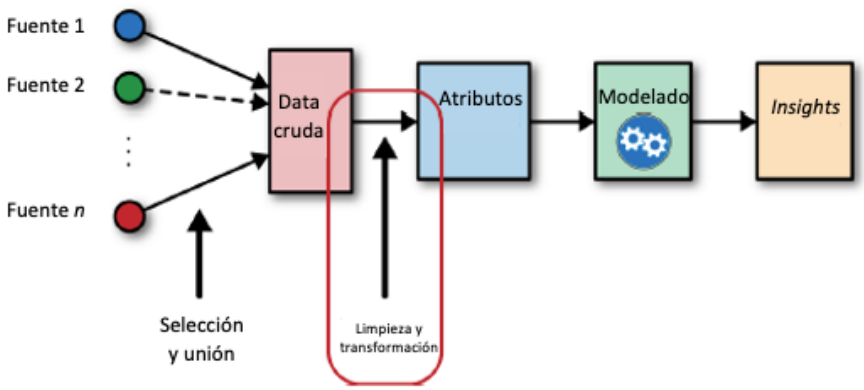


Figura 2.1: ingeniería de atributos situada en el flujo de Machine Learning

Es un paso crucial ya que los correctos atributos pueden facilitar el modelo y obtener así resultados de mejor calidad.

2.2.1. Estandarización del texto

Para el campo que contiene el texto del usuario, se procedió a hacer las siguientes modificaciones, de forma tal de estandarizar palabras y mejorar la performance del modelo:

- Pasar el texto a minúsculas.
- Eliminar caracteres alfanuméricos.
- Eliminar tildes.

De esta manera, la misma palabra fue computada como una, independientemente de si tenía mayúscula, tilde y/o carácter alfanumérico.

2.2.2. Tokenización del texto

Tokenizar [10] consiste en convertir una secuencia de caracteres en una secuencia de tokens. Cada token puede ser contado como una palabra. El tokenizador necesita saber qué caracteres indican que un token ha terminado y otro esté comenzando. Los caracteres de espacio son generalmente buenos separadores, al igual que los caracteres de puntuación.

Tokenizar cada uno de los comentarios de los usuarios permitió posteriormente aplicar operaciones para cada palabra, y computar la información de dicho texto.

2.2.3. Stemming y Lemmatization

Stemming [11] es una tarea de NLP que intenta reducir cada palabra a su lenguaje lingüístico básico, con la forma de la raíz de la palabra.

Se basa en la observación de que las palabras en los documentos a menudo tienen muchas variantes morfológicas. Por ejemplo, podemos usar las palabras “estando”, “estuve” y “estar” en el mismo documento. Estas palabras tienen claramente la misma raíz lingüística.

Stemming, permite procesar dichas palabras en una sola, donde toma la siguiente forma, citando el ejemplo anterior: “est”. De esta forma, todas las palabras “estando”, “estuve” y derivadas, van a ser transformadas a “est”.

Lemmatization, por el otro lado, permite procesar dichas palabras en una sola, donde toma la siguiente forma, citando al ejemplo anterior: “estar”. De esta forma, todas las palabras “estando”, “estuve” y derivadas, van a ser transformadas a “estar”.

Ponerlas juntas como si fueran ocurrencias de una sola palabra, permiten dar una fuerte indicación del contenido del documento mientras que cada palabra individualmente puede que no.

Para la presente tesis, se optó utilizar la opción de *Stemming*, ya que facilita la concentración de palabras, para un mejor grado de aprendizaje del modelo de *Machine Learning*.

Una de las mayores desventajas de esta heurística, es que ignora la similitud semántica entre palabras. Es decir, computa como palabras distintas a "auto" y "coche". Este conflicto fue tratado en la sección **2.2.8. Enfoque *Word2vec***.

2.2.4. Eliminación de *stop words*

Un enfoque ampliamente utilizado en el manipuleo de texto es la técnica de *stop words* [12], que consiste en eliminar palabras comúnmente usadas que no aportan valor para el modelo de *Machine Learning*. Para el lenguaje hispanohablante el cual se procedió a analizar en esta tesis, los ejemplos de *stop words* fueron: "de", "a", "el", "la", entre otros.

A su vez, se decidió eliminar la palabra "hola", ya que era una palabra típicamente utilizada por el usuario cuando abre un caso, y, al igual que las *stop words*, no aporta valor.

2.2.5. Creación de variable conteo de palabras y caracteres

Para facilitar la etapa de análisis exploratorio, se crearon dos *features* en el *dataset*:

- *Q_PALABRAS*: contiene la cantidad de palabras que utilizó el usuario.
- *Q_CARACTERES*: contiene la cantidad de caracteres que utilizó el usuario.

A su vez, estos nuevos *features* fueron tenidos en cuenta para el entrenamiento del modelo de *Machine Learning*.

2.2.6. Enfoque *Bag-Of-Words*

El enfoque *Bag-Of-Words* [13] permite convertir un documento de texto en un vector de cuenta. Dicho vector contiene una entrada para cada palabra posible en el vocabulario. Si la palabra "*lejos*" aparece tres veces en el documento, el vector de cuenta tiene un recuento de 3 en la posición correspondiente a esa palabra. Si una palabra del vocabulario no aparece en el documento, obtiene una cuenta de 0. Por ejemplo, el texto "*es un perro y es muy lindo*" tiene la representación de Bag-Of-Words mostrada a continuación:

Figura 2.2: conversión de texto al vector de Bag-Of-Words

Lo importante será entonces la geometría de la data en el espacio de variables. En el vector *Bag-Of-Words*, cada palabra se convierte en una dimensión del vector. Si hay n palabras en el vocabulario, entonces el documento será un punto ubicado en un espacio de n-dimensiones. A modo de representación, se muestra en la **Figura 2.3** tres textos en el espacio 3D correspondientes a las palabras "*perro*", "*muy*" y "*lindo*".

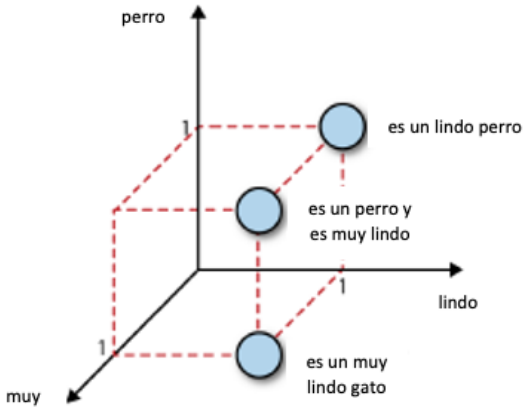


Figura 2.3: tres oraciones en el espacio 3D de variables

El enfoque *Bag-Of-Words* no es perfecto. Dividir una oración en palabras simples puede destruir el significado semántico. Por ejemplo "no está mal" semánticamente significa "decente" o incluso "bueno". Pero "no" y "malo" constituyen un sentimiento negativo. Sin embargo, se decidió usarlo ya que permite de forma simple convertir el texto en datos estructurados para los diferentes modelos de *Machine Learning*.

En la presente tesis, se decidió limitar el número de dimensiones, de forma tal de no aumentar la complejidad al modelo. En la sección **2.5. Cantidad de corte de palabras** se desarrolló el tema con mayor detalle.

2.2.7. Normalización Tf-Idf

Alice Zheng [14] define a la normalización Tf-Idf como un simple giro en el enfoque de *Bag-Of-Words*. Sus siglas significan *Term Frecuency – Inverse Document Frequency*. En lugar de mirar los recuentos de cada palabra en cada documento de un conjunto de

datos, mira un recuento normalizado donde cada palabra contada es dividida por el número de documentos en los que aparece esta palabra. Esto es:

$$bow\ (w,d) = \ veces\ palabra\ w\ aparece\ en\ el\ documento\ d\ tf - idf\ (w,d) = bow\ (w,d) * \frac{N}{\#\ documentos\ donde\ aparece\ w} \quad (2.1)$$

N es el número total de documentos en el conjunto de datos. La fracción *N/(# documentos donde aparece w)* es lo que se conoce como frecuencia inversa de documentos. Si una palabra aparece en muchos documentos, entonces su frecuencia inversa de documentos es cercana a 1. Si una palabra aparece en sólo unos pocos documentos, la frecuencia inversa de los documentos es mucho mayor.

Alternativamente, podemos tomar una transformada logarítmica en vez de usar el documento de frecuencia inversa. El logaritmo convierte el 1 en 0, y convierte números mucho más grandes (aquellos muy mayores a 1) a menores.

Si definimos Tf-Idf como:

$$tf - idf\ (w,d) = bow\ (w,d) * \ log\ (\frac{N}{\#\ documentos\ donde\ aparece\ w}) \quad (2.2)$$

entonces, una palabra que aparece en cada documento se eliminará de manera efectiva, y una palabra que aparece en muy pocos documentos tendrá un recuento aún mayor que antes.

2.2.8. Enfoque *Word2vec*

Como alternativa al enfoque de *Bag-of-Words* con normalización *Tf-Idf*, se empleó la técnica de *Word2vec*.

Word2vec [15] es una técnica de NLP donde trata a las palabras como una representación vectorial de n-dimensiones. A diferencia de la técnica de *Bag-of-Words, Word2vec* permite capturar no solo el significado de una palabra en el documento, sino también las similitudes semánticas y sintácticas, y las relaciones con otras palabras.

Su técnica de aprendizaje se basa en aprender las palabras embebidas utilizando una red neuronal de 2 capas. Como se ve en la **Figura 2.4**, existen 2 algoritmos de entrenamiento:

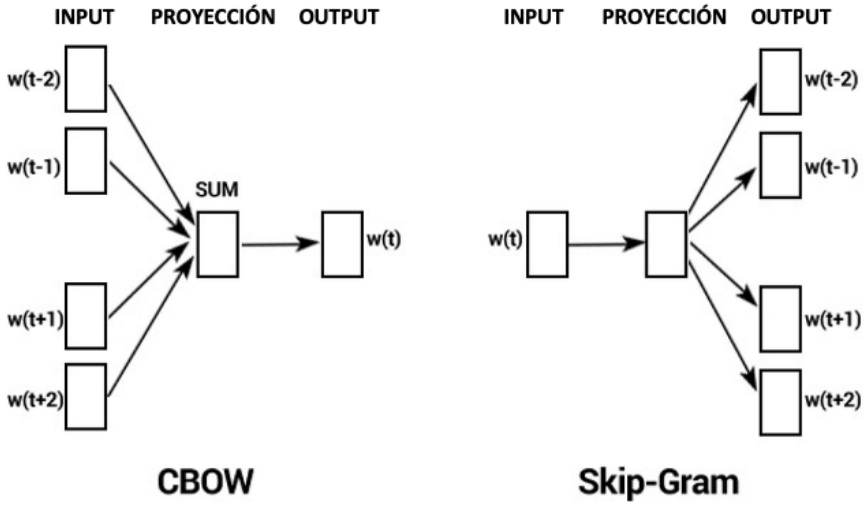


Figura 2.4: algoritmos CBOW y Skip-gram en Word2vec

La principal diferencia entre ambos algoritmos es que CBOW usa el contexto para predecir una palabra *target*, mientras que *Skip-gram* usa una palabra para predecir el contexto de una palabra *target*, y así capturar más de una relación semántica para una palabra.

Se hicieron uso de ambos algoritmos, y se seleccionó aquel que logró mejorar la performance del modelo.

Una vez entrenado el modelo *Word2vec* (con el vocabulario del *dataset* o bien importando un modelo pre-entrenado*)*, el modelo no solo permite estructurar el texto en un vector numérico de n-dimensiones, sino también estudiar las similitudes entre palabras.

Esta similitud de palabras se computa mediante la similitud coseno, cuya fórmula es la siguiente:

$$(A, B) = \frac{A.B}{\|A\|.\|B\|} \tag{2.3}$$

Palabras similares tendrán vectores con una distancia angular θ pequeña y un *cos(*θ*)* cercano a 1.

Dado que el texto que completa un usuario presenta un conjunto de palabras, el modelo *Word2vec* transformará cada una de las palabras en un vector de n-dimensiones y posteriormente, generará un vector de n-dimensiones que compute en cada una de sus dimensiones al promedio de los valores de las palabras en dicha dimensión. Por lo tanto, el modelo permitirá trabajar como *input* al conjunto de palabras escritas por el usuario, y como *output* tendrá un vector de n-dimensiones, que será tratado como n-variables del *dataset*.

2.2.9. One Hot Encoding

Además del texto escrito por el usuario, el *dataset* contiene variables categóricas. Por ejemplo, el país de contacto puede ser Argentina, México, Chile, etc. Por lo tanto, se necesitó un método de codificación para convertir esas categorías no numéricas en numéricas. Una técnica para dicha conversión fue la de asignar un número entero (ejemplo: 1) a cada una de las categorías posibles.

El método utilizado fue *One Hot Encoding* [16], donde cada bit representa una categoría posible. Si la variable no puede pertenecer a varias categorías a la vez, entonces solo un bit en el grupo puede estar "activado". Por lo tanto, una variable categórica con *k* categorías posibles se codifica como un vector de características de longitud *k*. En la **Figura 2.5** se representa un ejemplo para clarificar el concepto.

	e_1	e_2	e_3
San Francisco	1	0	0
New York	0	1	0
Seattle	0	0	1

Figura 2.5: One Hot Encoding de una variable categórica de 3 valores

One Hot Encoding es simple de implementar, pero usa un bit más del que es estrictamente necesario. Si vemos que *k-1* de los bits son 0, entonces el último bit debe ser 1 porque la variable debe tomar uno de los *k* valores. Matemáticamente, se puede escribir esta restricción como la suma de todos los bits debe ser igual a 1:

$$e_1 + e_2 + \cdots + e_k = 1 \tag{2.4}$$

2.3. Separación *train-set & test-set* y evaluación de performance

Conocido el *dataset* e identificada la variable categórica a predecir por los modelos, se procedió a separar el *dataset* en dos: *train-set* y *test-set*.

Esta separación se realizó para evaluar correctamente la performance de los modelos de clasificación en datos desconocidos y así entrenar los modelos en el *train-set*, ajustar los hiperparámetros mediante el aprendizaje supervisado por la variable a predecir, y posteriormente testear sus errores de predicción de la variable categórica en el *test-set*.

La razón por la cual se decidió emplear esta técnica se debe a que es una de las maneras más simples de simular los comportamientos sobre los datos desconocidos, y su costo de cómputo es relativamente bajo. La desventaja a tener en cuenta es que tiene una dependencia del azar, es decir, de cómo se separan las submuestras dentro del *train-set*.

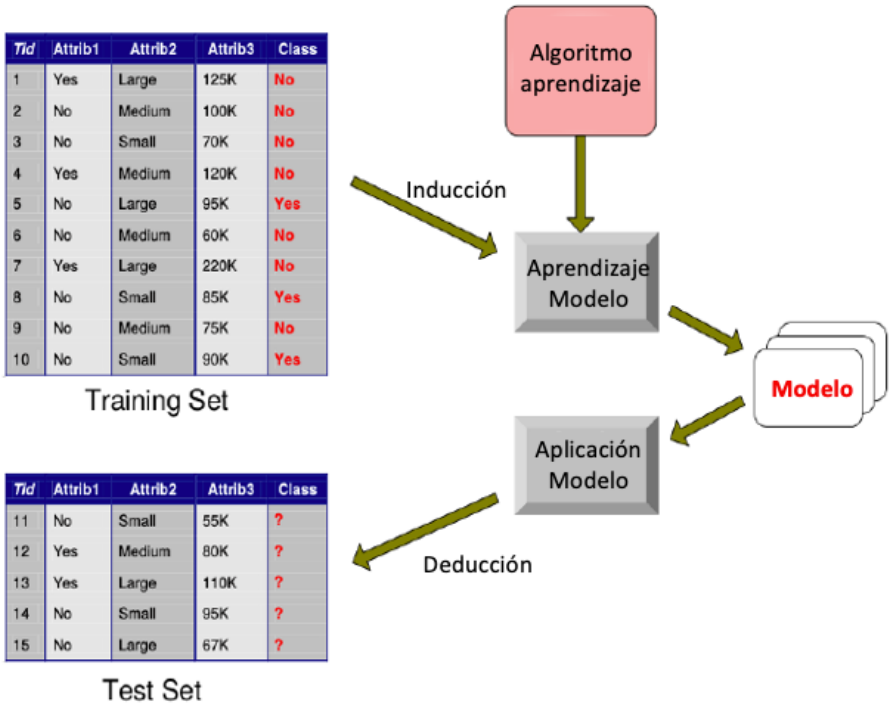


Figura 2.6: separación del dataset en train-set y test-set

Con respecto a esta etapa, se tomaron dos decisiones para los modelos a entrenar:

- La separación de *train-set* y *test-set* se realizó de manera aleatoria, con el objetivo de disminuir los posibles sesgos dentro del *dataset*, y maximizar la performance de los modelos.
- Asignarle el 30% del *dataset* al *test-set*, dado que se cuentan con suficiente volumen de datos, y así poder mejorar el cómputo de la performance de los modelos.

2.4. Riesgo de *Overfitting & Underfitting*

Una vez asignado el conjunto de datos *train-set* a los modelos de *Machine Learning*, se debió tener especial cuidado a la hora de entrenar los mismos. Ajustar demasiado los hiperparámetros a los datos del *train-set* puede hacer que el modelo de especial foco en reducir los errores dentro de dicho conjunto de datos, se vuelva un modelo muy complejo de entrenar y le de relevancia a variables poco importantes [17]. Por el otro extremo, puede ocurrir que los hiperparámetros se ajusten insuficientemente, y que el modelo termine siendo muy simple.

Dichos extremos se denominan como:

- **Underfitting:** se trata de un algoritmo demasiado rígido que no captura patrones relevantes, y consecuentemente tiene mala performance en el *train-set* y *test-set*. Un modelo que presenta *underfitting* suele tener baja flexibilidad, alto sesgo y baja varianza.

- **Overfitting:** se ajusta en exceso a las particularidades de los datos de entrenamiento, y consecuentemente tiene mala performance en el *test-set*. Un modelo que presente *overfitting* suele tener alta flexibilidad, bajo sesgo y alta varianza.

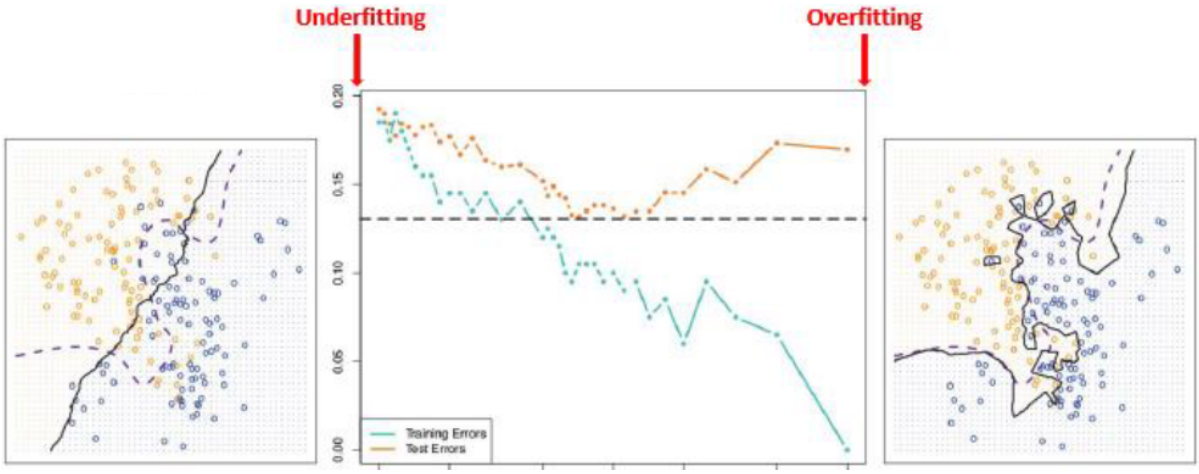


Figura 2.7: extremos de underfitting y overfitting

Tal como se puede visualizar en la **Figura 2.7**, se debieron buscar modelos que se ajustaran equilibradamente entre los extremos de *underfitting* y *overfitting*, y así lograr un modelo que permita tener buena performance en futuros datos, maximizando las probabilidades de predecir usuarios detractores.

