“Desarrollo de una Metodología Analítica para Identificar Factores Determinantes del Review Score en E-commerce: Aplicación a la Base de Datos de Olist, Brasil”

ACTIVIDAD FORMATIVA EQUIVALENTE A TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN ANALÍTICA DE NEGOCIOS

Estudiante: Ignacio Andrés Araneda Gallardo
Profesor Guía: Richard Weber Hass

Santiago, Septiembre 2025

Índice

RESUMEN EJECUTIVO 6
CAPÍTULO 1: INTRODUCCIÓN 7
CAPÍTULO 2: PLANTEAMIENTO Y FORMULACIÓN DEL PROBLEMA 9
CAPÍTULO 3: HIPÓTESIS 11
Preguntas de Investigación *11*
CAPÍTULO 4: OBJETIVOS 12
Objetivo Principal *12*
Objetivos Específicos *12*
CAPÍTULO 5: METODOLOGÍA 13
CAPÍTULO 6: MARCO TEÓRICO 16
CAPÍTULO 7: DESCRIPCIÓN DEL ENFOQUE DE SOLUCIÓN 21
CAPÍTULO 8: RESULTADOS 22
Exploración de Datos (EDA) *22*
Preparación de Datos e Ingeniería de Características *25*
CAPÍTULO 9: MODELACIÓN Y ANÁLISIS 27
Elección de mejor modelo para cálculo de puntaje de sentimiento de columna de comentarios *27*
Resultados de Métricas de Modelos de Clasificación Binaria Para Los Tres Escenarios *28*
Importancia de Características en Modelos de Árboles sin Puntaje de Sentimiento *31*
Importancia de Características en Modelos de Árboles con Sentiment_score_pysentimiento *32*
Importancia de Características en Modelos de Árboles con Sentiment_score_roberta *33*
Valores SHAP para los Tres Escenarios en Estudio *34*
Clusterización de Comentarios Mediante BERTopic *40*
CAPÍTULO 10: CONCLUSIONES 43
CAPÍTULO 11: BIBLIOGRAFÍA 44
CAPÍTULO 12: ANEXOS 49
A. Glosario de Términos *49*
B. Descripción de Variables Utilizadas *51*
C. Análisis de Datos Faltantes *52*
D. Gráficos Adicionales del Análisis Exploratorio de Datos (EDA) *53*
E. Tablas Detalladas de Valores SHAP *55*
F. Fragmentos de Código Utilizado en el Proyecto *62*

Índice de Tablas

Tabla 1: Cálculo de puntaje de sentimiento a partir de los comentarios mediante distintos modelos y métricas de performance frente a su relación con la característica 'review_score' del dataset. 28
Tabla 2. Resultados modelos de árboles y redes neuronales en clasificación binaria de 'review_score' sin incluir columna comentarios. 29
Tabla 3: Resultados modelos de árboles y redes neuronales en clasificación binaria de 'review_score' con puntaje de sentimiento de modelo pysentimiento. 29
Tabla 4: Resultados modelos de árboles y redes neuronales en clasificación binaria de 'review_score' con puntaje de sentimiento de modelo RoBERTa. 29
Tabla 5: Resultado clusterización por BERTopic con maximización de coeficiente de coherencia. 40
Tabla 6: Comentarios representativos por clústeres. 41
Tabla 7: Cantidad de datos nulos para la base de datos integrada con las 7 tablas del e-commerce Olist. 53

