UNIVERSIDAD TORCUATO DI TELLA

# Clasificación de mensajes dentro de la plataforma Properati: Un abordaje con NLP

**Maestría en Management + Analytics**

**Tesis**

**Alumno: Santiago Cisco**

**Tutor: Agustín Gravano**

## Resumen

El objetivo del siguiente trabajo es la elaboración de un modelo de Procesamiento de Lenguaje Natural (NLP) para clasificar los mensajes con consultas que envían los usuarios de Properati. Recurrimos a la metodología Bag of Words, utilizando un XGBoost como modelo para hacer las predicciones. Presentamos distintas técnicas de preprocesamiento de texto como tokenización, eliminación de stopwords, lematización, umbrales de frecuencia mínima para tokens y normalización TF-IDF. Hacemos pruebas sobre dos categorías para seleccionar las transformaciones que se efectúen al modelo. El modelo arroja, finalmente, una buena performance en las etiquetas de mayor frecuencia.



## Abstract

The aim of the following article is the elaboration of a Natural Language Processing (NLP) model to classify the messages with queries sent by the users of Properati. We use the Bag of Words methodology, with an XGBoost as the model to make predictions. We present different text preprocessing techniques such as tokenization, stopword elimination, lemmatization, minimum frequency thresholds for tokens and TF-IDF normalization. Tests will be made on two categories to select the transformations to be made to the model. Finally, the performance reached on the most abundant categories is satisfactory.

# Agradecimientos

En primer lugar, quiero agradecer a mi tutor, Agustín Gravano, por la paciencia y el acompañamiento durante todo este proceso.

Por otra parte, a Martín Sarsale por disponibilizarme los datos necesarios para poder realizar este trabajo.

Por último, a mis compañeros y seres queridos, por la ayuda brindada y el apoyo.

# Índice

# 1. Introducción

La irrupción de las nuevas tecnologías generó un incremento exponencial de las fuentes de datos que las organizaciones pueden aprovechar. La abundancia de información, sumado a las formas novedosas de explotación que suponen las técnicas *de Machine Learning* (aprendizaje automático) y *Deep Learning*, proporcionan a las empresas herramientas muy potentes para generar inputs para la toma de decisiones, además de posibilitar la automatización del trabajo que antes debía hacerse en forma manual.

Entre estas técnicas, el campo del Procesamiento de Lenguaje Natural (NLP) se especializa en lograr que las computadoras realicen tareas vinculadas al lenguaje humano (Jurafsky y Martin, 2019). El desafío del algoritmo es la transformación del texto en una representación semántica formal (Roukos, 2007: 617). Para mitigar este problema, muchas veces se recurre a limitar al modelo a un dominio o problema específico para modelar correctamente las características de esa tarea.

Lo que distingue al NLP de otros campos del aprendizaje automático es el conocimiento del lenguaje: Para que un algoritmo pueda trabajar con este, debe poder distinguir palabras y lidiar con características propias del lenguaje como la ortografía y la sintaxis (Jurafsky y Martin, 2019). Asimismo, además de cada palabra en sí, es importante tener en cuenta que el contexto en el que se encuentra puede alterar su significado (ibid.).

En este sentido, existen una serie de cuestiones que definieron los sistemas de NLP que se fueron desarrollando históricamente (Rabiner y Juang, 2007: 522): La unidad

de reconocimiento del lenguaje (desde palabras o secuencias de palabras, hasta sílabas o fonemas); el tamaño del vocabulario que el sistema puede procesar; la relación que el sistema atribuye a las palabras (tokens), es decir, desde sistemas que toman a las palabras como tokens independientes a los que establecen relaciones complejas para modelar su significado; la forma de interacción con el usuario (humano); y el medio.

Los datos, en este tipo de problemas, deben ser preprocesados teniendo en cuenta estas particularidades. Posteriormente, pueden aplicarse muchos de las técnicas comunes en otras áreas del aprendizaje automático como la separación en un conjunto de entrenamiento y otro de validación para entrenar y testear un modelo con el objetivo de, por ejemplo, clasificar mensajes de texto en diferentes categorías.

El objetivo de este trabajo, es el diseño de un modelo de NLP que permita clasificar los mensajes de clientes que entran a la plataforma Properati en una serie de tags no excluyentes, con el fin de automatizar la clasificación de los mismos. Esto permitiría a la empresa procesar el gran volumen de mensajes que recibe en forma diaria, permitiendo a las inmobiliarias que contratan sus servicios priorizar a la gente que busca propiedades en base a sus consultas, automatizar algunas respuestas o brindar una atención más personalizada.

El trabajo se estructurará de la siguiente manera: en la sección dos se ampliará la información sobre la empresa, se hará un análisis descriptivo del corpus de mensajes y de la distribución de las etiquetas (la variable dependiente). En la sección tres se presentará la metodología en la que está basada el modelo y se presentarán todos los preprocesamientos que se le harán a los mensajes para poder ser explotados. Además, se explicará el modelo elegido y las distintas métricas para poder evaluar su desempeño.

En la sección cuatro, se mostrarán las pruebas realizadas sobre los datos y los resultados de la corrida del modelo y su interpretación. Por último, en la sección cinco se presentarán las conclusiones del trabajo, incluyendo como el mismo puede contribuir a mejorar los servicios provistos por la empresa Properati.

# 2. Datos

## 2.1 Descripción del negocio

El presente trabajo se realizó con la base de datos de mensajes de la empresa Properati. La misma es una plataforma de compra/venta y alquiler de propiedades que conecta usuarios que buscan vender o alquilar una propiedad (por lo general, inmobiliarias) con potenciales clientes. La empresa cobra a los oferentes por contacto recibido. Además, opera en cinco países de Latinoamérica: Argentina, Colombia, Ecuador, Perú y Uruguay.

La plataforma ofrece una descripción básica de las propiedades (ubicación, precio, tamaño en ambientes y metros cuadrados), fotos (a veces se incluye un render interactivo en 3D) y una descripción breve. Por otra parte, los clientes pueden interactuar con el oferente de la propiedad a través de un sistema de mensajes en el que se le solicita al usuario nombre, mail y teléfono de contacto. Por defecto, si el usuario no escribe un mensaje en ese campo, se envía el siguiente texto al propietario: “Hola! Me interesa esta propiedad que vi en Properati.”.

La empresa busca diferenciarse de otras plataformas del mismo rubro a través de un enfoque centrado en el usuario y de la utilización de datos para la elaboración de métricas de tendencias de mercado, las cuales hace públicas en su página y en un blog, para ayudarlo en sus decisiones.

## 2.2 Necesidad de negocio

La empresa manifestó que las inmobiliarias que publican en el sitio, que son los clientes que monetiza Properati, reciben un gran volumen de mensajes por día. Si bien la empresa no brindó un número específico, se nos explicó que cada inmobiliaria puede llegar a recibir entre 100 y 200 por día. En la actualidad, Properati no brinda herramientas para organizar el flujo de

mensajes y su procesamiento el cual, por lo general, se realiza en forma manual por las inmobiliarias, lo que redunda en tiempos largos de respuesta a sus clientes.

En este sentido, Properati manifestó la necesidad de tener un sistema que procese los mensajes previamente de manera que las inmobiliarias tengan una herramienta que les permita entender qué mensajes tienen más valor y cuales menos. De esta manera, las mismas podrían ordenar su trabajo en forma más eficiente y reducir los tiempos de respuesta a los clientes con mayor potencialidad de realizar una operación, al priorizar este tipo de mensajes.

Por otra parte, la empresa manifestó que un modelo predictivo sería una herramienta mejor que otras opciones, como alguna categorización elegida por el usuario antes de enviar el mensaje, ya que quieren facilitarle al máximo la experiencia al mismo, evitándole tener que decidir a qué categorías pertenece su mensaje. Por otra parte, si el usuario no elige las categorías correctamente, es posible que la inmobiliaria pierda información valiosa de un potencial cliente.

Por último, un modelo entrenado para reconocer las distintas categorías a las cuales pertenece un mensaje, podría ser el primer paso para configurar un sistema de respuestas automáticas, configurados por la inmobiliaria, a los mensajes más simples, dejándoles únicamente los mensajes más complejos para responder.

## 2.3 Estructura del dataset

Se recibió por parte de la empresa una base de 5000 mensajes elegidos en forma aleatoria con los siguientes datos:

- ID: un código de identificación numérico único por cada mensaje.
- País: Indica el país en el que usuario que envió el mensaje está registrado.
- Mensaje: Variable “string” que contiene el mensaje que formuló el usuario.

Para la confección del modelo, se decidió trabajar sobre el mensaje en sí como variable independiente y descartar las variables de ID y país.

Por otra parte, la variable dependiente se definió mediante un ida y vuelta con el cliente. Se presentó la problemática y se le ofreció dos opciones: La primera fue generar una serie pequeña de categorías excluyentes para entrenar un modelo de clasificación multi clase; la segunda, generar una serie de etiquetas no excluyentes, para lo que se generarían una serie de modelos de clasificación binarios concatenados que predijeran si el mensaje pertenece o no a cada etiqueta.

El cliente finalmente optó por la segunda ya que el mensaje de un usuario de la plataforma podría contener más de una consulta. Por ejemplo, el siguiente mensaje contiene consultas acerca de expensas y medidas del departamento, además de requisitos para el alquiler:

> "Buenas tardes.
>
> Quería mayor información por este departamento, expensas, m2 y las condiciones de la garantía"

Properati manifestó en interés en abarcar la mayor cantidad de información posible sobre cada mensaje, por lo que se definieron como variable dependiente las siguientes etiquetas no excluyentes:

| Variable | Definición |
|---|---|
| Visitar | Cuando el cliente manifiesta el interés de visitar la propiedad publicada. |
| Dirección | Consultas sobre la dirección de la propiedad publicada. |
| Precio | Consultas sobre el precio de la propiedad publicada. |
| Disponibilidad | Consultas sobre si la propiedad se encuentra disponible actualmente. |
| Crédito | Consultas sobre la posibilidad de financiar la compra de la propiedad mediante un crédito hipotecario. |
| Dormitorios | Consultas sobre los dormitorios de la propiedad. |
| Expensas | Consultas sobre los costos mensuales de la propiedad (expensas, impuestos, limpieza, etc.). |
| Parking | Consultas sobre si la propiedad tiene un espacio para guardar un vehículo. |
| Medidas | Consultas sobre las medidas (usualmente superficie) de la propiedad. |
| Comunicación | Cuando el usuario quiere comunicarse con la inmobiliaria. |
| Requisitos | Consultas sobre los requisitos para poder alquilar o comprar la propiedad. |
| Más información | Cuando el usuario pide en forma genérica más información de la propiedad. |
| Fotos | Cuando el usuario pide más fotos de la propiedad. |
| Permuta | Consulta sobre si el propietario acepta permutar el inmueble por otro bien o propiedad. |
| Amoblada | Consultas sobre si la propiedad se entrega con muebles. |
| Piso | Cuando el usuario pregunta en qué piso se encuentra la propiedad. |
| Apto_Profesional | Cuando el usuario consulta si la propiedad es apta profesional. |

*Tabla 1: etiquetas utilizadas como variable dependiente*

En este sentido, un mensaje podría pertenecer a una, múltiples o ninguna categoría. En caso de que el mensaje pertenezca a la categoría correspondiente, la etiqueta tomará un valor de 1 para ese mensaje, de lo contrario, tomará un valor de 0.

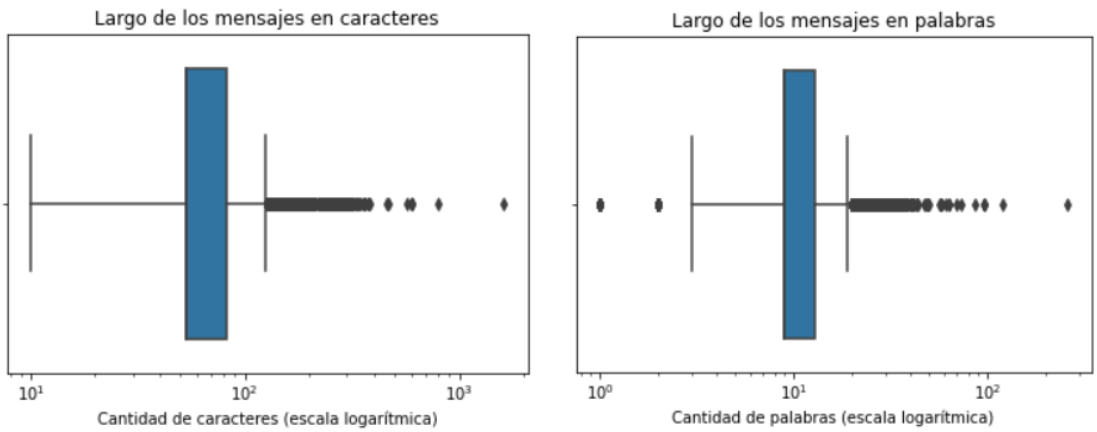
## 2.4 Análisis descriptivo de los mensajes

Los mensajes a procesar son strings cortos que tienen un largo promedio de 78 caracteres, con una mediana de 53. También, se observa que la media está en 12 palabras, con una mediana de 9. A modo ilustrativo, se muestran los siguientes ejemplos:

“Buenas tardes.

