Acámica

Data Science

Informe Final – Proyecto N°4

Ortolani Juan Martín

Marzo 2021

Objetivo

El objetivo de este proyecto será profundizar y mejorar la resolución de la anterior entrega N°3.

A partir del desarrollo presentado por mis compañeros en las demos, el material disponibilizado por Acámica y profundizando con bibliografía extra identifiqué algunos ajustes que podía añadir a mi proyecto.

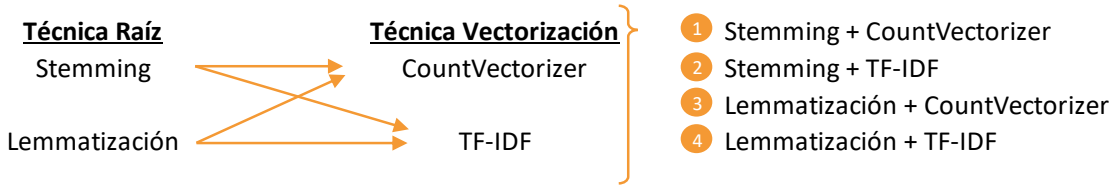
Para ubicarnos en contexto, el proyecto anterior se organizaba en tres partes:

- PARTE A: Exploración y Transformación de datos
- PARTE B: Modelos de Machine Learning
- PARTE C: Investigación/Interpretación de los datos

Para cada una de ellas encontré nuevas ideas/hipótesis que podrían traer mejoras:

- PARTE A: Análisis de bigramas y trigramas para verificar consistencia en el preprocesamiento.
- PARTE B: Hiperoptimización utilizando la librería Scikit-Optimize.
  Como pudimos comprobar, el tiempo de ejecución de la técnica de hiperoptimización seleccionada anteriormente es excesivamente elevado. Investigando en la web encontré la librería *skopt* que permite trabajar con optimización bayesiana, reduciendo considerablemente los tiempos de ejecución con resultados iguales o mejores a la técnica utilizada anteriormente.
- PARTE C: Terminamos la parte C del proyecto 3 con dos conclusiones:
  1. Los modelos presentan dificultad para distinguir los valores medios del rango de calificación (1 a 5).
  2. La combinación entre técnicas de vectorización y tokenización de mejor performance es Stemmización + TF-IDF.

Recordemos que en el proyecto 3 elaboramos un intensivo análisis para cada una de las combinaciones siguientes:



La conclusión 1 será el punto de partida para el proyecto 4. Aplicaremos la siguiente reclasificación para ver si arroja mejores resultados.

STARS	CALIFICACIÓN
1 2	MALO
3	REGULAR
4 5	BUENO

La conclusión 2 nos ahorrará tiempo cuando comencemos con el desarrollo de nuestros modelos de ML.

Carga del Dataset y Preprocesamiento de los Datos

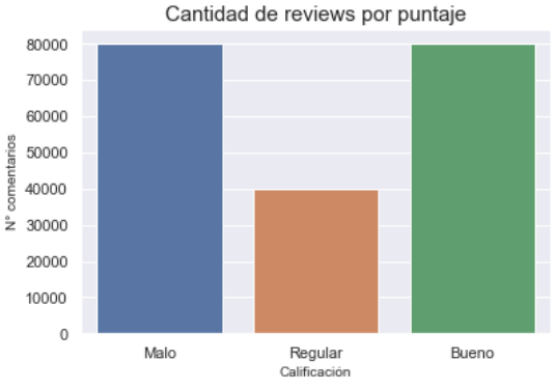
Las librerías utilizadas para la carga de datos y análisis exploratorio son Numpy, Pandas, Matplotlib, Seaborn, NLTK, re y Scikit-Learn.

