Reporte Proyecto 1
Inteligencia de Negocios - ISIS3301
Departamento de Sistemas y Computación
2021-20

Revisión de sentimientos en comentarios de libros

Integrantes:
Cristhian Forigua Díaz
Natalia Sanabría
Jorge Esguerra

Institución:
Universidad de los Andes

Fecha:
3 de octubre de 2021

Contenido

Comprensión del negocio y enfoque analítico

El negocio de venta de libros a través de la tienda de Amazon fue uno de los factores que propulsó a la empresa al éxito. Actualmente, Amazon tiene el servicio de Kindle, donde los lectores pueden comprar y leer libros en formatos digitales.

Desde hace años, los productos en Amazon Kindle pueden ser calificados y retroalimentados por los usuarios que los han comprado y han consumido su contenido. Esta información es crucial para el negocio de Amazon, puesto que le gustaría hacerse con los derechos de comercialización de libros con buena reputación dentro de sus clientes para impulsar sus ventas y mantenerse como líderes en la distribución y comercialización de libros. Esta información también es de interés para entidades como editoriales que, de tener una comprensión y un análisis sobre las reseñas de los usuarios, podrían buscar tener contacto con los autores de los libros que han tenido ventas y han sido calificados positivamente en la plataforma de Amazon Kindle. Finalmente, estos reviews también son de utilidad para los usuarios interesados en alguno de los libros.

Por lo anterior, hemos visto la oportunidad de crear una serie de modelos de Machine Learning para aprovechar los datos que se tienen de libros Kindle reseñados, buscando generar una herramienta que pueda ser de uso para entidades como Amazon y las editoriales y que pueda usarse para los objetivos de negocio que ellos consideren pertinentes.

Objetivos de negocio

- Clasificar productos del Kindle Store según sus reseñas, buscando determinar si los libros han tenido un buen o mal recibimiento por parte del público.

Para lograr lo anterior, es importante tener en cuenta los siguientes objetivos técnicos.

- Aplicar técnicas de machine learning para realizar un análisis sobre lenguaje natural y poder identificar sentimientos positivos y negativos a partir de texto plano.
- Realizar un proceso de preprocesamiento y preparación de los datos para los modelos de machine learning.
- Realizar un proceso de descripción y análisis de los datos para entender el contexto del problema.
- Realizar un análisis detallado de los resultados obtenidos por los algoritmos de machine learning y con base a ellos proporcionar recomendaciones al negocio.

Tarea de analítica de datos - Machine Learning

Según el contexto del problema y la base de datos seleccionada, se propone solucionar el problema como una tarea de clasificación, en donde, a partir de un texto plano en lenguaje natural, se pueda transformar a una representación de bolsa de palabras y pueda ser interpretado por un clasificador. La base de datos originalmente viene con puntajes del 1 al 5 para cada comentario. Sin embargo, hemos decidido transformar la variable de un problema de clasificación multiclase a un problema de clasificación binaria. Para esto, una reseña con un puntaje igual o inferior a 3, se asocia con un

sentimiento negativo (0), mientras que los puntajes mayores a 3 con un sentimiento positivo (1). Por lo anterior, la tarea se define como un problema de clasificación binario sobre muestras de texto plano.

Criterios de éxito

Teniendo en cuenta el contexto planteado, se establece como criterio de éxito un rendimiento mayor al 65% para cada clase. Ya que el problema que se presentará al final se resuelve mediante una clasificación binaria de textos, una selección aleatoria tendría un rendimiento del 50%. Por lo anterior, un rendimiento mayor al 65% ya es un modelo mejor que el azar. Así mismo, es importante el tiempo de respuesta de los modelos como criterio de éxito. Para este criterio, se establece que un modelo que genera una predicción para una nueva entrada en menos de 2 segundos es exitoso.

Tabla Requerimiento de Negocio - Requerimiento de ML.