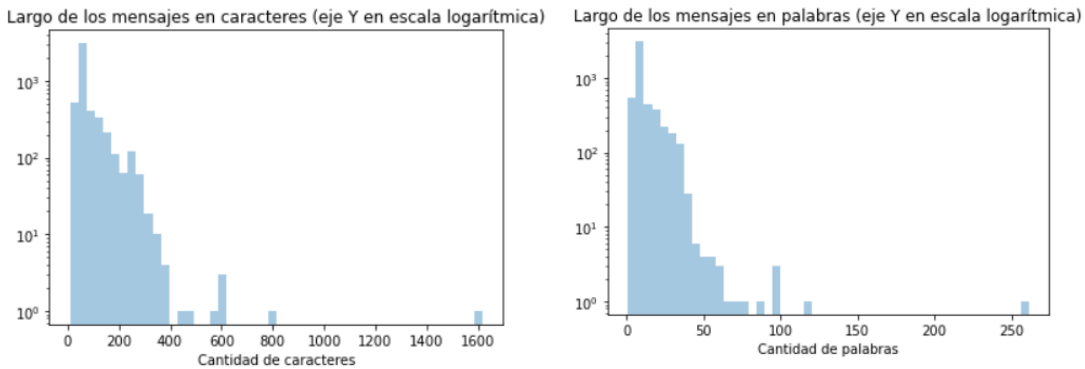
Quería mayor información por este departamento, expensas, m2 y las condiciones de la garantía”

“Buenos días, solicito por favor medidas y cuantas habitaciones consta. Gracias”




*Gráficos 2 y 3: Boxplots del largo de los mensajes en caracteres y en palabras. Escala logarítmica.*

Se puede observar (gráficos 2 y 3) cómo los primeros tres percentiles están por debajo de los 82 caracteres, si bien hay algunos outliers por encima de los 400. Por otra parte, al contar las palabras por mensaje, se pueden ver varios outliers por encima de los 50, lo que explica que la media esté cerca del tercer cuartil (13 palabras).




*Gráficos 3 y 4: Histogramas del largo de los mensajes en caracteres y en palabras. Escala logarítmica.*

Por otra parte, en los gráficos 3 y 4 puede verse cómo existe un pico muy fuerte de mensajes de largo de entre 50 y 75 caracteres. Medido en palabras, se observa de nuevo un pico en la distribución, esta vez en torno a las 9 palabras.

## 2.5 Distribución de etiquetas

Un primer problema que se encontró al observar el corpus de mensajes fue la gran proporción de mensajes por defecto: “Hola! Me interesa esta propiedad que vi en Properati.”, 2602 del total de 5009 en la base, que explica los picos en la distribución de los largos observados en la sección anterior. La gran proporción de mensajes sin ningún contenido informativo y que no pertenece a ninguna de las etiquetas disponibles podría introducir algo de ruido en la clasificación.

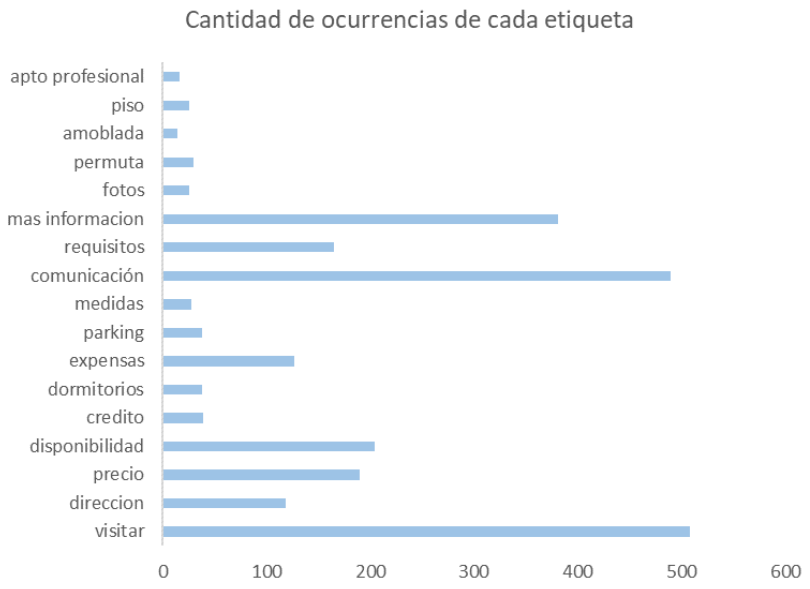



*Gráfico 5: Cantidad de ocurrencias por etiqueta*

Por otra parte, al observar la cantidad de ocurrencias de cada etiqueta (gráfico 5 y tabla 2) se puede ver que hay algunas que tienen un número muy bajo; por ejemplo, “apto profesional”, “piso”, “amoblada”, “permuta”, “fotos” y “medidas” tienen menos de 30

ocurrencias cada una, mientras que “crédito”, “dormitorios” y “parking” tienen tienen 39, 38 y 38 ocurrencias respectivamente. Debido a la proporción muy pequeña del corpus de mensajes que pertenece a estas etiquetas, podría suponer un problema para el modelo para clasificarlas.

| Etiqueta | Cantidad | Porcentaje del corpus |
|---|---|---|
| visitar | 508 | 10,14% |
| dirección | 118 | 2,36% |
| precio | 190 | 3,79% |
| disponibilidad | 204 | 4,07% |
| crédito | 39 | 0,78% |
| dormitorios | 38 | 0,76% |
| expensas | 127 | 2,54% |
| parking | 38 | 0,76% |
| medidas | 27 | 0,54% |
| comunicación | 490 | 9,78% |
| requisitos | 165 | 3,29% |
| más información | 381 | 7,61% |
| fotos | 25 | 0,50% |
| permuta | 30 | 0,60% |
| amoblada | 14 | 0,28% |
| piso | 25 | 0,50% |
| apto profesional | 16 | 0,32% |

*Tabla 2: Ocurrencias de cada etiqueta y porcentaje del corpus*

# 3. Metodología

## 3.1 Modelo Bag of Words

Para predecir las distintas etiquetas se recurrió a un modelo Bag of Words. Como se explicará a continuación, este tipo de modelos utiliza un vector de ocurrencias de palabras (*tokens*) de cada mensaje para poder predecir el valor que tomará su variable dependiente. Es decir, se considera el conteo de cada palabra como una variable independiente (Goldberg, 2017: 69).

En este sentido, se construye una matriz "document-term" en la cual cada fila de la matriz es un documento (en este caso, mensaje) y cada columna es una palabra del vocabulario total del modelo; como valores dentro de la matriz se toman a la cantidad de ocurrencias de cada palabra en cada documento. Al considerar a cada documento como un vector de ocurrencias de palabras, se puede considerar que dos documentos van a tender a ser similares si tienen palabras similares, es decir, si sus vectores son similares (Jurafsky y Martin, 2020: 102 y 103). De esta manera, el modelo puede aprender a inferir el valor de la variable dependiente a partir del valor de su vector.

La similitud entre dos vectores está definida por el ángulo formado entre los mismos, medido a través de la **similitud coseno**, que se calcula dividiendo el producto interno de dos vectores por la norma del primer vector dividido por la norma del segundo.

$$\frac{\vec{a} \cdot \vec{b}}{|\vec{a}||\vec{b}|} = \cos\theta$$

*Similitud coseno entre dos vectores*

De esta manera se puede captar la similitud de la frecuencia relativa entre las palabras de los documentos. Un valor cercano a 1 (0º entre los vectores) implica mayor similitud, mientras que un valor cercano a 0 (ángulo de 90º entre los vectores) implica que los documentos son muy diferentes[1].

En el ejemplo del gráfico 6, Jurafsky y Martin (2020: 102) comparan distintas obras de Shakespeare a partir de la ocurrencia de sólo dos términos: "battle" y "fool". En el mismo, las cuatro obras se representan como vectores bidimensionales de la frecuencia de esas palabras.

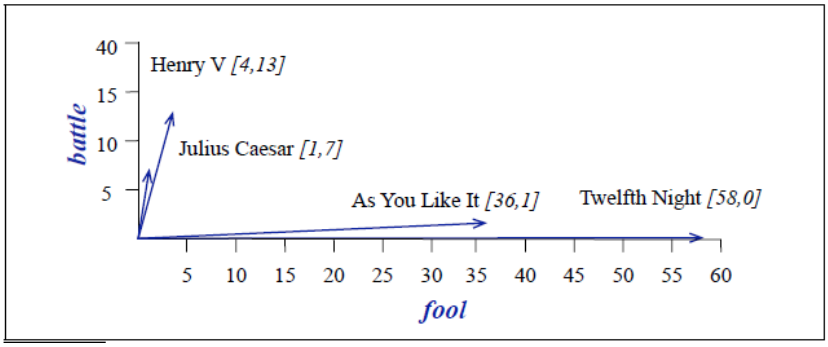



*Gráfico 6: Ejemplo en de documentos como vectores bidimensionales extraído de Jurafsky y Martin (2020: 102).*

En el ejemplo, pueden considerarse los documentos "Henry V" y "Julius Caesar" como más similares entre sí que con las obras "As you like it" y "Twelfth Night" ya que el ángulo entre sus vectores es menor que el ángulo con las otras dos obras.

Para trabajar esta metodología, se realizaron una serie de transformaciones a los mensajes, se los tokenizó y luego se los introdujo en un modelo por cada etiqueta para que prediga la probabilidad de que el mensaje pertenezca a la misma. A continuación, se presentarán todas las transformaciones hechas a los mensajes y el modelo elegido.

[1] En este tipo de problemas, es imposible un ángulo superior a 90º ya que no se consideran valores negativos para las frecuencias de las palabras.

## 3.2 Transformaciones

### 3.2.1 Transformaciones de formato

Para poder facilitar al modelo el aprendizaje de la relación entre determinados tokens y el valor de la variable dependiente de un determinado mensaje, es importante normalizar el formato del texto. Esto implica eliminar las mayúsculas, tildes y diéresis a fin de evitar que el modelo tome como palabras distintas a dos tokens por el uso de mayúsculas. Por otra parte, el eliminar las tildes ayuda a evitar el problema de que existan usuarios que acentúen su escritura y otros que no, algo común en internet.

También, se eliminaron los signos de puntuación y números de los mensajes para evitar que estos sean tomados como tokens.

### 3.2.2 Tokenización

Al recibir los mensajes, las computadoras toman los strings como secuencias de caracteres: A diferencia de los humanos, no pueden distinguir entre palabras. Para el procesamiento del lenguaje, por lo tanto, es necesario romperlos en palabras y signos de puntuación. A este proceso se lo llama “tokenización” (Bird et al, 2009: 80) y consiste en dividir el texto en “tokens”. Un “token” es una secuencia de caracteres que queremos tratar de manera agrupada (Bird et al, 2009: 7), por lo general, una palabra, aunque puede ser un conjunto de palabras.

### 3.2.3 Stopwords

Las “stopwords” son palabras de uso muy común en el lenguaje, que no tienen un significado interesante para los modelos de lenguaje natural. Eliminarlas puede ayudar a mejorar un poco la performance de estos modelos ya que son palabras que producen “ruido”: A pesar de que no tienen un significado que permita determinar que un mensaje pertenezca o no a una categoría, el modelo podría relacionarlas.

A fin de removerlas, se utilizó la lista de stopwords en español del paquete NLTK[2]. De esta lista, se quitó la palabra “donde”, ya que observamos en los datos que tiene una asociación importante con algunas etiquetas (como “dirección”), y se agregaron “favor” (por la expresión “por favor”), “hola” y “properati” que tienen un nivel de ocurrencia muy alto debido, sobre todo al mensaje por defecto (“Hola! Me interesa esta propiedad que vi en Properati.”) que incluye la plataforma. Además, estos tokens no tienen un valor informativo para el modelo.

### 3.2.4 Lematización

Para poder expresar distintas funciones en una oración, las palabras raíz (“lexemas”) sufren alteraciones o “flexiones” que posibilitan que las palabras se adapten al contexto del lenguaje. Sin embargo, a diferencia de las personas, un modelo puede no ser capaz de comprender que dos palabras con el mismo lexema (por ejemplo, gato y gatos) tienen el mismo significado. La lematización es un proceso que consiste en llevar las palabras de un documento a su lexema para eliminar este problema (Jurafsky y Martin, 2020: 20).

[2] Ver listado en anexo.

### 3.2.5 Frecuencia de tokens

Una de las desventajas de la metodología Bag of Words es que, por lo general, las matrices "document-term" resultantes tienen una dimensionalidad (cantidad de variables/términos) muy alta, dada la gran cantidad de palabras que puede tener el vocabulario presente en un modelo. Gran parte de los términos tienen una frecuencia muy baja y pueden no ser muy explicativos de la variable dependiente.

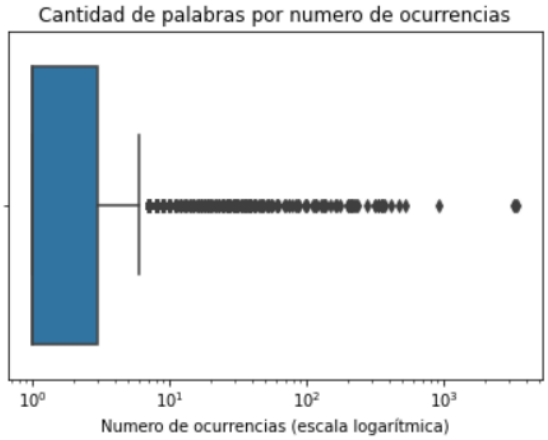



*Gráfico 7: Boxplot de cantidad de ocurrencias de cada palabra, zoom entre 0 y 100 ocurrencias (post quitado de stopwords y lematización)*

En el presente trabajo, puede verse en el gráfico 7 que, aun después de quitar stopwords y habiendo lematizado, la gran mayoría de los términos tiene un número bajo de ocurrencias: De los 2409 términos en el corpus, 1998 tienen menos de cinco ocurrencias; la mediana de ocurrencias está en 1 y el tercer cuartil en 3. Esto puede generar problemas ya que es posible que los términos de pocas apariciones estén generando ruido en el modelo, además de añadir mucha dimensionalidad al modelo innecesariamente, lo cual es costoso computacionalmente. Por esto, podría ser conveniente eliminar los tokens de menor frecuencia de un umbral determinado: Por ejemplo, si un token aparece una única vez o tan solo dos veces, podría ser

conveniente eliminarlo para mejorar la performance del modelo y reducir algo la cantidad de ruido en los datos ingresados.