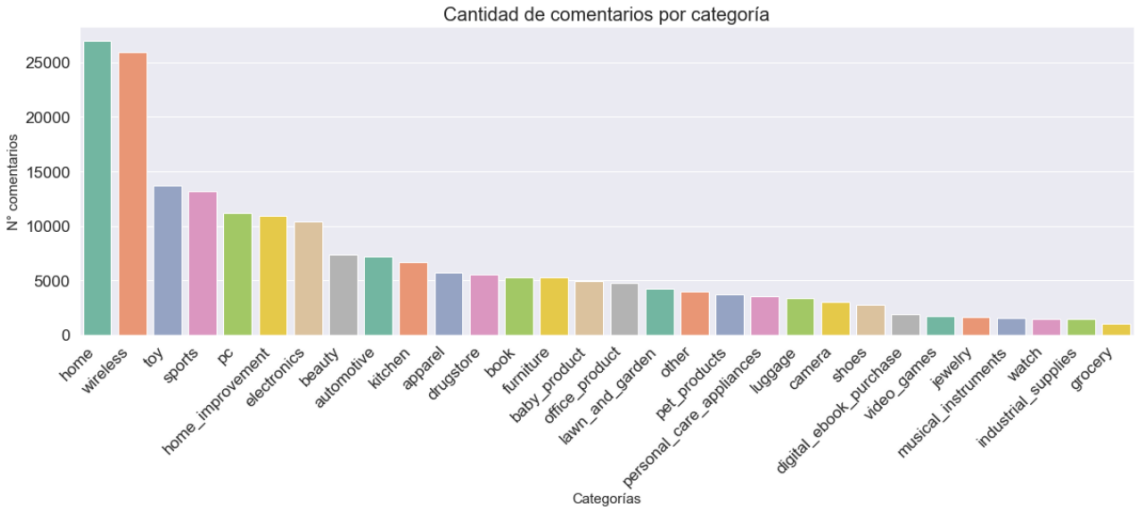
El dataset consta de 200.000 filas y 8 columnas que son:

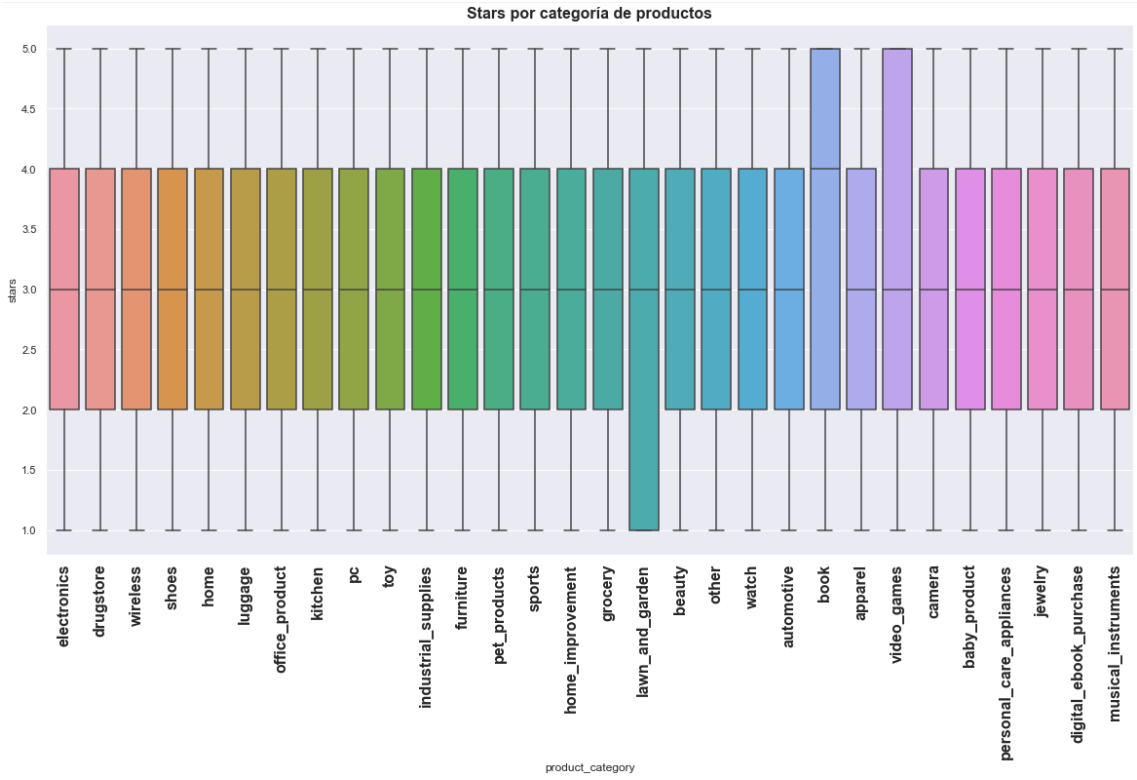
- review_id: N° de identificación de revisión.
- product_id: N° de identificación de producto.
- reviewer_id: N° de identificación de usuario.
- stars: Estrellas asignadas al producto por el usuario.
- review_body: Comentario.
- review_title: Título del comentario.
- language: Idioma del comentario.
- product_category: Categoría de producto.