Índice de Imágenes

Imagen 1: Gráfico de categorías de producto con peor ‘review score’ y gráfico con las diez categorías de producto con más ventas y sus respectivos ‘review score’. 22
Imagen 2: Distribución del puntaje de reseñas. 23
Imagen 3: Distribución de la variable ‘delivery_difference_days’ y gráfico de caja de ‘delivery_difference_days’ para reseñas negativos y positivos. 23
Imagen 4: Mapa de correlaciones entre variables numéricas. 24
Imagen 5: Nubes de palabras asociados a los comentarios, agrupados para ‘review score’ negativos y positivos. 24
Imagen 6: Importancia de características para modelos de árboles en escenario donde no se incluye el puntaje de sentimiento. 31
Imagen 7: Importancia de características para modelos de árboles que incluyen puntaje de sentimiento calculado por Pysentimiento. 32
Imagen 8: Importancia de características para modelos de árboles que incluyen puntaje de sentimiento calculado por RoBERTa. 33
Imagen 9: Fórmulas usadas para el cálculo de métricas de los valores SHAP de cada atributo. 34
Imagen 10: Valores Shap Promedio de cada caracteristica para modelos entrenados sin puntaje de sentimiento. 35
Imagen 11: Magnitud promedio de cada característica para modelos entrenados sin puntaje de sentimiento. 36
Imagen 12: Valores Shap Promedio de cada característica para modelos entrenados con puntaje de sentimiento de Pysentimiento. 37
Imagen 13: Magnitud promedio de cada característica para modelos entrenados con puntaje de sentimiento de Pysentimiento. 38
Imagen 14: Valores Shap Promedio de cada caracteristica para modelos entrenados con puntaje de sentimiento de RoBERTa. 39
Imagen 15: Magnitud promedio de cada característica para modelos entrenados con puntaje de sentimiento de RoBERTa. 39
Imagen 16: Gráfico de dispersión de gasto promedio por cliente 53
Imagen 17: Tabla comparativa entre estados de adelanto y retraso de pedidos. 54
Imagen 18: Cantidad de pedidos por cliente en una misma orden. 54
Imagen 19: Gráfico de dispersión del ‘review_score’ promedio de cada ‘seller ID’ presente en el e-commerce Olist. 55
Imagen 20: Distribución porcentual de calificaciones negativos y positivos para ordenes adelantadas y retrasadas. 55
Imagen 21: Valores SHAP modelo Random Forest escenario sin puntaje de sentimiento. 56
Imagen 22: Valores SHAP modelo Random Forest escenario sin puntaje de sentimiento. 56
Imagen 23: Valores SHAP modelo AdaBoost escenario sin puntaje de sentimiento. ...56
Imagen 24: Valores SHAP modelo GradientBoosting escenario sin puntaje de sentimiento. 57
Imagen 25: Valores SHAP modelo LSTM escenario sin puntaje de sentimiento. 57
Imagen 26: Valores SHAP modelo RNN escenario sin puntaje de sentimiento. 57

Imagen 27: Valores SHAP modelo XGBoost escenario con ‘sentiment_score_pysentimiento’ 58
Imagen 28: Valores SHAP modelo Random Forest escenario con ‘sentiment_score_pysentimiento’. 58
Imagen 29: Valores SHAP modelo AdaBoost escenario con ‘sentiment_score_pysentimiento’. 58
Imagen 30: Valores SHAP modelo Gradient Boosting escenario con ‘sentiment_score_pysentimiento’. 59
Imagen 31: Valores SHAP modelo LSTM escenario con ‘sentiment_score_pysentimiento’. 59
Imagen 32: Valores SHAP modelo RNN escenario con ‘sentiment_score_pysentimiento’. 59
Imagen 33: Valores SHAP modelo XGBoost escenario con ‘sentiment_score_roberta’. 60
Imagen 34: Valores SHAP modelo Random Forest escenario con ‘sentiment_score_roberta’. 60
Imagen 35: Valores SHAP modelo AdaBoost escenario con ‘sentiment_score_roberta’. 60
Imagen 36: Valores SHAP modelo Gradient Boosting escenario con ‘sentiment_score_roberta’. 60
Imagen 37: Valores SHAP modelo LSTM escenario con ‘sentiment_score_roberta’. . .61
Imagen 38: Valores SHAP modelo RNN escenario con ‘sentiment_score_roberta’. 61

RESUMEN EJECUTIVO

En el ecosistema de los comercios digitales, las calificaciones y reseñas de productos juegan un rol clave en la percepción del cliente, la reputación de los vendedores y la decisión de compra. Comprender los factores que influyen en las reseñas negativas y la satisfacción del cliente es crucial para plataformas de comercio electrónico que buscan mejorar la experiencia del usuario y fortalecer la percepción de este.

