PROYECTO FINAL DE INTRODUCCIÓN A LA MINERÍA DE DATOS

Nombre: Juan Torres **Cédula**: E-8-160393 **Grupo**: 1CA219

Materia: Introducción a la Ciencia de Datos **Profesor**: Juan Montenegro

1. INTRODUCCIÓN

1.1. ANTECEDENTES

Airbnb es una plataforma digital que conecta a personas que buscan alojamiento con anfitriones que ofrecen sus propiedades para estadías cortas.

Cuando potenciales huéspedes buscan un destino para quedarse, no solo toman en cuenta la ubicación, lugares o atracciones cercanas al alojamiento, la opinión de los demás usuarios es fundamental para que potenciales usuarios puedan hacerse una idea de si un lugar es bueno o no. Las reseñas definen cómo el alojamiento es visto por el público, y genera expectativas de lo que puede llegar a ser la experiencia. Un poco parecido a la dinámica que manejan ciertos hoteles y restaurantes con plataformas como TripAdvisor.

Por esa razón, los anfitriones tienen como prioridad mantener una buena impresión en sus huéspedes, lo cual se ve reflejado en las reseñas y opiniones que dejan en la página, pues saben que esto puede representar un flujo mayor de hospedajes en el futuro y un mejor *status* dentro de la plataforma. Mientras que, en el caso contrario, las malas reseñas pueden afectar su posición dentro de Airbnb y disminuir significativamente los huéspedes que reciben, y, por consiguiente, sus ingresos.

2. OBJETIVO

El objetivo de este proyecto será determinar qué factores que influyen en las calificaciones de los alojamientos dadas por los usuarios, aplicando técnicas de minería de texto y modelos de aprendizaje supervisado

3. FORMULACIÓN DEL PROBLEMA

Es por eso que nuestra variable elegida como target fueron las calificaciones de los usuarios, registradas en la columna de *review_scores_rating*.

3.1. JUSTIFICACIÓN

Se tomó esta decisión porque esta es la forma en que los huéspedes expresan su opinión general de cómo fue su experiencia en el alojamiento. Y se pueden desarrollar modelos de aprendizaje automático y *machine learing* que no solo nos ayuden a determinar qué factores tienen un mayor impacto en si la calificación será baja o alta. Además, para futuros estudios, nos permitiría realizar una proyección de cómo pueden verse afectadas las calificaciones en base a las decisiones y cambios que se realicen.

Cabe mencionar que la información que nos brindan las calificaciones se puede ver complementada por otros datos como los comentarios (*comments*), donde se profundiza y se da un mayor contexto de las razones detrás de la calificación. Por eso será fundamental el uso de la minería de texto, para obtener insights importantes que pueden ser útiles al momento de desarrollar nuestros modelos.

4. METODOLOGÍA Y RESULTADOS

4.1. CARGA Y EXPLORACIÓN DE DATOS

Nuestras fuentes de datos iniciales fueron dos los siguientes datasets:

a. listings.csv

Tiene 26,067 filas y 79 columnas. Contiene datos generales y características de los alojamientos, incluyendo: descripción del lugar y del vecindario, la ubicación, características inmuebles como el número de baños, el tipo de propiedad, número de habitaciones, camas, etc., así como información relacionada con el alojamiento como la disponibilidad, el costo y el mínimo de noches de estadía.

También incluye información útil sobre el arrendatario o *host,* como por ejemplo si tiene estatus de *super_host*, si esta verificado, si tiene foto de perfil, entre más información.

b. reviews.csv

Tiene 1,315,986 filas y 6 columnas Contiene la información de las reseñas escritas por los usuarios. Esta complementa los datos presentes en el dataset de listings.

Ninguno de los datasets tenía valores duplicados. Pero sí nos encontramos con muchos valores nulos, entonces se procedió con la limpieza inicial de los datasets.

4.2. LIMPIEZA DE DATASETS

Entre las estrategias que se utilizaron para la limpieza de datos podemos mencionar la imputación y la eliminación de valores faltantes.

Para **Listings**:

a) Eliminamos las columnas que tengan solo valores nulos o que sean innecesarias en este estudio.
b) Eliminamos las filas que no tengan precio. No las imputamos porque es algo que define el host, consideramos que no era conveniente. Pero, sí convertimos la columna price a valores numéricos, y eliminamos el signo de $
c) Eliminamos las filas vacías para los baños nos quedamos solo con el número de baños (bathroom_text), eliminando el texto (la palabra baño/s), pues esta contenía una información más completa con respecto a la cantidad de baños.

Ahora, había columnas de texto que podían ser importantes para el estudio, así que consideramos necesario tomarlas en cuenta incluso si estaban vacías, pues esto podía tener un significado. Por eso, en columnas como *description* o *comments*, completamos los espacios en blanco con indicaciones como: “Descripción no disponible”, “Sin comentarios”, entre otras.

d) Le asignamos el valor de "Descripción no disponible" a las filas vacías de la columna: 'neighborhood_overview'
e) Imputamos las columnas de texto vacías con 'No disponible' para las descripciones

Y por último, hubo columnas que sí consideramos necesario imputar, utilizando criterios como la moda, la mediana y la media.

f) Imputamos las columnas relacionadas con la infraestructura del Airbnb con la mediana, porque puede ser algo importante al momento de dejar un review. Por eso no lo eliminamos
g) Limpiamos y convertimos tasas porcentuales a decimales e imputamos con la media
h) Imputamos la columna '*host_response_time*' con la moda

Para **Reviews*:***

Solo había dos columnas con valores nulos: *comments* y *reviewer_name*. Imputamos la columna de comentarios, reemplazando los valores vacíos con un 'Sin comentarios'. Y decidimos eliminar la columna de *reviewer_name*, pues no era muy importante, es más, nos podría ayudar trabajar con reseñas anónimas.