Lo primero que hacemos es verificar que tipo de datos presentan las columnas para ver si es necesario hacer conversiones o no, luego verificamos el idioma de los comentarios para ver si tenemos más de un idioma en el dataset y finalmente buscamos si hay filas repetidas.

Una vez concluidos estos primeros pasos de preprocesamiento añadimos una columna llamada “calificacion” con los nuevos valores que mencionamos anteriormente y ploteamos algunos gráficos para entender mejor el conjunto de datos.







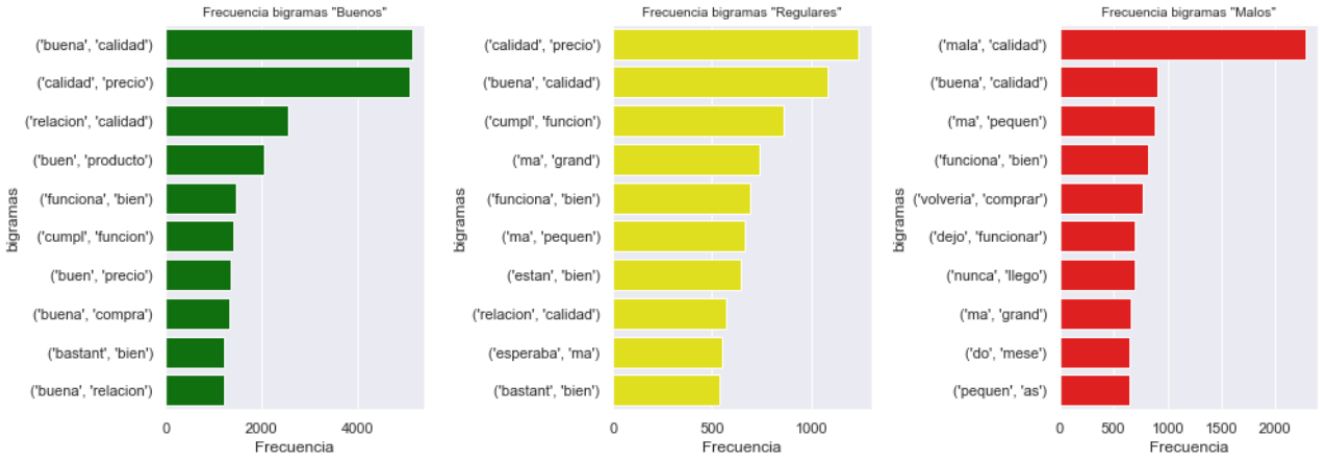
Continuando con el preprocesamiento importamos la librería re y normalizamos el dataset reemplazando los caracteres que no sean letras por espacios.

A continuación, se procede a tokenizar y filtrar stopwords utilizando la librería NLTK para luego aplicar Stemmización. El Stemming es un proceso eurístico que recorta la terminación de las palabras agrupándolas por su raíz. Como mencionamos en la introducción, ya vimos en el proyecto N° 3 que es la técnica que mejor resultados da.