### 3.2.6 Normalización TF-IDF

Al contrario que el problema descrito en la sección anterior, es probable que algunas palabras aparezcan con una muy alta frecuencia en todos los documentos, por ejemplo, las stopwords, aunque no necesariamente todas las palabras que aparecen con mucha frecuencia en un documento lo son. Si una palabra aparece con frecuencia demasiado alta, puede ser menos importante a la hora de clasificar un mensaje ya que pueden ser menos informativas que ocurren con una frecuencia menor (Jurafsky y Martin, 2020: 106).

| **Token** | **Cantidad de apariciones** |
|---|---:|
| propiedad | 3452 |
| interesar | 3328 |
| ver | 3266 |
| interesado | 933 |
| gracias | 520 |
| querer | 478 |
| ser | 412 |
| saber | 368 |
| celular | 363 |
| proyecto | 352 |
| contacto | 350 |
| persona | 350 |
| si | 336 |
| enviado | 332 |
| informacion | 318 |
| inmobiliario | 313 |
| buen | 280 |
| visitar | 236 |
| preferir | 231 |
| podriar[3] | 226 |

*Tabla 3: Listado de tokens más frecuentes*

[3] Al lematizar, algunos lemmas pueden no ser los gramaticalmente correctos. Sin embargo, como el lematizador lleva a todos los términos a la misma raíz, esto no debería afectar en la predicción del modelo.

En este trabajo, después de aplicar lematización y eliminación de stopwords, existen aún palabras de muy alta frecuencia, como, por ejemplo “propiedad”, “interesar”, “ver”, “interesado” o “gracias”. Es probable que estas palabras introduzcan algo de ruido en las predicciones del modelo. Para reducir este problema, se puede recurrir a la normalización TF-IDF.

El problema de alta frecuencia puede producirse de dos maneras: La primera es que un mismo término aparezca demasiadas veces en uno o más documentos; la segunda es que existan términos que aparezcan en muchos documentos. Para abordar el primer problema, se recurre a la normalización TF (Term Frecuency). Siguiendo a Jurafsky y Martin (2020: 107), en vez de tomarse la frecuencia bruta de cada término, se le aplica un logaritmo base 10, para penalizar las de mayor aparición. Se le agrega un uno para evitar un error si es que un término aparece 0 veces en un documento.

$$\mathrm{tf}_{t,d} = \log_{10}(\mathrm{count}(t,d)+1)$$

*Normalización TF, donde se calcula la frecuencia del término “t” en el documento “d”. Fuente: Jurafsky y Martin (2020: 107).*

Por otra parte, el término IDF (Inverse Document Frecuency) le da más peso a aquellos términos que aparecen en menos documentos. Los términos que aparecen sólo en algunos documentos son más útiles para diferenciarlos del resto que aquellos términos que aparecen en casi todo el corpus (Jurafsky y Martin, 2020: 107). El término IDF se calcula dividiendo el total de documentos de la colección por la cantidad de documentos en los que aparece ese término: SI una palabra aparece en todos los documentos, este término adoptará el valor de 1, si aparece en tan solo un documento, el valor del total de documentos. Se la suele penalizar con un logaritmo base 10 por el alto número de documentos que puede haber en un corpus.

$$\text{idf}_t = \log_{10}\left(\frac{N}{\text{df}_t}\right)$$

*Normalización IDF, donde "N" es el número total de documentos en el corpus y "DFt" la cantidad de documentos en los cuales el término "t" aparece. Fuente: Jurafsky y Martin, 2020: 107.*

La normalización TF-IDF combina los dos términos de la siguiente manera, donde Wt,d es la frecuencia ponderada del término "t" en el documento "d":

$$w_{t,d} = \text{tf}_{t,d} \times \text{idf}_t$$

*Norma TF-IDF. Fuente: Jurafsky y Martin, 2020: 108.*

## 3.3 Modelo elegido

El modelo elegido para poder obtener las predicciones es el Extreme Gradient Boosting (XGBOOST). Se procesarán los mensajes con las transformaciones descritas en la sección anterior y se los introducirá en 17 modelos secuenciales (uno por cada categoría a predecir) para estimar si cada uno de ellos pertenece o no a cada una de las categorías. Se analizará, asimismo, si el modelo tiene la capacidad de predecir todas las categorías ya que, como se observó en algunos casos, la frecuencia es demasiado baja.

Por otra parte, a la hora de tomar decisiones sobre transformaciones a realizar a los mensajes, se experimentará con las dos categorías de mayor frecuencia ("comunicación" y "visitar"), ya que la mayor disponibilidad de los datos facilita medir el impacto de cada transformación. Para esto, se dividirá el conjunto de datos en tres subconjuntos: Un conjunto de entrenamiento (con el cual se entrenarán los modelos a testear), un conjunto de validación (para medir la performance de los modelos y tomar decisiones), y un conjunto de testeo (con el cual se medirá la performance final del modelo). Esta división será explicada más adelante, pero es importante para evitar inflar los resultados obtenidos en los modelos y obtener una idea más real de cómo el modelo podría performar en producción. Por último, el modelo entregable al cliente será entrenado con todos los datos.

Para evaluar la performance de los modelos se recurrirá a las siguientes métricas: Accuracy; Área bajo la curva; Precisión; Recall; y F1 Score. Más adelante se profundizará sobre cada una y se analizarán ventajas y desventajas para este problema en particular. Además, se explicará por qué se tomó la decisión de que la métrica más útil para resolver este problema es el F1 score.

### 3.3.1 XGBoost

XGBoost es una implementación de la familia de algoritmos de "boosting". La misma es una técnica de ensamblaje de árboles de decisión, es decir, el modelo resultante es producto del ensamblado de árboles más pequeños en forma secuencial. Cada árbol es entrenado utilizando información de los árboles entrenados anteriormente (James et al, 2017: 321).

A diferencia de entrenar un solo árbol de decisión de gran tamaño para resolver el problema, lo cual podría generar el problema de sobre ajustar a los datos y una variabilidad alta, se entrenan varios arboles pequeños para que el modelo vaya aprendiendo lentamente. La ventaja de entrenar muchos árboles y ponderar sus decisiones, es que se reduce el error por variabilidad.

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all $i$ in the training set.
2. For $b = 1, 2, \ldots, B$, repeat:
   (a) Fit a tree $\hat{f}^b$ with $d$ splits ($d+1$ terminal nodes) to the training data $(X, r)$.
   (b) Update $\hat{f}$ by adding in a shrunken version of the new tree:

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x). \tag{8.10}$$

   (c) Update the residuals,

$$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i). \tag{8.11}$$

3. Output the boosted model,

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x). \tag{8.12}$$

*Algoritmo de boosting. Fuente: James et al, 2017: 323.*

Cómo se ve en el algoritmo, se entrena un primer árbol con d+1 (típicamente un número bajo) de nodos terminales. Se entrena luego un segundo árbol pequeño que se concentra sobre

los residuos del árbol anterior: Se actualizan los pesos de las observaciones de manera que las que el primero predijo mal van a tener un peso mayor. A continuación, se actualizan los residuos y se repite el proceso. El resultado final será un conjunto de árboles de decisión pequeños ponderados por la performance que tuvieron para poder predecir el problema.

Los hiperparámetros del XGBoost a optimizar son los siguientes[4]:

| **Hiperparámetros** | **Descripción** | **Valor por defecto** |
|---|---|---|
| eta | Factor de regularización de los pesos de cada feature en cada vuelta de la iteración | 0,3 |
| max_depth | Máxima profundidad de cada árbol | 6 |
| min_child_weight | Cantidad mínima de observaciones ponderadas en cada hoja | 1 |
| subsample | Ratio de muestreo tomado por el algoritmo del total de los datos para su entrenamiento | 1 |
| gamma | Mínima reducción de la función de pérdida exigida para aceptar un corte de cada árbol | 0 |
| colsample_bytree | Ratio de muestreo del total de variables para ser utilizado a la hora de entrenar cada árbol | 1 |
| alpha | Regularización L1 sobre los pesos | 0 |
| lambda | Regularización L2 sobre los pesos | 1 |

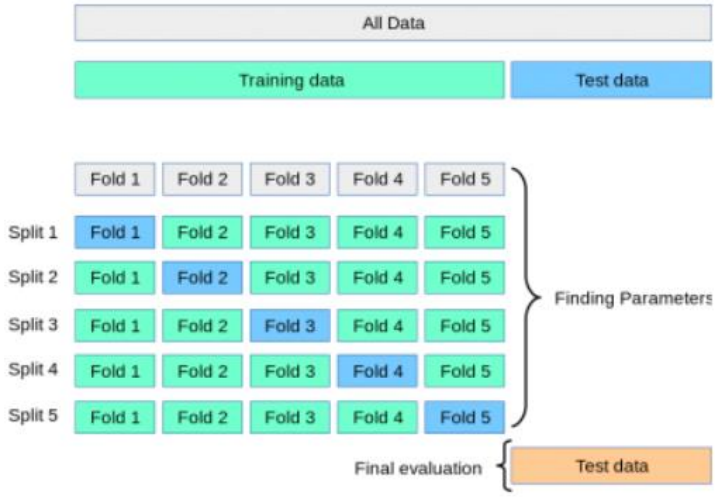
Normalmente, los hiperparámetros de un modelo se optimizan mediante Cross Validation. Esta técnica consiste en dividir el set de datos en un número K de subsets para utilizar K-1 subsets para entrenar un modelo y el subset restante para evaluarlo. Esto se itera K veces, utilizando en cada iteración un subset distinto como conjunto de validación.

En primer lugar, se construye una grilla con posibles valores de cada uno de los hiperparámetros; a continuación, se divide el set de datos en un número K de subsets y se entrena como se explicó anteriormente, utilizando una selección aleatoria de la lista de valores de cada hiperparámetro[5]. Este proceso se repite de manera que todos los K subsets de datos se

[4] Fuente: Documentación XGBoost. Disponible en https://xgboost.readthedocs.io/en/latest/parameter.html.

[5] A este tipo de búsqueda se lo denomina "random search". También es posible evaluar cada combinación de valores de hiperparámetros en la grilla (grid search); sin embargo, este método es demasiado costoso computacionalmente, por lo que se suele optar por el anterior.

utilicen como conjunto de validación. De esta manera, se obtiene una aproximación a cómo performaría el modelo, con la ventaja de haber utilizado todos los datos para entrenarlo y para validarlo.




*Ejemplo de un proceso de Cross Validation con un K = 5*[6]

Este proceso se repite un número de veces predefinido por el usuario y se obtiene el set de valores de hiperparámetros que tuvo mejor performance. Más adelante, se mostrará cómo se realizó este proceso para poder optimizar hiperparámetros en el modelo.

Por otra parte, algo muy útil a la hora de resolver problemas con un dataset muy desbalanceado es jugar con los pesos de las observaciones, lo que permite reducir el sesgo del modelo a predecir hacia la clase mayoritaria. Para esto, es conveniente ajustar el hiperparámetro scale_pos_weight, que balancea el peso de las observaciones de la clase positiva y la clase negativa. Un valor típico a utilizar, en este caso, es el conteo de las instancias negativas dividida el conteo de las instancias positivas[7].

[6] Extraído de https://towardsdatascience.com/cross-validation-and-hyperparameter-tuning-how-to-optimise-your-machine-learning-model-13f005af9d7d

[7] Fuente: Documentación de XGBoost, disponible en https://xgboost.readthedocs.io/en/latest/parameter.html. Ver en particular la descripción del hiperparámetro “scale_pos_weight”.

## 3.4 Métricas para la evaluación de los modelos

A continuación, se describirán las distintas métricas para evaluación de los modelos contemplados y se explicará por qué se eligió una de ellas en particular.

### 3.4.1 Accuracy

Esta métrica consiste simplemente en dividir la cantidad total de predicciones correctas sobre el total de las predicciones hechas. Su ventaja es que es muy clara e interpretable. La desventaja es que puede no ser muy sensible al sesgo en un modelo cuyas clases están muy desbalanceadas: Si se tiene una clase con una ocurrencia de, por ejemplo, 1%, si el modelo predice a todas las observaciones como de la clase mayoritaria tendrá una accuracy de 99%, lo cual puede no reflejar la performance real del modelo ya que, a pesar de que la métrica dio un resultado muy alto, no se captó ninguna observación de la clase minoritaria.