Este proyecto propone una metodología integral para analizar y predecir calificaciones de productos en plataformas de comercio electrónico, combinando técnicas de *machine learning* y procesamiento de lenguaje natural (NLP). La propuesta permite identificar los determinantes clave de las malas calificaciones, integrando variables estructurales y aspectos emocionales presentes en los comentarios de los usuarios.

La metodología se aplica al caso de la plataforma Olist (Brasil), considerando variables del producto, del cliente, del vendedor, de los comentarios y de las órdenes. Se incorpora análisis de sentimientos mediante los modelos de procesamiento de lenguaje natural PySentimiento y RoBERTa para capturar el impacto emocional en las calificaciones. Se entrenan modelos de clasificación binaria, donde *Random Forest* obtiene el mejor desempeño (precisión y F1-score de 91,1%).

Los resultados muestran que ‘delivery_difference_days’ y el puntaje de sentimiento de los comentarios son las variables más influyentes en la predicción de reseñas negativas. Además, ‘delivery_status’ y ‘order_per_customer’ presentan una contribución positiva sobre esta clase. El análisis SHAP confirma el impacto del tiempo de entrega y el sentimiento de los comentarios, mientras que *BERTopic* revela que los principales focos de insatisfacción se relacionan con problemas logísticos y calidad del producto.

Este trabajo entrega una herramienta analítica replicable que puede ser aplicada en distintos entornos digitales para anticipar insatisfacciones, optimizar procesos y fortalecer la experiencia del cliente.

CAPÍTULO 1: INTRODUCCIÓN

El comercio electrónico ha transformado profundamente la manera en que los consumidores interactúan con las empresas, generando un cambio sustancial en las dinámicas del mercado global. En Brasil, este fenómeno ha adquirido una relevancia particular, posicionándose como una de las principales actividades económicas digitales en la región. Según Americas Market Intelligence (2023), el país contaba con aproximadamente 128 millones de compradores en línea, lo que representaba más del 70% de la población adulta, con un gasto promedio anual de USD 981 por usuario. Este crecimiento exponencial no solo ha abierto nuevas oportunidades para las empresas, sino que también ha intensificado los desafíos relacionados con la experiencia del cliente, siendo la logística uno de los aspectos más críticos. Los retrasos en las entregas y el incumplimiento de expectativas impactan directamente en la percepción del consumidor, afectando no solo las calificaciones asignadas, sino también la confianza en las plataformas de comercio electrónico (Dündar & Öztürk, 2020; Yuan & Qiao, 2018).

La elección de este tema responde a la creciente importancia del comercio electrónico en economías emergentes como la brasileña, y a la necesidad de comprender qué factores determinan la satisfacción del cliente. En este contexto, las calificaciones de productos funcionan como indicadores clave, al reflejar tanto la calidad del servicio como la percepción del consumidor. Estas reseñas no solo influyen en las decisiones de compra de otros usuarios, sino que también ofrecen a las empresas una fuente valiosa de retroalimentación sobre el desempeño de sus operaciones (Zhao et al., 2020; Eid, 2011). Comprender qué variables inciden en estas valoraciones es fundamental para optimizar la experiencia del cliente y mejorar la competitividad en un mercado cada vez más dinámico.

Los comentarios de los usuarios aportan una dimensión adicional de análisis, al permitir explorar cómo las emociones expresadas en los textos reflejan sus experiencias durante el proceso de compra. Por esta razón, se propone el uso de herramientas analíticas avanzadas que no solo midan el impacto de los retrasos en las calificaciones, sino que también permitan analizar el contenido textual de las reseñas para profundizar en la percepción del cliente (Jain & Roy, 2024; Chehal et al., 2022).