2.5. Cantidad de corte de palabras

Dado que gran parte del *dataset* contiene texto tokenizado, con un volumen extenso de vocabulario, se empleó una técnica diferente a las descriptas anteriormente. La misma consiste, para el enfoque *Bag-of-Words*, en filtrar aquellos tokens que tengan una aparición menor a una cantidad fija *Q_CORTE* dentro del corpus. Con esta técnica, se logró disminuir la carga de cómputo de los modelos, reduciendo la dispersión de los datos y maximizando la performance de los modelos.

Con el fin de optimizar este parámetro *Q_CORTE*, se corrieron los modelos y sus hiperparámetros óptimos para los siguientes valores de *Q_CORTE*: 5, 20 y 100, quedándose con aquel que mejore la performance del modelo.

2.6. Modelos

Con el objetivo de entrenar un modelo de clasificación con buena capacidad de predicción, se utilizaron cuatro algoritmos de aprendizaje supervisado:

- Modelo de regresión logística LOGIT
- Modelo de *Naïve Bayes*
- Modelo de *Random Forest*

- Modelo *XG Boost*

Para estos cuatro algoritmos, se buscaron los hiperparámetros óptimos de cada uno, de forma tal de obtener la máxima capacidad predictiva, y se seleccionó aquel que logró el mayor grado de precisión.

2.6.1. Aprendizaje supervisado

La mayoría de los problemas de aprendizaje estadístico pueden pertenecer a una de estas dos categorías: supervisado o no supervisado.

Para un aprendizaje supervisado [18], cada observación de la medición del predictor x_i, $i= 1, \ldots, n$ tendrá asociado una respuesta y_i. El objetivo de dicho aprendizaje es el de ajustar un modelo que relacione la respuesta con la del predictor, con el objetivo de predecir con precisión la respuesta para futuras observaciones (predicción) o bien comprender mejor la relación entre la respuesta y los predictores (inferencia).

Por el contrario, el aprendizaje no supervisado describe una situación de mayor complejidad donde para cada observación $i= 1, \ldots, n$, observamos un vector de medidas x_i pero sin respuesta asociada y_i. No es posible ajustar un modelo de regresión lineal ya que no hay una variable de respuesta a predecir. La situación se denomina no supervisada ya que se carece de una variable de respuesta que pueda supervisar el análisis.

Dado que se contó con información real clasificada según si fue o no detractor, se abordó el problema con aprendizaje supervisado.

2.6.2. Problema de clasificación

Lo que caracteriza al aprendizaje supervisado es que permite predecir una variable. Dicha variable puede ser continua o categórica. En los casos donde se busca predecir una variable continua, se plantea un problema de regresión; en los casos donde la variable es categórica, de clasificación.

Como en la presente tesis se buscó predecir si un usuario pertenecerá a la categoría 'detractor', se abordó el aprendizaje supervisado como un problema de clasificación.

2.6.3. Modelo de regresión logística LOGIT

Los modelos de regresión logística [19], permiten modelar la probabilidad de que un valor *Y* corresponda a una categoría en particular.

En problemas de clasificación binarios, se modela de la siguiente manera:

$$E\ (Y \mid X_1, \ldots, X_p) = P\{Y = 1 \mid X_1, \ldots, X_p\} \tag{2.5}$$

Por medio de la regresión logística, obtenemos:

$$log\left(\frac{P(Y=1 \mid X_1,\ldots,X_p)}{P(Y=0 \mid X_1,\ldots,X_p)}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p \quad (2.6)$$

De este modo, se obtiene un log-ratio de probabilidades en función lineal de las *X's*. A este ratio se lo denomina como *log-odds* o LOGIT.

La regresión logística es simple ya que es un clasificador lineal. Dada su simplicidad, es un buen primer clasificador para entrenar. Como input, toma una combinación ponderada de las características de entrada y las pasa a través de una función sigmoide [20]. La función transforma una entrada de número real, *x*, en un número entre 0 y 1.

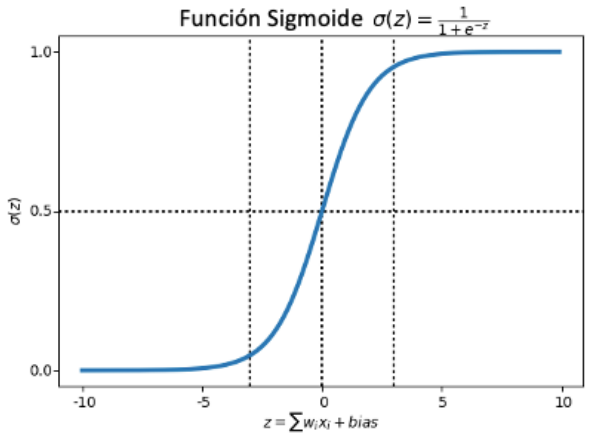


Figura 2.8: función sigmoide

El modelo tiene un conjunto de parámetros *w*, que representa la pendiente del aumento alrededor del punto medio 0,5. El término de intercepción *bias* denota el valor de entrada donde la salida de la función cruza el punto medio.

Un clasificador logístico predecirá la clase positiva si la salida sigmoide es mayor que 0,5, y con la clase negativa en caso de que sea menor. Al variar *w* y *bias*, uno puede controlar dónde ocurre ese cambio en la decisión, y qué tan rápido debe responder la decisión a los valores de entrada cambiantes alrededor de este punto medio.

2.6.4. Modelo de *Naïve Bayes*

El modelo de *Naïve Bayes* [21] permite encontrar la clase más probable de una observación, empleando para ello el Teorema de *Bayes*.

En problemas de clasificación multi-clase, se modela de la siguiente manera:

1. *Sean k clasificadores mutuamente excluyentes y exhaustivos c_1, c_2, …, c_k con probabilidades $P(c_1)$, $P(c_2)$, …, $P(c_k)$, respectivamente, y con n atributos a_1, a_2, …, a_n con valores de v_{i1}, v_{i2}, …, v_{in} para una dada observación i, respectivamente. La*

probabilidad de pertenecer a la clase k para una dada observación i es proporcional a:

$$P\ (c_k \backslash x_i) \cong P(c_k) * P(a_1 = v_{i1} \wedge a_2 = v_{i2} \wedge \dots a_n = v_{in} \setminus c_k) \quad (2.7)$$

2. *Si se asume que los n atributos son independientes, se puede reformular la expresión matemática a:*

$$P\ (c_k \backslash x_i) \cong P(c_k) * P(a_1 = v_{i1} \backslash\ c_k) * P(a_2 = v_{i2} \backslash\ c_k) * \dots P(a_n = v_{in} \setminus c_k) \quad (2.8)$$

3. *Para cada observación i, se calcula la probabilidad para cada una de las k categorías.*

4. *La clase predicha de la observación i será aquella con el mayor valor de probabilidad calculado.*

Si bien este modelo no suele alcanzar gran performance dado que no alcanza a descubrir relaciones complejas entre variables, se decidió utilizarlo como segundo clasificador ya que, al igual que el modelo de regresión logística LOGIT, es un modelo simple y rápido de entrenar.

2.6.5. Modelo de *Random Forest*

2.6.5.1. Árbol de decisión

Para explicar el modelo *Random Forest*, hay que comenzar por la definición de un árbol de decisión [22]. El árbol de decisión emplea el siguiente algoritmo:

1. Se divide el espacio predictor – es decir, el set de posibles valores para X_1, X_2, …, X_p – en *J* diferentes y no superpuestas regiones, R_1, R_2, …, R_J a través de particiones binarias recursivas con las características de ser (i) *Top-Down*, ya que comienzan las particiones desde una sola región y (ii) *Greedy*, ya que en cada partición se toma la mejor en ese paso en particular (y no la mejor para futuras particiones)

2. Cada partición binaria del espacio se realiza con el criterio de reducir el índice de Gini definido por:

$$G = \sum_{k=1}^{K} \widehat{p_{mk}}\ (1 - \widehat{p_{mk}}) \quad (2.9)$$

Donde $\widehat{p_{mk}}$ corresponde a la proporción de observaciones de la clase *k* en la región *m*. El índice de Gini permite medir la pureza de una clase dentro de una región, donde toma un valor pequeño si el nodo contiene observaciones predominantes de una clase *k*.

3. Dado que muchas particiones conducirán a aumentar la complejidad del modelo (y por ende el riesgo de *overfitting*), se fija un criterio de parada en las particiones. En la presente tesis, se emplearon los siguientes parámetros:
   - Mínimo número de observaciones que deben existir en una región para que una partición ocurra.
   - Mínimo número de observaciones que deben existir en la última región.
   - Máxima profundidad de cortes permitida.
   - Parámetro de complejidad α:

$$G' = G + \alpha \mid T \mid \tag{2.10}$$

   Donde $\mid T \mid$ indica el número de nodos terminales del árbol *T*. El parámetro α permite controlar el trade-off entre la complejidad del árbol y el ajuste al train-set.

4. Para cada observación que cae en la región R_J, se calcula como valor predicho a la mayoría de las observaciones que caen dentro de dicha región R_J.

La **Figura 2.9** detalla las particiones del espacio de un árbol de decisión:

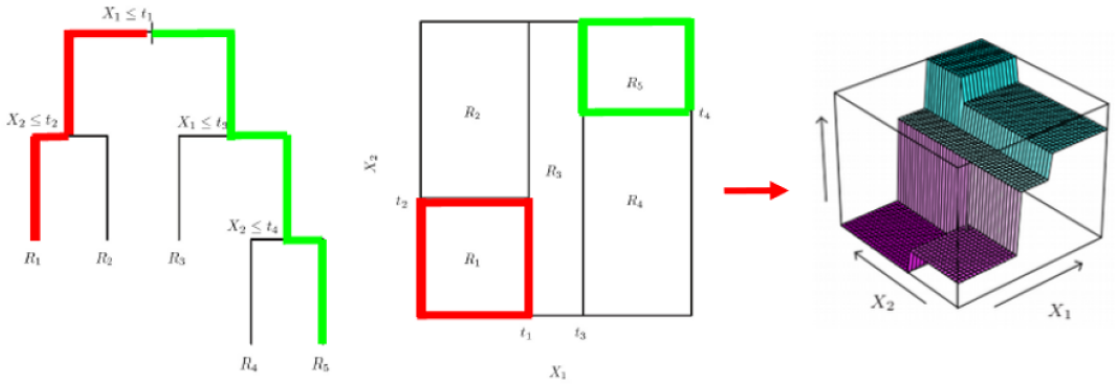


Figura 2.9: funcionamiento de un árbol de decisión

2.6.5.2. Algoritmo *Bagging* y muestreo *Bootstrap*

Si bien los árboles de decisión poseen bajo sesgo, sufren de alta varianza. El algoritmo *Bagging* [23], por consiguiente, consiste en separar el *train-set* en muestras aleatorias con reposición (denominadas *Bootstrap*) y entrenar en cada muestra un árbol de decisión. De esta manera, se logra reducir la varianza y por ende, aumentar la capacidad predictiva del modelo.

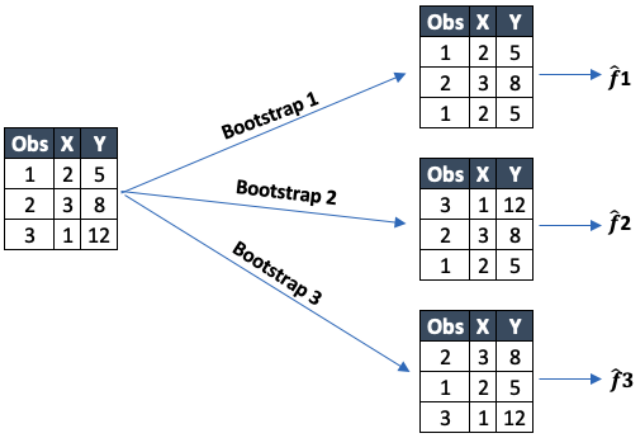


Figura 2.10: Bagging y muestreo Bootstrap

Como se puede ver en la **Figura 2.10**, el valor predicho será calculado por la votación de la clase predicha por la mayoría de los árboles.

2.6.5.3. Algoritmo Random Forest

El algoritmo *Bagging* posee la desventaja de que los árboles entrenados están fuertemente correlacionados, ya que cada uno de ellos usa los mismos predictores, por lo que todos los árboles terminan siendo similares entre sí.

Promediar varios árboles correlacionados entre sí no conduce a reducir significativamente la varianza, pero sí se reduce en el caso de que los árboles no estén correlacionados. El modelo de *Random Forest* [24] se basa en este principio, en donde construye árboles de decisión con muestreo *Bootstrap*, con muestras de *m* predictores del conjunto completo de *p* predictores. La división permite usar solo uno de esos *m* predictores. Una muestra nueva de *m* predictores se toma en cada división, y típicamente se escogen $m \approx \sqrt{p}$ – es decir, el número de predictores considerados en cada división es aproximadamente igual a la raíz cuadrada del número total de predictores. Una vez completados todos los árboles, el algoritmo toma como clase predicha, a aquella que surge de la votación de la mayoría de los árboles.

En otras palabras, al construir un *Random Forest*, en cada división del árbol, el algoritmo no puede considerar la mayoría de los predictores disponibles. Supongamos que hay un predictor muy fuerte en el conjunto de datos, junto con un número de otros predictores moderadamente fuertes. Luego, en la colección de los árboles *bagged*, la mayoría o todos los árboles utilizarán este potente predictor en la división superior. En consecuencia, todos los árboles *bagged* se verán bastante similares entre sí. Por lo tanto, las predicciones de los árboles *bagged* estarán altamente correlacionados.

Random Forest soluciona dicho problema obligando a cada división a considerar sólo un subconjunto de los predictores. Por lo tanto, en promedio $\frac{p-m}{p}$ de las divisiones ni siquiera considerarán el predictor fuerte, por lo que otros predictores tendrán más

posibilidades. Podemos pensar a este proceso como una decorrelación de árboles, lo que hace que el promedio de los árboles resultantes sea menos variable y por lo tanto más confiable.

Como se puede ver en la **Figura 2.11**, la principal diferencia entre modelos *Bagging* y *Random Forest* es la elección del tamaño del subconjunto de predictores *m*.

Figura 2.11: muestreo doble de Random Forest vs Bagging

El uso de un valor pequeño de *m* en la construcción de un *Random Forest* normalmente será útil cuando tenemos una gran cantidad de predictores correlacionados.

2.6.6. Modelo *Gradient Boosting XG Boost*

El modelo *XG Boost* pertenece a la familia de algoritmos *Boosting* [25] y consiste en crear varios árboles de decisión a partir del *train-set*, y entrenarlos secuencialmente. La particularidad que tiene es que cada árbol consume información del árbol entrenado previamente.

De esta manera, posee un funcionamiento de tipo ensamble, en donde se combina las predicciones de modelos más pequeños, tomando como predicción final alguna de las combinaciones de cada árbol.

El algoritmo que trabaja es el siguiente:

1. *Sea $\hat{f}(x) = 0$ y $r_i = y_i$ para todo i del train-set, donde $\hat{f}(x)$ corresponde al valor predicho inicial, r_i al valor residual de lo que aún no se predijo e y_i a la variable objetivo.*

2. *Para b= 1, 2, …, B cantidad de árboles se repiten los siguientes tres pasos:*

    a. *Se ajusta el árbol $\hat{f}(b)$ con d separaciones (d+1 nodos terminales) al train-set (X, r).*

b. *Se actualiza $\hat{f}(b)$ agregando una versión reducida del nuevo árbol:*

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \widehat{f}^{b}(x) \tag{2.11}$$

donde el parámetro λ corresponde al parámetro shrinkage o tasa de aprendizaje, que controla la velocidad de aprendizaje del modelo.

c. *Se actualizan los valores residuales,*

$$r_i \leftarrow r_i - \lambda \widehat{f}^{b}(x_i) \tag{2.12}$$

3. *El output del modelo Boosting será:*

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \widehat{f}^{b}(x_i) \tag{2.13}$$

A diferencia de ajustar un solo árbol de decisión a los datos, lo que equivale a ajustar los datos de forma difícil y potencialmente con *overfitting*, el modelo *Boosting* ajusta un árbol utilizando los residuos actuales, en lugar del resultado *Y* como respuesta. Luego, agrega este nuevo árbol de decisión en la función ajustada con el objetivo de actualizar los valores residuales. Cada uno de estos árboles pueden ser relativamente pequeños con sólo unos pocos nodos terminales, determinados por el parámetro *d* del algoritmo.

Ajustando pequeños árboles a los valores residuales, se va lentamente optimizando $\hat{f}$ en áreas donde no performa bien. El parámetro de aprendizaje λ relentiza el proceso, permitiendo que más árboles de diferentes formas ataquen los valores residuales.

A diferencia del modelo de *Random Forest*, donde cada árbol de entrena de manera independiente de los demás, en el modelo *Boosting* el entrenamiento de cada árbol depende de los árboles previamente entrenados.

2.6.7. Importancia de variables

Los modelos de árboles utilizados (*Random Forest* y *XG Boost*) tienen la particularidad de permitir computar la importancia de cada uno de los *features* del *dataset* respecto a la variable a predecir, lo cual ayuda a la hora de realizar un análisis exploratorio de los datos.

Para calcular la importancia de cada variable en un modelo de clasificación [26], se utilizó el índice de Gini (medida de desigualdad de clases), el cual su valor decrece por divisiones sobre un predictor dado, promediado sobre todos los árboles *B*. Por lo tanto, las variables con la mayor disminución media en el índice de Gini serán aquellas con mayor importancia.

En la sección **4.6. Valuación de importancia de variables** se mostró el resultado, con las variables más significativas para predecir probabilidad de detracción.

2.6.8. Métricas de performance

Para poder medir la performance de cada uno de los modelos descriptos en la sección **2.6. Modelos,** y así poder elegir aquel de mayor performance, se utilizaron las métricas de *precision*, *recall,* F1 *score, accuracy* y área bajo la curva ROC [27]. Estas métricas fueron aplicadas dentro del *test-set*.

Como métrica *core* de análisis y comparación de la performance de los modelos, se utilizó el área bajo la curva ROC ya que, como se verá a continuación, permite independizarse de una probabilidad de corte.

Para acompañar la definición de cada una de ellas, se muestra a continuación una matriz de confusión [28], la cual permite identificar 4 cuadrantes según la clase actual y la predicha por el modelo.

		Clase predicha		Instancias totales
		+	−	
Clase actual	+	TP	FN	P
	−	FP	TN	N

Figura 2.12: matriz de confusión

Donde:

- **TP (*True Positives*):** número de instancias positivas que son clasificadas como positivas.
- **FP (*False Positives*):** número de instancias negativas que son clasificadas como positivas.
- **FN (*False Negatives*):** número de instancias positivas que son clasificadas como negativas.
- **TN (True Negatives):** número de instancias negativas que son clasificadas como negativas
- **P (TP+FN):** número total de instancias positivas
- **N (FN+TN):** número total de instancias negativas

2.6.8.1. Precision

Precision permite medir la proporción de instancias clasificadas como positivas que son realmente positivas.

Para calcular la *precision*, se usó la siguiente fórmula:

$$precision = \frac{TP}{TP + FP} \tag{2.14}$$

2.6.8.2. Recall

Recall permite medir la proporción de instancias positivas que son correctamente clasificadas como positivas.

Para calcular el *recall*, se usó la siguiente fórmula:

$$recall = \frac{TP}{TP + FN} \tag{2.15}$$

2.6.8.3. F1 *score*

F1 *score* permite combinar las medidas *precision* y *recall* en un solo valor. Esto es práctico ya que hace más fácil poder comparar el rendimiento combinado de ambas métricas.