### 3.4.2 Área bajo la curva (AUC)

Esta métrica se utiliza en problemas de clasificación y relaciona dos razones: La razón de verdaderos positivos (True Positive Rate, en adelante TPR) y la razón de falsos positivos (False Positive Rate, en adelante FPR) (Goadrich, 2006: 2). El TPR mide la fracción de las observaciones positivas que fueron predichas correctamente. El FPR mide la fracción de los ejemplos negativos que fueron clasificados erróneamente como positivos. Las dos métricas toman valores entre el 0 y el 1.

| | actual positive | actual negative |
|---|---|---|
| predicted positive | $TP$ | $FP$ |
| predicted negative | $FN$ | $TN$ |

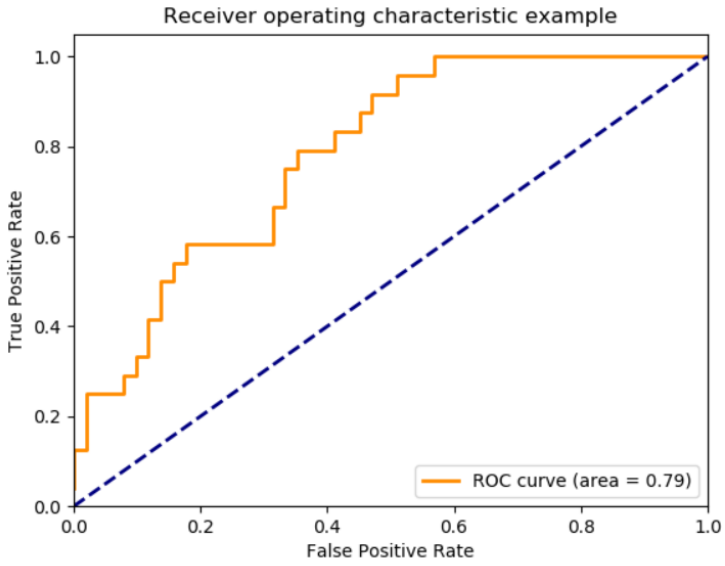
(a) Confusion Matrix

| | | |
|---|---|---|
| Recall | = | $\frac{TP}{TP+FN}$ |
| Precision | = | $\frac{TP}{TP+FP}$ |
| True Positive Rate | = | $\frac{TP}{TP+FN}$ |
| False Positive Rate | = | $\frac{FP}{FP+TN}$ |

(b) Definitions of metrics

*Definiciones de TPR, FPR, recall y precisión. Fuente: Goadrich, 2006: 4*

Para computar el AUC, el FPR y el TPR son computados a distintos umbrales y se crea una curva. El valor que toma el AUC es el área por debajo de la misma y puede interpretarse como la probabilidad de que un clasificador binario tome una observación positiva elegida aleatoriamente y la clasifique como tal.




*Ejemplo de gráfico de curva entre TPR y FPR (curva ROC) con la que se calcula el AUC[8].*

[8] Extraído de https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234

Se considera que un valor de 0.5 de AUC es igual al azar. Valores por debajo de ese umbral implica que el modelo performa peor que el azar, mientras que un valor cercano a 1 es un buen modelo medido por esta métrica.

### 3.4.3 Precisión y recall

Precisión mide la fracción de ejemplos clasificados como positivos que son realmente positivos. Recall, en cambio, mide la fracción de ejemplos positivos que son clasificados como tales (Goadrich, 2006: 2). La ventaja de utilizar alguna de estas métricas está en que mide que tan bueno es un modelo en predecir una clase determinada.

Existe por lo general un trade-off entre estas dos métricas: Si se busca un precisión alto, es posible que se elija un umbral de probabilidad más alto para determinar que una observación es de esa clase. Esto puede generar que algunas observaciones que efectivamente era de esa clase, pero cuya probabilidad de salida del modelo fue baja, sean clasificadas como negativas. Viceversa, si se busca incrementar el recall, es posible que se tenga que escoger un umbral de probabilidad más bajo, sacrificando algo de precisión en el camino.

### 3.4.4 F1 score

Esta métrica reconcilia precisión y recall: Es sensible tanto a qué tan preciso un clasificador es (es decir, a la proporción de observaciones que clasificó correctamente) y qué tan robusto es (que no haya perdido un número significativo de observaciones). La métrica va de 0 a 1, en donde mientras más cercano a 1 el valor, mejor es el clasificador. Se calcula de la siguiente manera:

$$F1 = 2 * \frac{1}{\frac{1}{precision} + \frac{1}{recall}}$$

*F1 score*[9]

Esta métrica puede ser muy útil para resolver el problema que estamos abordando, ya que nos interesa medir si el modelo logra clasificar correctamente la clase positiva, más que una ponderación de la clase positiva con la negativa. Al estar tan desbalanceados los datos, el modelo puede tender a dar valores sistemáticamente altos de AUC o de Accuracy a pesar de no clasificar correctamente las clases positivas. Por otra parte, si bien precisión y recall (medidas en la clase positiva) son métricas útiles para este problema, es ideal considerar una métrica que sea sensible a ambas para que se considere tanto la precisión como la robustez del modelo.

**Es por esto que se utilizará esta métrica para tomar decisiones sobre parámetros, procesamiento de datos y elección del modelo final.**

[9] Extraído de: https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234

## 3.5 Separación en Entrenamiento, Validación y Test

A fin de entrenar el modelo, se armarán tres conjuntos distintos: En primer lugar un conjunto de test, o Held-Out Set, con una selección aleatoria del 10% de los datos. Este conjunto no formará parte de ninguna prueba y se utilizará únicamente para medir la performance final de cada modelo.

La importancia de hacer esta separación inicial es porque al entrenar modelos sobre un conjunto de entrenamiento y ejecutar las pruebas sobre un conjunto de validación para tomar decisiones en cuanto a hiperparámetros o procesamiento de datos, es posible que se esté sobre ajustando sobre los datos de validación, inflando artificialmente la performance del modelo. El objetivo de este primer conjunto es medir la performance real del modelo en datos representativos que no hayan sido utilizados en ningún momento de la confección del modelo (James et al, 2017: 30).

Una segunda separación consiste en dividir el 90% restante de los datos en un conjunto de entrenamiento (70%) y otro de validación (30%) de manera aleatoria. Los datos de entrenamiento serán utilizados para entrenar los modelos en cada prueba. Estos modelos serán testeados en los datos de validación, a fin de elegir los parámetros que minimicen el error en validación en cada instancia. Como se explicó anteriormente, las pruebas serán realizadas en las dos categorías más importantes: “comunicación” y “visitar”.

Los hiperparámetros de cada modelo serán optimizados vía Cross Validation sobre el conjunto de entrenamiento y luego testeados en validación. Una vez que se encuentren los parámetros que minimicen el error en validación (medido con F1 score), los modelos serán entrenados con el 90% de los datos y serán testeados con los datos de testeo para medir la performance del modelo.

# 4. Resultados

## 4.1 Testeo y selección de transformaciones

A continuación se expondrán los resultados del testeo de las siguientes transformaciones elegidas: Exclusión de stopwords, uso de lematizador, quitar palabras menos frecuentes y normalización TF-IDF. Para probarlas, se normalizó el texto de los mensajes (quita de tildes y mayúsculas) además de tokenizarlo. Las transformaciones fueron comparadas mediante el uso de Cross Validation con la métrica F1, como se explicó en la sección anterior, pero se mostrarán otras métricas que permiten medir la performance del modelo.

### 4.1.1 Stopwords y lematización

*Comunicación*

Para establecer una base a fin de probar la performance del modelo incluyendo las transformaciones, se corrió uno sin ninguna de estas. Como la proporción de observaciones en el conjunto de entrenamiento que pertenecen al tag "comunicación" es de aproximadamente un 10% del total (306 de 3156), se corrió un XGBoost con el parámetro de scale_pos_weight = 10, para darle un peso un poco mayor a las observaciones positivas. La performance del modelo fue la siguiente:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 0.97 | 0.99 | 0.98 | 1216 |
| 1.0 | 0.86 | 0.75 | 0.80 | 137 |
| accuracy | | | 0.96 | 1353 |
| macro avg | 0.92 | 0.87 | 0.89 | 1353 |
| weighted avg | 0.96 | 0.96 | 0.96 | 1353 |

el AUC del modelo es: 0.9703046965040338

*Tabla 4: Resultado del modelo "comunicación" sin lematizar ni quitar stopwords*

Se alcanza un buen valor de F1 score sin necesidad de realizar casi transformaciones, un valor de AUC de 0.97 y una accuracy de 0.96. En este primer modelo puede verse lo expuesto en la sección anterior: Estas dos últimas métricas pueden no ser del todo precisas para medir la real performance del modelo debido al fuerte desbalance entre las clases. Puede evidenciarse algo de sesgo al ver que la precisión y el recall de la clase positiva son bastante más bajos que los de la clase negativa.

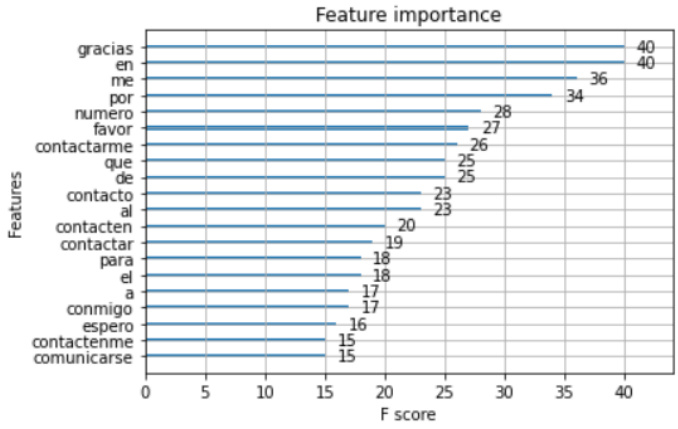



*Gráfico 8: Palabras más importantes para el modelo*

Por otra parte, al analizar las palabras que el modelo consideró más importantes para poder clasificar, puede observarse en el gráfico 8 que en los primeros lugares se encuentran algunas que no parecen tener mucha relación con la categoría pero parecerían ser muy importantes: “gracias”, algunas stopwords como “en”, “me” y “por”, la palabra “favor”. Es posible que estas palabras tan abundantes estén causando ruido en la predicción del modelo, impactando en su performance.

A continuación, se probó el mismo modelo pero con el lematizador Stanza en español, con las modificaciones descritas anteriormente.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 0.98 | 0.99 | 0.98 | 1223 |
| 1.0 | 0.86 | 0.82 | 0.84 | 130 |
| accuracy | | | 0.97 | 1353 |
| macro avg | 0.92 | 0.90 | 0.91 | 1353 |
| weighted avg | 0.97 | 0.97 | 0.97 | 1353 |

el AUC del modelo es: 0.9797377193534185

*Comunicación: prueba con lematizador pero sin quitar stopwords.*

Puede verse cómo el score F1 de la categoría positiva aumenta de 0.80 a 0.84, por lo que la lematización parece tener un impacto positivo en el modelo. El aumento se explica principalmente por un incremento de 0.07 puntos en el recall. Puede observarse, además, que, si bien existe un incremento en el AUC del modelo, este es extremadamente marginal.

Algo similar ocurre cuando se quitan los stopwords del modelo (sin aplicar lematización):

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 0.99 | 0.99 | 0.99 | 1229 |
| 1.0 | 0.85 | 0.85 | 0.85 | 124 |
| accuracy | | | 0.97 | 1353 |
| macro avg | 0.92 | 0.92 | 0.92 | 1353 |
| weighted avg | 0.97 | 0.97 | 0.97 | 1353 |

el AUC del modelo es: 0.9709277146381795

*Comunicación: sin lematización pero con quita de stopwords*

En este caso, el F1 Score aumenta de 0.80 a 0.85 en la categoría positiva. De nuevo, casi todo el incremento es explicado por un incremento en el recall, aunque esta vez aumentó marginalmente la precisión.

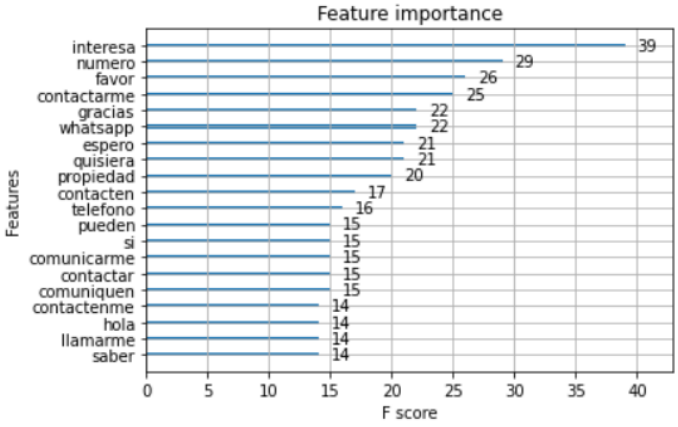



*Gráfico 9: Palabras más importantes para el modelo, con quita de stopwords*

Resulta interesante observar en el gráfico 9 que al quitar stopwords el modelo empieza a utilizar palabras más relacionadas con la etiqueta: “numero”, “contactarme”, “whatsapp” y “contacten” parecen palabras mucho más informativas que las que figuraban en el modelo base. Esto puede deberse a que la eliminación de stopwords podría estar quitando algo del ruido que generaban en el modelo, haciendo que el F1 de la categoría positiva aumente, al darle más peso, como explicamos anteriormente, a palabras más informativas para poder clasificar a los mensajes en una u otra categoría.

Cuando se utilizan tanto lematizador como quita de stopwords, la performance del modelo aumenta nuevamente, pasando el F1 score de 0.80 a 0.86. Incluso se alcanza una performance superior que considerando el lematizador (0.84) o la quita de stopwords (0.85) por separado.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 0.99 | 0.98 | 0.98 | 1220 |
| 1.0 | 0.85 | 0.86 | 0.86 | 133 |
| accuracy | | | 0.97 | 1353 |
| macro avg | 0.92 | 0.92 | 0.92 | 1353 |
| weighted avg | 0.97 | 0.97 | 0.97 | 1353 |

el AUC del modelo es: 0.9752619253050661

*Comunicación: Performance utilizando lematizador y quitando stopwords*

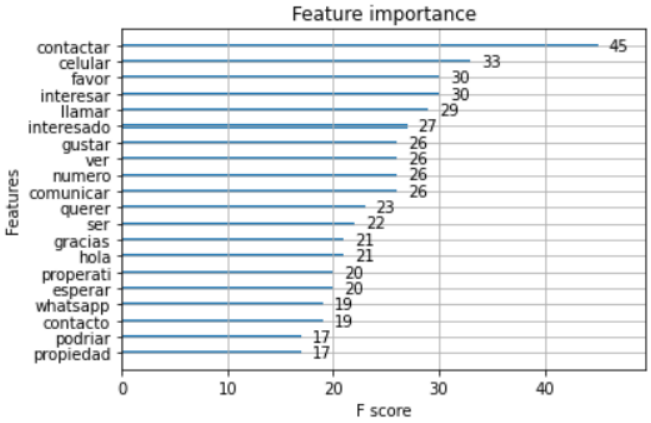



*Gráfico 10: Palabras más importantes para el modelo "comunicación", con quita de stopwords y utilizando lematizador*

Al analizar los tokens más importantes para el modelo (gráfico 10) nuevamente nos encontramos con palabras relacionadas semánticamente con la categoría a predecir, como "contactar", "celular", "llamar", "numero" o "comunicar". Esto, nuevamente, confirma que estas dos técnicas mejoran la performance del modelo.