Luego de la limpieza, nos quedamos con 21,971 filas y 54 columnas en el dataset de Listings, y con 1,208,139 filas y 6 columnas en el dataset de Reviews.

Para finalizar, unimos ambos datasets utilizando como clave común las columnas de id (df_listing) y listing_id (df_reviews) quedándonos con un total de 1,208,139 filas, sin ningún valor nulo y 57 columnas.

4.3. FORMULACIÓN DEL PROBLEMA

Como el objetivo principal del proyecto es aplicar técnicas de minería de texto y modelos de aprendizaje supervisado para determinar los factores que influyen en las calificaciones de los alojamientos de Airbnb México dadas por los usuarios, definimos como variable objetivo para entrenar el modelo, la variable de *review_scores_rating*. Para esto, creamos una columna llamada *Target*, que tomaría valores de 1 o 0, dependiendo del valor de la calificación. Se le asignó el valor 1, cuando las calificaciones fueran mayores o iguales a 4.5, y el valor del cero cuando fueran menores a 4.5.

Sin embargo, sí notamos un significativo desbalance en la distribución de clases, pues ahora teníamos alrededor de 1,172,950 registros con el valor 1 y solo 35,189 con el valor cero. Esto podría provocar que nuestro modelo se vea afectado, ya que una vez entrenado, podría estar segado por este desbalance, y podría terminar prediciendo bien los resultados de calificaciones altas (los números 1), pero no tan bien las bajas (números ceros).

4.4. FILTRADO DEL TEXTO

a. SELECCIÓN DE COLUMNAS

Antes de comenzar con el preprocesamiento del texto, seleccionamos las columnas que consideramos inicialmente relevantes para nuestro estudio, para así reducir el tamaño del dataset y trabajar de manera más eficiente y óptima. Filtramos el dataset para quedarnos solo con las columnas seleccionadas y lo guardamos como: Datos Completos Finales.

b. FILTRADO POR IDIOMA

- DETECCIÓN DE IDIOMAS

El dataset de Datos Completos Finales, contenía más de un millón de registros, incluso después de haber aplicado varios filtros y de haber limpiado varias veces el dataset para reducirlo lo más posible. Trabajar con tal cantidad

de datos seguía siendo complicado debido a las limitaciones de nuestro equipo, por eso fue necesario reducirlo aún más, y para ello decidimos filtrar los datos por idioma. Para esto, desarrollamos otro código, donde:

Definimos la función *detectar_idioma()* para identificar los idiomas presentes en la columna comments del dataset.

Para poder trabajar de la mejor manera posible tal cantidad de datos, dividimos el dataset en lotes de 100,000 usando *chunksize*. Y por medio de un ciclo for, se fueron cargando los los bloques hasta completar el dataset.

Luego, volvimos a eliminar los comentarios vacíos, en caso de que hubiera quedado alguno de la limpieza anterior, y seleccionamos muestras aleatorias de 10,000 comentarios de cada bloque. Con la función detectar_idioma() que ya habíamos definido, identificamos los idiomas presentes en las muestras y contamos cuantos comentarios había por idioma. Una vez se alcanzaron 10,000 comentarios analizados, se paró el proceso.

Los resultados nos indicaron que el idioma más frecuente era el **inglés**, con **5,679** comentarios, con seguido del español, con **3,864**. Decidimos trabajar con los comentarios en inglés.

- LIMPIEZA DEL TEXTO

Sin embargo, nos dimos cuenta de que el texto necesitaba una nueva limpieza, pues había ciertos aspectos como espacios dobles, HTML, emojis, símbolos raros y repetidos, que podían afectar nuestro estudio.

- FILTRADO DE COMENTARIOS TOTALMENTE EN INGLÉS

Debido a que había comentarios que tenían más de un idioma, creamos la función **ingles()**.

Esta función usaba detect_langs() para poder identificar el idioma “predominante” en el comentario y pasarlo por varios “filtros” para quedarnos al final con los comentarios que estuvieran 100% en inglés:

1) Si había dos idiomas, y el segundo idioma superaba el 10% de probabilidad, se descartaba el comentario. Si no, pasaban al siguiente filtro.
2) Si el idioma predominante no era el inglés o el inglés tenía menos del 90% de probabilidad. Se descartaba el comentario
3) Si contenía palabras frecuentes en español, en este contexto, como: lugar, gracias, etc., se descartaba el comentario.

Para poder aplicar la función **ingles()** para filtrar el dataset, se repitió el mismo proceso que se utilizó en la detección de idiomas, para leer el dataset completo: dividir en lotes de 100,000, con la diferencia de que esta vez sí se trabajó en todas las filas, no en una muestra.

Se guardaron los comentarios válidos de cada lote y se agruparon todos en un nuevo dataset que llamamos: 'Archivo_Filtrado_EN.csv', con **405,048 comentarios**, el total de comentarios en inglés. A este archivo, le aplicamos el preprocesamiento de texto.

4.5. PREPROCESAMIENTO DEL TEXTO

Durante el preprocesamiento del texto, limpiamos nuevamente los comentarios, utilizando la función preprod_txt, con la cual se le dio un formato más ordenado al texto. Se transformó todo el texto a minúsculas, eliminamos los símbolos y las etiquetas HTML que pudieron quedar.

Después, utilizamos *word_tokenize* para separar el texto en palabras individuales o tokens, y eliminamos las *stopwords* en inglés, elegimos palabras como “the”, “is”, “and”, las cuales no tenían un significado real para el estudio. Por último, llevamos todas las palabras a su forma raíz usando el lemmatizer.