Oportunidad o Problema de Negocio	Entender cuáles de los productos que están siendo ofrecidos reciben buenas calificaciones de los usuarios respecto a los que no.	
Descripción del requerimiento desde el punto de vista de aprendizaje de máquina	Para una entrada en texto plano en lenguaje natural X, se quiere encontrar una etiqueta y que corresponde a si el comentario refleja un sentimiento negativo ($y = 0$) o un sentimiento positivo ($y = 1$).	
Detalles de la actividad de minería de datos		
Tarea	**Técnica**	**Algoritmo e hiper-parámetros utilizados (con la justificación respectiva)**
Preprocesamiento de los datos	Libreria nltk	Se realizaron algoritmos de limpieza de los datos, tokenización y normalización para encontrar los espacios de representación.
Búsqueda del espacio de representación	Bag of Words (BoW)	Se utilizaron algoritmos de codificación como el Tfidf Vectorizer y Count Vectorizer. Se seleccionaron estos 2 algoritmos ya que representan la frecuencia de las palabras dentro de un texto lo cual puede proveer mayor información sobre los comentarios. Así mismo, se utilizaron todos los hiper parámetros por defecto en las implementaciones de sklearn.
Clasificación	Regresión Logística	

Clasificación	Naive Bayes Classifier	MultinomialNB es un clasificador que funciona muy bien con espacios de representación discretos, como es el caso de la técnica de bolsa de palabras. Se probarán hiper-parámetros como el *alpha,* el cual es un parámetro adaptativo para la suavidad y *fit_prior,* el cual determina si debe aprender o no las probabilidades previas de las clases.
Clasificación	Random Forest Classifier	Random Forest es un algoritmo que puede usarse como clasificador o regresor, en este caso se usa para la tarea de clasificación. Los hiperparámetros para este modelo son: -max_depth, la profundidad máxima de cada uno de los árboles de decisión que componen al bosque. Asignamos 15 como profundidad máxima dado que la infraestructura de cómputo donde entrenamos los modelos no permitió aumentar el parámetro más, so pena de demorarse varios días el cálculo del modelo. Gridsearch determinó que el modelo con profundidad 15 es mejor que con profundidad 8. - n_estimators: La cantidad de árboles que componen al bósque. El valor óptimo fue hallado por el método de GridSearch, siendo 100. - max_features: Cantidad de features diferentes que cada árbol puede utilizar para realizar el proceso de separación de los datos. Según gridsearch, el valor óptimo es de sqrt(#features).

Comprensión de los datos

Los datos provienen de Kaggle, (Amazon Reviews: Kindle Store Category). Este dataset es un conjunto de reseñas de libros que se venden mediante Amazon Kindle Store.

El dataset es una tabla, formato csv, donde cada fila es una reseña con múltiples metadatos y datos de cada reseña. No obstante, para este proyecto, las columnas relevantes de los datos son:
overall: calificación otorgada al producto en la reseña. Valor numérico de 1 a 5.

reviewText: El texto plano de la reseña. El dataset contiene 983.000 entradas de reseñas de diversos libros. Los datos contienen reseñas desde Mayo de 1996 hasta Julio de 2014. Cada libro reseñado tiene al menos 5 reseñas, y cada usuario que tiene reseñas en el dataset tiene al menos 5 de ellas.

Perfilamiento de los datos:

En primera instancia, se realizó un perfilamiento de los datos para tener una primera idea de estos. En un inicio, la base de datos cuenta con **10 columnas** y **982619 filas.** Dentro de las variables, se encuentra un puntaje de 1 a 5 asociado a cada comentario, un resumen del comentario y otros datos relacionados con las fechas y usuarios que realizaron los comentarios. Sin embargo, las únicas variables útiles para la solución del problema son el comentario, su puntaje y su resumen. En la figura 1 se muestra un análisis inicial de los datos numéricos y de las columnas. Se puede observar que para la variable del puntaje ("overall") se tienen únicamente números del 1 al 5, por lo que no hay valores atípicos.

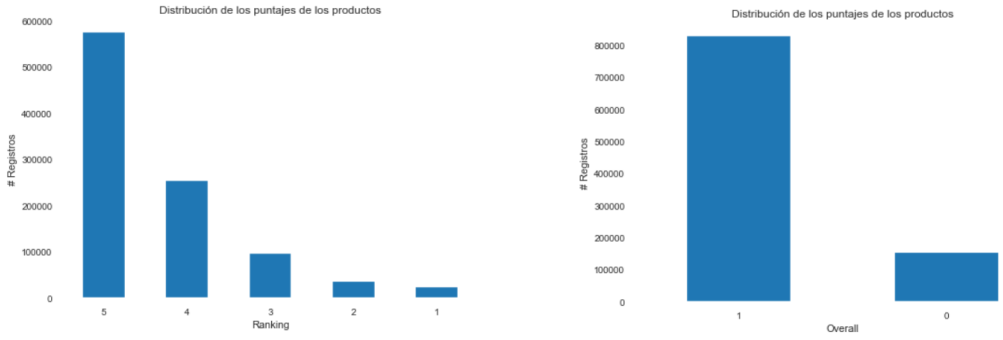
	Unnamed: 0	overall	unixReviewTime
count	982619.000000	982619.000000	9.826190e+05
mean	491309.000000	4.347801	1.378183e+09
std	283657.816418	0.955056	2.298476e+07
min	0.000000	1.000000	9.522144e+08
25%	245654.500000	4.000000	1.365206e+09
50%	491309.000000	5.000000	1.383523e+09
75%	736963.500000	5.000000	1.396051e+09
max	982618.000000	5.000000	1.406074e+09