### *Visitar*

Se procedió con la misma metodología para analizar la performance en la categoría "visitar". También se eligió un valor de 10 en el valor de scale_pos_weight ya que la frecuencia de aparición de esta categoría es similar. Sin embargo, en este caso, la performance del modelo empeora al aplicar lematización y al quitar stopwords. Si se observa el ejemplo base (sin lematizar ni quitar stopwords), la performance del modelo ya es bastante buena:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 0.99 | 1.00 | 0.99 | 1228 |
| 1.0 | 0.95 | 0.94 | 0.94 | 125 |
| accuracy | | | 0.99 | 1353 |
| macro avg | 0.97 | 0.97 | 0.97 | 1353 |
| weighted avg | 0.99 | 0.99 | 0.99 | 1353 |

el AUC del modelo es: 0.9963811074918567

*Visitar: performance del modelo sin lematizar ni quitar stopwords*

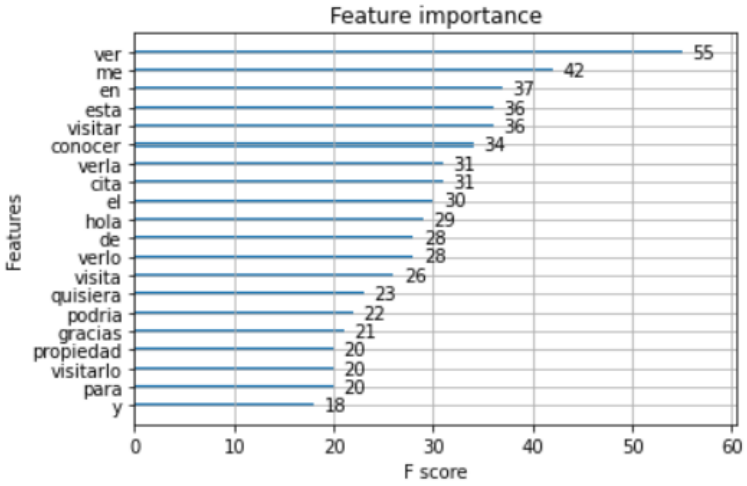



*Gráfico 11: Visitar: Palabras más importantes para el modelo*

Si bien se nota algo de ruido en las palabras que el modelo elige para poder clasificar el mensaje (gráfico 11), puede observarse un score F1 muy alto (0.94). Además, la palabra más importante (“ver”), está relacionada con la categoría. También aparecen palabras como “visitar”, “conocer”, “verla” y “cita” que se relacionan con la misma. Sin embargo, al aplicar lematización, la performance del modelo cae bastante. Lo mismo ocurre quitando stopwords.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 0.99 | 0.99 | 0.99 | 1218 |
| 1.0 | 0.89 | 0.94 | 0.91 | 135 |
| accuracy | | | 0.98 | 1353 |
| macro avg | 0.94 | 0.96 | 0.95 | 1353 |
| weighted avg | 0.98 | 0.98 | 0.98 | 1353 |

el AUC del modelo es: 0.9953597275436356

*Performance "visitar" aplicando lematización*

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 1.00 | 0.99 | 0.99 | 1230 |
| 1.0 | 0.88 | 0.95 | 0.91 | 123 |
| accuracy | | | 0.98 | 1353 |
| macro avg | 0.94 | 0.97 | 0.95 | 1353 |
| weighted avg | 0.98 | 0.98 | 0.98 | 1353 |

el AUC del modelo es: 0.9935620331813073

*Performance "visitar" quitando stopwords*

En ambos casos, el score F1 cae de 0.94 a 0.91, explicado sobre todo por una caída en la precisión. Si aplicamos ambas técnicas juntas, la performance cae aún más:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 0.99 | 0.99 | 0.99 | 1216 |
| 1.0 | 0.88 | 0.92 | 0.90 | 137 |
| accuracy | | | 0.98 | 1353 |
| macro avg | 0.93 | 0.95 | 0.94 | 1353 |
| weighted avg | 0.98 | 0.98 | 0.98 | 1353 |

el AUC del modelo es: 0.994459517864003

*Performance "visitar" aplicando lematización y quitando stopwords*

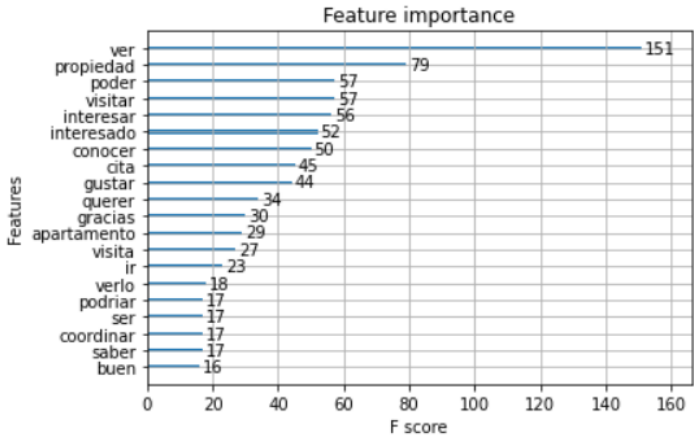



*Gráfico 12: Palabras más importantes del modelo "visitar" aplicando lematización y quita de stopwords*

En este caso, el F1 score baja a 0.90. Por otra parte, en el gráfico 12 puede apreciarse que el token "ver" adquiere aún más peso que en el modelo base, pero aparecen palabras que quizás están menos relacionadas con la categoría, como "interesar", "interesado" o "propiedad" entre las primeras categorías.

Teniendo en cuenta los resultados obtenidos, se tomó la decisión de realizar la quita de stopwords y la lematización a los mensajes del modelo. En primer lugar, la caída del score F1 en "Visitar" (0.04) es de menor magnitud que el incremento en el score de comunicación (0.06). Por otra parte, la performance del modelo en visitar sigue siendo muy alta a pesar de la caída del score (el F1 termina en 0.9). Finalmente, al observar las palabras que el modelo utiliza para clasificar los mensajes, en ambos casos aparecen más arriba palabras con mayor relación al campo semántico de la categoría una vez aplicadas las transformaciones. Esto podría ayudar a reducir la posibilidad de sobreajuste a los datos utilizados para entrenar el modelo para las categorías restantes.

### 4.1.2 Eliminación de palabras más frecuentes

Cómo se mencionó anteriormente, una posibilidad es introducir un umbral mínimo de apariciones que una palabra debe tener para que sea considerada como variable del modelo. Es decir, si un token tiene un número de apariciones en todo el conjunto de entrenamiento por debajo de ese umbral, se la elimina.

Se testearon, en ambas categorías, distintos umbrales para entender cómo influye en la performance de los modelos. Para lograr esto, se midió la cantidad de apariciones de cada palabra en el conjunto de entrenamiento. A continuación, se eliminaron del conjunto de entrenamiento y del de validación todas las palabras que tuvieran un valor de apariciones menor al umbral establecido en el conjunto de entrenamiento. Los umbrales testeados fueron 0 (mantener todas las palabras), 1 (eliminar todas las palabras que no aparecieran en el conjunto de entrenamiento), 2 (eliminar todas las palabras que aparecieran menos de 2 veces en el conjunto de entrenamiento), 3 (eliminar todas las palabras que aparecieran menos de 3 veces en el conjunto de entrenamiento) y 4 (eliminar todas las palabras que aparecieran menos de 4 veces en el conjunto de entrenamiento).

*Comunicación*

Al evaluar la eliminación de las palabras menos frecuentes en los umbrales mencionados, se obtuvieron los siguientes resultados[10]:

[10] Las métricas de precisión, recall y F1 se hicieron sobre la categoría positiva.

| Umbral | Precisión | Recall | AUC | F1 |
|---|---|---|---|---|
| 0 | 0,85 | 0,83 | 0,9857 | 0,84 |
| 1 | 0,86 | 0,84 | 0,9798 | 0,85 |
| 2 | 0,89 | 0,85 | 0,975 | 0,87 |
| 3 | 0,86 | 0,86 | 0,9661 | 0,86 |
| 4 | 0,84 | 0,81 | 0,9233 | 0,82 |

*Tabla 4: Resultados de eliminación de palabras menos frecuentes para el tag "comunicación" con distintos umbrales*

En la tabla 4 puede observarse que el modelo alcanza su mejor performance con un umbral de 2. A partir de allí la métrica F1 empieza a caer. Puede verse cómo al eliminar las palabras menos frecuentes, incrementa la precisión y el recall del modelo hasta alcanzar ese umbral.

*Visitar*

Al evaluar la eliminación de palabras menos frecuentes en esta categoría, se observa una mayor estabilidad en los resultados:

| Umbral | Precisión | Recall | AUC | F1 |
|---|---|---|---|---|
| 0 | 0,91 | 0,88 | 0,9819 | 0,89 |
| 1 | 0,89 | 0,89 | 0,9920 | 0,89 |
| 2 | 0,86 | 0,92 | 0,9920 | 0,89 |
| 3 | 0,89 | 0,91 | 0,9946 | 0,90 |
| 4 | 0,92 | 0,89 | 0,9897 | 0,90 |

*Tabla 5: Resultados de eliminación de palabras menos frecuentes para el tag "visitar" con distintos umbrales*

En este caso, la tabla 5 muestra cómo la métrica F1 permanece casi estable en los cinco ejemplos. Se puede ver cómo existe una caída en la precisión, que es compensada con un incremento en el recall. Ambas métricas alcanzan su mínimo y su máximo respectivamente con un umbral de 2.

*Cantidad de tokens*

Por otra parte, es interesante observar cómo cae la cantidad de tokens en el corpus a medida que se incrementa el umbral (gráfico 13).

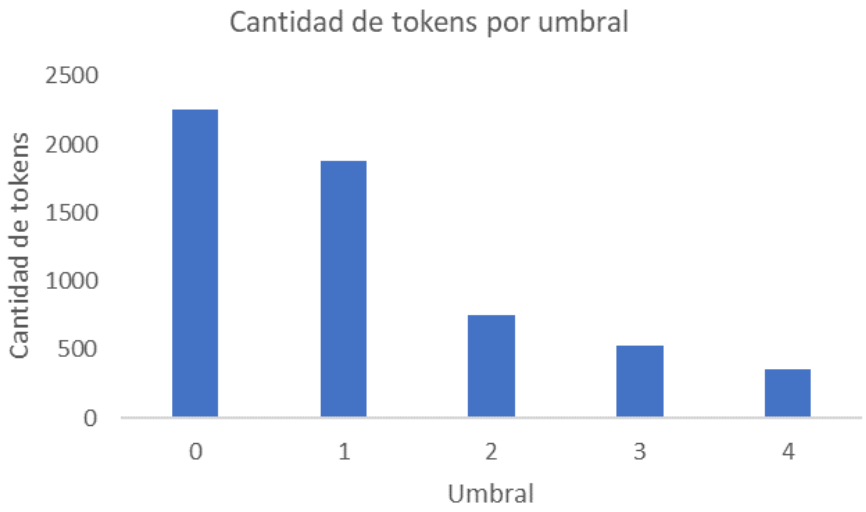



*Gráfico 13: Cantidad de tokens por umbral*

La cantidad de tokens cuando no se aplica ningún umbral es 2252. Este número cae rápidamente hasta alcanzar los 361 tokens con un umbral de 4, el máximo probado. Se tomará un umbral de 2 para entrenar el modelo, ya que en este punto la categoría "comunicación" alcanzó su mejor performance (en el caso de "visitar" se mantuvo estable en todos los valores). En ese umbral, la cantidad de tokens es de 754.

### 4.1.3 Normalización TF-IDF

Se probó en ambas categorías la performance del modelo aplicando normalización TF-IDF. Para esto se corrió un modelo base sin normalizar y luego un modelo base con la normalización aplicada, y se comparó su performance utilizando la métrica F1 de la categoría

positiva. Para correr estos modelos, se mantuvieron las transformaciones elegidas en secciones anteriores (lematización, quita de stopwords y quita de términos de menor frecuencia de 2).

| | Sin normalización | Con normalización |
|---|---|---|
| **Comunicación** | 0,86 | 0,84 |
| **Visitar** | 0,91 | 0,94 |

*Tabla 6: Testeo normalización TF-IDF en categorías comunicación y visitar*

Se observa en la tabla 6 un impacto ambiguo de la normalización en ambas categorías. Mientras que, en el caso de comunicación, la performance cae de 0,86 a 0,84, en el caso de visitar la misma incrementa de 0,91 a 0,94. En este caso, al igual que con la quita de stopwords y lematización, se decidió aplicar la normalización TF-IDF. En primer lugar, porque el incremento en performance de "visitar" es marginalmente mayor a la caída de performance en "comunicación". Por otra parte, "comunicación" aún mantiene un nivel de performance alto (0.84) después de aplicar la normalización. Por último, la normalización TF-IDF es una técnica reconocida para quitar el ruido que podrían generar los tokens de mayor frecuencia en este tipo de modelos. Si bien este no parecería ser un problema en la categoría "comunicación", podría ayudar a mejorar otras categorías que dependan excesivamente de algunos términos para generar la clasificación (como es el caso de "visitar") con un costo de performance mínimo para las categorías que no presentan este problema.