Para calcular el F1 *score*, se usó la siguiente fórmula:

$$F1\ score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{2.16}$$

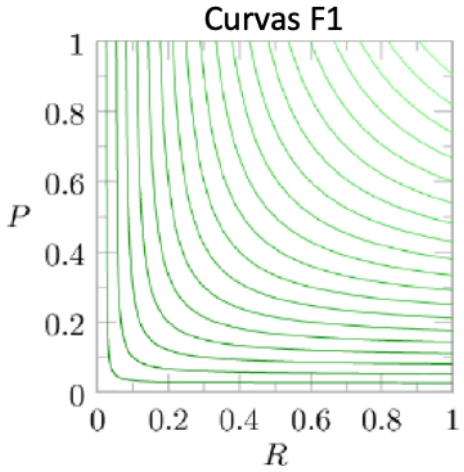


Figura 2.13: curvas F1 score

2.6.8.4. Accuracy

Accuracy mide el porcentaje de casos que el modelo ha acertado. Esto es, la proporción de instancias que son correctamente clasificadas.

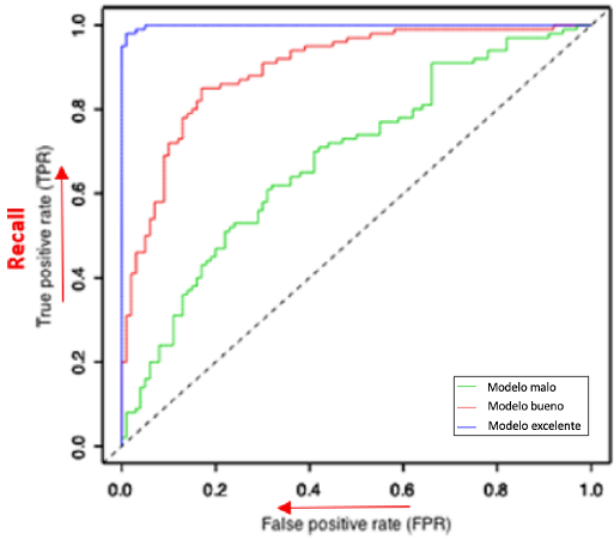
Para calcular el *accuracy*, se usó la siguiente fórmula:

$$Accuracy = \frac{TP + TN}{P + N} \tag{2.17}$$

2.6.8.5. Área bajo la curva ROC

Las métricas descriptas previamente tienen el problema que dependen de una probabilidad de corte. Es decir, a partir de qué probabilidad (suele tomarse 0,5) el modelo debe clasificar por una u otra clase.

La métrica área bajo la curva ROC (*Receiver Operating Characteristics*) permite independizarse de esta probabilidad. En la misma, se grafica el ratio FP en el eje horizontal, y el ratio TP en el eje vertical. Cada punto del gráfico puede ser escrito como el par de valores (*x*,*y*) indicando que el ratio FP corresponde al valor *x* y el ratio TP al *y*. Para clarificar la definición, se muestra la representación de los ejes en la **Figura 2.14.**



2.14: métrica área bajo la curva ROC

El área bajo la curva (AUC) permite medir la capacidad de predicción del clasificador. Cuanto mayor sea, mejor será el modelo prediciendo. Un AUC grande implicará que el modelo va a estar separando mejor la curva del positivo y negativo. Es decir, que el clasificador estará separando bien las clases: que a los que tendrán *Y=1* los predecirá alto, y a los que *Y=0*, bajo.

2.6.9. Optimización de hiperparámetros

Con el objetivo de obtener un área bajo la curva ROC maximizado para cada uno de los modelos descriptos en la sección **2.6. Modelos**, se emplearon técnicas de optimización de forma tal de iterar sobre distintos valores de área bajo la curva ROC en función de los hiperparámetros de cada modelo.

Existen varios enfoques para buscar los hiperparámetros óptimos. Entre ellos, se encuentran las técnicas de *grid search* y *random search* [29].

El enfoque *grid search* consiste en definir una grilla de posibles hiperparámetros y utilizar un algoritmo para probar el rendimiento del modelo correspondiente entrenado con cada uno de los posibles hiperparámetros establecidos. Una vez entrenado y testeado, se eligen aquellos valores que maximicen la performance.

A diferencia de *grid search* donde los hiperparámetros están predefinidos, en *random search* los hiperparámetros son elegidos aleatoriamente *n*-veces. Luego de que el modelo es entrenado y testeado con los *n*-sets de hiperparámetros elegidos aleatoriamente, se eligen aquellos valores que maximicen la performance.

El enfoque que se utilizó para la búsqueda de hiperparámetros óptimos fue el de *grid search*, ya que se caracteriza por su simpleza a la hora de ejecutar computacionalmente el algoritmo.

A continuación, se describirán todos los hiperparámetros trabajados en cada modelo, y su significado.

2.6.9.1. Optimización para Word2vec

A continuación se describen los parámetros que empleó el modelo *Word2vec*, optimizados para mejorar la capacidad de predicción del algoritmo de *Machine Learning*.

- Cantidad de dimensiones que tendrá el vector de la palabra embebidas. Se tomó un valor de 30 dimensiones.
- Máxima distancia entre la palabra target y el resto de las palabras que la rodea. Se tomó un valor de 10 palabras.
- Mínima aparición de la palabra que se considerará a la hora de entrenar el modelo; si una palabra tiene una ocurrencia menor a este parámetro, el modelo la ignora. Se tomó un valor de 5.
- Número de particiones durante el entrenamiento. Se tomó un valor de 3.
- Algoritmo de entrenamiento: si el modelo emplea el algoritmo CBOW o *Skip-gram*. Se tomó al algoritmo *Skip-gram*.

2.6.9.2. Optimización para regresión logística y *Naïve Bayes*

Dado que se emplearon los modelos de regresión logística y *Naïve Bayes* como dos primeros modelos de predicción básicos, no se setearon hiperparámetros óptimos.

La razón de esta decisión se debe a que los próximos modelos (*Random Forest* y *XG Boost*) poseen técnicas más avanzadas que maximizan la performance.

2.6.9.3. Optimización para *Random Forest*

El hiperparámetro que se decidió optimizar en el modelo de *Random Forest* fue:

- **Cantidad de árboles.**

La decisión se debe a que es el parámetro que mayor consecuencia trae al área bajo la curva ROC, con bajo costo de cómputo.

Aumentar o disminuir la cantidad de árboles del modelo, implica que el modelo tendrá más o menos árboles que predicen la categoría de los datos de entrenamiento. El modelo de *Random Forest* tomará como categoría predicha a aquella que surja como voto de mayoría de todos los árboles de decisión.

Es decir, se fue iterando el valor del área bajo la curva ROC óptimo variando los valores de los árboles de decisión de 250 a 2000, con corte cada 250.

En la sección **5.6. Análisis de resultados** se detalló el hiperparámetro óptimo seleccionado.

2.6.9.4. Optimización para *XG Boost*

Los hiperparámetros que se decidieron optimizar en el modelo de *XG Boost* [30] fueron:

- ***Tasa de aprendizaje:*** permite controlar el ratio de que cada *boosting* aprende. Sus valores típicos varían de 0,01 a 0,001. Valores altos pueden causar mayor probabilidad de *overfitting*.
- ***Mínima reducción de pérdida:*** fija la reducción mínima de pérdidas para realizar una nueva partición de un nodo del árbol. Sus valores típicos varían de 0 a infinito. Valores bajos pueden causar mayor probabilidad de *overfitting*.
- **Cantidad de árboles:** permite controlar el número de árboles que entrenará el modelo. A diferencia de Random Forest, aumentar el número de árboles genera mayor probabilidad de *overfitting*.
- ***Muestreo de parámetros por árbol:*** permite controlar las variables a muestrear y considerar en cada árbol. Sus valores típicos varían de 0 a 1. Valores altos pueden causar mayor probabilidad de *overfitting*.

- ***Profundidad máxima:*** permite controlar la profundidad máxima de los árboles. Sus valores típicos varían de 0 a infinito. Valores altos pueden causar mayor probabilidad de *overfitting*.

En la sección **5.6. Análisis de resultados** se detallaron los hiperparámetros óptimos seleccionados.

Capítulo 3
Datos

3.1. Alcance del estudio

Como primer paso, se definió el alcance de los datos a recolectar. Para optimizar el entrenamiento del modelo de *Machine Learning*, se definieron criterios para filtrar la información disponibilizada por la empresa de *e-commerce*, a través de cuatro grandes filtros:

1. **Temporal:** para evitar cualquier efecto de estacionalidad del año, se tomó como límite temporal al año 2020 completo.
2. **Negocio:** la empresa presenta dos grandes negocios (*Fintech* y *e-commerce*). Para un mayor grado de interpretabilidad del modelo y de los datos, se decidió estudiar todos aquellos casos correspondientes a *e-commerce*.
3. **Lenguaje:** la empresa presenta dos grandes centros de atención (Brasil e Hispanos). Dado que el modelo de *Machine Learning* tendrá como input texto, y dicho texto tendrá asociación con sentimientos del usuario, se decidió limitar el estudio a todos los casos abiertos con lenguaje hispano, y así evitar que una misma palabra escrita en diferentes idiomas, tenga distintas interpretaciones para el modelo.
4. **Tipos contacto:** la empresa presenta dos grandes problemáticas en *customer service* (casos y reclamos). Se decidió excluir todos los reclamos abiertos, ya que el representante de *customer service* tiene poca incidencia, y depende más de las contrapartes en cuestión (comprador y vendedor).

La importante empresa de *e-commerce* proveyó el acceso a los datos bajo un acuerdo de confidencialidad, donde se tuvo la limitación de no mostrar información sensible.

3.2. Acceso a los datos

Una vez definido el alcance de los datos a estudiar, se procedió a acceder a los mismos. Para los datos buscados, la empresa presenta los mismos almacenados a través de bases de datos relacionales [31]. Estas bases siguen un modelo relacional el cual consta de:

- **Registros/Tuplas:** cada caso con *customer service* es almacenado como un registro diferente en una tabla estructurada.
- **Atributos:** valores que se almacenan para cada registro (ejemplo: rol, tipo de usuario, canal de contacto, texto escrito, etc.).
- **Relaciones:** conjunto de registros para los cuales se almacenan los mismos atributos. Ya que se quiere entrenar el modelo con información de NPS, se

procede a relacionar la tabla de casos abiertos en *customer service* con encuestas respondidas de NPS.

- **Clave primaria:** es el atributo que identifica unívocamente a cada registro. En el caso estudiado, fue el ID del caso abierto con customer service.
- **Clave foránea:** es la clave primaria de otro registro referenciado por el registro actual. Dado que en ambas tablas (casos abiertos, y respuesta de encuestas) se *linkean* según un único registro, la clave foránea fue el ID del caso.

Luego de haber identificado las tablas a las cuales se les iba a extraer la información, se definió el lenguaje de consulta de datos a utilizar. Para el mismo, se utilizó SQL (*Structure Query Language*) [32] para descargar la información de interés para tres tablas estructuradas (I. Casos abiertos, II. Respuestas de encuestas de NPS y III. Texto escrito en formulario) a través de la confección de una *query* (ver **Apéndice I**).

De esta manera, se pudo exportar el *dataset* de estudio, en formato .csv. El mismo se encuentra al final del documento con una muestra de los datos (ver **Apéndice II**).

3.3. Estructura de los datos

El *dataset* descargado a través de SQL con el cual se procedió a entrenar el modelo de *Machine Learning*, presentó 15 variables y 100.000 registros.

Las variables fueron seleccionadas de forma tal que cuenten con información del usuario al momento de contacto con *customer service*. Es decir, sus principales características como cliente, independientemente del posterior trato del caso por parte del representante.

Al final del documento se detallan cada una de ellas (ver **Apéndice III**).

En el **Capítulo 4**, se procedió a hacer un estudio resumido de cada una de ellas.

Capítulo 4
Análisis descriptivo

4.1. Descripción del análisis

En esta sección, el objetivo del análisis descriptivo fue:

- Recolectar y ordenar la información por medio de gráficas y medios visuales.
- Extraer las características más representativas de una colección de datos.
- Describir tendencias.
- Encontrar *insights* que puedan servir a la hora del análisis predictivo y prescriptivo.

4.2. Vocabulario y cantidad de caracteres

El corpus de estudio del *dataset* a analizar poseyó un vocabulario de **67.184** palabras diferentes. Es decir, para la base de estudio, los usuarios emplearon un vocabulario de **67.184** palabras para abrir un caso con *customer service*.

A su vez, el corpus de estudio estuvo compuesto por **24.576.067** caracteres.

4.3. Distribución de valores y proporción de detractores por *feature*

En la siguiente sección, se hizo un estudio de cada *feature* en base a dos gráficos seleccionados:

- Histograma de frecuencias de cada *feature*, ordenada de mayor a menor.
- Proporción de detractores para los valores de cada *feature*.

4.3.1. Análisis según nota y tipo de NPS

En la siguiente figura, se visualiza cómo se distribuyeron las notas de respuesta de encuesta de NPS. Como *insight* del mismo, se pudo ver cómo las respuestas de los usuarios estuvieron concentradas en ambos extremos (0 y 10), con mayor densidad en el 10.

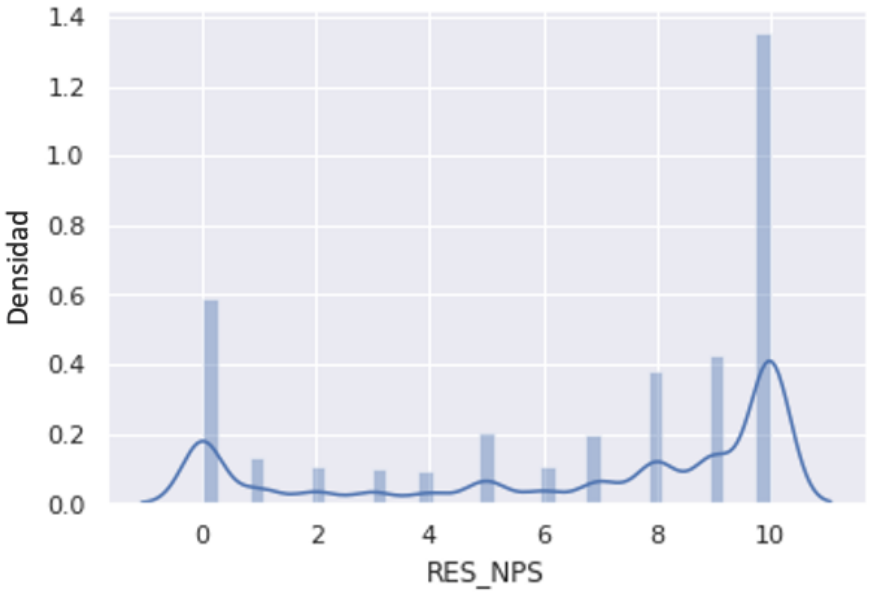


Figura 4.1: distribución de la respuesta de NPS

A su vez, si agrupamos las respuestas de NPS según si pertenecen a la categoría de detractor, neutro o promotor, y visualizamos su frecuencia relativa en el *dataset*, se obtuvo el siguiente Pareto de distribución:

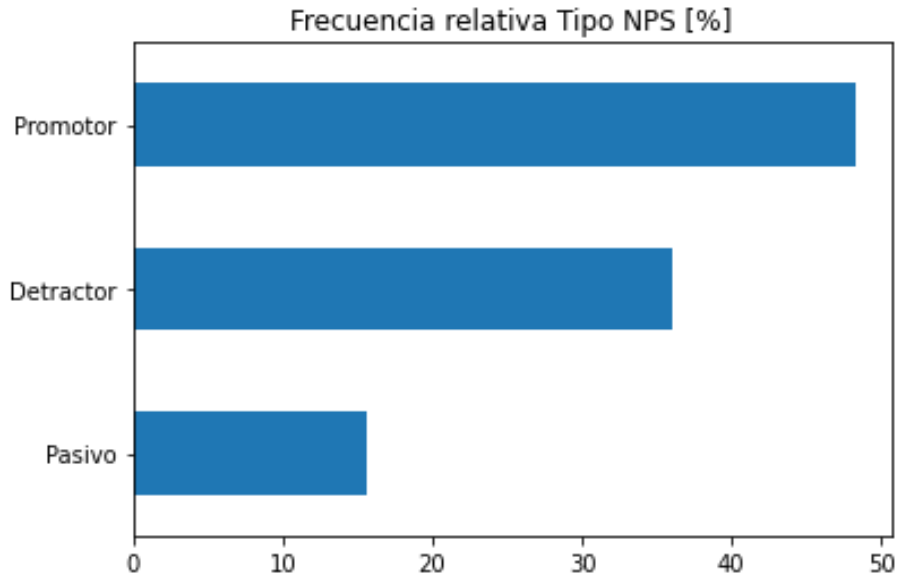


Figura 4.2: frecuencia relativa [%] de tipo de respuesta de NPS

Como el objetivo propuesto fue predecir a todos los usuarios con probabilidad de detracción, el *insight* de casi 40% de detractores permitió asegurarnos que nuestra base de análisis no tuvo baja población de una categoría en particular, por lo que no se precisó realizar muestras con sobrerepresentatividad.

De esta forma, se simplificó al modelo y se lo entrenó de forma tal que prediga variables de dos categorías: 'detractor' y 'no detractor'.

4.3.2. Análisis según mes de contacto

En la siguiente figura, se visualiza cómo se distribuyeron los casos con respuesta de NPS a lo largo de los meses del año 2020. Como *insight* del mismo, se obtuvo que los meses de mayor contacto fueron mayo, junio y julio con aproximadamente el 40% de los casos con respuesta de NPS del año, con un promedio mensual de 8.000 casos con respuesta de NPS. Por último, dado que no existieron meses con menos de 4.000 encuestas, se aseguró una buena representatividad de la métrica de NPS a lo largo del año.

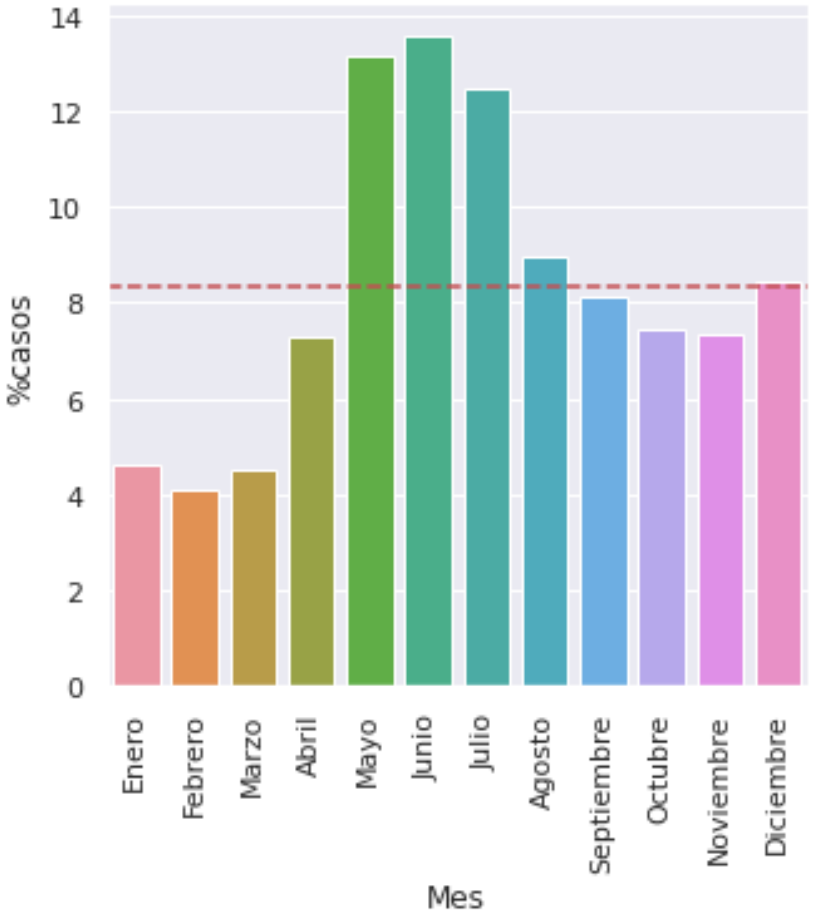


Figura 4.3: frecuencia relativa [%] de mes de contacto

A continuación se visualiza cómo se distribuyeron la proporción de detractores por mes a lo largo del año.