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 982619 entries, 0 to 982618
Data columns (total 10 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   Unnamed: 0      982619 non-null  int64
 1   asin            982619 non-null  object
 2   helpful         982619 non-null  object
 3   overall         982619 non-null  int64
 4   reviewText      982597 non-null  object
 5   reviewTime      982619 non-null  object
 6   reviewerID      982619 non-null  object
 7   reviewerName    978803 non-null  object
 8   summary         982618 non-null  object
 9   unixReviewTime  982619 non-null  int64
dtypes: int64(3), object(7)
memory usage: 75.0+ MB
```

Figura 1. Entendimiento de las variables de la base de datos

Adicionalmente, se realizó un análisis de la completitud de los datos. Se encontró que la variable de "*ReviewerName*" tenía 3816 valores nulos. Sin embargo, como esta variable no aporta información relevante, se eliminó del registro de datos. Así mismo, se encontraron 22 registros con comentarios nulos y 1 registro que no tenía resumen. Para los casos anteriores, se eliminaron los registros dichos valores nulos.

Finalmente, se estudió la distribución de la variable objetivo, "*overall*". Dichos resultados se muestran en la figura 2. Se puede ver que en la representación original los datos están altamente desbalanceados, siendo 5 el puntaje mayoritario frente a las demás clases.



Representación original Representación Binaria

Figura 2. (Izq) Distribución de los datos en su representación original. (Der) Representación de los datos en su representación binaria.

Como se quiere analizar los sentimientos a través de los comentarios, se realizó una transformación a la variable de salida para que tuviera sentido con el contexto de negocio. Ahora, en lugar de identificar la puntuación del comentario, se quiere identificar si existe un sentimiento negativo (0) o positivo (1) asociado a dicho comentario. Con la transformación anterior, el problema pasa de ser un problema multiclase a un problema de clasificación binaria. En la Figura 1 se puede observar la distribución para la etiqueta transformada. Sin embargo, se observa que el desbalance de clases persiste aún la representación binaria.

Preprocesamiento de los datos:

En primera instancia, los comentarios pasaron por un proceso de limpieza en donde se eliminan los carácteres que no son ASCII, se remueven los signos de puntuación, se reemplaza la representación numérica de los números por su versión en caracteres, se estandarizan a minúscula y se eliminan todas las "*stopwords*", palabras que no tienen un significado importante, de los comentarios. Posteriormente, se realizó un proceso de tokenización con los textos sin contracciones. Al final de este proceso, cada comentario se representaba como una lista de las palabras con un significado semántico en el idioma. Finalmente, cada una de estas palabras pasó por un proceso de normalización a través de 2 técnicas.

Una vez realizado el preprocesamiento, las palabras deben ser traducidas al espacio de representación para que puedan ser interpretadas por los modelos. Para el caso de este proyecto, se utilizó el espacio de representación de "*Bag of Words (BoW)*". Sin embargo, para obtener dicho espacio de representación, se utilizaron 2 codificadores diferentes. Utilizamos las implementaciones de *CountVectorizer* y el *TfidfVectorizer* para codificar el espacio de representación. Fueron estas representaciones las que fueron utilizadas para entrenar los modelos de clasificación.