## 4.2 Optimización de hiperparámetros

Para optimizar hiperparámetros en cada uno de los modelos, se recurrió a la técnica de "random search" descripta anteriormente. Para ello, se armó una grilla única con valores de cada uno de los hiperparámetros y se corrieron 30 modelos por cada una de las etiquetas. Cada

modelo utilizó una configuración aleatoria de los hiperparámetros de la grilla (mostrada en la Tabla 7). Finalmente, se escogió la configuración con mayor performance en cada etiqueta.

Con el fin de validar la performance de cada uno, se recurrió a la técnica de Cross Validation con tres cortes, dado que algunas etiquetas tenían un número muy bajo de ocurrencias por lo que se prefirió que hayan menos cortes pero un mayor número de datos para validar. Por el mismo motivo, los cortes fueron elaborados mediante la técnica de "stratified K fold": Al momento de dividir los datos se busca mantener la distribución de clases que existe en el conjunto original en cada corte. Esto es especialmente útil en problemas con clases desbalanceadas.

Por otra parte, la métrica que se utilizará para comparar la performance de los modelos es el score F1 macro de Scikit Learn. Para calcular esta métrica, se toma el score F1 de la clase positiva y de la clase negativa y se los promedia. La elección de este score se debió a que este promedio no está ponderado por la frecuencia de cada clase, por lo que es más sensible a la clase menos frecuente en problemas en los que se tienen clases desbalanceadas[11].

La grilla de parámetros utilizada fue la siguiente:

| Hiperparámetro | Grilla de valores posibles |
|---|---|
| eta | [0.1, 0.2, 0.3, 0.4] |
| max_depth | [1, 2, 3, 4] |
| min_child_weight | [1, 3, 5, 7] |
| subsample | [0.7, 0.8, 0.9, 1] |
| gamma | [0, 0.2, 0.5, 1] |
| colsample_bytree | [0.7, 0.8, 0.9, 1] |
| alpha | [0, 0.1, 0.25, 0.5, 1] |
| lambda | [0, 0.5, 1, 2] |

*Tabla 7: Grilla de valores posibles de hiperparámetros a testear*

[11] Fuente: Documentación de la métrica F1 Score de Scikit Learn. Disponible en https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html#sklearn.metrics.f1_score

Los resultados obtenidos para las categorías “comunicación” y “visitar” fueron los siguientes[12] (se exponen las cinco combinaciones de hiperparámetros con mejor performance en Cross Validation en orden descendente)[13]:

| Comunicación | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| scale_pos_weight | subsample | min_child_weight | max_depth | lambda | gamma | eta | colsample_bytree | alpha | mean_test_score |
| 10 | 0.8 | 1 | 3 | 1 | 0 | 0.2 | 1 | 0.25 | 0.911823 |
| 10 | 0.9 | 1 | 3 | 1 | 0.5 | 0.2 | 0.7 | 0.5 | 0.910384 |
| 10 | 1 | 1 | 4 | 0 | 0 | 0.3 | 1 | 0.5 | 0.909611 |
| 10 | 0.9 | 3 | 1 | 0 | 0.5 | 0.3 | 0.7 | 0.5 | 0.901660 |
| 10 | 1 | 3 | 1 | 0.5 | 1 | 0.2 | 1 | 0 | 0.901415 |

*Tabla 8: Optimización de hiperparámetros de “comunicación”*

| Visitar | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| scale_pos_weight | subsample | min_child_weight | max_depth | lambda | gamma | eta | colsample_bytree | alpha | mean_test_score |
| 10 | 1 | 3 | 3 | 0 | 0 | 0.3 | 0.7 | 1 | 0.955773 |
| 10 | 1 | 3 | 2 | 1 | 0.2 | 0.4 | 0.8 | 0.1 | 0.953156 |
| 10 | 0.7 | 3 | 2 | 0.5 | 1 | 0.2 | 0.9 | 0 | 0.951792 |
| 10 | 0.7 | 5 | 2 | 0.5 | 0.2 | 0.2 | 0.8 | 0 | 0.944957 |
| 10 | 0.9 | 7 | 3 | 0.5 | 0.2 | 0.4 | 1 | 1 | 0.943977 |

*Tabla 9: Optimización de hiperparámetros de “visitar”.*

## 4.3 Resultados de los modelos

A continuación, se analizarán los resultados de los modelos. Se analizará su performance en validación y en test (held out set) con diversas métricas. Por otra parte, se analizará descartar algunos modelos cuya performance sea particularmente baja. También, se analizará si conviene o no incluir la variable del país del usuario que emitió el mensaje.

[12] En el caso del hiperparámetro “scale_pos_weight” se utilizó, en cada categoría, un valor aproximado del número de instancias negativas dividido el número de instancias positivas, como se explicó anteriormente.

[13] Los resultados del resto de las categorías están disponibles en el Apéndice.

### 4.3.1 Variable “País”

Se testeó el efecto de incluir o excluir el código del país en el cual el usuario que envió el mensaje está registrado. Para ello, se creó una variable *dummy* por cada valor que podría tomar la variable “país”: Argentina, Colombia, Perú, Ecuador o Uruguay. Por cada registro, la variable tomará el valor de 1 si es que el mensaje fue emitido por un usuario de ese país y 0 si es que el usuario que emitió ese mensaje no es de ese país.

Como con el resto de las pruebas, ésta se realizó sobre las categorías “comunicación” y “visitar”. Por otra parte, los modelos se testearon con las transformaciones probadas anteriormente y con los hiperparámetros optimizados para cada categoría.

Se probó el resultado del modelo en la categoría “comunicación” incluyendo la información del país del emisor y sin incluirlo:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 0.98 | 0.98 | 0.98 | 1207 |
| 1.0 | 0.85 | 0.87 | 0.86 | 146 |
| accuracy | | | 0.97 | 1353 |
| macro avg | 0.92 | 0.93 | 0.92 | 1353 |
| weighted avg | 0.97 | 0.97 | 0.97 | 1353 |

el AUC del modelo es: 0.9655264382426713

*Performance del modelo en categoría “comunicación” incluyendo información del país.*

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 0.99 | 0.98 | 0.98 | 1207 |
| 1.0 | 0.85 | 0.88 | 0.86 | 146 |
| accuracy | | | 0.97 | 1353 |
| macro avg | 0.92 | 0.93 | 0.92 | 1353 |
| weighted avg | 0.97 | 0.97 | 0.97 | 1353 |

el AUC del modelo es: 0.962147177991397

*Performance del modelo en categoría "comunicación" excluyendo información del país.*

Puede verse que casi no hay diferencias entre el modelo que incluye la variable país y el que la excluye. Lo mismo puede verse en la categoría "visitar":

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 1.00 | 0.98 | 0.99 | 1213 |
| 1.0 | 0.84 | 0.97 | 0.90 | 140 |
| accuracy | | | 0.98 | 1353 |
| macro avg | 0.92 | 0.98 | 0.95 | 1353 |
| weighted avg | 0.98 | 0.98 | 0.98 | 1353 |

el AUC del modelo es: 0.994441173006713

*Performance del modelo en categoría "visitar" incluyendo información del país.*

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 1.00 | 0.98 | 0.99 | 1213 |
| 1.0 | 0.84 | 0.97 | 0.90 | 140 |
| accuracy | | | 0.98 | 1353 |
| macro avg | 0.92 | 0.97 | 0.94 | 1353 |
| weighted avg | 0.98 | 0.98 | 0.98 | 1353 |

el AUC del modelo es: 0.9951065834412907

*Performance del modelo en categoría "visitar" excluyendo información del país.*

Como la performance del modelo se mantiene prácticamente igual con y sin esta variable, se decidió excluirla del modelo.

### 4.3.2 Performance de los modelos e interpretación de los resultados

Se corrieron los modelos de cada categoría y se analizó su performance en entrenamiento, validación y en el conjunto de testeo (*held out set*) en diversas métricas. Por otra parte, al ver los resultados, se analizó si conservar todas las categorías o descartar aquellas que no se pueden predecir con la precisión suficiente.

A continuación, se presenta la performance de cada categoría en su modelo correspondiente, ordenados de manera descendente por su score F1 en validación. Se aplicaron todas las transformaciones testeadas e hiperparámetros optimizados en cada categoría:

| | Validación | | | | | Held out set | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **Precisión** | **Recall** | **F1** | **AUC** | **Frec.** | **Precisión** | **Recall** | **F1** | **AUC** | **Frec.** |
| **Visitar** | 0.90 | 0.95 | 0.92 | 0.99 | 140 | 0.91 | 0.94 | 0.92 | 0.99 | 52 |
| **Piso** | 1.00 | 0.86 | 0.92 | 0.99 | 7 | 1.00 | 1.00 | 1.00 | 1.00 | 2 |
| **Expensas** | 0.94 | 0.88 | 0.91 | 0.98 | 33 | 0.91 | 0.91 | 0.91 | 0.99 | 11 |
| **Comunicación** | 0.85 | 0.84 | 0.85 | 0.97 | 146 | 0.82 | 0.87 | 0.84 | 0.97 | 53 |
| **Más Información** | 0.83 | 0.88 | 0.85 | 0.97 | 104 | 0.91 | 0.84 | 0.88 | 0.97 | 38 |
| **Disponibilidad** | 0.77 | 0.88 | 0.82 | 0.99 | 52 | 0.80 | 0.95 | 0.87 | 0.98 | 21 |
| **Fotos** | 0.86 | 0.75 | 0.80 | 0.95 | 8 | 0.25 | 0.33 | 0.29 | 0.72 | 3 |
| **Dirección** | 0.76 | 0.84 | 0.79 | 0.98 | 37 | 0.88 | 0.88 | 0.88 | 0.98 | 8 |
| **Requisitos** | 0.76 | 0.74 | 0.75 | 0.97 | 43 | 0.72 | 0.93 | 0.81 | 0.99 | 14 |
| **Permuta** | 0.70 | 0.78 | 0.74 | 0.97 | 9 | 0.50 | 0.75 | 0.60 | 0.99 | 4 |
| **Parking** | 1.00 | 0.56 | 0.71 | 0.96 | 9 | 1.00 | 1.00 | 1.00 | 1.00 | 1 |
| **Precio** | 0.68 | 0.71 | 0.69 | 0.97 | 48 | 0.67 | 0.70 | 0.68 | 0.96 | 20 |
| **Dormitorios** | 0.62 | 0.57 | 0.59 | 0.67 | 14 | 0.33 | 0.67 | 0.44 | 0.77 | 3 |
| **Apto profesional**[14] | 0.50 | 0.40 | 0.44 | 0.86 | 5 | | | | | **0** |
| **Crédito** | 0.36 | 0.56 | 0.43 | 0.79 | 9 | 0.33 | 0.40 | 0.36 | 0.81 | 5 |
| **Medidas**[14] | 0.36 | 0.38 | 0.37 | 0.93 | 13 | | | | | **0** |
| **Amoblada**[14] | 0.00 | 0.00 | 0.00 | 0.93 | 2 | | | | | **0** |

*Tabla 10: Resultados y frecuencia de la categoría positiva de los modelos por etiqueta en validación y held out set.*

Como puede apreciarse en la Tabla 10, la performance de los modelos fue muy dispar. Por un lado, las categorías "comunicación", "visitar", "dirección", "disponibilidad", "expensas", "requisitos" y "más información" tuvieron una performance buena en ambos conjuntos. En estos casos, no existieron diferencias muy significativas entre los resultados en validación y en el *held out set*, con la excepción de "dirección", en la que la performance en el segundo conjunto fue bastante superior a la performance en el primero. La causa de esto podría ser a que por la baja frecuencia de esta categoría en el conjunto de testeo (únicamente 8 apariciones) este resultado haya estado inflado por casualidad.

Por otra parte, las palabras utilizadas por cada modelo guardan relación con el campo semántico de cada categoría. Si tomamos, por ejemplo, categorías distintas a "comunicación" o "visitar", las cuales fueron analizadas exhaustivamente a lo largo de este trabajo, se pueden

[14] Para los casos de "medidas", "amoblada" y "apto profesional" no existían apariciones dentro del conjunto de Held Out, por lo que no tenía sentido calcular las métricas en el mismo.

encontrar tokens con un significado cercano al esperado entre las variables más importantes que utiliza un modelo para definir si un mensaje pertenece a una categoría o no.

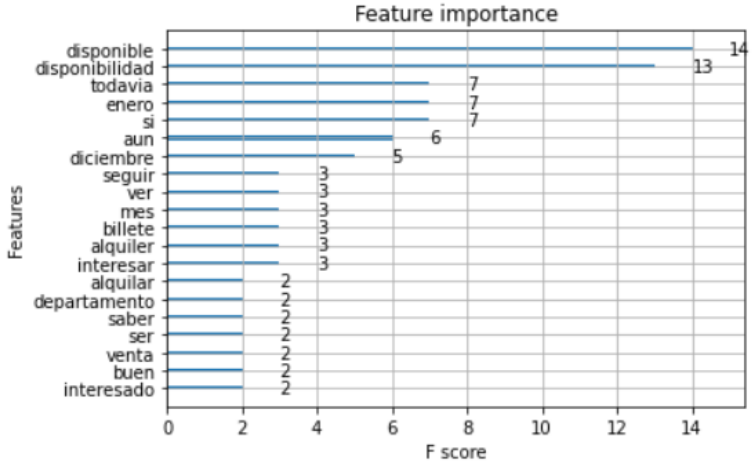



*Gráfico 14: Variables más importantes para la categoría "disponible"*

En el gráfico 14 puede verse cómo para la categoría "disponible" los tres tokens más importantes para definir si el mensaje pertenece a esa categoría son "disponible", "disponibilidad" y "todavía" que son tres palabras muy asociadas al campo semántico de la categoría. Más abajo pueden verse otras palabras como "aun" o "seguir" (el token lematizado de la palabra "sigue") que también están asociados a la etiqueta.