El resultado de este proceso lo guardamos en una nueva columna llamada **clean_text**. Esta versión limpia del texto es mucho más útil para trabajar las futuras etapas del proyecto, como el análisis de sentimientos, la extracción de características con TF-ID, y para el entrenamiento de los modelos, ya que disminuye el ruido, la ambigüedad y la redundancia.

4.6. VECTORIZACIÓN

Convertimos los valores de la columna de texto limpio (**clean_text**) a una forma numérica, de manera que pudiéramos considerarla al momento de trabajar con los modelos de aprendizaje automático. Para esto, utilizamos TF-ID Vectorizer, ya que era el más recomendado para trabajar opiniones, que transformó el texto en una matriz de 405,048 filas y 47,283 columnas

Esto nos permite identificar las palabras más signficativas de una reseña, sin dejar que las palabras más comunes en el dataset afecten el análisis. Es por eso que a las palabras más comunes del dataset, como *great* o *place*, las cuales podemos ver en la figura 1, se les asignó un peso más bajo dentro de la matriz.

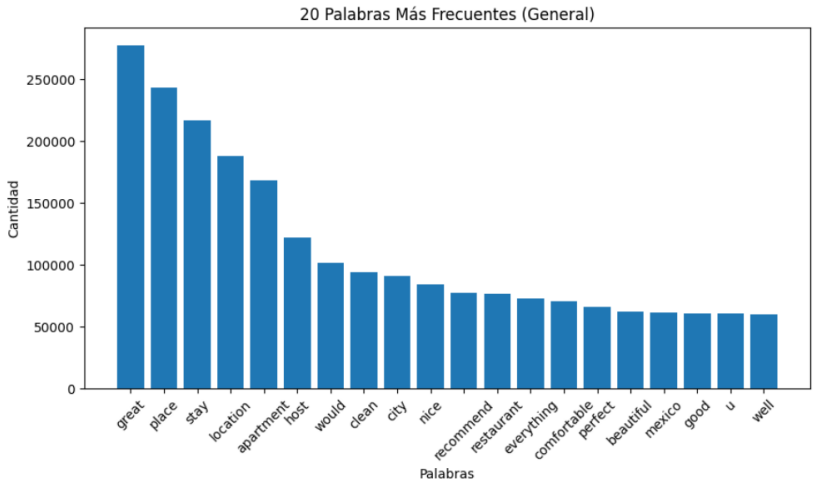

Figura 1: 20 palabras más frecuentes del dataset

Se probó el vectorizador con diferentes configuraciones. Lo probamos con unigramas y bigramas usando ngram_range(1,2) y con las palabras más frecuentes, diciéndole que ignorara a las que estuvieran presentes en el 80% del dataset, pero conservando aquellas que estuvieran por lo menos en dos comentarios. En total se generaron: 47,283 unigramas, 1,658,591 bigramas y 25,142 palabras frecuentes. Estas matrices se utilizaron como datos de entrada para los modelos de clasificación.

4.7. VISUALIZACIONES

- **WordCloud**

Al generar el WordCloud, podemos confirmar que las palabras más comunes en los comentarios incluyen términos como: place, stay, location y great, lo que es un indicativo de que la mayoría de las reseñas son positivas, pues palabras que podrían indicar algo negativo, como noise, hot o loud, están de un tamaño más pequeño.

Figura 2: Wordcloud de los Comentarios

- **Histogramas**

En los histogramas por clase, se analizaron las palabras que más se repetían bajo ciertas condiciones de las variables de superhost y de room_type