Con los pasos anteriores ya estaríamos en condiciones de vectorizar y comenzar a entrenar modelos de ML, pero antes, vamos a realizar un análisis de n-gramas.

Los n-gramas son básicamente la unión de uno o varios tokens o caracteres vecinos. Dicho análisis es muy interesante ya que difícilmente las críticas puedan resumirse en unidades semánticas individuales, menos aún en el idioma español. Creando esta unión de tokens podré agregar cuerpo a la crítica para entender más en detalle la intención de las mismas y verificar la consistencia de lo desarrollado hasta el momento.

Bigrams



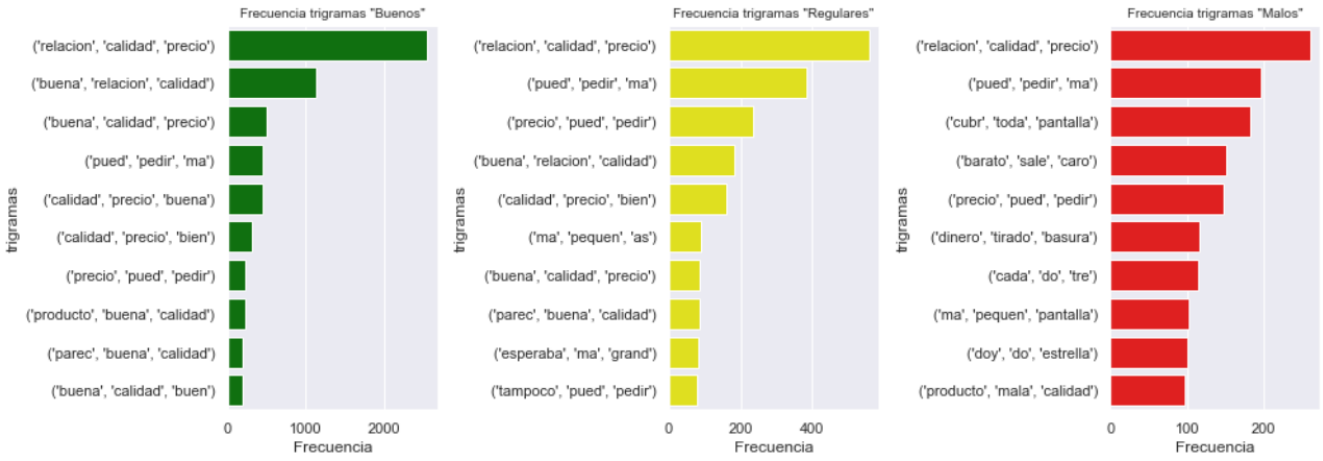
Encuentro coherencia en los brigramas.

Si bien hay varias cadenas que tal vez se repitan, podemos ver diferencias en la frecuencia. Por ejemplo, para el caso de las críticas "Buenas" vemos una clara predominancia de las cadenas 'calidad-precio' y 'buena-calidad'. Además, se observa que en la mayoría tenemos la palabra 'buen' o 'bien' lo cual podría decirse que es correcto.

Respecto de los bigramas "Regulares", continúa siendo la calificación más confusa de las tres pero también mantiene coherencia. Aquí ya vemos que la frecuencia es más pareja y aparece menos veces la palabra 'bien' o 'buen'.

Finalmente, si vemos los bigramas "Malos" vemos que la frecuencia también está más distribuída pero tiene más peso la cadena 'mala-calidad'.

Trigrams



Los trigramas mantienen la coherencia que ya vimos anteriormente.

Algunos llaman particularmente la atención ya que son bien característicos de la calificación y refuerzan lo que vimos en los bigramas. Por ejemplo:

- 'tampoco-puedo-pedir' → "Regular"
- 'barato-sale-caro' → "Malo"
- 'peor-compra-hecho' → "Malo"
- 'producto-nunca-llego' → "Malo"

Vectorización

Volviendo una vez más a lo mencionado en la introducción, dijimos que parte de las conclusiones del proyecto N°4 eran que la técnica de vectorización de mejor performance era TF-IDF.