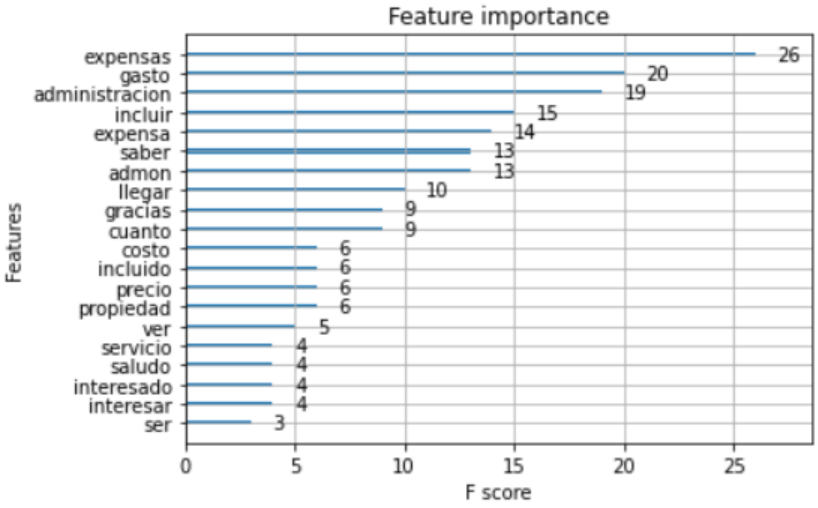



*Gráfico 15: Variables más importantes para la categoría "expensas"*

De modo similar, en el gráfico 15 pueden verse como para la categoría "expensas", varios de los tokens más importantes son palabras relacionadas con la categoría, como "expensas", "gastos", "administración" o "expensa".

En la tabla 10 también se muestran varias categorías que no tuvieron una buena performance: "crédito", "dormitorio", "medidas", "amoblada" y "apto profesional". Estas categorías tenían muy baja frecuencia: 39, 38, 27, 14 y 16, respectivamente. Esta puede ser una causa de su mala performance, ya que, el modelo no termina de tener suficientes datos como para aprender.

Asimismo, existe un conjunto de categorías en las cuales, si bien la performance fue buena, la frecuencia de apariciones no es lo suficientemente alta como para creer que los resultados son confiables. Este es el caso de las categorías "parking", "fotos"[15], "permuta" y "piso". En estos casos, los conjuntos de validación tenían 9, 8, 9 y 7 observaciones respectivamente para poder medir la performance del modelo. Por este motivo, podría ser que

[15] En el caso de "fotos", la buena performance se dio sólo en validación, reforzando que hay que tomar la buena performance de estas categorías de manera cuidadosa.

los resultados altos hayan sido azarosos y podría esperarse una performance más baja en datos reales.

# 5. Conclusiones

Un desafío importante de este trabajo fue la definición del problema junto a Properati. El desafío a abordar en este trabajo no fue un requerimiento directo del cliente, sino que fue el resultado del intercambio entre el tesista y dos representantes de la empresa en el que se postuló una necesidad de negocio que existía y se buscó la mejor manera de abordarlo. Para ello, una de las partes más importantes, tanto en términos de tiempo como de centralidad en el trabajo, fue el análisis de los mensajes y la comprensión de las etiquetas.

Por otra parte, nos sorprendió que la variable país no haya resultado significativa para los modelos analizados. Entendemos que esto podría deberse a que dentro de los países con los que trabaja la empresa, la forma de referirse a las categorías es lingüísticamente similar. Esta variable podría ser más importante si se compararan países con un lenguaje más diferente.

Asimismo, consideramos importante aclarar que la decisión de utilizar la métrica F1 en vez de AUC fue acertada. Cómo se ve en la tabla 10, en prácticamente todos los modelos el puntaje AUC dio muy alto (por encima de 0.9), sin embargo, esto no fue sinónimo de buena performance del modelo en todos los casos. La métrica para medir la calidad de un modelo debe siempre ser elegida en función del tipo de problema y de una decisión de negocio: En este caso, Properati está interesada en detectar los mensajes que pertenecen a cada una de las etiquetas definidas, por lo que el foco debe estar puesto en entender si el modelo logra predecir con precisión y exhaustividad la clase minoritaria, algo que la métrica F1 capta mejor que el AUC.

Por último, durante su realización surgieron varias limitaciones. En primer lugar, existe una distancia entre lo que el cliente deseaba como producto óptimo y lo que era técnicamente posible. Dada la distribución que naturalmente tienen los mensajes redactados por los usuarios, existen algunas etiquetas con muy baja frecuencia, por lo que es sumamente difícil entrenar a un modelo que pueda predecirlas con precisión: Es el caso de “crédito”, “dormitorio”, “medidas”, “amoblada” y “apto profesional”.

En segundo lugar, la poca disponibilidad de datos para validar los resultados, generaron que en otras categorías (“parking”, “fotos”, “permuta” y “piso”) presentaran resultados dudosamente altos, por lo que su buena performance debería ser tomada con cuidado. Al no tener un volumen alto de mensajes de estas categorías para poder verificar que el modelo esté clasificando bien, el buen resultado podría haber sido azaroso. Es importante destacar, sin embargo, que la última decisión sobre qué nivel de performance es lo suficientemente aceptable como para implementarse es una decisión de negocio. Es por este motivo que se corrieron todos los modelos y se indicó la performance alcanzada para que sea la empresa la que decida que etiquetas mantendrá y cuáles descartará en función de los resultados obtenidos.

En tercer lugar, existió una limitación desde el punto de vista técnico. Si bien los modelos de Bag of Words no son el estado del arte en la actualidad, ya que existen modelos más potentes basados en redes neuronales, como por ejemplo BETO, es importante destacar el buen resultado que tuvo en este caso. Algo a tener en cuenta cuando se aborda un problema de una empresa para la cual uno no trabaja, es que uno no conoce al equipo que se va a encargar de la implementación del proyecto. Los modelos Bag of Words, si bien tienen una performance por lo general inferior a modelos más avanzados, son mucho más fáciles de comprender en el caso que se requiera modificarlos, ya que están basados en modelos de *machine learning* más tradicionales (en este caso XGBoost). Al haber obtenido una performance lo suficientemente buena, entendemos que es preferible privilegiar un código fácil de abordar con una buena performance a otro difícil de abordar, pero con una performance algo mejor.

## Próximos pasos

A la luz de los análisis realizados y del modelo desarrollado, se identificaron algunos posibles puntos sobre los que se podría trabajar en el futuro para profundizar esta solución. En primer lugar, como se mencionó anteriormente, las categorías utilizadas en este trabajo fueron

construidas junto a personal de Properati en base al conocimiento que ellos tenían del negocio. Sin embargo, es posible que, con algún tipo de análisis de clustering, puedan encontrarse otras categorías de interés para las inmobiliarias. En particular, podrían utilizarse técnicas como Latent Dirichlet Allocation (LDA) para poder encontrar grupos de tokens que podrían aportar un valor que no había sido considerado.

Asimismo, uno de los problemas que se enfrentaron en este trabajo fue la poca disponibilidad de mensajes para algunas categorías que produjeron resultados de validación dudosamente altos en algunos casos ("parking", "fotos", "permuta" y "piso"). Por otra parte, en otros casos, ni siquiera fue posible entrenar un modelo con una performance aceptable ("crédito", "dormitorio", "medidas", "amoblada" y "apto profesional").

Una posible forma de abordar esta limitación podría ser mediante la utilización de modelos de aprendizaje semi supervisado. La ventaja de estos modelos es que pueden trabajar a partir de una pequeña cantidad de datos clasificados con un dataset más grande sin clasificar. Existen algunos ejemplos que utilizan redes adversarias, en los cuales el modelo puede aprender a clasificar también los mensajes de una determinada categoría que presenten una serie de perturbaciones, de esta manera, mejorando el aprendizaje.

Esto último es importante ya que este modelo fue entrenado con una muestra de los datos. Utilizar este tipo de técnicas puede ser muy útil para escalar los datos del modelo, que, a fines de realizar este trabajo, fue entrenado con una muestra pequeña de los mensajes que tiene la compañía a disposición. Utilizar una base mayor, en este sentido, podría mejorar los resultados.

# Anexo

## Listado de stopwords del paquete NLTK

A continuación, se listan las palabras que componen el listado de stopwords del paquete NLTK de Python, que fueron las eliminadas de los mensajes en el modelo.

{'a', 'al', 'algo', 'algunas', 'algunos', 'ante', 'antes', 'como', 'con', 'contra', 'cual', 'cuando', 'de', 'del', 'desde', 'donde', 'durante', 'e', 'el', 'ella', 'ellas', 'ellos', 'en', 'entre', 'era', 'erais', 'eran', 'eras', 'eres', 'es', 'esa', 'esas', 'ese', 'eso', 'esos', 'esta', 'estaba', 'estabais', 'estaban', 'estabas', 'estad', 'estada', 'estadas', 'estado', 'estados', 'estamos', 'estando', 'estar', 'estaremos', 'estará', 'estarán', 'estarás', 'estaré', 'estaréis', 'estaría', 'estaríais', 'estaríamos', 'estarían', 'estarías', 'estas', 'este', 'estemos', 'esto', 'estos', 'estoy', 'estuve', 'estuviera', 'estuvierais', 'estuvieran', 'estuvieras', 'estuvieron', 'estuviese', 'estuvieseis', 'estuviesen', 'estuvieses', 'estuvimos', 'estuviste', 'estuvisteis', 'estuviéramos', 'estuviésemos', 'estuvo', 'está', 'estábamos', 'estáis', 'están', 'estás', 'esté', 'estéis', 'estén', 'estés', 'fue', 'fuera', 'fuerais', 'fueran', 'fueras', 'fueron', 'fuese', 'fueseis', 'fuesen', 'fueses', 'fui', 'fuimos', 'fuiste', 'fuisteis', 'fuéramos', 'fuésemos', 'ha', 'habida', 'habidas', 'habido', 'habidos', 'habiendo', 'habremos', 'habrá', 'habrán', 'habrás', 'habré', 'habréis', 'habría', 'habríais', 'habríamos', 'habrían', 'habrías', 'habéis', 'había', 'habíais', 'habíamos', 'habían', 'habías', 'han', 'has', 'hasta', 'hay', 'haya', 'hayamos', 'hayan', 'hayas', 'hayáis', 'he', 'hemos', 'hube', 'hubiera', 'hubierais', 'hubieran', 'hubieras', 'hubieron', 'hubiese', 'hubieseis', 'hubiesen', 'hubieses', 'hubimos', 'hubiste', 'hubisteis', 'hubiéramos', 'hubiésemos', 'hubo', 'la', 'las', 'le', 'les', 'lo', 'los', 'me', 'mi', 'mis', 'mucho', 'muchos', 'muy', 'más', 'mí', 'mía', 'mías', 'mío', 'míos', 'nada', 'ni', 'no', 'nos', 'nosotras', 'nosotros', 'nuestra', 'nuestras', 'nuestro', 'nuestros', 'o', 'os', 'otra', 'otras', 'otro', 'otros', 'para', 'pero', 'poco', 'por', 'porque', 'que', 'quien', 'quienes', 'qué', 'se', 'sea', 'seamos', 'sean', 'seas', 'sentid', 'sentida', 'sentidas', 'sentido', 'sentidos', 'seremos', 'será', 'serán', 'serás', 'seré', 'seréis', 'sería', 'seríais', 'seríamos',

'serían', 'serías', 'seáis', 'siente', 'sin', 'sintiendo', 'sobre', 'sois', 'somos', 'son', 'soy', 'su', 'sus', 'suya', 'suyas', 'suyo', 'suyos', 'sí', 'también', 'tanto', 'te', 'tendremos', 'tendrá', 'tendrán', 'tendrás', 'tendré', 'tendréis', 'tendría', 'tendríais', 'tendríamos', 'tendrían', 'tendrías', 'tened', 'tenemos', 'tenga', 'tengamos', 'tengan', 'tengas', 'tengo', 'tengáis', 'tenida', 'tenidas', 'tenido', 'tenidos', 'teniendo', 'tenéis', 'tenía', 'teníais', 'teníamos', 'tenían', 'tenías', 'ti', 'tiene', 'tienen', 'tienes', 'todo', 'todos', 'tu', 'tus', 'tuve', 'tuviera', 'tuvierais', 'tuvieran', 'tuvieras', 'tuvieron', 'tuviese', 'tuvieseis', 'tuviesen', 'tuvieses', 'tuvimos', 'tuviste', 'tuvisteis', 'tuviéramos', 'tuviésemos', 'tuvo', 'tuya', 'tuyas', 'tuyo', 'tuyos', 'tú', 'un', 'una', 'uno', 'unos', 'vosotras', 'vosotros', 'vuestra', 'vuestras', 'vuestro', 'vuestros', 'y', 'ya', 'yo', 'él', 'éramos'}

## Resultados de optimización de hiperparámetros via Cross Validation

A continuación, se listan los resultados de la optimización de hiperparámetros por categoría vía Cross Validation.