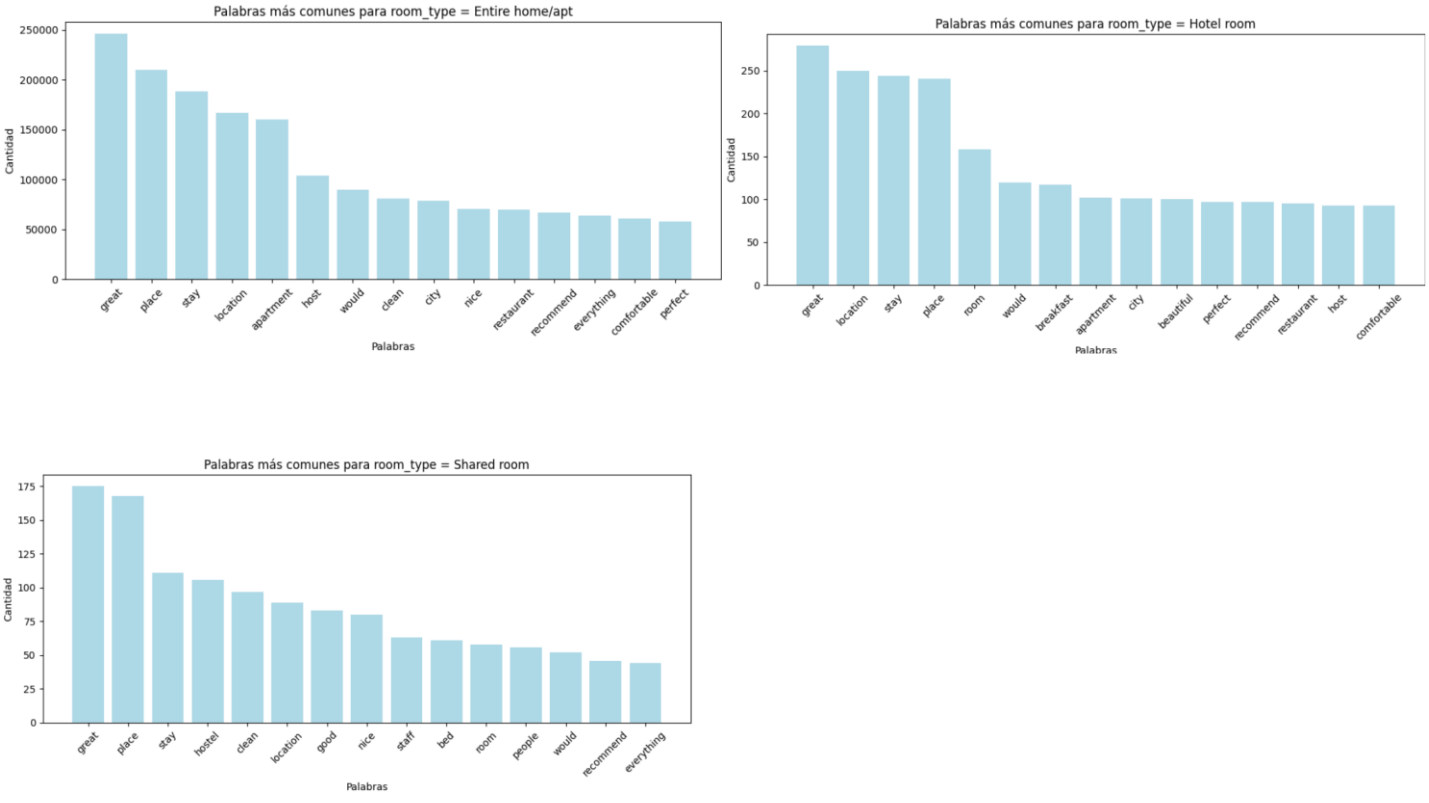

Figura 3: Histogramas para room_type

Algo que me pareció interesante fue que para el caso del room_type, si bien en general las palabras se mantienen constantes, la presencia de la palabra “perfect” se repitió únicamente en aquellas propiedades que ofrecían mayor privacidad e independencia, como: habitaciones de hotel, aparamentos o casas completas, pero no en habitaciones compartidas. Esto nos lleva a pensar que la privacidad y la independencia son cualidades muy valoradas por los usuarios al momento de buscar un alojamiento.

5. PROCESAMIENTO PREVIO DE DATOS ESTRUCTURADOS

Para nuestro caso, ya previamente habíamos realizaRdo una pre-evaluacion de las columnas y eliminamos las no relevantes. Adicional a eso, ya habíamos realizado el tratamiento de valores nulos en la limpieza de datos.

Por lo que, antes de la codificacion volvimos a reducir el dataframe con las columnas más relevantes y las que dejaremos para el modelado, incluyendo: 'review_scores_rating', 'description', 'comments', 'neighborhood_overview', 'amenities', 'comentario_limpio', entre otras.

Esto ayuda a preparar los datos estructurados para combinarse con los datos de texto al entrenar el modelo. Convertimos el texto en números que los modelos pueden entender, y la normalización evita que las variables más grandes dominen el proceso de entrenamiento.

Luego, se llevó a cabo la codificación de las variables categórica, incluyendo: *room_type, property_type* y *host_is_superhost*.

Para las dos primeras, se utilizó la codificación one-hot a través de la función pd. get_dummies(), creando columnas binarias que representan cada categoría. Y la tercera, se convirtió a formato binario (1 y 0) de manera manual, lo que hace su uso más sencillo en modelos matemáticos. Antes tenía valores de cierto y falso.

Luego, se procedió a normalizar las variables numéricas como bathrooms_text, price y number_of_reviews. Para ello, se utilizó StandardScaler(), que ajusta los valores para que tengan una media de 0 y una desviación estándar de 1. Esta transformación se aplicó directamente sobre las columnas correspondientes del DataFrame df_codificado.

Este procedimiento fue esencial ya que los modelos de machine learning, en particular aquellos que son lineales como la regresión logística, reaccionan a la magnitud de los datos. Sin el método de normalización, las variables que tienen valores más altos podrían influir de manera desproporcionada en el aprendizaje del modelo. Estandarizando todas las variables numéricas, se asegura que cada una tenga un impacto equilibrado en el proceso de enseñanza y se incrementa la efectividad general del pipeline.

6. INTEGRACIÓN DE TEXTO CON VARIABLES ESTRUCTURADAS

Para integrar los comentarios de texto y las variables estructuradas en un solo conjunto de datos, se utilizó un ColumnTransformer. Esta herramienta facilitó la aplicación de distintas transformaciones a diferentes tipos de columnas: la columna de texto: *clean_text*, fue transformada a vectores usando TF-ID Vectorizer, las columnas numéricas como *price*, *number_of_reviews* y *bathrooms_text*, fueron normalizadas usando StandardScaler, y las

variables categóricas como *room_type, property_type* y *host_is_superhost* se transformaron mediante OneHotEncoder.

Una vez realizadas estas transformaciones, se creó un conjunto de datos de entrenamiento usando *train_test_split*, dividiendo los datos en un 80% para entrenamiento y un 20% para prueba. Además, se incluyó la opción *stratify* para asegurar que la proporción de clases (*target*) permaneciera equilibrada en ambos grupos. Este paso fue fundamental para que los modelos pudieran aprender de manera justa sobre ambas clases del problema.

5. CONSTRUCCIÓN DEL MODELO PREDICTIVO

Se implementaron tres modelos: **Regresión Logística**, **Random Forest** y **Gradient Boosting**. Para cada uno, se utilizó un Pipeline que incluía el preprocesamiento de los datos, el balanceo de clases mediante SMOTE y el modelo específico, lo cual permitió un flujo de trabajo claro, repetible y sin filtraciones de información entre los entrenamientos y las pruebas.

Para determinar la mejor configuración de cada modelo, se empleó GridSearchCV, realizando una búsqueda de hiperparámetros con validación cruzada (CV=3) y tomando como métrica principal el AUC-ROC, dada la desigualdad en las clases. Esta optimización garantizó que cada modelo estuviera adecuadamente ajustado antes de ser evaluado.