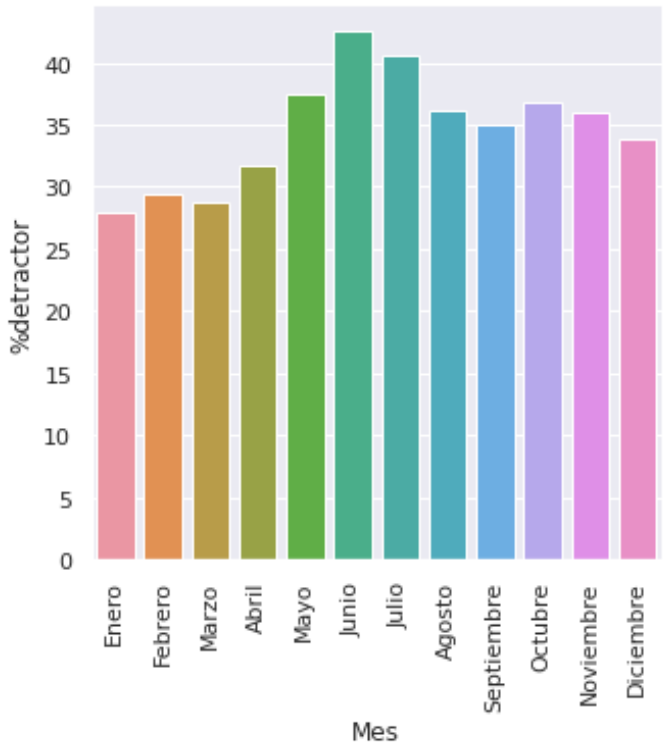


Figura 4.4: proporción de detractores por mes de contacto

Como *insight* del análisis por mes, se obtuvo que la proporción de detractores varió de acuerdo al mes. Como hipótesis a desarrollar, se incorporó la variable mes al *dataset*, para analizar si existieron estacionalidades en detractores a lo largo del año.

En cuanto a la explicación de negocio del aumento en la cantidad de contactos por el canal *offline* en los meses de mayo, junio y julio, y su correspondiente aumento en detractores, se consultó con el equipo de *customer service* para un mayor entendimiento. Este efecto se debió principalmente al aumento desmedido de la demanda post cuarentena obligatoria en todos los países. Este aumento fue ampliamente superior al proyectado por la empresa, por lo que la velocidad en incorporar nuevos representantes de *customer service* fue insuficiente para amortiguar el aumento en la demanda. Como consecuencia, todos los contactos que no se lograron atender en los canales *online* fueron derivador al canal *offline*. Este aumento de los casos entrantes en el canal *offline* trajo consecuencias negativas ya que el tiempo medio de respuesta aumentó, afectando la experiencia del usuario y provocando un aumento en detractores. A partir del mes de agosto se logró alcanzar un volumen suficiente de representantes de *customer service*, lo cual permitió atender más casos de los usuarios en los canales *online*, y reducir la cantidad de casos derivados a los canales *offline*.

4.3.3. Análisis según tipo de consulta

El *dataset* a analizar presentó consultas para las dos unidades de negocio de la empresa de *e-commerce*, donde (i) *Marketplace* corresponde a dudas de usuarios comprando o vendiendo en la plataforma y (ii) Logística corresponde a dudas de usuarios acerca del estado de su compra o venta.

En primer paso, se visualizó cómo se distribuyeron los casos con encuestas de NPS entre estas dos diferentes unidades de negocio. El resultado se visualiza en la siguiente figura.

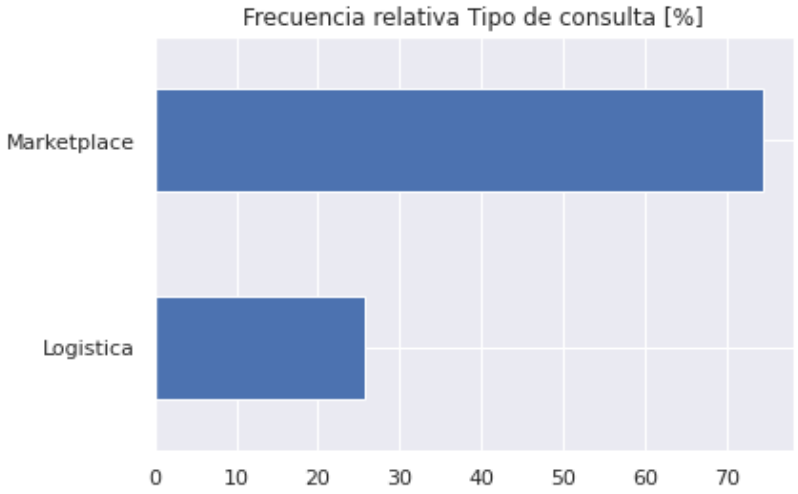


Figura 4.5: frecuencia relativa [%] de tipo de consulta

Tal como se puede ver, el 70% de las consultas correspondieron a la compra o venta en la plataforma, y el 30% restante al estado del envío.

Dado que la variable a predecir fue si es o no un futuro detractor, a continuación se visualiza la proporción de detractores por tipo de consulta.

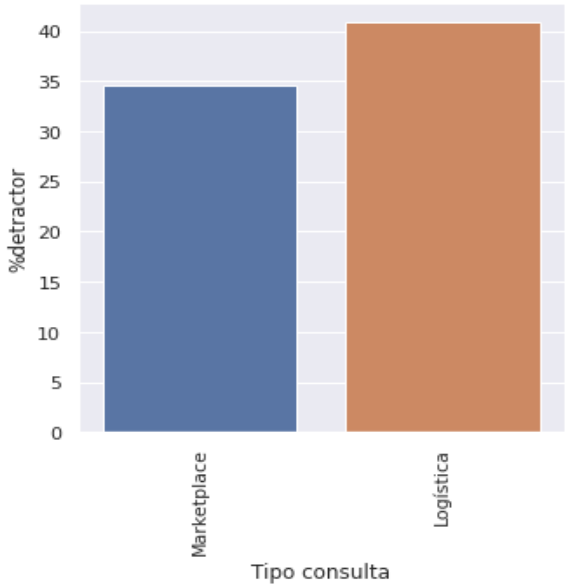


Figura 4.6: proporción de detractores por tipo de consulta

Como *insight* del análisis por tipo de consulta, se obtuvo que en los tipos de consulta de Logística, existieron mayores detractores que en *Marketplace*. Se consideró esta variable para el análisis, ya que el tipo de consulta puede dar una buena capacidad de predicción al modelo.

4.3.4. Análisis según país de contacto

En la siguiente figura, se visualiza cómo se distribuyeron los contactos con respuesta de NPS entre los distintos países de la empresa de *e-commerce*, ordenados de mayor a menor del total de casos.

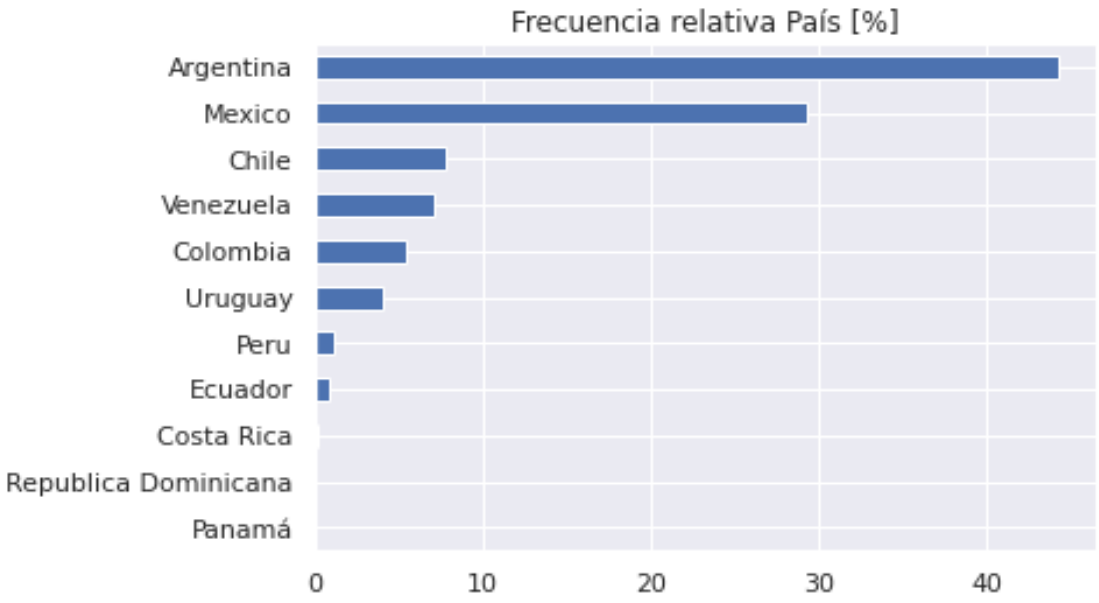


Figura 4.7: frecuencia relativa [%] de país de contacto

Como primer *insight*, se pudo ver que el 80% de los casos tienen origen de contacto en Argentina y México. El 20% restante, estuvo distribuido en 9 países.

Aclarado el peso que presenta cada país, se pasó a visualizar la proporción de detractores que presentan cada uno de los países.

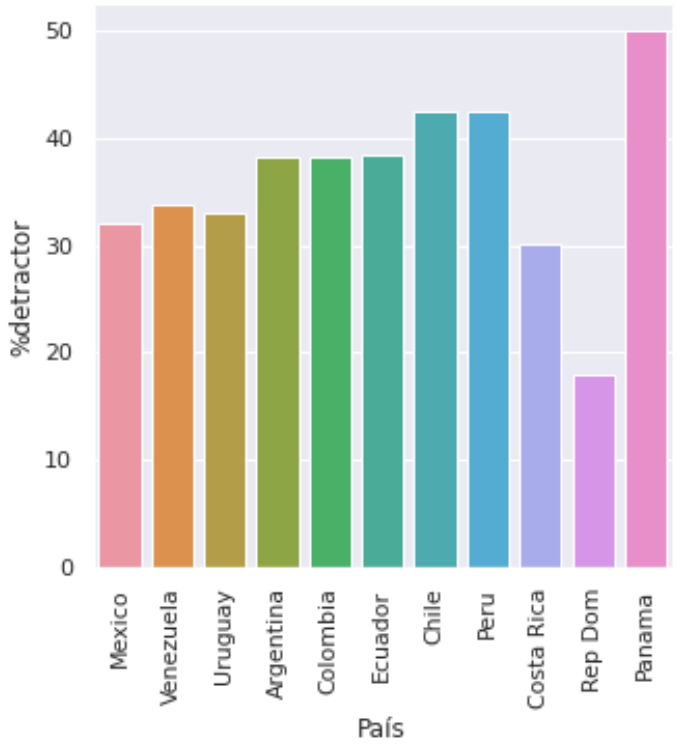


Figura 4.8: proporción de detractores por tipo de país de contacto

Los países de Argentina y México, que representan el 80% de los casos, presentaron distintas proporciones de detractores. Esto, a priori, significó que hubo variabilidad de la variable a predecir según el país, con lo cual podría favorecer a la capacidad predictiva del modelo.

4.3.5. Análisis según rol de usuario

La empresa de *e-commerce* brinda el servicio de *customer service* para sus tipos de usuarios, que son (i) *Buyer*, con consultas acerca del estado de su compra o bien de dudas en la plataforma, y (ii) *Seller*, con consultas acerca del estado de su venta o bien de dudas en la plataforma.

A continuación, se visualiza cómo se distribuyeron las consultas creadas, en función de si fueron creados por el *Buyer* o el *Seller*.

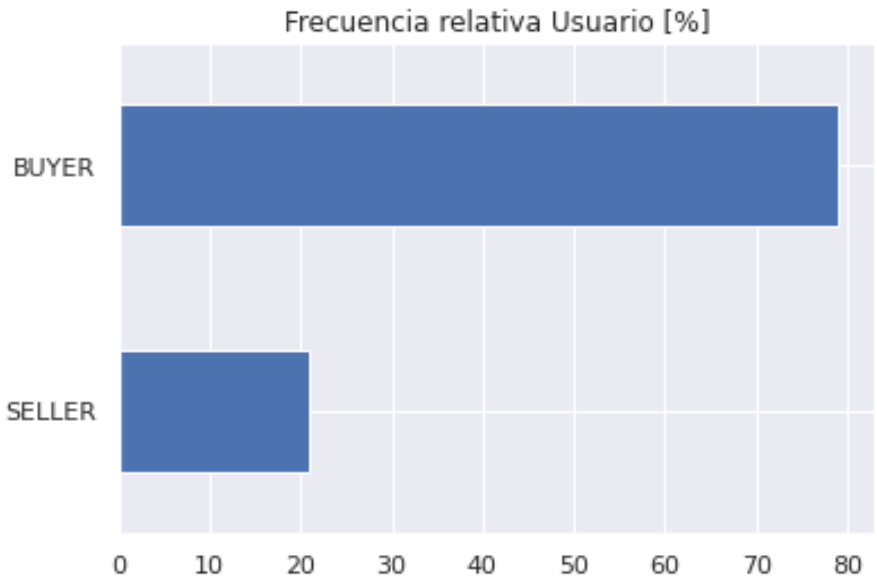


Figura 4.9: frecuencia relativa [%] del rol del usuario

Posteriormente, se visualiza cómo se distribuyeron su proporción en detractores para ambos roles de usuarios.

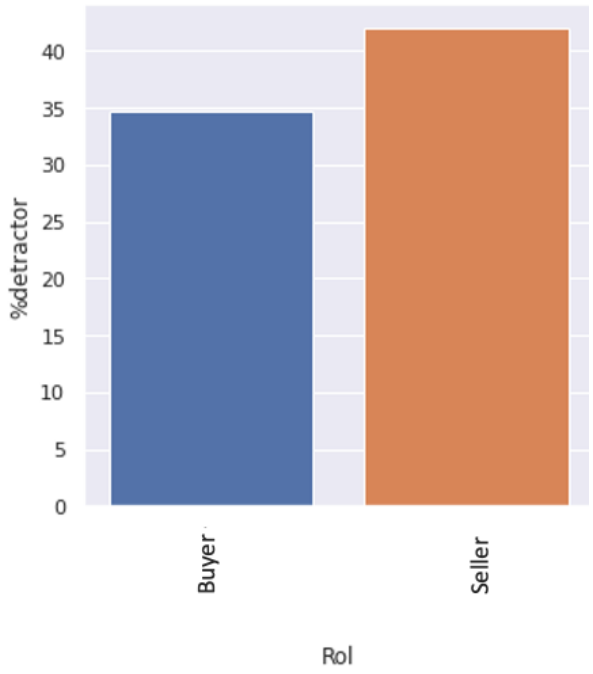


Figura 4.10: proporción de detractores por rol de usuario

Como *insight* del mismo, se obtuvo que hubo mayor probabilidad de que un vendedor se convierta en detractor. Asimismo, se la consideró dentro del *dataset* ya que es de importancia para el negocio y el modelo.

4.3.6. Análisis según nivel de usuario

La empresa de *e-commerce* brinda incentivos a sus dos tipos de usuarios, en función de la cantidad de compras o ventas que realizan. Los niveles varían en función del rol de la siguiente manera, ordenados de menor a mayor nivel:

- ***Buyer*:** Loyal 1, Loyal 2, Loyal 3, Loyal 4, Loyal 5 y Loyal 6
- ***Seller*:** Longtail, Midtail fijo, Midtail variable, ML_TOP_ON, ML_TOP_OFF, Best seller y Mercado líder.

En la siguiente figura, se visualiza cómo se distribuyeron las consultas creadas por nivel de usuario.

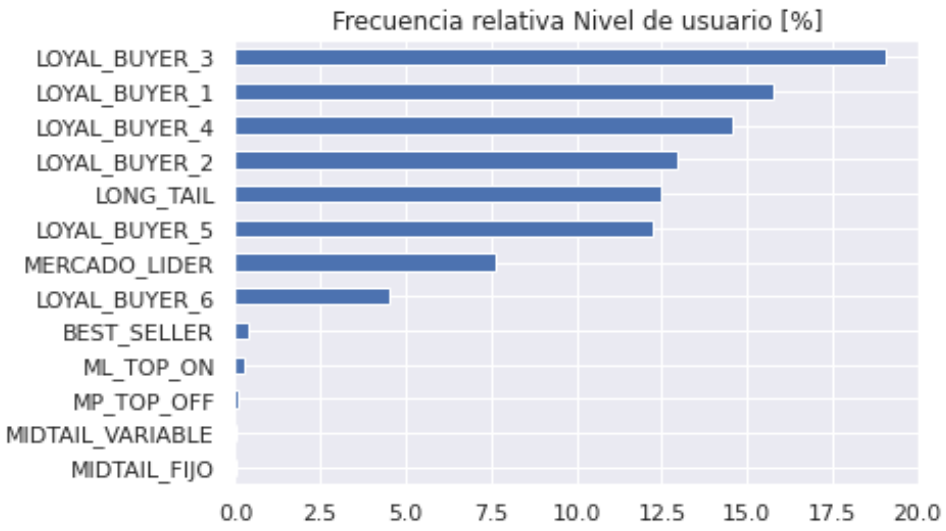


Figura 4.11: frecuencia relativa [%] del nivel de usuario

Como se puede ver, el *Buyer* se contactó en todos sus niveles de *loyalty*, mientras que los tipos de *Seller* que se contactaron fueron principalmente *Long Tail* y Mercado Líder.

A continuación, se visualiza cómo fue la proporción de detractores en cada uno de los niveles de usuarios, agrupados por similitud de proporciones.

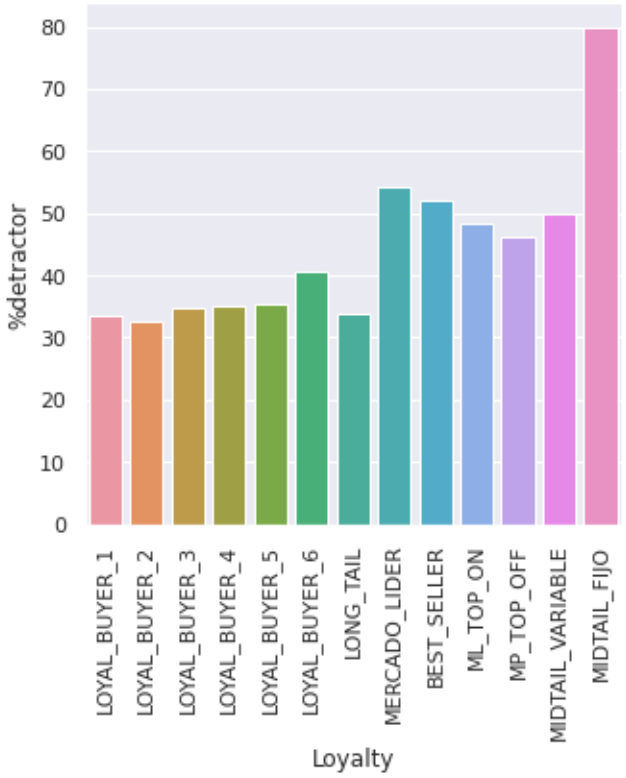


Figura 4.12: proporción de detractores por nivel de usuario

Como *insight* del mismo, se obtuvo que a mayor nivel de *loyalty* del *Buyer* o *Seller*, la proporción de detractores se incrementó. Es decir que, a priori, la exigencia que tuvieron con *customer service* fue mayor, y la consulta más específica. Con lo cual, se consideró a la variable dentro del *dataset*.

4.3.7. Análisis según proceso atendido

La empresa de *e-commerce* tiene caminos de contacto donde el usuario debe responder una serie de preguntas previas *multiple-choice*, hasta que logra escribir su caso con *customer service*. Esta serie de preguntas permite conocer la naturaleza del caso previo a que sea abierto. Estas distintas naturalezas se las conoce como procesos, y son una variable que permite conocer la temática del caso al representante de antemano. Como se puede visualizar en la siguiente figura, existieron 8 procesos.