Este trabajo se sustenta en el potencial de las técnicas avanzadas de análisis de datos y procesamiento de lenguaje natural (NLP) para abordar problemáticas complejas en entornos digitales. En particular, el uso de modelos de *machine learning* y algoritmos basados en BERT (*Bidirectional Encoder Representations from Transformers*) permite un análisis detallado del contenido emocional de los comentarios, relacionándolo con distintos factores que inciden en la experiencia del cliente, tales como aspectos logísticos, características del producto, calidad del servicio y cumplimiento de los tiempos de entrega (Wang et al., 2023; Singh et al., 2022). Este enfoque no solo mejora la capacidad de las empresas para identificar áreas críticas, sino que también les proporciona herramientas para anticiparse a posibles calificaciones negativas y tomar decisiones informadas orientadas a mejorar la satisfacción del usuario.

El objetivo principal de este estudio es desarrollar una metodología que permita identificar las características más relevantes que influyen en las calificaciones otorgadas

por los clientes, tanto positivas como negativas. Para ello, se plantean dos preguntas centrales: ¿Qué variables aumentan la probabilidad de recibir una calificación negativa? y ¿Qué información relevante puede extraerse del contenido de los comentarios, y cómo se relaciona con la satisfacción del usuario?

La hipótesis central de este estudio es que el análisis del contenido emocional de los comentarios permite identificar información clave sobre la experiencia de compra, lo que no solo contribuye a explicar las calificaciones otorgadas, sino que también proporciona señales predictivas útiles para comprender y anticipar el comportamiento de compra de futuros clientes (Shan et al., 2025; Jim et al., 2024).

La investigación se desarrolla utilizando la base de datos de Olist disponible en Kaggle, el cual contiene información detallada sobre fechas de entrega, calificaciones, comentarios de los usuarios y características de los productos. La metodología incluye el preprocesamiento y exploración de datos, el análisis de la relación entre los tiempos de entrega y las calificaciones, la aplicación de técnicas de NLP para cuantificar el contenido emocional de los comentarios, y el desarrollo de modelos de clasificación binaria que distingan entre calificaciones positivas y negativas. Finalmente, se analizan los resultados obtenidos para determinar las variables con mayor impacto en las calificaciones, y se explora cómo el contenido de los comentarios contribuye a comprender mejor la satisfacción del cliente. Se espera que los hallazgos permitan generar recomendaciones orientadas a optimizar la atención al cliente y fortalecer la competitividad en un entorno digital en constante evolución.

CAPÍTULO 2: PLANTEAMIENTO Y FORMULACIÓN DEL PROBLEMA

En el contexto del comercio electrónico, las empresas operan en un entorno digital donde la experiencia del cliente y la opinión expresada después de la compra —a través de reseñas y calificaciones— se ha convertido en un activo estratégico. A medida que el mundo del e-commerce ha experimentado un crecimiento sostenido, también lo ha hecho el volumen de datos generados por los usuarios, especialmente en forma de comentarios textuales y puntuaciones. Este aumento plantea un problema concreto de toma de decisiones: cómo transformar este volumen creciente de datos no estructurados en conocimiento accionable que apoye decisiones operativas y estratégicas, como la priorización de mejoras logísticas, la atención al cliente o el diseño de campañas personalizadas (Deniz et al., 2022; Daza et al., 2024).

La decisión empresarial que se busca respaldar en este proyecto está directamente relacionada con la anticipación y prevención de experiencias negativas mediante el uso de técnicas analíticas. En muchos casos, las empresas cuentan con las calificaciones y comentarios, pero carecen de metodologías automáticas que integren esta información con variables logísticas o de producto para tomar decisiones de forma oportuna y eficiente (Yuan & Qiao, 2018; Zaghloul et al., 2024).