6. EVALUACIÓN DE CADA MODELO

La evaluación se centró en medir cuán bien cada modelo podía diferenciar entre comentarios con calificaciones altas (target=1) y bajas (target=0). Para esto, se utilizaron diferentes criterios.

6.1. Evaluación de Modelo

El **modelo Random Forest** se destaca como el que ofrece el mejor rendimiento general según el informe de clasificación. Con un nivel de **exactitud del 85%**, un **F1-score de 0.91** para la clase 1, sobresale por su habilidad para identificar correctamente la clase mayoritaria (*target = 1),* que abarca más del 98% del conjunto total de datos.

Incluso frente a la clase minoritaria (*target = 0*), ofrece un **F1-score de 0.14**, el mejor entre los tres modelos. Esto es especialmente importante en datasets desbalanceados, ya que indica que, aunque su desempeño en la clase 0 no es perfecto, es superior al resto.

Reporte:

	precision	recall	f1-score	support
0	0.07	0.79	0.14	1214
1	1.00	0.85	0.92	79796
accuracy			0.85	81010

Por otro lado, el modelo de **Regresión Logística tiene el rendimiento más débil**, lo que lo hace menos confiable para detectar calificaciones negativas, que son las más importantes para análisis de satisfacción o detección de problemas en Airbnb.

Reporte de clasificación:

	precision	recall	f1-score	support
0	0.05	0.81	0.09	1214
1	1.00	0.76	0.87	79796
accuracy			0.77	81010

6.2. MATRICES DE CONFUSIÓN

La **matriz de confusión** facilita la evaluación clara de cuántas predicciones fueron acertadas y cuántas fallidas. **Random Forest es el más efectivo de los modelos**, ya que alcanza 67,869 verdaderos positivos y disminuye los falsos negativos a 11,927, además de clasificar correctamente 959 de 1,214 casos negativos (clase 0). Aunque no es ideal para esta clase menos frecuente, presenta un buen equilibrio, con errores moderadamente distribuidos entre ambas clases. Por otro lado, Gradient Boosting mejora la identificación de la clase 0 al contar con 1,078 verdaderos negativos y solo 136 falsos positivos, mostrando un mayor enfoque en la detección de calificaciones negativas o eventos raros.

En contraste, **Regresión Logística es el modelo menos efectivo**, ya que solo logra identificar correctamente 989 casos de la clase 0 y presenta 18,797 falsos negativos, lo cual es problemático si la clase 0 representa un comportamiento crítico (como reseñas negativas o quejas de mal servicio). Esta distribución indica que el modelo casi ignora la clase minoritaria y tiene éxito solo debido a su inclinación hacia la mayoría. En cambio, Random Forest y Gradient Boosting demuestran una mejor capacidad para manejar el desequilibrio.

6.3. CURVA ROC Y AUC

La **curva ROC** y la medición que la acompaña, el **AUC (Área Bajo la Curva)**, muestran cuán eficaz es un modelo para distinguir entre distintas categorías sin depender de un límite de decisión específico. Gradient Boosting destaca en esta medición con un AUC de 0. 9198, lo que indica que tiene la mayor capacidad para separar clases de manera probabilística. Random Forest le sigue con un AUC de 0. 9053, lo que también indica un rendimiento sólido y confiable. Finalmente, Regresión Logística tiene un AUC de 0. 8713, que, aunque es aceptable, está claramente por debajo de los otros modelos.

En base a estos resultados, podemos decir que **Random Forest se posiciona como el modelo más completo**, confiable y aplicable para este caso de análisis de las calificaciones de los usuarios de Airbnb.

7. VISUALIZACIÓN DE RESULTADOS

Para visualizar el contenido textual, se crearon nubes de palabras segmentadas según la variable objetivo. Algo que llama la atención que que las mismas palabras destacan para ambos targets.

Esperábamos que los comentarios de las calificaciones bajas incluyeran palabras “negativas”, que nos ayudaran a identificar problemas a corregir. Pero no fue así.

Una hipótesis sugiere que los comentarios en los targets 0 pueden ser positivos, sobre todo porque puede tratarse de buenas calificaciones que estén por debajo de 4.5, por ejemplo: 4.0 en adelante, y por eso los

comentarios incluyen palabras “positivas” como *great* y *recommend*. No necesariamente el hecho de que el target sea 0, significa que será una mala o una baja calificación. O esto también podría deberse al desbalance de clases. Esto podemos confirmarlo con el análisis de sentimientos.