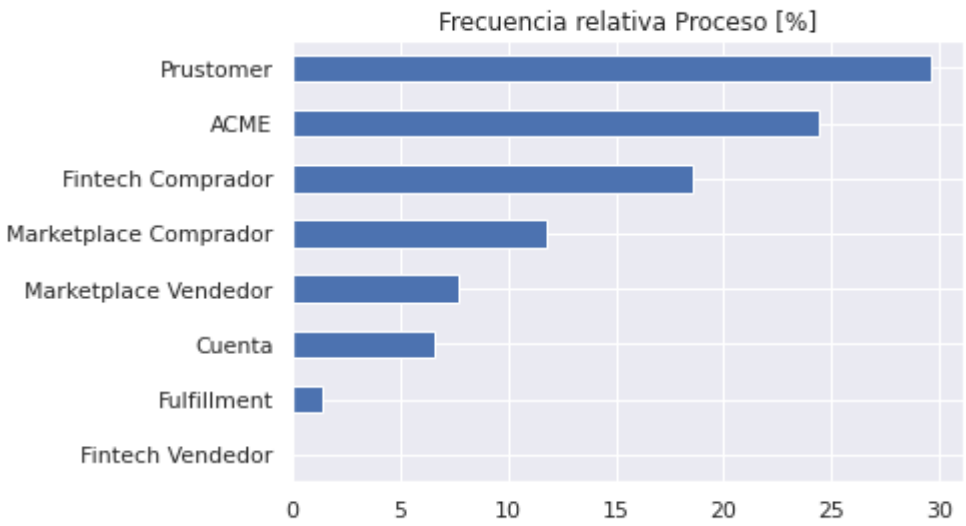


Figura 4.13: frecuencia relativa [%] del proceso

A continuación, se visualiza cómo se distribuyó la proporción de detractores por proceso.

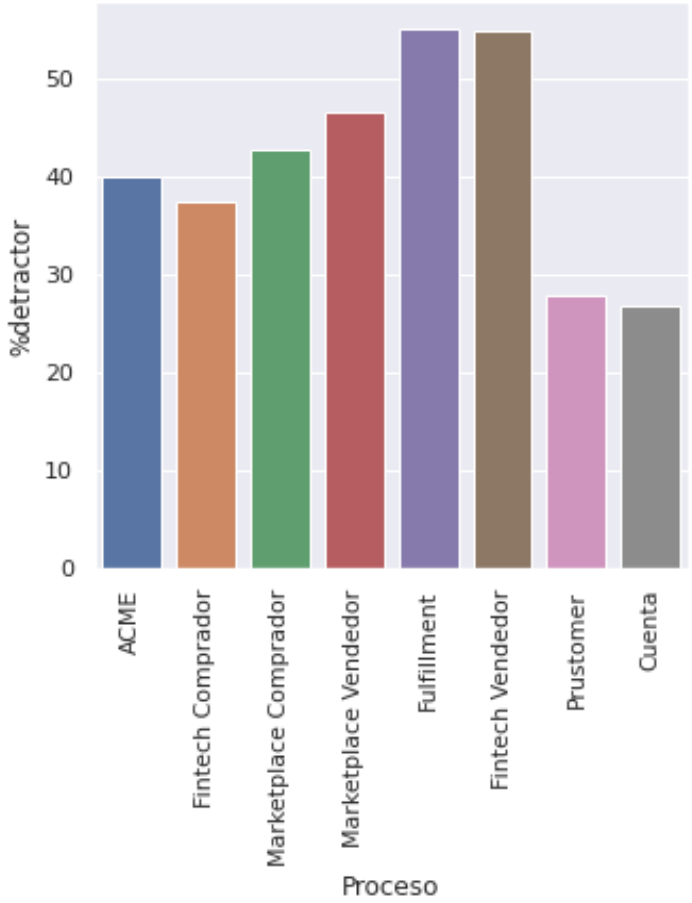


Figura 4.14: proporción de detractores por proceso

Como *insight* del mismo, se obtuvo que la proporción de detractores tuvo una varianza entre los distintos procesos. Con lo cual, se consideró este *feature* para el análisis.

4.3.8. Análisis según canal del equipo de atención

En conjunto con la tipificación del caso a un proceso, se le asigna un determinado canal de atención, donde cada canal presenta varios representantes. Como se puede visualizar en la siguiente figura, existieron 7 equipos de atención, donde el 80% estuvo concentrado en Multicanal Chat y Offline, mientras que el 20% restante en los otros 5 equipos.

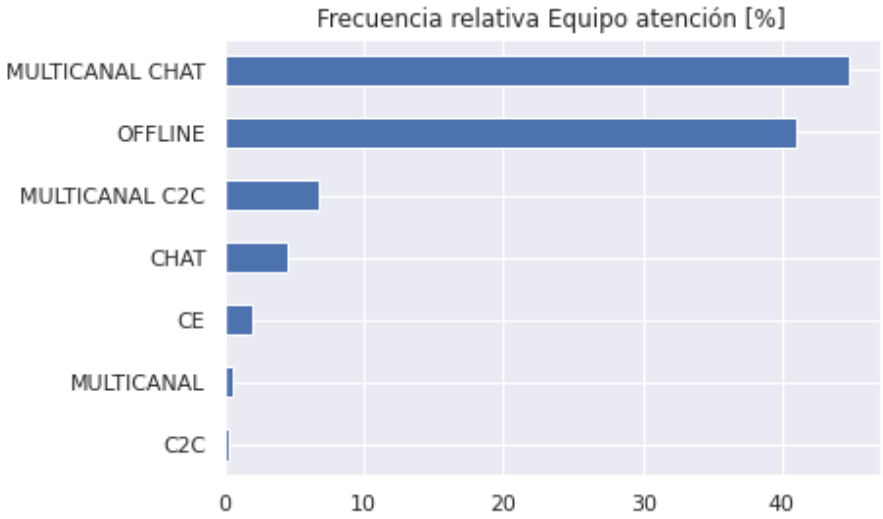


Figura 4.15: frecuencia relativa [%] del equipo de atención

A continuación, se visualiza cómo varió la proporción de detractores por equipo de atención.

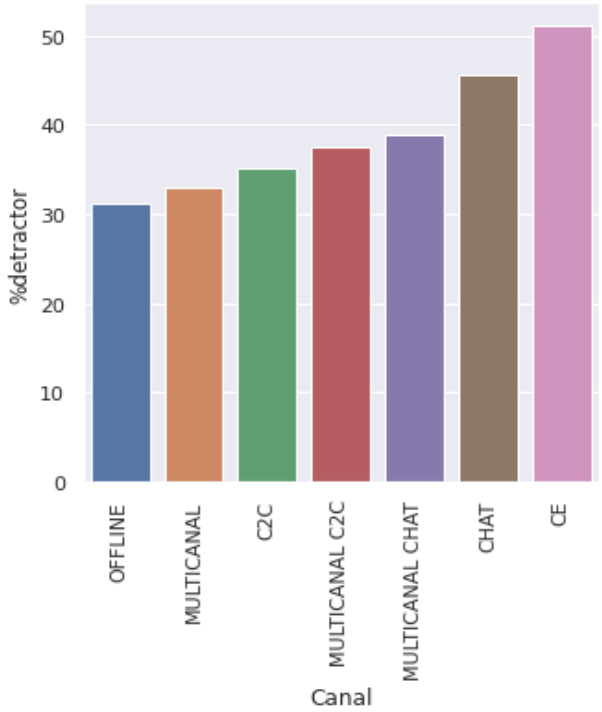


Figura 4.16: proporción de detractores por equipo de atención

Se consideró esta variable dentro del *dataset* dada la variabilidad en detractores.

4.3.9. Análisis según formulario de ingreso de contacto

Luego de responder una serie de preguntas, el usuario ingresa mediante un formulario específico, donde puede comenzar a escribir el texto contando su consulta. El *dataset* analizado, contó con 187 formularios de contacto.

Se visualiza la proporción de contactos entre estos 187 formularios, ordenados de menor a mayor.

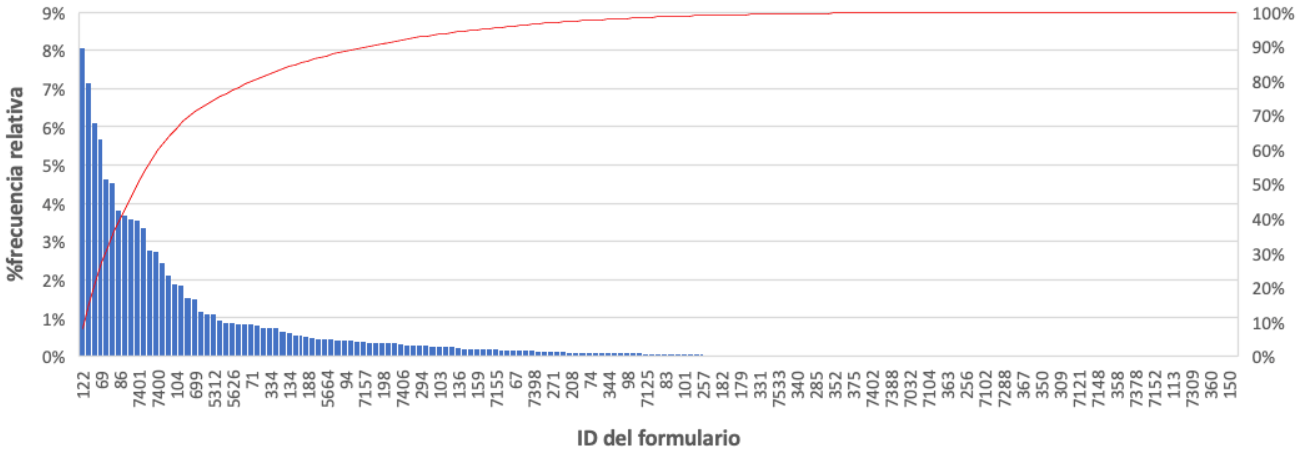


Figura 4.17: frecuencia relativa [%] del formulario de contacto

Como se puede ver en la figura, existió una distribución entre los distintos formularios, donde casi la mitad de los mismos, poseyó muy pocos contactos. Dado que esta variable tiene muchas categorías, y los contactos están distribuidos en parte de los mismos, se decidió quitar este *feature* del modelo y así evitar dar una mayor complejidad al modelo.

4.4. Distribución de cantidad de palabras y caracteres

Una vez tokenizado el texto, aplicado el *stemming* y eliminado todos sus stop *words*, se procedió a visualizar la cantidad de palabras escritas por usuario a la hora de contactarse vía formulario con *customer service* en conjunto con la cantidad de caracteres que utilizó.

4.4.1. Distribución en el total de usuarios

En la siguiente figura, se visualiza cómo se distribuyeron la cantidad de palabras escritas por un usuario cuando se contactó con *customer service*. Como *insight*, se pudo ver que la mayoría de los casos tuvieron entre 10 y 25 palabras.

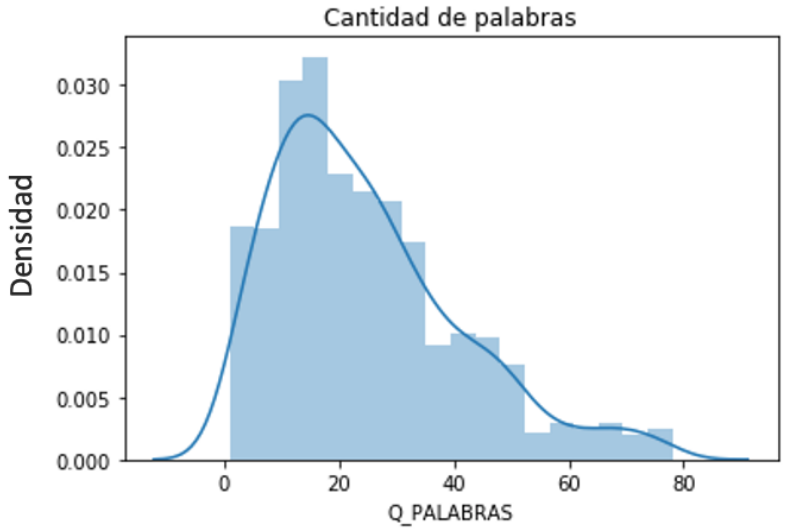


Figura 4.18: distribución de cantidad de palabras por caso

Pasando al estudio de la cantidad de caracteres, se realizó el mismo gráfico para estudiar su distribución. Como se puede visualizar, la mayoría de los usuarios escribieron entre 50 y 300 caracteres.

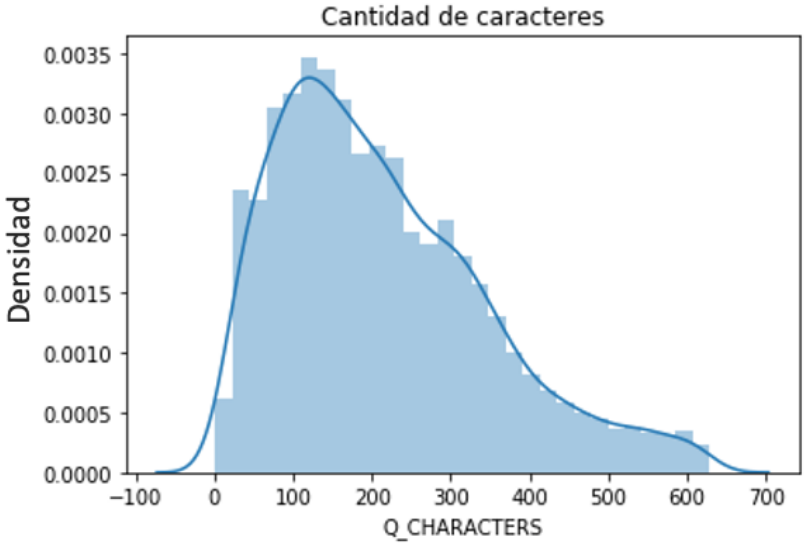


Figura 4.19: distribución de cantidad de caracteres por caso

4.4.2. Distribución según la categoría de NPS

En la siguiente figura, se visualiza cómo se distribuyeron la cantidad de palabras utilizadas por caso, según categoría de NPS. Como *insight* del mismo, se obtuvo que los usuarios promotores tuvieron en promedio menor cantidad de palabras empleadas, pero sin diferencias estadísticamente significativas, asegurando un intervalo de confianza del 95%.

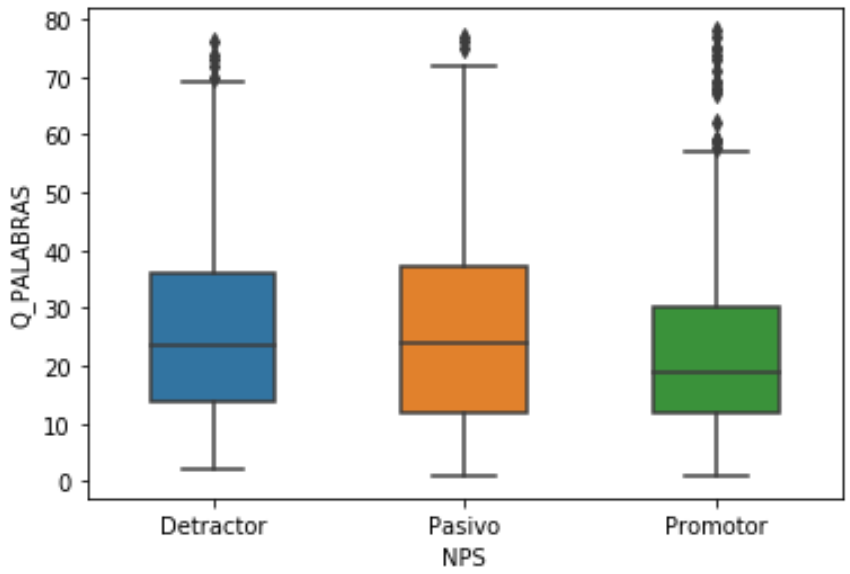


Figura 4.20: distribución de cantidad de palabras por caso, con apertura categoría de NPS

Por lo tanto, se tomó la agrupación de NPS en conjunto con la cantidad de palabras, para incorporarlo al *dataset*.

Con respecto a la cantidad de caracteres, se pudo observar que también existió diferencia entre la cantidad de caracteres en promedio, según si fue detractor o no.

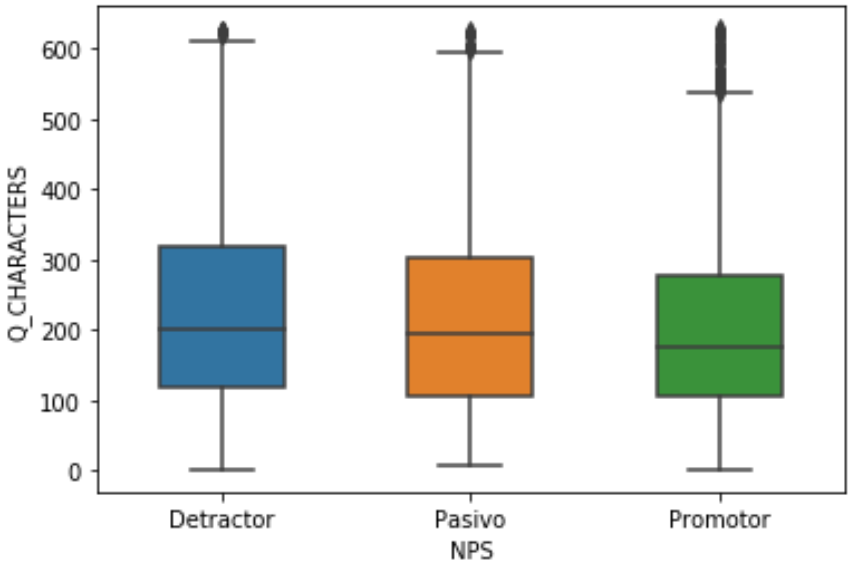


Figura 4.21: distribución de cantidad de caracteres por caso, con apertura categoría de NPS

Al igual que la variable cantidad de palabras, se incorporó esta variable al *dataset*, para evaluar si mejoró la performance del modelo.

4.5. Distribución principales palabras usadas y sus sinónimos

4.5.1. Distribución en usuarios promotores

En la siguiente figura, se visualiza cuáles fueron las principales palabras más usadas de los promotores. Como *insight* del mismo, se obtuvo que las palabras usadas con mayor frecuencia fueron: salir, emergencia, cuenta, lejos, estaré, entre otras.

Figura 4.22: principales palabras utilizadas por promotores

A su vez, la representación gráfica permitió visualizar diferencias en frecuencias de palabras en función del tamaño. Por lo tanto, se vio que los promotores utilizaron con frecuencia ciertas palabras, y otras no.

Tomando las palabras ‘emergencia y ‘salir’, se muestra a continuación las palabras similares que arrojó el modelo *word2vec*:

EMERGENCIA	SALIR
[('covid', 0.8342214226722717), ('sanitaria', 0.8207283020019531), ('familiar', 0.7943732738494873), ('coronavirus', 0.7789534330368042), ('pandemia', 0.7759039402008057), ('laborales', 0.7725362181663513), ('contingencia', 0.7692261934280396), ('contigencia', 0.7661978006362915), ('fallecio', 0.7634221315383911), ('enfermo', 0.7615442872047424)]	[('ir', 0.8777965307235718), ('viajar', 0.8755033016204834), ('ausentarme', 0.8698362708091736), ('trasladarme', 0.8643907308578491), ('contigencia', 0.8500609397888184), ('arriesgarme', 0.8471658825874329), ('concurrir', 0.8459184169769287), ('devido', 0.842607319355011), ('covidno', 0.8414656519889832), ('lejos', 0.8413859605789185)]

Figura 4.23: palabras embebidas para principales palabras promotor

Tal como se describió en la sección **2.2.8. Enfoque *Word2vec***, se pudo observar para la palabra ‘emergencia’ que tuvo un valor cercano a 1 con la palabra ‘covid’. Es decir, el modelo computó con buena similitud dichas palabras, y las tomó como similares. Lo mismo se vio con ‘salir’, con una similitud coseno de 0.88 con la palabra ‘ir’.