La literatura reciente respalda esta necesidad. Diversos estudios han mostrado que las reseñas no solo permiten evaluar la satisfacción del cliente, sino también detectar fallas logísticas, brechas entre lo prometido y lo recibido, e incluso predecir eventos como devoluciones o abandono de la plataforma (Zhao et al., 2023; Wu et al., 2023). Sin embargo, como destaca Rahman et al. (2024), existen desafíos técnicos y organizacionales que limitan la aplicación efectiva del análisis de sentimientos en el comercio electrónico: dificultad para comprender el contexto o el sarcasmo, manejo de lenguaje informal y emojis, adaptación a distintos dominios, y la falta de interpretabilidad en modelos avanzados.

A esto se suman limitaciones señaladas por estudios recientes como el desarrollado por Ipa et al. (2024), que indican que las reseñas en el comercio electrónico suelen estar escritas en lenguaje coloquial, con errores gramaticales, jerga o incluso cambios de idioma dentro de una misma oración, lo que afecta la precisión de los modelos de clasificación y genera barreras para una automatización confiable. Por otro lado, investigaciones como las de Chehal et al. (2021) destacan que el análisis de sentimientos ha evolucionado hacia enfoques de mayor granularidad, como el análisis por aspecto, donde se asocia cada opinión a atributos específicos del producto, permitiendo una lectura más profunda y útil para decisiones prácticas. No obstante, tal como advierten Elzeheiry et al. (2023), aún persiste una brecha metodológica en la integración entre estas opiniones no estructuradas y los datos estructurados del negocio, lo que impide transformar el conocimiento generado en acciones concretas.

Desde el punto de vista de la Analítica de Negocios, este problema representa una oportunidad significativa para aplicar modelos de clasificación y análisis de sentimientos que permitan automatizar la interpretación de reseñas. Al integrar estos resultados con variables como tiempos de entrega, tipo de producto o método de pago, se pueden

generar alertas tempranas, mejorar la asignación de recursos y elevar la percepción del servicio al cliente (Deniz et al., 2022). La propuesta se alinea con los objetivos de la analítica descriptiva, predictiva y prescriptiva, al permitir no solo comprender lo que ocurre, sino anticiparse y tomar acción informada (Daza et al., 2024).

No obstante, este proyecto considera algunas restricciones importantes. Se trabajará con la base de datos de Olist (disponible en la plataforma Kaggle), la cual —si bien es amplio y representativo— presenta limitaciones como la falta de datos actualizados, ausencia de los nombres de productos o ubicación geográfica exacta. Además, se limitará el análisis al contenido textual en portugués, lo que implica restricciones en cuanto a la capacidad de generalización del modelo a otros idiomas o plataformas. Asimismo, se priorizarán modelos interpretables y viables para implementación, considerando un balance entre precisión, capacidad explicativa y recursos computacionales.

En resumen, el presente proyecto busca abordar un problema real y relevante para la industria del e-commerce: la necesidad de tomar decisiones más informadas y proactivas a partir del análisis automatizado de comentarios y calificaciones de los clientes, utilizando herramientas de NLP, clasificación y visualización integradas en una solución analítica con impacto práctico.

CAPÍTULO 3: HIPÓTESIS

El análisis del contenido emocional de los comentarios y la clasificación de las calificaciones realizadas por los clientes permiten identificar las variables más influyentes en la percepción del servicio y del producto, y ofrecen señales predictivas que facilitan la anticipación del comportamiento de compra de futuros clientes. Específicamente, se espera que:

Preguntas de Investigación

- ¿Qué factores asociados al proceso de compra pueden influir en la percepción y satisfacción del cliente en plataformas de e-commerce?
- ¿Hasta qué punto es posible predecir evaluaciones negativas de los clientes a partir de información disponible antes y después de la entrega?
- ¿En qué medida los comentarios de los clientes permiten comprender las causas subyacentes de su insatisfacción, considerando la ausencia de contacto directo entre comprador y vendedor?