TF-IDF significa "Term Frequency-Inverse Document Frequency"

→ Term Frequency: Cuántas veces un término aparece en un documento. La importancia de esto es que se supone que mientras más veces aparece, más importante es ese término en ese documento.

Sin Embargo, hay términos (como los artículos "la", "el", o las preposiciones "a", "por") que tienen alta frecuencia en todos los textos. Si les asignamos un puntaje alto, no ganaremos información con respecto a cuál es el tópico del documento.

Aquí participa la segunda parte de la palabra:

→IDF (Inverse Document Frequency): Esto significa que se compara la frecuencia de la palabra en ese documento con la frecuencia en todo el corpus de documentos. A mayor frecuencia de la palabra en todos los documentos, menor puntaje tendrá. Así evitamos puntuar alto a stop words o palabras que no agregan información sobre ese documento específico.

Modelado

Ahora si ya estamos en condiciones de comenzar con los modelos.

Utilizaremos los mismos modelos que se utilizaron en el proyecto N°3 (Random Forest, XGBoost y SVC) y como dijimos en la introducción utilizaremos la librería skopt para optimizaciones.

Scikit-Optimize es una librería open-source basada en Scipy, NumPy y Scikit-Learn que proporciona un conjunto de herramientas pueden utilizarse para el ajuste de hiperparámetros.

Skopt precisa que sean definidas cuatro características:

1) Espacio: Puede contener una o varias dimensiones y existen diferentes funciones para definirlo. En este caso utilizaremos:

- Integer: Dimensión de espacio de búsqueda que puede tomar valores enteros.
- Categorical: Dimensión del espacio de búsqueda que puede adoptar valores categóricos.

2) BayesSearchCV: La clase BayesSearchCV proporciona una interfaz similar a GridSearchCV o RandomizedSearchCV pero realiza una optimización bayesiana sobre hiperparámetros. A diferencia de GridSearchCV, no se prueban todos los valores de los parámetros, sino que se muestrea un número fijo de configuraciones de parámetros a partir de las distribuciones especificadas. El número de ajustes de parámetros que se prueban viene dado por n_iter.

3) Función objetivo: Esta es una función que será llamada por el procedimiento de búsqueda, recibe valores de hiperparámetros como entrada del espacio de búsqueda y devuelve la pérdida (cuanto menor, mejor). Esto significa que durante el proceso de optimización, entrenamos el modelo con valores de hiperparámetros seleccionados y predecimos la característica

objetivo y luego evaluamos el error de predicción y se lo devolvemos al optimizador. El optimizador decidirá qué valores comprobar y volver a iterar.

4) Optimizador: es la función que realiza el proceso de optimización de hiperparámetros bayesianos. La función de optimización itera en cada modelo y el espacio de búsqueda para optimizar y luego minimiza la función objetivo. Hay diferentes funciones de optimización proporcionadas por la biblioteca scikit-Optimize, en este caso utilizaremos gp_minimize (optimización bayesiana mediante procesos gaussianos).

Habiendo aplicado el proceso de hiperoptimización para cada modelo podemos ver que el de mejor performance es el SVC con un accruacy de 64,1% cuando sus hiperparámetros toman los valores kernel=”rbf” y C=7.

Conclusión

Como principales conclusiones podemos decir que todas las hipótesis planteadas al comienzo son verdaderas:

A. Una nueva clasificación ayudará a obtener mejores resultados ✓
   - Mejor Acc Anterior: 0.405
   - Mejor Acc Actual: 0.641

B. La nueva técnica de hiperoptimización demorará menos tiempo ✓
   - Tiempo Anterior: `Wall time: 1h 34min 36s`
   - Tiempo Actual: `Wall time: 14min 49s`

C. Nueva hiperoptimización puede arrojar resultados iguales o mejores ✓
   - Mejor Acc Anterior: 0.405
   - Mejor Acc Actual: 0.641

D. Bigrams y Trigrams para verificar consistencia ✓
   - Fue un análisis muy enriquecedor y efectivamente nos permitió comprobar consistencia.