| Direccion | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| scale_pos_weight | subsample | min_child_weight | max_depth | lambda | gamma | eta | colsample_bytree | alpha | mean_test_score |
| 40 | 0.9 | 1 | 4 | 0 | 0 | 0.3 | 0.8 | 1 | 0.877829 |
| 40 | 0.7 | 1 | 3 | 0 | 0 | 0.4 | 1 | 1 | 0.874526 |
| 40 | 1 | 3 | 4 | 1 | 0.2 | 0.3 | 0.7 | 0.5 | 0.870422 |
| 40 | 1 | 3 | 2 | 0 | 0.2 | 0.4 | 0.9 | 0 | 0.869564 |
| 40 | 0.9 | 3 | 1 | 1 | 0 | 0.1 | 0.7 | 0 | 0.864082 |

| Precio | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| scale_pos_weight | subsample | min_child_weight | max_depth | lambda | gamma | eta | colsample_bytree | alpha | mean_test_score |
| 25 | 1 | 3 | 2 | 0.5 | 0.2 | 0.3 | 0.7 | 1 | 0.852273 |
| 25 | 0.9 | 3 | 3 | 0 | 0.5 | 0.4 | 0.9 | 0.25 | 0.846141 |
| 25 | 0.7 | 1 | 3 | 0.5 | 1 | 0.1 | 1 | 0.1 | 0.838396 |
| 25 | 1 | 1 | 3 | 2 | 0.2 | 0.1 | 0.9 | 0 | 0.836663 |
| 25 | 0.8 | 5 | 4 | 1 | 0 | 0.2 | 0.9 | 0.5 | 0.836389 |

**Disponibilidad**

| scale_pos_weight | subsample | min_child_weight | max_depth | lambda | gamma | eta | colsample_bytree | alpha | mean_test_score |
|---|---|---|---|---|---|---|---|---|---|
| 25 | 0.9 | 1 | 1 | 1 | 0.2 | 0.4 | 0.9 | 0.5 | 0.930800 |
| 25 | 0.7 | 3 | 1 | 2 | 0.2 | 0.3 | 1 | 0.1 | 0.920058 |
| 25 | 1 | 5 | 1 | 0.5 | 0 | 0.2 | 0.7 | 0.25 | 0.919221 |
| 25 | 1 | 7 | 4 | 2 | 0.5 | 0.4 | 0.9 | 0.1 | 0.918292 |
| 25 | 0.7 | 3 | 3 | 2 | 0.5 | 0.4 | 0.7 | 0.1 | 0.918205 |

**Crédito**

| scale_pos_weight | subsample | min_child_weight | max_depth | lambda | gamma | eta | colsample_bytree | alpha | mean_test_score |
|---|---|---|---|---|---|---|---|---|---|
| 131 | 0.9 | 1 | 3 | 0.5 | 0.5 | 0.2 | 0.8 | 0 | 0.726956 |
| 131 | 1 | 5 | 4 | 2 | 0.2 | 0.3 | 0.7 | 0.25 | 0.688182 |
| 131 | 0.9 | 3 | 2 | 2 | 1 | 0.3 | 1 | 0.5 | 0.683141 |
| 131 | 1 | 5 | 4 | 0.5 | 1 | 0.1 | 0.7 | 1 | 0.638885 |
| 131 | 0.9 | 7 | 3 | 1 | 0 | 0.2 | 0.7 | 0.25 | 0.675533 |

**Dormitorios**

| scale_pos_weight | subsample | min_child_weight | max_depth | lambda | gamma | eta | colsample_bytree | alpha | mean_test_score |
|---|---|---|---|---|---|---|---|---|---|
| 130 | 0.9 | 1 | 2 | 0 | 1 | 0.3 | 0.9 | 0.1 | 0.767357 |
| 130 | 0.8 | 3 | 4 | 1 | 0.5 | 0.2 | 0.7 | 1 | 0.755689 |
| 130 | 0.8 | 7 | 3 | 0 | 1 | 0.1 | 0.8 | 1 | 0.749878 |
| 130 | 0.7 | 5 | 3 | 2 | 0.2 | 0.2 | 0.7 | 0.5 | 0.749482 |
| 130 | 0.8 | 1 | 1 | 0 | 0 | 0.2 | 0.8 | 0.1 | 0.745376 |

**Expensas**

| scale_pos_weight | subsample | min_child_weight | max_depth | lambda | gamma | eta | colsample_bytree | alpha | mean_test_score |
|---|---|---|---|---|---|---|---|---|---|
| 38 | 0.7 | 3 | 2 | 0 | 1 | 0.3 | 0.8 | 1 | 0.928689 |
| 38 | 0.7 | 3 | 4 | 2 | 0.2 | 0.2 | 0.7 | 0.5 | 0.926259 |
| 38 | 0.7 | 1 | 3 | 0.5 | 0 | 0.3 | 0.8 | 0.5 | 0.925675 |
| 38 | 0.9 | 5 | 1 | 0 | 1 | 0.2 | 0.7 | 0 | 0.924219 |
| 38 | 0.7 | 3 | 2 | 2 | 0 | 0.2 | 0.8 | 0.25 | 0.918039 |

**Parking**

| scale_pos_weight | subsample | min_child_weight | max_depth | lambda | gamma | eta | colsample_bytree | alpha | mean_test_score |
|---|---|---|---|---|---|---|---|---|---|
| 121 | 0.9 | 1 | 3 | 1 | 0 | 0.3 | 0.8 | 0.1 | 0.868965 |
| 121 | 0.7 | 1 | 4 | 2 | 1 | 0.4 | 0.8 | 0 | 0.844066 |
| 121 | 0.7 | 3 | 3 | 1 | 0.5 | 0.4 | 0.8 | 0.25 | 0.838854 |
| 121 | 0.8 | 7 | 4 | 1 | 0.5 | 0.4 | 0.8 | 0.5 | 0.820674 |
| 121 | 0.9 | 5 | 2 | 2 | 1 | 0.3 | 0.7 | 1 | 0.806332 |

**Medidas**

| scale_pos_weight | subsample | min_child_weight | max_depth | lambda | gamma | eta | colsample_bytree | alpha | mean_test_score |
|---|---|---|---|---|---|---|---|---|---|
| 166 | 1 | 1 | 3 | 0.5 | 0.2 | 0.1 | 0.7 | 1 | 0.537709 |
| 166 | 1 | 7 | 1 | 1 | 1 | 0.2 | 0.8 | 0.1 | 0.531708 |
| 166 | 1 | 5 | 1 | 2 | 1 | 0.2 | 0.8 | 0.1 | 0.531708 |
| 166 | 1 | 5 | 1 | 1 | 0.2 | 0.4 | 0.9 | 0.25 | 0.528623 |
| 166 | 0.9 | 1 | 1 | 0.5 | 0.5 | 0.3 | 0.7 | 0 | 0.528404 |

**Requisitos**

| scale_pos_weight | subsample | min_child_weight | max_depth | lambda | gamma | eta | colsample_bytree | alpha | mean_test_score |
|---|---|---|---|---|---|---|---|---|---|
| 30 | 0.8 | 1 | 4 | 0.5 | 0.2 | 0.2 | 1 | 1 | 0.902233 |
| 30 | 1 | 3 | 3 | 2 | 0 | 0.2 | 1 | 0.5 | 0.897668 |
| 30 | 0.7 | 3 | 4 | 1 | 0.2 | 0.3 | 0.7 | 0.1 | 0.892624 |
| 30 | 1 | 5 | 2 | 1 | 0.5 | 0.2 | 0.9 | 0.5 | 0.888475 |
| 30 | 1 | 1 | 1 | 0.5 | 0.2 | 0.4 | 0.8 | 0.25 | 0.888405 |

**mas informacion**

| scale_pos_weight | subsample | min_child_weight | max_depth | lambda | gamma | eta | colsample_bytree | alpha | mean_test_score |
|---|---|---|---|---|---|---|---|---|---|
| 12 | 0.9 | 3 | 3 | 1 | 0.5 | 0.4 | 0.7 | 0.5 | 0.933549 |
| 12 | 0.7 | 3 | 2 | 0.5 | 1 | 0.4 | 0.9 | 0.25 | 0.933353 |
| 12 | 0.9 | 1 | 1 | 2 | 0 | 0.1 | 0.9 | 0.5 | 0.933303 |
| 12 | 0.8 | 3 | 2 | 2 | 0.5 | 0.2 | 0.7 | 1 | 0.932103 |
| 12 | 0.7 | 3 | 3 | 2 | 1 | 0.3 | 0.7 | 1 | 0.931215 |

**Fotos**

| scale_pos_weight | subsample | min_child_weight | max_depth | lambda | gamma | eta | colsample_bytree | alpha | mean_test_score |
|---|---|---|---|---|---|---|---|---|---|
| 200 | 0.8 | 3 | 1 | 0 | 0.2 | 0.3 | 0.9 | 0.1 | 0.750668 |
| 200 | 0.9 | 1 | 2 | 2 | 0.5 | 0.3 | 0.9 | 0.5 | 0.737142 |
| 200 | 0.8 | 3 | 2 | 0 | 1 | 0.1 | 0.7 | 1 | 0.730037 |
| 200 | 0.8 | 5 | 1 | 0.5 | 1 | 0.4 | 0.7 | 0.25 | 0.726334 |
| 200 | 0.7 | 3 | 3 | 2 | 0.2 | 0.3 | 0.7 | 0 | 0.722617 |

**Permuta**

| scale_pos_weight | subsample | min_child_weight | max_depth | lambda | gamma | eta | colsample_bytree | alpha | mean_test_score |
|---|---|---|---|---|---|---|---|---|---|
| 170 | 1 | 1 | 1 | 2 | 1 | 0.3 | 1 | 0.1 | 0.877221 |
| 170 | 0.7 | 1 | 4 | 0.5 | 1 | 0.1 | 0.7 | 0.25 | 0.830765 |
| 170 | 0.7 | 1 | 3 | 0.5 | 1 | 0.1 | 0.9 | 0.25 | 0.830765 |
| 170 | 0.9 | 5 | 1 | 0 | 1 | 0.3 | 0.8 | 0.1 | 0.815871 |
| 170 | 0.7 | 3 | 4 | 1 | 0.5 | 0.2 | 0.7 | 0.1 | 0.803670 |

| Amoblada | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| scale_pos_weight | subsample | min_child_weight | max_depth | lambda | gamma | eta | colsample_bytree | alpha | mean_test_score |
| 640 | 0.7 | 5 | 3 | 1 | 0 | 0.3 | 0.9 | 0.5 | 0.777540 |
| 640 | 0.7 | 1 | 4 | 0.5 | 0.2 | 0.4 | 0.9 | 0 | 0.777540 |
| 640 | 0.7 | 5 | 4 | 1 | 1 | 0.4 | 0.8 | 0 | 0.721746 |
| 640 | 1 | 3 | 2 | 0 | 0 | 0.4 | 1 | 0.1 | 0.666190 |
| 640 | 1 | 5 | 3 | 1 | 0 | 0.4 | 0.9 | 0 | 0.666190 |

| Piso | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| scale_pos_weight | subsample | min_child_weight | max_depth | lambda | gamma | eta | colsample_bytree | alpha | mean_test_score |
| 165 | 0.8 | 5 | 2 | 0.5 | 1 | 0.4 | 0.7 | 0 | 0.889656 |
| 165 | 0.7 | 5 | 1 | 0.5 | 0.5 | 0.1 | 0.9 | 0.25 | 0.889656 |
| 165 | 0.7 | 7 | 1 | 0.5 | 0.2 | 0.1 | 1 | 0.5 | 0.889656 |
| 165 | 1 | 3 | 1 | 0 | 0.5 | 0.2 | 1 | 0.1 | 0.889656 |
| 165 | 0.9 | 7 | 2 | 2 | 0.5 | 0.2 | 1 | 1 | 0.889656 |

| Apto profesional | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| scale_pos_weight | subsample | min_child_weight | max_depth | lambda | gamma | eta | colsample_bytree | alpha | mean_test_score |
| 280 | 1 | 3 | 3 | 0.5 | 0.5 | 0.2 | 0.7 | 1 | 0.596291 |
| 280 | 1 | 5 | 1 | 0 | 0.5 | 0.4 | 0.8 | 1 | 0.581427 |
| 280 | 0.7 | 1 | 4 | 0.5 | 0.2 | 0.1 | 1 | 0.25 | 0.572961 |
| 280 | 1 | 1 | 4 | 0 | 0.2 | 0.3 | 1 | 0 | 0.572722 |
| 280 | 0.8 | 5 | 3 | 0.5 | 0.5 | 0.4 | 1 | 0 | 0.565155 |

# Bibliografía

Bird, Steven, Klein, Ewan, Loper, Edward (2009), *Natural Language Processing with Python*, O'Reilly, Sebastopol.

Cohen, Jacob (1960), "A coefficient of agreement for nominal scales." *Educational and psychological measurement,* 20.1: 37-46.

Goadrich, Mark (2006), "The Relationship Between Precisión-Recall and ROC Curves", *Proceedings of the 23rd International Conference on Machine Learning,* Association for Computing Machinery.

Goldberg, Yoav (2017), *Neural Network Methods for Natural Language Processing*, Morgan & Claypool Publishers, Toronto.

James, Gareth, Witten, Daniela, Hastie, Trevor, Tibshirani, Robert (2017), *An Introduction to Statistical Learning*, Springer, New York.

Jurafsky, Daniel, Martin, James H. (2020), *Speech and Language Processing: An Introduction to Language Processing, Computational Linguistics, and Speech Recognition*, Pearson Prentice Hall, New Jersey.

Rabiner, Laurence y Juang, Biing-Hwang (2007), “Historical Perspective of the field of ASR/NLU” en Benesti, Jacob, Mohan Sondhi, M., Huang, Yiteng (2007), *Springer Handbook of speech processing*, Springer, Berlin.

Roukos, Salim (2007), “Natural Language Understanding” en en Benesti, Jacob, Mohan Sondhi, M., Huang, Yiteng (2007), *Springer Handbook of speech processing*, Springer, Berlin.