Figura 4: Wordcloud Target 1

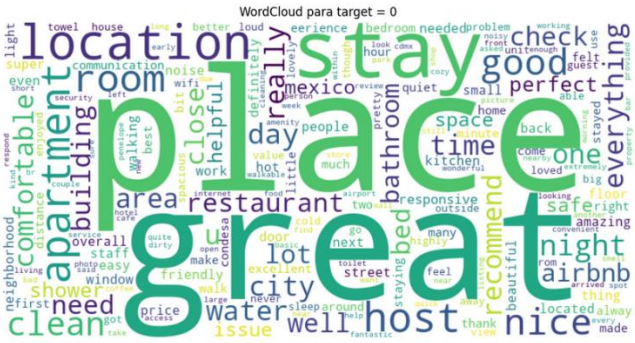

Figura 5: Wordcloud target 0

8. ANÁLISIS DE SENTIMIENTOS

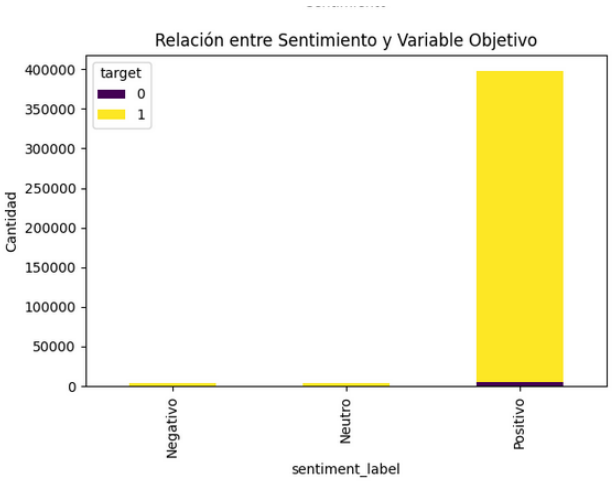

Figura 6: Relación entre Sentimiento y Target

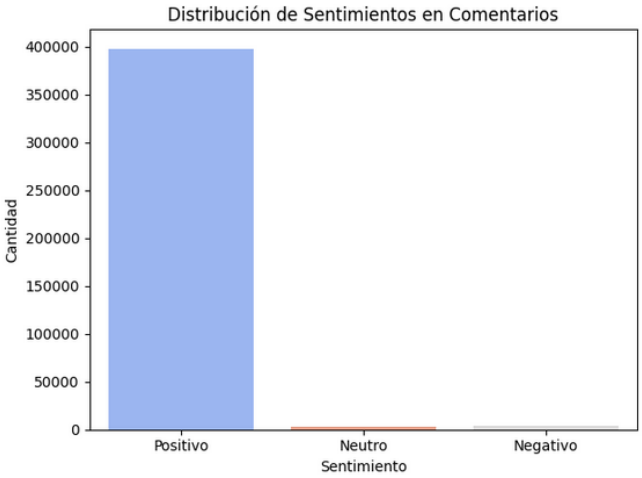

Figura 7: Distribución de Sentimientos en Comentarios

Las gráficas presentan una clara preferencia hacia comentarios con una connotación positiva, que están mayormente vinculados a una evaluación alta (**target = 1**). En la figura 6, que relaciona el sentimiento con la variable objetivo, se puede ver que casi todos los comentarios favorables están clasificados como target 1, mientras que los negativos o neutros son muy pocos, aunque también tienden a asociarse más con el valor 1.

El segundo gráfico, en la Figura 7, que se enfoca en la distribución de sentimientos, respalda esta tendencia: la mayor parte de los comentarios son positivos, con escasos comentarios neutros o negativos. Esto sugiere un evidente desbalance en las clases, lo cual es especialmente importante si se piensa usar los datos para crear un modelo de clasificación, ya que podría causar un sobreajuste y disminuir la habilidad del modelo para generalizar con nuevos datos.

En resumen, los datos indican que el sentimiento positivo está fuertemente relacionado con calificaciones altas, lo cual podría ser útil para desarrollar modelos que pronostiquen valoraciones o identifiquen comentarios significativos. Sin embargo, esto también supone un riesgo si no se gestionan correctamente las clases menos representadas, ya que un modelo entrenado en este escenario podría volverse insensible a señales negativas o neutras.

Dado que la variable *target* se refiere a calificaciones (0 para bajas, 1 para altas), este análisis apoya la idea de que los comentarios positivos tienden a favorecer una mejor evaluación. A pesar de ello, para garantizar la solidez del modelo predictivo o del análisis de impacto, sería aconsejable utilizar técnicas de equilibrado o ponderación para corregir el desbalance observado.

9. **FACTORES MÁS INFLUYENTES**

Ya pudimos determinar que el modelo más eficiente para trabajar el conjunto de datos fue el de Random Forest, lo utilizaremos para determinar cuáles fueron los factores más influyentes en el estudio, aplicando **SHAP**.

9.1 SHAP

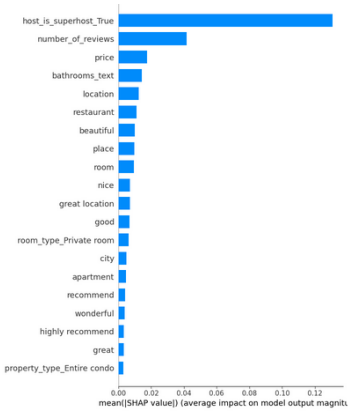

Figura 8: Top 20 características más importantes según Random Forest.

En la Figura 8 podemos ver las variables que el modelo Random Forest identificó como las más influyentes para predecir si una reseña será positiva (target = 1). En la Tabla 1, se profundiza en las más importantes:

Tabla 1: Las características más influyentes según Random Forest

VARIABLE	DESCRIPCIÓN	IMPACTO	EXPLICACIÓN
host_is_superhost_True	Nos indica si el anfitrión tiene status de "**superhost**"	Es la variable **más influyente**. Contribuye con un **+0.130682** a que el resultado del modelo se incline hacia la clase **1.**	Los **superhost** ofrecen mejores experiencias
number_of_reviews	Es el número de reviews para ese alojamiento	Es la segunda variable **más influyente**. Contribuye con un **+0.041701** a que el resultado del modelo se incline hacia la clase **1.**	Si les gustó el lugar, lo van a recomendar y deján más reseñas positivas
price	El precio del alojamiento	Es la segunda variable **más influyente**. Contribuye con un **+0.017343** a que el resultado del modelo se incline hacia la clase **1**	La relación calidad-precio es importante para los huéspedes.