4.5.2. Distribución en usuarios detractores

En la siguiente figura, se visualiza cuáles fueron las principales palabras más usadas de los detractores. Como *insight* del mismo, se obtuvo que las palabras usadas con mayor frecuencia fueron: infracción, día, ayuda, nadie, reclamo, necesito, entre otras.

Figura 4.24: principales palabras utilizadas por detractores

Dada la diferencia de tamaño en la representación gráfica, se pudo observar que hubieron ciertas palabras que se repitieron con frecuencia en los detractores.

Tomando las palabras 'infracción' y 'ayuda', se muestra a continuación las palabras similares que arrojó el modelo *word2vec*:

INFRACCION	AYUDA
[('entiendo', 0.9033202528953552), ('infraccionme', 0.8958151340484619), ('massaludos', 0.8919004797935486), ('cometida', 0.8845995664596558), ('infraccion gracias', 0.8830133676528931), ('cometo', 0.8775806427001953), ('exactamente', 0.85960453748703), ('infreccion', 0.8592917919158936), ('masque', 0.8590579032897949), ('infracion', 0.857257604598999)]	[('ayuden', 0.9037750363349915), ('resolver', 0.8368480801582336), ('apoyo', 0.8223572969436646), ('ayude', 0.8135861158370972), ('asistencia', 0.8094859719276428), ('liberando', 0.8046591877937317), ('resolverlo', 0.8041691184043884), ('colaboracion', 0.7942233085632324), ('inconveniente', 0.7867115139961243), ('porfavor', 0.7804961800575256)]

Figura 4.25: palabras embebidas para principales palabras detractor

Como se puede ver, el modelo permitió captar similitudes de las palabras ('ayuda' con 'asistencia') y también similitudes sintácticas ('infracción' con 'infración').

4.5.3. Distribución en usuarios pasivos

En la siguiente figura, se visualiza cuáles fueron las principales palabras más usadas de los pasivos. Como *insight* del mismo, se obtuvo que las palabras usadas con mayor frecuencia fueron: buenas, pagué, pasa, hice, cuenta, entre otras.

Figura 4.26: principales palabras utilizadas por pasivos

Al igual que con promotores y detractores, se observó que hubo una distribución mayor en ciertas palabras.

En cuanto a las principales palabras de los pasivos, se destacaron ‘pague’ y ‘buenas’. A continuación, se visualizan las principales palabras que presentaron mayor similitud con las mismas.

PAGUE	BUENAS
[('abone', 0.8870272040367126), ('paguepesos', 0.8619922995567322), ('deaprox', 0.8446907997131348), ('page', 0.8272998332977295), ('cobraron', 0.8261891007423401), ('pagando', 0.8242610096931458), ('pagoy', 0.8240630626678467), ('compre', 0.8166693449020386), ('elegipagos', 0.8165570497512817), ('paguecuotas', 0.8160592317581177)]	[('tardes', 0.9882378578186035), ('noches', 0.9844357967376709), ('tardesles', 0.8934407234191895), ('estimados', 0.8673906922340393), ('escribo', 0.8393687009811401), ('comentarles', 0.8383338451385498), ('disculpe', 0.8307315707206726), ('consultarles', 0.8159298896789551), ('buen', 0.8129243850708008), ('buenos', 0.8069524168968201)]

Figura 4.27: palabras embebidas para principales palabras pasivo

Como se puede ver, la palabra ‘pague’ presentó mayor similitud con ‘abone’, mientras que ‘buenas’ se asoció con ‘tardes’/’noche’, lo cual hizo sentido ya que suele ir acompañada de dichas palabras.

4.6. Valuación de importancia de variables

Para las variables descriptas en la sección **3.3. Estructura de los datos**, se evaluó el grado de importancia que presenta cada una de ellas frente a la variable a predecir “Detractor” / “No detractor”. A continuación, se muestran los resultados obtenidos para las primeras 20 variables más significativas del *dataset*, con el modelo *Word2vec* aplicado junto al algoritmo *XG Boost* óptimo.

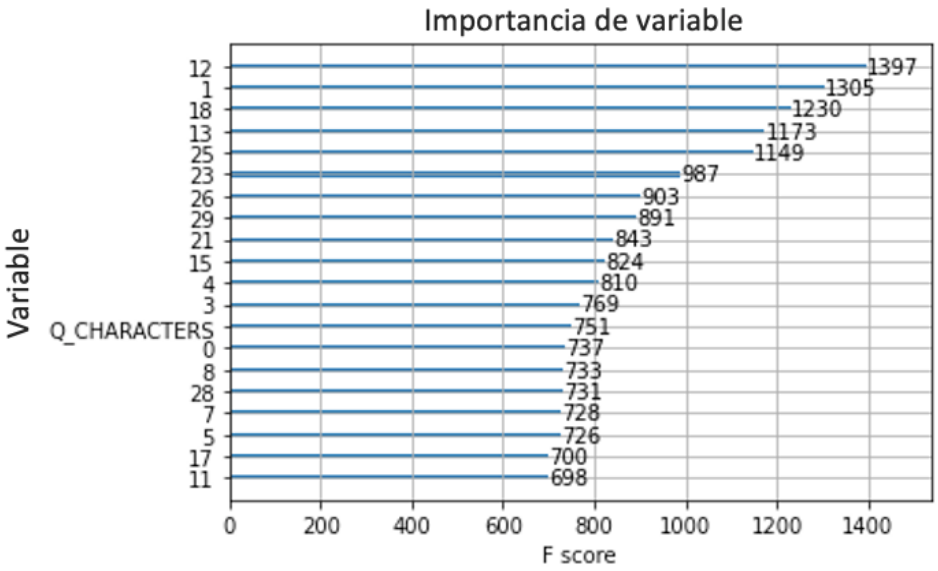


Figura 4.28: importancia de variables del dataset respecto a la variable a predecir

Como se describió en la sección **2.2.8. Enfoque *Word2vec***, se generó a través del enfoque *Word2vec* un vector de 30 dimensiones que representó al texto escrito por el usuario. Dichas dimensiones fueron tomadas dentro del *dataset* como los *features* denominados ‘0’ a ‘29’.

Como *insight*, se obtuvo que las principales variables de mayor capacidad de predicción fueron las dimensiones creadas por el texto mediante el modelo *Word2vec*. En puesto número 13 apareció la variable *Q_CHARACTERS* generada, con lo cual permitió obtener como conclusión que la cantidad de caracteres que utilizó el usuario pudo ayudar a predecir si fue un posible detractor.

Capítulo 5
Resultados

5.1. Descripción del análisis predictivo

Una vez realizado el análisis descriptivo e identificados los mejores *insights* y ajustes a realizar con el *dataset*, se procedió a realizar el análisis predictivo. Dicho análisis incluye anticiparse a futuros eventos y tendencias, regresiones multivariadas y minería de datos.

Para esta sección, el objetivo del análisis fue:

- Definir la variable del *dataset* a predecir.
- Mediante enfoque de *Machine Learning*, entrenar diferentes modelos que predigan dicha variable, logrando los mayores niveles de certeza posibles.
- Medir resultados con máximo área bajo la curva ROC para cada uno de los algoritmos.
- Asegurarse de emplear las técnicas correctamente, sin riesgo a cometer *data leakage.*

5.1.1. Identificación de la variable a predecir

Tal como se describió en la sección **1.5. Objetivo**, la variable que fue predicha por el modelo fue la variable **NPS** descripta en la sección **3.3. Estructura de los datos**.

Dicha variable es una variable categórica, cuyos valores posibles son los de 'Promotor', 'Neutro' y 'Detractor'. A efectos del análisis, y dado que se busca dar un enfoque de "contención" a los posibles usuarios detractores que se contacten con *customer service*, se decidió tratar a la variable categórica como una variable binaria, con 2 posibles valores: 'Detractor' y 'No detractor'.

Esta decisión permitió que los modelos tomen especial foco a los posibles detractores, y así mejorar la capacidad predictiva de los mismos.

5.2. Modelo de regresión logística LOGIT

Se tomó como primer modelo de *Machine Learning* a entrenar al modelo de regresión logística LOGIT, descripto en la sección **2.6.3. Modelo de regresión logística LOGIT**.

Una vez entrenado el modelo con la muestra de datos de entrenamiento, se procedió a evaluar su performance en la muestra de validación. Los resultados de este primer

modelo con mejor performance fueron los presentados en las **Figuras 5.1**, **5.2** y **5.3**, y la técnica que fue utilizada para tratar el texto fue la de *Bag-of-Words*.

	precision	*recall*	f1-*score*	*support*
Detractor	0.63	0.28	0.39	10815
No_Detractor	0.69	0.91	0.78	19185
accuracy			0.68	30000
promedio macro	0.66	0.59	0.59	30000
promedio ponderado	0.67	0.68	0.64	30000

Figura 5.1: precision, recall y f1-score obtenido en LOGIT

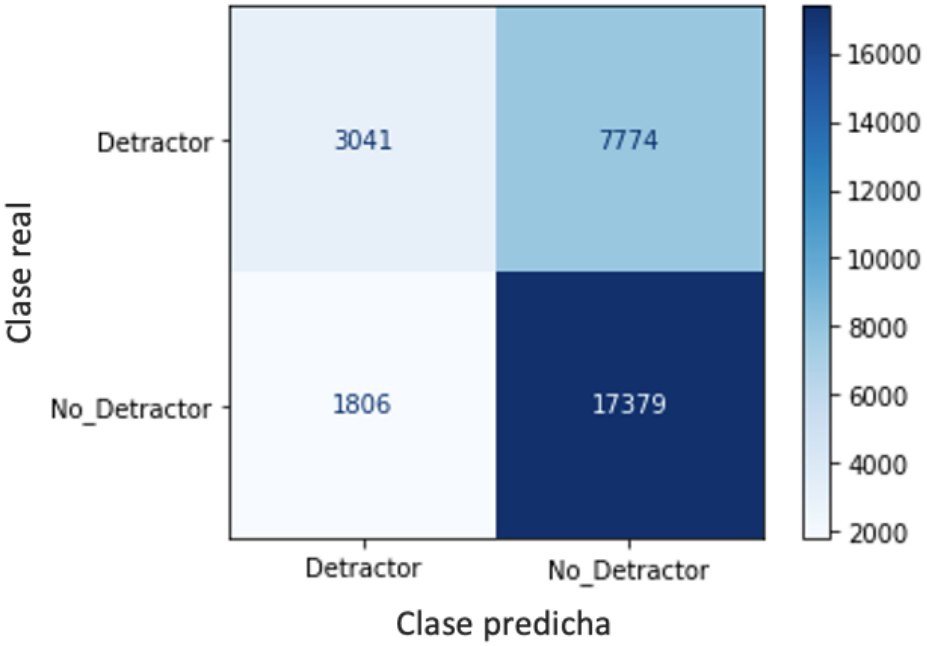


Figura 5.2: matriz de confusión obtenida en LOGIT

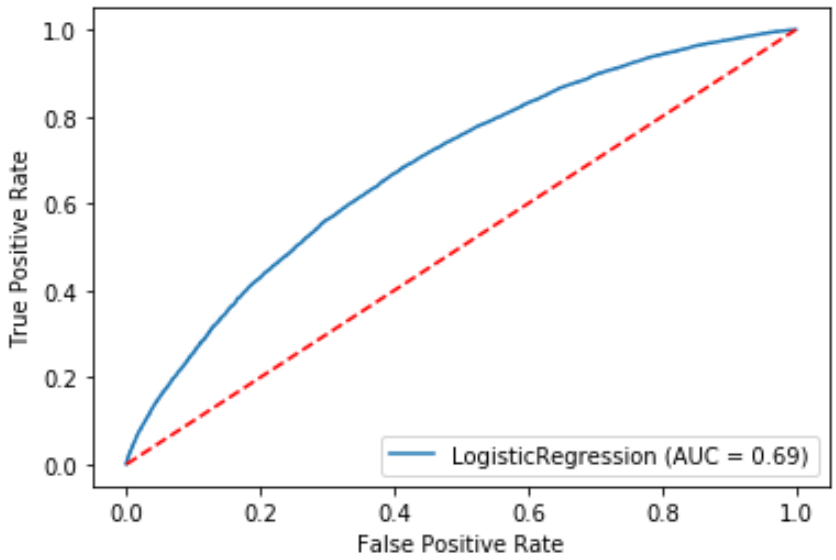


Figura 5.3: área bajo la curva ROC obtenida en LOGIT

Como se puede ver en los resultados, este primer modelo obtuvo un área bajo la curva ROC del 69%.

5.3. Modelo de *Naïve Bayes*

Se tomó como segundo modelo de *Machine Learning* a entrenar al modelo de *Naïve Bayes,* descripto en la sección **2.6.4. Modelo de *Naïve Bayes***.

Una vez entrenado el modelo con la muestra de datos de entrenamiento, se procedió a evaluar su performance en la muestra de validación. Los resultados de este segundo modelo con mejor performance fueron los presentados en las **Figuras 5.4**, **5.5** y **5.6**, y la técnica que fue utilizada para tratar el texto fue la de *Word2vec*.

	precision	*recall*	f1-*score*	*support*
Detractor	0.47	0.52	0.49	10797
No_Detractor	0.71	0.67	0.69	19160
accuracy			0.62	29957
promedio macro	0.59	0.59	0.59	29957
promedio ponderado	0.62	0.62	0.62	29957

Figura 5.4: precision, recall y f1-score obtenido en Naïve Bayes

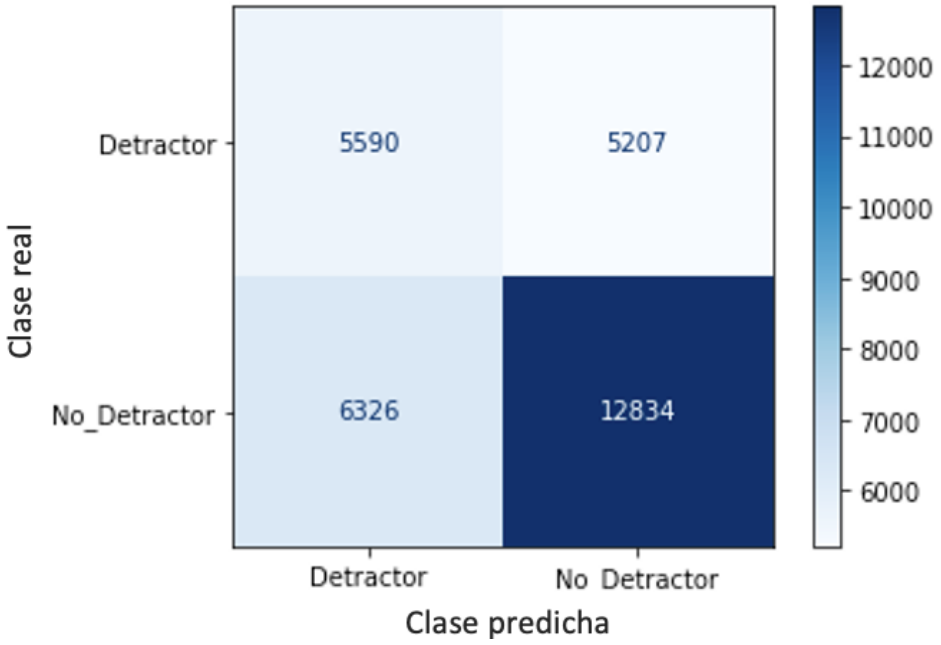


Figura 5.5: matriz de confusión obtenida en Naïve Bayes

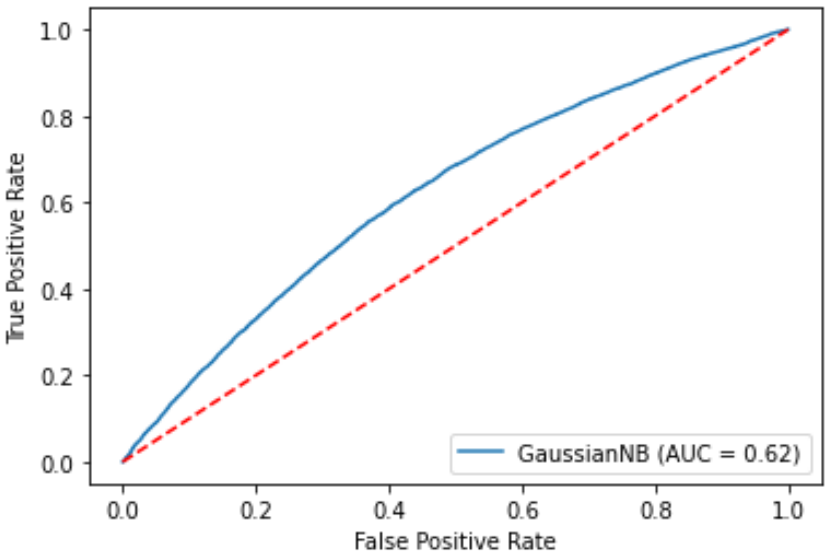


Figura 5.6: área bajo la curva ROC obtenida en *Naïve Bayes*

Como se puede ver en los resultados, este segundo modelo obtuvo un área bajo la curva ROC del 62%, siendo 7pp inferior al modelo de regresión logística LOGIT.

5.4. Modelo de *Random Forest*

Como tercer modelo alternativo, se tomó el modelo de *Random Forest* descripto en la sección **2.6.5. Modelo de Random Forest**.

Una vez entrenado el modelo con la muestra de datos de entrenamiento, se procedió a evaluar su performance en la muestra de validación. Los resultados de este modelo con mejor performance fueron los presentados en las **Figuras 5.7**, **5.8** y **5.9**, y la técnica que fue utilizada para tratar el texto fue la de *Word2vec*.