Como el problema de clasificación binaria sigue siendo un problema desbalanceado, utilizamos la estrategia de "*under-sampling*" para disminuir la brecha entre las clases. Con este propósito, muestreamos los datos para que exista una relación 2:1 entre la clase mayoritaria y la minoritaria. Con este nuevo segmento de datos se entrenaron los modelos.

3. Modelado y evaluación

3.1 MultinomialNB

La selección de este modelo se fundamenta en la representación de los datos. Este modelo es bueno para trabajar con datos en representación categórica, como es el caso del conteo de palabras. Gracias a esto, este clasificador tiene un alto potencial para procesar el lenguaje natural.

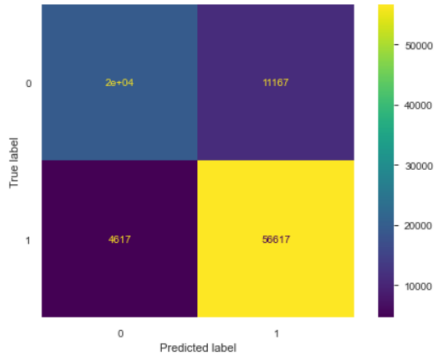
El modelo de multinomialNB utiliza el algoritmo de Bayes para datos que están distribuidos multinominalmente. Dentro del contexto de análisis de lenguaje natural, con un vocabulario de tamaño k y y clases, la distribución se parametriza con un conjunto de vectores $\theta_y = (\theta_{y1}, \dots, \theta_{yk})$, donde θ_{yi} es la probabilidad $P(x_i|y)$ de la i-ésima

característica. Los parámetros θ_y son estimados a través de una versión suavizada de la máxima verosimilitud de la forma:

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n}$$

Donde N_{yi} es la cantidad de veces que aparece la característica i en una muestra de la clase y en un conjunto de entrenamiento y N_y es el total de veces de todas las características de la clase y [1].

El modelo de MultinomialNB obtuvo una exactitud del 86% y 83% sobre los conjuntos de entrenamiento y test respectivamente. Lo anterior indica que el modelo no se sobreajustó a los datos. Así mismo, en la figura 3 se puede observar la matriz de confusión y el rendimiento por clase del modelo. Se puede observar que el modelo tiene un rendimiento inferior para la clase 0, lo cual se debe al desbalance en esta clase. Sin embargo, con el muestreo propuesto, se alcanza un rendimiento aceptable para ambas clases. Así mismo, los valores de precisión y cobertura son favorables, lo que indica que el modelo, lo que identifica, lo hace correctamente y su proporción con respecto a lo que debería identificar es alta. Sin embargo, la cobertura de la clase es 0, indicando que el modelo no detecta tan bien todo lo que debería identificar.



Exactitud sobre entrenamiento: 0.86
Exactitud sobre test: 0.83

	precision	recall	f1-score	support
0	0.81	0.64	0.71	30771
1	0.84	0.92	0.88	61234
accuracy			0.83	92005
macro avg	0.82	0.78	0.80	92005
weighted avg	0.83	0.83	0.82	92005

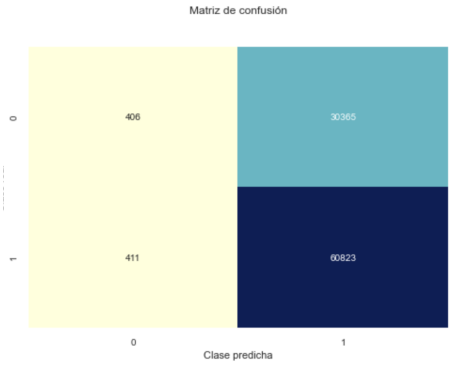
Figura 3. Resultados del modelo multinomialNB.

Así mismo, para llegar al modelo presentado anteriormente, se utilizó la técnica de GridSearch y se utilizaron diferentes valores para la fuerza del grado de suavidad y si el modelo predecía o no las probabilidades previas de las clases.

3.2 Logistic Regression

Una regresión logística es un método estadístico que permite hacer una clasificación binaria a través de una función logística (o función sigmoide). Dicha función está definida por la siguiente ecuación:

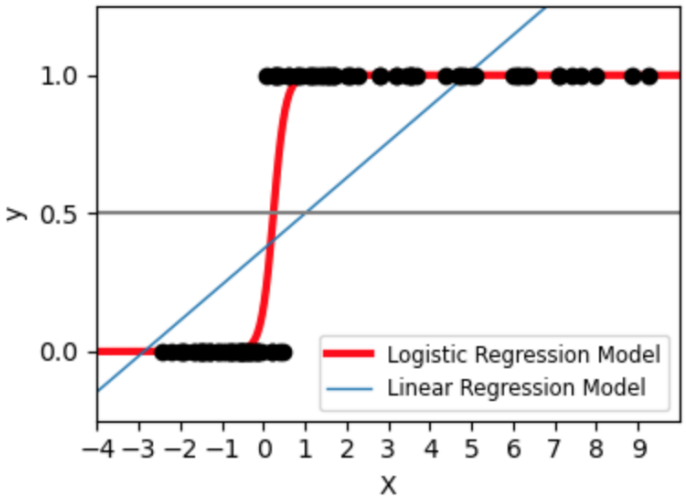
$$f(x) = 1/1 + e^{-x}$$



	precision	recall	f1-score	support
0	0.01	0.50	0.03	817
1	0.99	0.67	0.80	91188
accuracy			0.67	92005
macro avg	0.50	0.58	0.41	92005
weighted avg	0.98	0.67	0.79	92005

Figura 4. Resultados del modelo Logistic Regression