10. DIAGRAMA DE PIPELINE DEL PROYECTO

El flujo completo del proyecto se puede resumir en un proceso que incluye las siguientes etapas:

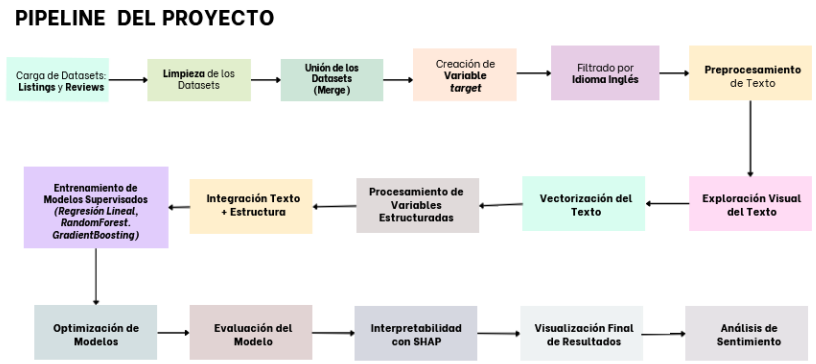

Figura 5: Pipeline del proyecto

11. REFLEXIÓN CRÍTICA

11.1. CONCLUSIONES Y RECOMENDACIONES

En base a los resultados obtenidos, podemos concluir que los factores que más influyen para obtener calificaciones positivas son el precio y el status de superhost del anfitrión, lo que nos lleva a intuir que los huéspedes valoran mucho la relación calidad-precio. El modelo que nos permitió obtener estos resultados fue el Random Forest, el modelo más eficiente para trabajar estos datos. Sin embargo, si analizamos la naturaleza del dataset, podemos decir que el marcado desbalance de clases que presentó, sí afectó nuestros resultados. En futuros estudios trataríamos de trabajar en eso.

11.2. SELECCIÓN DE VARIABLES Y TEXTO

Escogimos la columna *comments* como nuestra fuente principal de texto, porque esta nos da un mejor contexto y profundiza más en la opinión de los usuarios. Nos permiten entender cómo se sintieron las personas dentro del Airbnb, qué les gustó, qué no les gustó, en las propias palabras del usuario. Son un indicador crucial para anticipar la calificación que darán.

En relación con las variables estructuradas, elegimos las features que podían afectar directamente la experiencia del usuario y como se sentía dentro del Airbnb, como, por ejemplo: el tipo de habitación, número de baños, tipo de propiedad, entre otras.

Se observó, por medio de las Wordclouds, que la mayoría de las palabras presentes en el dataset eran indicadores de reseñas positivas, pues veíamos palabras como great, en un tamaño grande, y palabras que indicaban algo negativo, en un tamaño más pequeño. Lastimosamente, nos dimos cuenta de que esto podía deberse al desbalance de clases del dataset.

11.3. JUSTIFICACIÓN DE MODELOS

a) REGRESIÓN LOGÍSTICA

La **Regresión Logística** es un modelo estadístico tradicional que se utiliza para **clasificaciones binarias**. Su fundamento principal es calcular la probabilidad de que una observación pertenezca a una categoría, modelando esa probabilidad mediante una función logística, que tiene forma sigmoidea. Es un modelo lineal, lo que implica que se considera que existe una conexión lineal entre las variables independientes y la probabilidad del suceso. En situaciones como el análisis de opiniones o valoraciones, puede servir como un punto de partida para identificar los factores que influyen en una opinión positiva o negativa.

VENTAJAS

Entre sus principales ventajas, resalta su facilidad de uso y velocidad. Es muy fácil de comprender: los coeficientes muestran la magnitud y el efecto de cada variable en la probabilidad de la categoría deseada. También es eficiente desde el punto de vista computacional, funciona adecuadamente con conjuntos de datos pequeños y no exige muchos recursos. Por esta razón, a menudo se utiliza como el modelo de referencia en diversos problemas de clasificación.

LIMITACIONES

Sin embargo, sus limitaciones son notables cuando se enfrenta a problemas complejos o a datos desequilibrados. La regresión logística no puede captar relaciones no lineales a menos que se introduzcan manualmente variables polinomiales o de interacción. También es considerablemente sensible al desequilibrio de clases, como en el caso en que la clase 0 representa un porcentaje muy bajo. Esto resulta en una tasa alta de falsos negativos y un bajo recall para la clase menos común, como se notó en este análisis.

b) RANDOM FOREST

El **Random Forest** es un modelo que se basa en árboles de decisión. Consiste en crear varios árboles utilizando subconjuntos aleatorios del conjunto de datos y luego promediar los resultados, en el caso de clasificación, se toma la opción más votada. Esta técnica permite captar relaciones no lineales, manejar interacciones entre variables y mejorar la capacidad del modelo para generalizar. Es uno de los modelos más comunes en el mundo real debido a su resistencia y facilidad de uso.

VENTAJAS

Entre sus principales beneficios se encuentra su habilidad para trabajar con datos desequilibrados, relaciones complicadas y conjuntos de datos con muchas variables sin requerir normalización o transformación previa. No solo proporciona un buen rendimiento en términos de exactitud, sino que también es resistente al sobreajuste gracias a la combinación de varios árboles. Además, es compatible con métodos de interpretación como SHAP, lo que ofrece explicaciones detalladas para ambas clases cuando se utiliza de manera correcta.

LIMITACIONES

Aunque fue el modelo que mejor rendimiento tuvo, su principal limitación es el alto costo computacional. Entrenar y realizar predicciones con numerosos árboles puede hacerse lento en conjuntos de datos muy grandes. Y es algo que nos pasó durante el desarrollo de este proyecto.

c) GRADIENT BOOSTING

Gradient Boosting es un método de ensamblaje que entrena árboles de decisión de forma secuencial; cada árbol nuevo busca corregir los errores producidos por el anterior. A diferencia de Random Forest, que promedia los árboles, este método mejora gradualmente para reducir una función de pérdida, como la pérdida logarítmica o el error cuadrático. Esta capacidad de rectificar los errores de forma escalonada permite que sea un modelo altamente preciso y afinado, especialmente útil para identificar patrones sutiles, como quejas o calificaciones inusuales.