Dado que se trata de un modelo de mayor grado de precisión, se procedió a entrenar el mismo con los hiperparámetros óptimos, probando las distintas combinaciones de hiperparámetros descriptos en la sección **2.6.9.3. Optimización para Random Forest** de forma tal de maximizar el área bajo la curva ROC .

	precision	*recall*	f1-*score*	*support*
Detractor	0.61	0.27	0.38	10797
No_Detractor	0.69	0.90	0.78	19160
accuracy			0.68	29957
promedio macro	0.65	0.59	0.58	29957
promedio ponderado	0.66	0.68	0.64	29957

Figura 5.7: precision, recall y f1-score en Random Forest

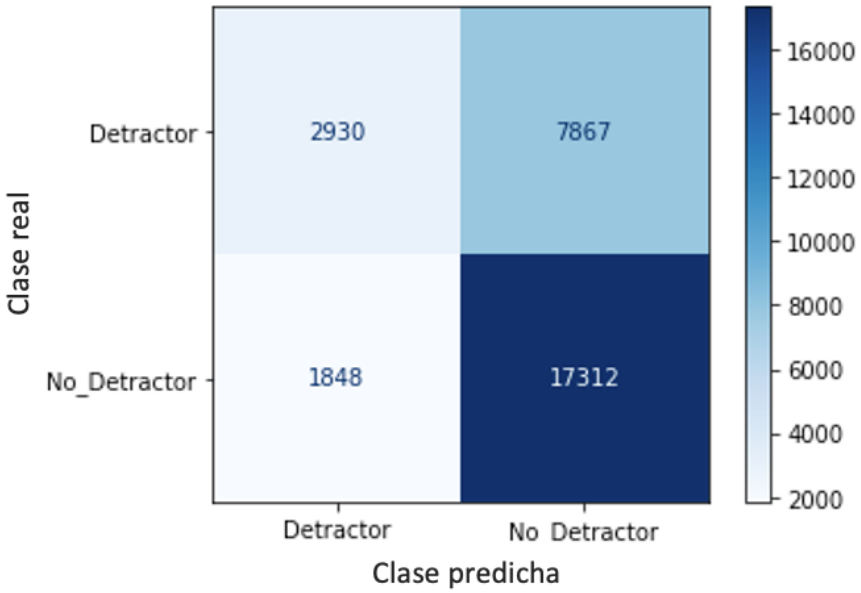


Figura 5.8: matriz de confusión obtenida en Random Forest

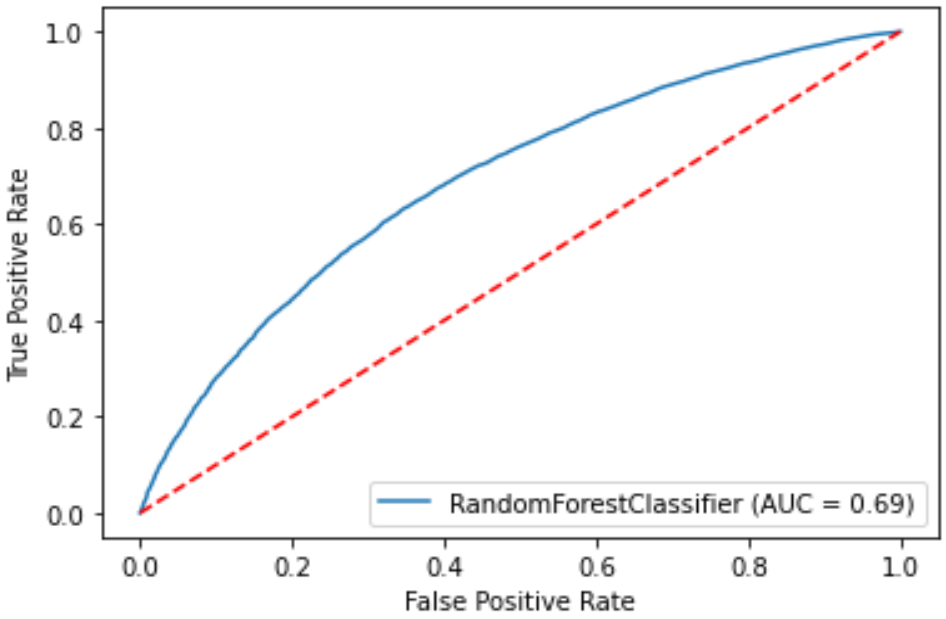


Figura 5.9: área bajo la curva ROC obtenida en Random Forest

Como se puede ver en los resultados, este segundo modelo obtuvo un área bajo la curva ROC de 69%. Es decir, sin grandes diferencias respecto al modelo de regresión logística LOGIT.

5.5. Modelo *XG Boost*

Como cuarto modelo de *Machine Learning* a entrenar, se tomó el modelo *XG Boost* descripto en la sección **2.6.6. Modelo Gradient Boosting *XG Boost***.

Una vez entrenado el modelo con la muestra de datos de entrenamiento, se procedió a evaluar su performance en la muestra de validación. Los resultados de este modelo con

mejor performance fueron los presentados en las **Figuras 5.10**, **5.11** y **5.12**, y la técnica que fue utilizada para tratar el texto fue la de *Word2vec*.

Dado que se trata de un modelo de mayor grado de precisión, se procedió a entrenar el mismo con los hiperparámetros óptimos, probando las distintas combinaciones de hiperparámetros descriptos en la sección **2.6.9.4. Optimización para *XG Boost*** de forma tal de maximizar el área bajo la curva ROC.

	precision	*recall*	f1-*score*	*support*
Detractor	0.62	0.30	0.40	10797
No_Detractor	0.69	0.90	0.78	19160
accuracy			0.68	29957
promedio macro	0.65	0.60	0.59	29957
promedio ponderado	0.67	0.68	0.64	29957

Figura 5.10: precison, recall y f1-score en XG Boost

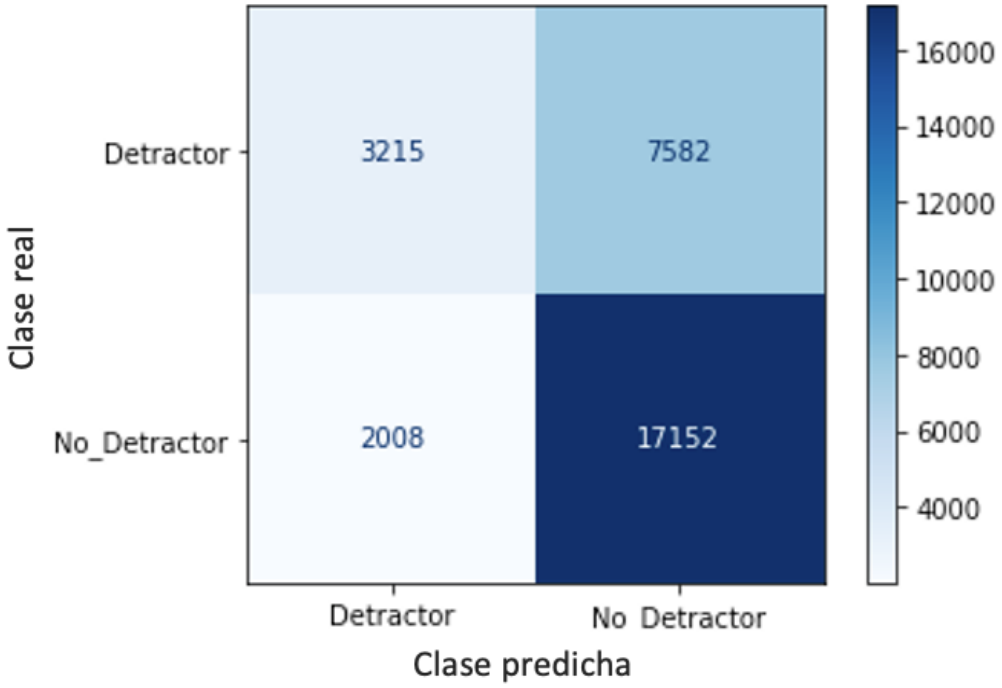


Figura 5.11: matriz de confusión obtenida en XG Boost

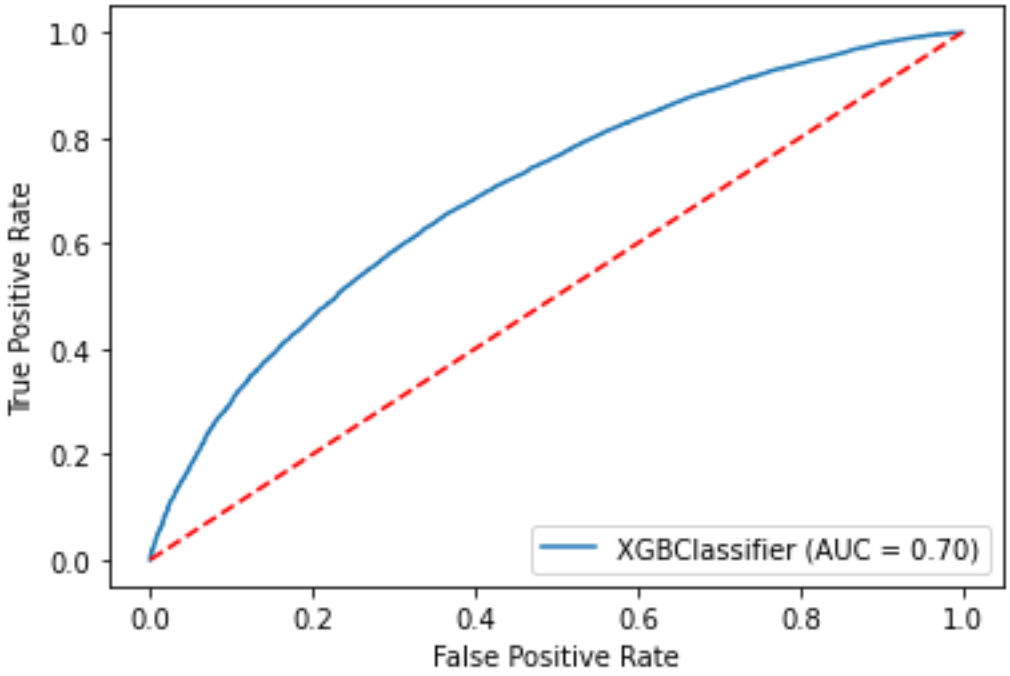


Figura 5.12: área bajo la curva ROC obtenida en XG Boost

Como se puede ver en los resultados, este tercer modelo obtuvo una precisión superior, con un área bajo la curva ROC de 70%, superando a la performance del modelo de *Random Forest* y LOGIT por 1pp.

Además, podemos observar que si comparamos los resultados del modelo de *XG Boost* con el de *Random Forest*, el modelo *XG Boost* tuvo mayor *precision* y *f1-score*.

5.6. Análisis de resultados

Se realizaron 26 corridas de modelos, probando todas las posibles combinaciones del uso o no del enfoque *Bag-of-Words* (con las variaciones de *Q_CORTE* y normalización *Tf-Idf*), el uso o no del enfoque *Word2vec* (importando un modelo pre-entrenado en español o utilizando un modelo entrenado con el *dataset*), los diferentes algoritmos (Regresión Logística, *Naïve Bayes*, *Random Forest* y *XG Boost*) y los hiperparámetros óptimos.

A su vez, se comenzó empleando el 20% del *dataset* (20.000 registros) y una vez identificados la grilla de parámetros que maximizan la performance, se empleó el 100% del *dataset* (100.000 registros).

Se muestra en la **Figura 5.10** los resultados de la mejor performance de cada uno de los modelos. Se detalla al final del documento el resultado de las 26 corridas realizadas (ver **Apéndice IV**).

Enfoque BOW	Q_CORTE (BOW)	Norm Tf-Idf (BOW)	Enfoque Word2vec	Modelo importado (Word2vec)	Modelo	Hiperparámetros	ROC
Sí	20	No	No	-	Regresión Logística	-	0,685
No	-	-	Sí	No	Naïve Bayes	-	0,623
No	-	-	Sí	No	Random Forest	1750 árboles	0,690
No	-	-	Sí	No	XG Boost	0,01 eta / 0,4 colsample / 1500 árboles / 6 max_depth / 10 gamma	**0,701**

Figura 5.10: resultados de área bajo la curva ROC para las mejores corridas

Como recomendación final, se decidió optar por un algoritmo *XG Boost*, que presenta un área bajo la curva ROC del 70%, y es entrenado con un enfoque *Word2vec*.

A su vez, se evaluó la performance del mismo en el *train-set*, y se obtuvo un área bajo la curva ROC de 0,75. Con lo cual, no se cuenta con un riesgo alto de *over fitting*.

Capítulo 6
Conclusiones

6.1. Descripción del análisis prescriptivo

El análisis prescriptivo incluye encontrar una o varias soluciones entre una gama de variantes con el objetivo de optimizar los recursos y aumentar la eficiencia operativa, luego de haber realizado previamente los análisis descriptivo y predictivo.

En esta sección, el objetivo del análisis fue:

- Hallar optimizaciones bajo restricciones de negocios.
- Señalar el camino que realmente conviene seguir.
- Cuantificar el impacto económico de la propuesta.

6.2. Recomendaciones de negocio

Tras el análisis desarrollado en el **Capítulo 5,** se encontró que, mediante el uso de técnicas de *Machine Learning*, es posible medir las variables que arroja un usuario antes de ser atendido por un representante de customer service (proceso, país, nivel, texto, etc.) y predecir el grado de satisfacción (si será o no un detractor) con una capacidad de predicción del 70%.

Esto sugiere que hay una oportunidad a la hora de predecir si un usuario tendrá una mala experiencia, y poder reaccionar en tiempo y forma con el objetivo de evitar la mala experiencia y así convertirlo en promotor.

Tal como se investigó con los *managers* de *customer service,* la actualidad de esta importante empresa de *e-commerce* se basa en un sistema donde los usuarios se contactan vía los canales *offline* y los casos son creados dentro de las distintas colas de atención, pero asignados aleatoriamente entre los representantes de *customer service*, sea cual fuera su grado de *expertise*.

La recomendación de negocio que se propone es mejorar el sistema de asignación aleatoria actual, incorporándole una lógica de asignación inteligente que tome como *inputs* a (i) las probabilidades de detracción de los usuarios que abren un caso en el canal offline de *customer service*, y (ii) la cantidad de representantes de *customer service* disponibles para atender casos, ordenados de mayor a menor *expertise*.

Este nuevo sistema propuesto tendría como lógica asignar los casos con mayor probabilidad de detracción a los mejores representantes de *customer service*, de forma tal de mitigar el riesgo de detracción. Y para los casos con baja probabilidad de

detracción, asignárselos a los representantes de menor nivel de seniority, logrando mantener la experiencia del usuario y fomentar el desarrollo del representante de *customer service*.

6.3. Limitaciones

En línea con la recomendación de negocio, se debe hacer foco en las limitaciones que trae el análisis de la presente tesis. Existen oportunidades para mejorar el análisis predictivo y prescriptivo. A continuación, se enumeran los principales tres caminos que se podrían recorrer.

Para comenzar, el *scope* de análisis está limitado a los países Hispanohablantes de la empresa de *e-commerce*, y a los contactos en los canales offline. Se tomó este grupo para unificar lenguaje, pero de avanzar, se debería analizar la performance del modelo en los casos abiertos en portugués y evaluar si no varía considerablemente la performance, o bien desarrollar modelos por separado especializados en cada lenguaje.

Si bien la etapa predictiva logra alcanzar resultados favorables, llegando a una capacidad de predicción del 70%, existe oportunidad en mejorar este grado de precisión. Como próximos pasos del análisis, podría mejorar la capacidad predictiva explorando aún más cantidad de variables del usuario, aumentar el volumen de datos y explorar más técnicas de *Machine Learning*, e incluso de *Deep Learning*.

Por último, se hizo especial foco en el desarrollo de un modelo que prediga probabilidad de detracción del usuario. La solución de negocio de asignar los recursos de mayor *seniority* a aquellos casos con mayor probabilidad de detracción debe ser considerada con cuidado, ya que podría no ser la óptima, y depender de otras variables. Como solución a este inconveniente y próximos pasos, se debería estudiar el sistema de asignación como un problema de optimización mediante técnicas de programación lineal, tomando varios inputs, donde uno de ellos sea las probabilidades de detracción que arroja el modelo desarrollado.

6.4. Estimación del impacto económico

En conjunto con la recomendación de negocio, se procedió a cuantificar el valor económico de implementar este nuevo modelo de asignación. Para el mismo, se tomaron una serie de supuestos hasta llegar a un impacto anual en dólares de implementar el nuevo modelo.

6.4.1. Supuestos tomados

A continuación se listan los supuestos tomados para el cálculo del impacto económico de la iniciativa:

- Implementar el nuevo sistema de asignación logrará un aumento de 12pp del NPS, ya que el NPS de los representantes *juniors* de *customer service* será igual al NPS de los *team leaders* mostrado en la **Figura 1.3**.
- En los casos donde el modelo prediga que el usuario no será detractor y en la realidad lo es, el sistema asignará al usuario un representante de bajo *skill* y el usuario terminará siendo detractor en el 100% de las veces.
- En los casos donde el modelo prediga que el usuario será detractor y en la realidad también lo es, el sistema asignará al usuario un representante de alto *skill* y el usuario terminará siendo promotor en el 70% de las veces (se asume una efectividad del 70% dado que igual puede seguir siendo detractor o bien puede ser neutro).
- En los casos donde el modelo prediga que el usuario no será detractor y en la realidad tampoco lo es, el sistema asignará al usuario un representante de bajo *skill* y el usuario terminará siendo promotor en el 70% de las veces.
- En los casos donde el modelo prediga que el usuario será detractor y en la realidad no lo es, el sistema asignará al usuario un representante de alto *skill* y el usuario terminará siendo promotor en el 70% de las veces, pero a un costo por la mala asignación del recurso.
- El costo anual por una mala asignación del recurso por usuario será de 500 USD.
- Un usuario promedio gastará 670 USD por año. Se toma este valor ya que es el valor promedio de un usuario en Amazon [33], empresa referente de *e-commerce*.
- Un usuario promotor gastará un 50% más de lo que gasta en promedio un usuario.
- Un usuario detractor gastará un 50% menos de lo que gasta en promedio un usuario.
- Un usuario neutro gastará lo mismo que gasta en promedio un usuario.
- Un usuario en promedio abre 10 casos por año en el canal *offline*.
- Un usuario que se contacte y tenga una buena o mala experiencia, quedará con dicha experiencia percibida a lo largo del año.

6.4.2. Análisis de impacto

Como primer paso, se consideró la matriz de confusión del modelo óptimo de *XG Boost* descripto en la sección **5.5. Modelo *XG Boost*** para identificar a los usuarios que serán promotores, detractores y neutros, calculando el peso que representa cada uno de los cuadrantes, tal como se indica en la **Figura 6.1**.

		Clase predicha	
		Detractor	No Detractor
Clase real	Detractor	3.215 (11%)	7.582 (25%)
	No Detractor	2.008 (7%)	17.152 (57%)

Figura 6.1: cantidad de casos (y porcentajes) de matriz de confusión XG Boost

Luego, se analizó cada uno de los cuadrantes de la matriz de confusión:

- El 11% será asignado a un representante de alto *skill* y terminará siendo promotor en el 70% de los casos. Se llamó a este valor como $\%P_{DD}$ y su valor fue aproximadamente 7%.
- El 25% será asignado a un representante de bajo *skill* y terminará siendo detractor en la totalidad de los casos. Se llamó a este valor como $\%D_{DN}$ y su valor fue de 25%.
- El 7% será asignado a un representante de alto *skill* y terminará siendo promotor en el 70% de los casos, pero a un dado costo. Se llamó a este valor como $\%P_{ND}$ y su valor fue aproximadamente 5%.
- El 57% será asignado a un representante de bajo *skill* y terminará siendo promotor en el 70% de los casos. Se llamó a este valor como $\%P_{NN}$ y su valor fue aproximadamente 39%.

Con estos cuatro valores, se procedió a calcular el NPS del nuevo modelo de asignación tal que:

$$NPS_{propuesta} = (\%P_{DD} + \%P_{ND} + \%P_{NN}) - \%D_{DN} = 26\% \quad (6.1)$$

Luego, se procedió a calcular el nuevo mix de neutros del nuevo modelo de asignación, completando el 100% de la suma de los promotores y detractores del nuevo modelo de asignación, tomando un valor de 24%. Se llamó a este valor como $\%N_{propuesta}$.

Como se detalló en la sección **6.4.1. Supuestos tomados**, se consideró el gasto anual en dólares promedio de un usuario igual a 670 USD. Se llamó a este valor como G_{anual} y se llamaron a los valores de gasto anual en dólares de un usuario promotor, detractor y neutro como G_P, G_D y G_N respectivamente, tal que:

$$G_P = 1{,}5 * G_{anual} = 1.005\ USD \quad (6.2)$$

$$G_D = 0{,}5 * G_{anual} = 335\ USD \quad (6.3)$$

$$G_N = G_{anual} = 670\ USD \quad (6.4)$$

Con el objetivo de calcular la ganancia anual con el nuevo sistema de asignación, se consideraron los valores de:

- Cantidad de casos abiertos por año en el canal offline, tomando un valor de 2.700.000 casos, tal como se detalló en la **Figura 1.2**.
- Cantidad de casos abiertos por año en promedio por usuario, tomando un valor de 10 casos, tal como se detalló en la sección **6.4.1. Supuestos tomados**.
- Costo anual de una mala asignación de recurso, tomando un valor de 500 USD por usuario, tal como se detalló en la sección **6.4.1. Supuestos tomados**.

Con estos valores, se procedió a calcular la ganancia anual del nuevo sistema de asignación tal que:

$$Ganancia_{propuesta=}$$

$$\left[\%P_{DD} * G_P + \%D_{DN} * G_D + \%P_{ND} * (G_P - costo) + \%P_{NN} * G_P + \%N_{propuesta} * G_N\right] * \frac{\#casos\ anuales\ abiertos}{\#\ caso\ anuales\ por\ usuario} = 197{,}8 \text{ millones de USD} \quad (6.5)$$

Una vez obtenida la ganancia anual del nuevo sistema de asignación, se procedió a calcular la ganancia anual del sistema de asignación actual. Para ello, se consideraron los valores conocidos de:

- %mix de promotores actual, tomando un valor de 50%, conocido del *dataset* analizado. Se llamó a este valor como *%P*.
- %mix de detractores actual, tomando un valor de 36%, conocido del *dataset* analizado. Se llamó a este valor como *%D*.
- %mix de neutros actual, tomando un valor de 14%, conocido del *dataset* analizado. Se llamó a este valor como *%N*.