Como se puede ver, la regresión logística logró una exactitud del 67% sobre el conjunto de prueba, pero como se puede apreciar, se presentaron demasiados falsos positivos. Asimismo, es claro que este modelo tiende a categorizar los resultados como comentarios positivos y a obviar, casi por completo, la clase de sentimientos negativos. Una posible razón de este ajuste es que gran parte de los datos, al procesarlos, quedaron en la mitad de la curva donde es necesario tomar una decisión de clasificación que no es tan evidente como cuando los datos tienden a infinito o menos infinito y por lo tanto su clasificación es más sencilla.



3.3 Random Forest Classifier

RandomForest es un algoritmo que se puede usar tanto para regresión como para clasificación. En este caso, se utiliza en el contexto de clasificación, dado el planteamiento del problema y los objetivos que se quieren alcanzar. Este modelo utiliza

el concepto de la sabiduría de las masas, al construir un ensemble de múltiples árboles de clasificación que determinan individualmente el valor de etiqueta, en este caso binaria, de una entrada de datos particular. La clasificación que haya sido arrojada por la mayor cantidad de árboles en el bosque será la clasificación final de ese dato.

Decision Tree Classifier es el árbol de decisión que utiliza Random Forest. Su principio de funcionamiento es separar conjuntos de datos basado en una regla como la entropía o el gini, que son medidas de la ganancia de información que se da cuando un conjunto de datos se subdivide bajo una cierta regla de separación.

$$Gini = 1 - \sum_{i=1}^{n} p^2(c_i)$$

$$Entropy = \sum_{i=1}^{n} -p(c_i)log_2(p(c_i))$$

where $p(c_i)$ is the probability/percentage of class c_i in a node.

Figura 6: ecuación de gini y de entropía, criterios de separación de árbol de decisión.

El modelo de RandomForestClassifier arrojó los siguientes resultados sobre el conjunto de evaluación. Tuvo una exactitud del **67**% sobre el conjunto de datos de evaluación, lo que se debe en mayor medida a que el **67**% de los textos eran de sentimiento positivo, y el modelo prácticamente siempre predice que cualquier texto es positivo. Este modelo está sub-ajustado (**underfit**).

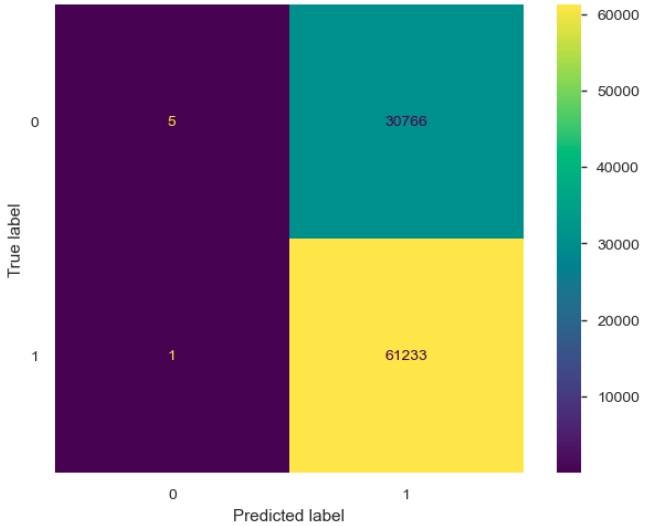


Figura 5. Resultados del modelo RandomForest.

Se evidencia que el modelo tiene un sesgo altísimo, de sobre el 99%, de asignar a cualquier texto la clasificación de sentimiento positivo (1). Esto se debe al hiperparámetro de la profundidad máxima de cada árbol del ensemble, que tuvo un valor óptimo de 15. Los árboles de decisión, y por consiguiente los Random Forest, son

subóptimos para tareas de clasificación con matrices de entrada dispersas, como ocurre en este caso, debido a que la distribución de palabras calculada por los métodos de TFID y Count vectorizer son extremadamente dispersas (sparse). Con lo cual, es prácticamente imposible que un árbol de decisión logre una clasificación adecuada de los textos, pues tiene a lo sumo 15 tokens para decidir si el texto es positivo o no.

Se contempló correr el modelo de RandomForest con parámetro de max_depth ilimitado, pero durante la ejecución, el método de gridsearch no arrojaba resultados después de dejar 4 horas de cómputo. Para iteraciones posteriores, se recomienda ejecutar el modelo de Random Forest con una profundidad máxima más alta, cercana a los 50 o incluso 100 niveles de profundidad, pero con una máquina con un mejor poder de procesamiento para poder observar resultados más pronto.

Dado que el modelo de Naive Bayes Classifier había alcanzado un desempeño de más del 80%, consideramos que era ilógico poner a correr por más de varios días un gridsearch sobre RandomForestClassifier, sabiendo que el modelo de Naive Bayes se demora apenas un minuto en ser construído.

Distribución del trabajo en equipo