VENTAJAS

Una de sus principales ventajas es su notable capacidad de predicción, especialmente con conjuntos de datos desbalanceados o de relaciones no lineales. Puede manejar diversas funciones de pérdida, incorpora regularización y puede optimizar tanto la precisión como la sensibilidad. Su rendimiento es tan elevado que a menudo se prefiere en competencias de aprendizaje automático, como las de Kaggle. Además, es compatible con herramientas que facilitan la interpretación, como SHAP.

LIMITACIONES

Por otro lado, su complejidad representa su mayor limitación. Gradient Boosting requiere una selección cuidadosa de los hiperparámetros, como la tasa de aprendizaje, el número de árboles y su profundidad, para evitar el sobreajuste. Es más sensible a los ruidos y valores atípicos, y su proceso de entrenamiento puede ser lento, sobre todo con grandes volúmenes de datos. Además, la interpretación de los resultados puede ser menos clara en comparación con modelos como la regresión logística.

11.4. EVALUACION DE RESULTADOS

Los tres modelos cometieron específicamente errores de **falsos negativos**, lo que significa que clasificaron comentarios como clase 0 cuando en realidad eran clase 1. A pesar de que se utilizó **SMOTE** para equilibrar las clases, los problemas persistieron. El modelo que mayores dificultades presentó fue el de **regresión logística**.

La tendencia o patrón principal que notamos fue el evidente desbalance de clases en los datos, lo cual afectó significativamente nuestros resultados, sin importar cuanto limpiáramos nuestro dataset, la diferencia seguía siendo muy amplia. Más de el 90% de los datos eran de clase 1 y un pequeño porcentaje de clase 0. Esto provocó que nuestros modelos tuvieran un sesgo hacia las calificaciones altas, lo cual podemos ver reflejado en nuestro análisis de sentimientos y otras visualizaciones de los datos.

11.5. REFLEXIÓN ÉTICA Y DE USO RESPONSABLE

El uso irresponsable e inadecuado de los modelos predictivos en plataformas como Airbnb puede aumentar las desigualdades sociales o económicas de sus usuarios. Por ejemplo, si un modelo que no ha sido entrenado correctamente discriminara en contra de ciertos vecindarios o tipos de propiedades podría dejar fuera o perjudicar injustamente a algunos anfitriones o huéspedes. También hay el peligro de que estas herramientas refuercen estereotipos si los datos y las variables utilizadas no se revisan de manera adecuada.

Para prevenir prejuicios, especialmente en modelos que procesan texto, es esencial hacer un preprocesamiento cuidadoso y adecuado de los datos, para así garantizar que haya diversidad en los conjuntos de entrenamiento. Además, es relevante examinar la interpretabilidad, como hicimos con SHAP, y revisar constantemente los modelos. De esta manera podemos llevar a cabo las correcciones necesarias para evitar prejuicios o sesgos.

11.6. LIMITACIONES DE ESTUDIO

Durante el proyecto, nos encontramos con varias restricciones. Una de ellas fue la falta de experiencia directa con la plataforma Airbnb. Al no ser usuarios de la plataforma ni tener un conocimiento profundo de cómo funciona internamente, tuvimos que analizar los datos y las variables solo desde un ángulo técnico, sin disponer de una perspectiva del usuario o del negocio que nos ayudara a entender mejor algunas decisiones.

No obstante, la limitación más significativa fue de carácter técnico: manejar un volumen tan alto de datos fue complicado debido a las capacidades limitadas de nuestro equipo. Ciertos procesos, como la transformación del texto o el entrenamiento de modelos con validación cruzada, demandaban una cantidad considerable de memoria y recursos de procesamiento, lo que hizo que muchas pruebas fueran lentas e incluso se congelaran. Esto nos obligó a tomar decisiones como el código de Filtrado del Idioma Inglés para reducir hasta la mínima expresión los datasets.

Por último, considero que trabajar con un dataset tan desbalanceado, sí fue una limitante, porque afectó nuestras conclusiones y hallazgos.

11.7. IA REGENERATIVA

A. Riesgos de utilizar IA sin comprensión

Hoy en día la inteligencia artificial tiene muchas aplicaciones en diferentes campos, y avanza cada vez más todos los días. Pero, es importante saber que, sin criterio, ni comprensión de lo que se está trabajando, la IA deja de ser un apoyo y se convierte en un riesgo potencial. Si bien estos modelos están entrenados para aprender y están cargados con información, muchas veces ellos no comprenden el contexto ni el objetivo que tenemos nosotros como analistas, no conocen ni comprenden nuestras prioridades, lo que queremos lograr, lo que nos funciona y lo que no.

Reconozco las ventajas que ofrecen las herramientas de inteligencia artificial, sobre todo en términos de eficiencia, pero es fundamental tener unas bases sólidas de conocimiento y comprensión de lo que se está trabajando para interpretar lo que nos da, de lo contrario no sabremos si está haciendo o no un buen trabajo. Al final del día, nosotros debemos tomar la decisión final, y responder por el trabajo que entregamos, no

podemos delegar la responsabilidad o culpar a estos modelos, pues por algo se nos asignó la tarea a nosotros, es nuestra responsabilidad hacer un buen trabajo.

B. Qué aprendimos al trabajar sin generadores automáticos

Existe una gran comunidad de programadores, analistas e ingenieros de sistemas que ponen su conocimiento a disposición del público en foros de internet, páginas, videos y libros que fueron un gran apoyo para nosotros al momento de desarrollar el código y comprender los conceptos fundamentales necesarios para llevar a cabo el proyecto. Esto requirió de un arduo trabajo de investigación, de mucho ensayo y error, hasta poder concretar un código funcional que cumpliera con los requisitos del proyecto.