Con estos valores, se procedió a calcular la ganancia anual del sistema de asignación actual tal que:

$$Ganancia_{actual=}$$

$$[\%P * G_P + \%D * G_D + \%N * G_N] * \frac{\#casos\ anuales\ abiertos}{\#\ caso\ anuales\ por\ usuario} = 193{,}6 \text{ millones de USD} \quad (6.6)$$

Conocidas las ganancias del nuevo sistema de asignación y del sistema de asignación actual, se procedió a calcular el impacto económico del nuevo sistema de asignación, tal que:

$$Impacto\ económico\ anual_{propuesta} = Ganancia_{propuesta} - Ganancia_{actual} = 4{,}2\ millones\ de\ USD \quad (6.7)$$

Por último, se procedió a calcular el impacto de NPS del nuevo sistema de asignación, tal que:

$$Impacto\ NPS_{propuesta} = NPS_{propuesta} - NPS_{actual} = 12\% \quad (6.8)$$

Como se puede observar, el impacto económico del nuevo sistema de asignación será de 4,2 millones de USD anuales, y el aumento de NPS será de 12pp, tal como se detalló en la sección **6.4.1. Supuestos tomados**.

6.4.3. Análisis del costo de mala asignación de recursos

Si bien en la sección **6.4.1. Supuestos tomados** se asumió un costo anual de mala asignación por usuario de 500 USD, se analizó el máximo costo tolerado por el cual el impacto económico anual de la propuesta sea positivo y brinde ganancias.

El resultado obtenido fue que para costos anuales inferiores a 844 USD, el nuevo modelo de asignación generará un impacto económico mejor al sistema de asignación actual. Con costos superiores a este valor, el modelo de asignación propuesto generará ganancias inferiores al modelo aleatorio, con lo cual no tendrá sentido implementar.

6.5. Conclusiones finales

Como conclusión final de la tesis, y tras haber obtenido resultados aplicables por medio de técnicas de *Machine Learning*, se puede confirmar que los datos no estructurados son una fuente rica de información esperando a ser utilizados.

En entornos de la industria de *e-commerce* cada vez más enfocados al usuario, la experiencia del usuario no debería perder importancia con el crecimiento del negocio. Utilizar NLP puede dar una gran ventaja competitiva en *customer service*, y anticiparse a los posibles descuidos que se le da al usuario.

Si bien la presente tesis tiene como foco de estudio las etapas descriptivas y predictivas, el estudio prescriptivo podría incorporar un modelo de asignación que tome como input las probabilidades de detracción que arroja el modelo de *XG Boost*, y así cumplir con el objetivo de estar enfocado en el usuario, optimizar el uso del recurso del representante de *customer service*, y que la experiencia del usuario no se vea afectada.

Bibliografía

[1] Miguel, N. (2017). La atención al cliente en la era del eCommerce, < https://www.oleoshop.com/blog/atencion-al-cliente-ecommerce >

[2] Becerra, J. (2021). Ecommerce: La experiencia del usuario es más importante que el producto, < https://www.america-retail.com/ecommerce/ecommerce-la-experiencia-del-usuario-es-mas-importante-que-el-producto/ >

[3] Forrester (2012). Consumers Drive Channel Preference To Achieve Effortless Customer Service, < https://go.forrester.com/blogs/12-06-20-consumers_drive_channel_preference_to_achieve_effortless_customer_service/>

[4] Reichheld, F. (2003). The One Number You Need to Grow

[5] Müller, O., Brocke, J. and Junglas, I. (2016). Using Text Analytics to Derive Customer Service Management Benefits from Unstructured Data, <https://www.researchgate.net/profile/Jan-Vom-Brocke/publication/312293790_Using_Text_Analytics_to_Derive_Customer_Service_Management_Benefits_from_Unstructured_Data/links/5a340267aca27247eddc21bc/Using-Text-Analytics-to-Derive-Customer-Service-Management-Benefits-from-Unstructured-Data.pdf >

[6] Hui, S. and Jha, G. (2000). Data mining for customer service support, < https://sci-hub.do/https://www.sciencedirect.com/science/article/pii/S0378720600000513 >

[7] Auguste, J., Charlet, D., Damnati, G., Bechet, F. and Favre, B. Can we predict self-reported customer satisfaction from interactions?, < https://sci-hub.do/https://ieeexplore.ieee.org/abstract/document/8683896/ >

[8] SAS. Procesamiento del lenguaje natural – Qué es y por qué es importante, <https://www.sas.com/es_ar/insights/analytics/what-is-natural-language-processing-nlp.html>

[9] Zheng, A. and Casari A. (2016). Feature Engineering for Machine Learning. Principles and techniques for data scientists, pp. vii

[10] Zheng, A. and Casari A. (2016). Feature Engineering for Machine Learning. Principles and techniques for data scientists, pp. 52

[11] Bramer, M. (2007). Principles of Data Mining. Stop words and Stemming, pp. 242-243

[12] Bramer, M. (2007). Principles of Data Mining. Stop words and Stemming, pp. 245

[13] Zheng, A. and Casari A. (2016). Feature Engineering for Machine Learning. Tf-Idf: Bag-of-X: Turning Natural Text into Flag Vectors pp. 42-45

[14] Zheng, A. and Casari A. (2016). Feature Engineering for Machine Learning. Tf-Idf: A Simple Twist on Bag-of-Words, pp. 61-63

[15] Li, Z. (2019). A Beginner's Guide to Word Embedding with Gensim Word2Vec Model, <https://towardsdatascience.com/a-beginners-guide-to-word-embedding-with-gensim-word2vec-model-5970fa56cc92>

[16] Zheng, A. and Casari A. (2016). Encoding Categorical Variables, pp. 78

[17] Bramer, M. (2007). Principles of Data Mining. Avoiding Overfitting of Decision Trees, pp. 119-120

[18] Gareth, J. (2013). An Introduction to Statistical Learning with Applications in R. Supervised versus Unsupervised Learning, pp. 35-37

[19] Gareth, J. (2013). An Introduction to Statistical Learning with Applications in R. Logistic Regression, pp. 142-144

[20] Zheng, A. and Casari A. (2016). Feature Engineering for Machine Learning. Classification with Logistic Regression, pp. 66-67

[21] Bramer, M. (2007). Principles of Data Mining. Naïve Bayes Classifiers, pp. 24-31

[22] Gareth, J. (2013). An Introduction to Statistical Learning with Applications in R. Classification Trees, pp. 314-318

[23] Gareth, J. (2013). An Introduction to Statistical Learning with Applications in R.

[24] Gareth, J. (2013). An Introduction to Statistical Learning with Applications in R. Random Forests, pp. 324-325

[25] Gareth, J. (2013). An Introduction to Statistical Learning with Applications in R. Boosting, pp. 325-328

[26] Gareth, J. (2013). An Introduction to Statistical Learning with Applications in R. Variable Importance Measures, pp. 323-324

[27] Bramer, M. (2007). Principles of Data Mining. True and False Positives and Negatives, pp. 174-176

[28] Bramer, M. (2007). Principles of Data Mining. Performance Measures, pp. 176-185

[29] Zheng, A. and Casari A. (2016). Feature Engineering for Machine Learning. Tunning Logistic Regression with Regularization, pp. 68-72

[30] Gareth, J. (2013). An Introduction to Statistical Learning with Applications in R. Boosting, pp. 325-329

[31] Codd, E. (1970). Modelo relacional de Codd, estructuras y relaciones, <http://www.lsi.us.es/docencia/get.php?id=3183>

[32] Chamberlin, D. (1974). Information technology — Database languages — SQL — Part 1: Framework (SQL/Framework), < https://www.iso.org/standard/63555.html>

[33] Sickler, J. (2020). What is Customer Lifetime Value?, < https://terakeet.com/blog/customer-lifetime-value/>

Apéndice

I) Query SQL:

```
create multiset volatile table DET, no log as (

SELECT
  NPS.cas_case_id as CASE_ID,
  CAST(INC.contact_date_id AS DATE)/100+190000 as MES_ID,
  INC.INCOMING_TYPE,
  INC.CS_BU as BU,
  INC.CS_CENTRO as CENTRO,
  INC.sit_site_id,
  INC.cas_contact_origin as ORIGEN_CONTACTO,
  INC.cas_form_id as FORM_ID,
  (CASE WHEN BODY_TXT is null then 'Sin texto'
        WHEN BODY_TXT is not null then 'Con texto'
        else 'NA' end) as Flag_texto,
  (CASE WHEN INC.CAS_USER_SEGMENT in (INC.ML_SELLER_NAME,INC.MP_SELLER_NAME) THEN 'SELLER'
        WHEN INC.CAS_USER_SEGMENT=INC.ML_BUYER_NAME THEN 'BUYER'
        ELSE 'REVISAR' end) as Rol,
  INC.cas_user_segment as NIVEL_USUARIO,
  INC.proceso_agrupado_nps as PROCESO_AGRUPADO,
  INC.user_team_channel,
  INC.USER_ROLE as EXPERIENCIA_REP,
  NPS.RES_NPS,
  (CASE WHEN NPS.RES_NPS in (0,1,2,3,4,5,6) then '-1'
        WHEN NPS.RES_NPS in (7,8) then '0'
        WHEN NPS.RES_NPS in (9,10) then '1'
        ELSE 'NA' END) AS NPS,
  TXT.fc_body_txt as BODY_TXT


FROM
  WHOWNER.BT_CX_CONTACTS INC
  INNER JOIN WHOWNER.BT_CX_NPS_DETAIL NPS on NPS.CAS_CASE_ID = INC.CAS_CASE_ID
  INNER JOIN WHOWNER.LK_CX_FIRST_CONTACT TXT on TXT.cas_case_id = NPS.cas_case_id

WHERE
  CENTRO= 'Hispanos'

  and MES_ID in ('202001','202002','202003','202004','202005','202006','202007','202008','202009','202010','202011','202012')
  and INC.INCOMING_TYPE in ('OUTGOING_FIRST_CONTACT') --me quedo con los contactos que generaron OFC
  and INC.QUEUE_ID not in ('109','110') --excluyo colas de bugs
  and INC.FLAG_EXCLUDE_NUMERATOR_CR = '0' --excluyo los casos que no se consideran en el CR
  and INC.origin_table in ('BT_CX_INCOMING_CR')
  and proceso_agrupado is not null
  and proceso_agrupado not in ('Wallet','CBT','Cards','No Team','Point') --excluyo procesos de MP de bajo peso
  and ORIGEN_CONTACTO in ('FORM')
  and TXT.FC_TYPE in('Form')
  and Rol not in ('REVISAR')
  and Flag_texto= 'Con texto'
  and USER_TEAM_CHANNEL is nul null
  and USER_TEAM_CHANNEL not in ('SIN_CLASIFICAR')
  AND FORM_ID is not null

) with data primary index(CASE_ID) on commit preserve rows;

Sel count(*) from DET;
Sel * from DET;
```

II) Sampling del dataset:

CASE_ID	MES_ID	BU	CENTRO	SIT SITE ID	ORIGEN CONTACTO	FORM_ID	ROL	NIVEL USUARIO	PROCESO AGRUPADO	USER TEAM CHANNEL	EXPERIENCIA REP	RES NPS	NPS	BODY_TXT
59484548	Mayo	Marketplace	Hispanos	Chile	FORM	73	BUYER	LOYAL_BUYER_4	Fintech Comprador	OFFLINE	JR	4	Detractor	Completa el formulario para que atendamos tu caso. Te….
64341397	Julio	Marketplace	Hispanos	Mexico	FORM	7018	SELLER	MERCADO_LIDER	Marketplace Vendedor	OFFLINE	JR	0	Detractor	Pregúntanos lo que quieras: Hola necesito ayuda con el ..
58074772	Mayo	Marketplace	Hispanos	Argentina	FORM	187	BUYER	LOYAL_BUYER_2	Prustomer	OFFLINE	JR	5	Detractor	Tu publicación no cumplía con nuestras políticas…
71925096	Septiembre	Marketplace	Hispanos	Argentina	FORM	122	BUYER	LOYAL_BUYER_2	Prustomer	OFFLINE	SR	6	Detractor	Completá el formulario para que podamos darte más información. Mensaje: Hola. …
47848176	Enero	Marketplace	Hispanos	Argentina	FORM	64	BUYER	LOYAL_BUYER_5	Fintech Comprador	OFFLINE	JR	7	Pasivo	Completa el formulario para que atendamos tu caso. Te responde…
87320293	Diciembre	Marketplace	Hispanos	Mexico	FORM	73	BUYER	LOYAL_BUYER_4	Fintech Comprador	MULTICANAL CHAT	AGENT	0	Detractor	Completa el formulario para que atendamos tu caso. …
68722767	Agosto	Logistica	Hispanos	Mexico	FORM	73	BUYER	LOYAL_BUYER_2	ACME	MULTICANAL CHAT	JR	0	Detractor	Completa el formulario para que atendamos tu caso. Te responderemos en …
59751560	Mayo	Logistica	Hispanos	Mexico	FORM	7157	BUYER	LOYAL_BUYER_2	ACME	MULTICANAL CHAT	JR	10	Promotor	¿Dónde quieres recibir la compra? El paquete ya está en camino…
83923792	Diciembre	Marketplace	Hispanos	Mexico	FORM	7018	BUYER	LOYAL_BUYER_3	Prustomer	MULTICANAL CHAT	AGENT	10	Promotor	Pregúntanos lo que quieras: Por que me suspendieron mi cuenta si en ningún …
60649641	Mayo	Marketplace	Hispanos	Argentina	FORM	189	BUYER	LOYAL_BUYER_1	Prustomer	OFFLINE	SS	7	Pasivo	Completa el formulario para que atendamos tu caso. Te responderemos en lo….
54393538	Marzo	Marketplace	Hispanos	Argentina	FORM	188	BUYER	LOYAL_BUYER_2	Prustomer	OFFLINE	JR	9	Promotor	Completa el formulario para que atendamos tu caso. Te respond…
78380856	Octubre	Logistica	Hispanos	Mexico	FORM	7018	SELLER	MERCADO_LIDER	Fulfillment	CE	JR	4	Detractor	Pregúntanos lo que quieras: hola buen dia, quisiera mandar unos productos a fu…
61440134	Junio	Marketplace	Hispanos	Colombia	FORM	73	BUYER	LOYAL_BUYER_3	Marketplace Comprador	MULTICANAL CHAT	JR	8	Pasivo	Completa el formulario para que atendamos tu caso. Te responderemos en los próxi….
72105055	Septiembre	Marketplace	Hispanos	Chile	FORM	187	BUYER	LOYAL_BUYER_4	Prustomer	OFFLINE	JR	10	Promotor	Tu publicación no cumplía con nuestras políticas. Completa el formulario y po...
79168087	Noviembre	Marketplace	Hispanos	Colombia	FORM	699	BUYER	LOYAL_BUYER_3	Prustomer	OFFLINE	JR	8	Pasivo	Completa el formulario para que atendamos tu caso. Te responderemos en los próximos 2 días…

III) Descripción del dataset:

VARIABLE	TIPO	VALORES	DESCRIPCIÓN
CASE_ID	Integer	71925096, 64341397, …	ID del caso
MES_ID	Integer	202001, 202002, …	Mes del caso
BU	String	Marketplace, Logística	Tipo de consulta, si es por *Marketplace* o cuestiones logísticas
CENTRO	String	Hispanos	Lenguaje del caso (hispano)
SIT_SITE_ID	String	Chile, México, Argentina, Colombia, Venezuela, Ecuador, Uruguay, Perú, Costa Rica, República Dominicana, Panamá, Bolivia, Paraguay, Guatemala	País del caso
ORIGEN_CONTACTO	String	FORM	Origen de contacto (por formulario)
FORM_ID	Integer	73, 7018, …	ID de la pregunta del Formulario
ROL	String	Comprador, Vendedor	Rol del usuario
NIVEL_USUARIO	String	BEST_SELLER, LONG_TAIL, LOYAL_BUYER_1, LOYAL_BUYER_2, LOYAL_BUYER_3, LOYAL_BUYER_4, LOYAL_BUYER_5, LOYAL_BUYER_6, MERCADO_LIDER, MODTAIL_FIJO, MIDTAIL_VARIABLE, ML_TOP_ON, MP_TOP_OFF	Nivel del usuario
PROCESO_AGRUPADO	String	ACME, Cuenta, Fintech comprador, Fintech vendedor, Fulfillment, Marketplace comprador, Marketplace vendedor, prustomer	Problemática del caso
USER_TEAM_CHANNEL	String	C2C, CE, CHAT, MULTICANAL, MULTICANAL C2C, MULTICANAL CHAT, OFFLINE	Canal de atención del usuario
EXPERIENCIA_REP	String	AGENT, JR, SP, SR, SS, TEAM_LEADER, TL	Nivel de seniority del representante
RES_NPS	Integer	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10	Nota de la encuesta
NPS	String	Detractor, Pasivo, Promotor	Tipo de respuesta de NPS
BODY_TXT	String	Hola, hice una compra que…	Texto escrito por el usuario en el formulario

IV) Resultados área bajo la curva ROC:

#	Volumen dataset	Enfoque BOW	Q_CORTE (BOW)	Norm Tf-Idf (BOW)	Enfoque Word2vec	Modelo importado (Word2vec)	Algoritmo Machine Learning	Hiperparámetros óptimos	ROC
1	20.000	Sí	5	No	No	-	Regresión Logística	-	0,554
2	20.000	Sí	20	No	No	-	Regresión Logística	-	0,554
3	20.000	Sí	100	No	No	-	Regresión Logística	-	0,554
4	**100.000**	**Sí**	**20**	**No**	**No**	**-**	**Regresión Logística**	**-**	**0,685**
5	100.000	No	-	-	Sí	No	Regresión Logística	-	0,674
6	100.000	No	-	-	Sí	Sí	Regresión Logística	-	0,642
7	100.000	Sí	20	No	No	-	Naïve Bayes	-	0,584
8	100.000	Sí	20	Sí	No	-	Naïve Bayes	-	0,621
9	**100.000**	**No**	**-**	**-**	**Sí**	**No**	**Naïve Bayes**	**-**	**0,623**
10	20.000	Sí	5	No	No	-	Random Forest	2250 árboles	0,678
11	20.000	Sí	20	No	No	-	Random Forest	1750 árboles	0,680
12	20.000	Sí	100	No	No	-	Random Forest	2750 árboles	0,679
13	20.000	Sí	5	Sí	No	-	Random Forest	1250 árboles	0,674
14	20.000	Sí	20	Sí	No	-	Random Forest	1750 árboles	0,677
15	20.000	Sí	100	Sí	No	-	Random Forest	1500 árboles	0,675
16	100.000	Sí	20	No	No	-	Random Forest	1750 árboles	0,688
17	**100.000**	**No**	**-**	**-**	**Sí**	**No**	**Random Forest**	**"**	**0,690**
18	100.000	No	-	-	Sí	Sí	Random Forest	"	0,661
19	20.000	Sí	5	No	No	-	XG Boost	0,01 eta / 0,4 colsample / 1500 árboles / 6 max_depth / 10 gamma	0,685
20	20.000	Sí	20	No	No	-	XG Boost	"	0,677
21	20.000	Sí	100	No	No	-	XG Boost	"	0,680
22	20.000	Sí	5	Sí	No	-	XG Boost	"	0,681

23	20.000	Sí	20	Sí	No	-	XG Boost	"	0,677
24	20.000	Sí	100	Sí	No	-	XG Boost	"	0,680
25	**100.000**	**No**	**-**	**-**	**Sí**	**No**	**XG Boost**	**"**	**0,701**
26	100.000	No	-	-	Sí	Sí	XG Boost	"	0,689