Para poder desarrollar este proyecto se procedió a asignar los roles sugeridos por el enunciado, de tal forma que cada integrante debiera responder unas responsabilidades específicas.

Líder del proyecto: Cristhian Forigua
Líder de datos y analítica: Jorge Esguerra
Líder de negocio: Natalia Sanabria

La asignación de tareas específicas se presenta en la tabla a continuación.

Tarea	Responsable	Horas estimadas	Horas reales
Preprocesamiento de los datos	Cristhian Forigua y Jorge Esguerra	6h	12h
Implementación del Modelo 1 - Logistic Regression	Cristhian Forigua	3h	3h
Implementación del Modelo 2 - Naive Bayes Classifier	Natalia Sanabria	3h	3h
Implementación del Modelo 3 - RandomForestClassifier	Jorge Esguerra	3h	3h
Redacción del informe	Todos	6h	8h

Elaboración del video	Todos	1h	1h

Distribución de puntos: 1/3 de los puntos equitativos por persona.

Conclusiones:

El modelo Naive Bayes Classifier es poderoso para las tareas de procesamiento de lenguaje natural.

Para esta tarea, definitivamente el modelo más apropiado, en términos de exactitud y también de tiempo de ejecución fue Naive Bayes. En términos de tiempos de ejecución, una vez terminado el preprocesamiento, tanto Naive Bayes como la Regresión Logística demoraban apenas algunos minutos, mientras que RandomForestClassifier y tardó más de una hora en calcularse, con hiperparámetros adecuados. Sin embargo, se lograron ajustes por mucho superiores en el NB respecto a la regresión logística. Esto lo hace el modelo más apropiado para hacer parte de un pipeline de machine learning en producción, en donde el negocio probablemente querrá re-entrenar el modelo cuando se dispongan de más datos de reseñas.

El modelo de Naive Bayes es el que recomendamos para continuar con fases posteriores del proyecto. Podría eventualmente hacerse una aplicación que permita, en tiempo real, introducir una reseña y calcular el sentimiento de la misma. De esta manera, podría pedirse a expertos literarios la realización de reseñas sobre libros, buscando evaluar cuáles van a ser los más exitosos, y con ello tomar decisiones de negocio que permitan aumentar las ventas. Esto podría aprovecharlo tanto Amazon Kindle como las editoriales, al buscar hacerse con los derechos de distribución y de publicación de diversas obras prometedoras, respectivamente.

En cuanto al modelo de Regresión Logística, se logró un ajuste bueno en un tiempo razonable (aunque incompetente respecto al modelo de NB). No obstante, no se descarta su uso, especialmente en el contexto real en el que la mayor parte de los datos pertenece a la categoría de sentimiento positivo, en donde podría ser posible obtener un mejor desempeño.

A futuro, es importante reevaluar la hiperparametrización del modelo de RandomForestClassifier. La limitante de este proyecto, en cuanto a este modelo, fue el poder de cómputo de la infraestructura donde fueron entrenados. Los hiper parámetros tuvieron que limitarse, y no se probaron todas las posibles combinaciones para buscar mejorar el desempeño, puesto que el tiempo de ejecución de estos modelos incrementa radicalmente por cada valor adicional considerado en los métodos de grid search. Esto explica el sub-ajuste del modelo de RandomForestClassifier, que definitivamente no es óptimo para esta tarea de clasificación de reseñas, puesto que el cálculo de múltiples árboles de decisión de profundidades mayores a 15 toma un tiempo muy largo, lo que sería perjudicial si el modelo quisiera re-entrenarse en el futuro o si hiciera parte de un pipeline ejecutado periódicamente.

Link a repositorio proyecto: https://github.com/cdforigua05/Proyecto1_BI