CAPÍTULO 4: OBJETIVOS

Objetivo Principal

Desarrollar una metodología que permita identificar las principales características que impactan en las calificaciones de los clientes en plataformas de comercio electrónico, integrando variables logísticas, de compra y percepciones expresadas en los comentarios, con el fin de comprender los factores que afectan la experiencia del usuario.

Objetivos Específicos

1. Comprender cómo variables asociadas al proceso de compra y entrega inciden en la satisfacción o insatisfacción del cliente.
2. Explorar el vínculo entre el contenido emocional y temático de los comentarios y las calificaciones otorgadas por los usuarios.
3. Evaluar el potencial de modelos predictivos para anticipar reseñas negativas a partir de datos estructurados y no estructurados.
4. Identificar oportunidades de mejora en la experiencia del cliente a partir de los hallazgos del análisis, con foco en la gestión operativa y la comunicación con el usuario.

CAPÍTULO 5: METODOLOGÍA

El presente capítulo describe la metodología empleada para desarrollar un modelo predictivo que permita identificar las principales características que influyen en las calificaciones de productos en la plataforma de comercio electrónico Olist. El enfoque adoptado se estructura en etapas lógicas que abarcan desde la integración y transformación de datos estructurados y no estructurados, hasta la construcción y evaluación de modelos predictivos. Esta metodología busca responder a los objetivos planteados, integrando aspectos logísticos, del proceso de compra y del contenido de los comentarios, con el fin de comprender los factores que afectan la experiencia del usuario.

1. **Marco Conceptual**

   Se establecieron los fundamentos teóricos y técnicos del estudio, considerando la aplicación de modelos de clasificación binaria y técnicas de procesamiento de lenguaje natural (NLP) para el análisis de reseñas.

2. **Integración de Datos**

   Se consolidaron siete tablas de la base de datos de Olist mediante claves primarias para construir una base estructurada. Debido a la falta de granularidad geográfica, se excluyó la información sobre ubicaciones de bodegas.

3. **Análisis Exploratorio de Datos (EDA)**

   Se analizaron distribuciones, relaciones entre variables y patrones relevantes en las calificaciones. Se exploraron diferencias por categorías de productos, tiempos de entrega, y se generaron nubes de palabras diferenciadas según el tipo de reseña.

4. **Limpieza y Transformación de Datos**

   Se eliminaron valores nulos, registros inconsistentes y variables irrelevantes. Se construyeron variables derivadas como ‘delivery_difference_days’, ‘delivery_status’, volumen del producto y agregados por cliente para capturar patrones de comportamiento. Se agruparon las calificaciones 1 y 2 como negativas y 3 a 5 como positivas, en línea con estudios previos.

5. **Preprocesamiento de Comentarios**

   Se limpiaron y estandarizaron los textos eliminando ruido, corrigiendo ortografía y aplicando lematización. Esto permitió preparar los comentarios para su análisis semántico posterior.

6. **Clusterización Temática con BERTopic**

La clusterización temática con BERTopic se aplicó para identificar patrones no supervisados en las reseñas, transformando texto no estructurado en temas interpretables (ej.: 'problemas de logística', 'calidad del producto'). Esto permitió enriquecer el análisis predictivo al agregar variables categóricas derivadas de los *clusters* y priorizar focos de acción en el negocio, complementando el análisis de sentimiento. Se agruparon los comentarios mediante *embeddings* generados con Sentence-BERT y el algoritmo HDBSCAN. Se evaluaron distintas configuraciones de *clusters* y se seleccionó la más coherente según el coeficiente C_v.

7. Análisis de Resultados por Tema

Se interpretaron los temas identificados, relacionándolos con calificaciones promedio y proporción de reseñas negativas. Se extrajeron palabras clave por *clúster* para comprender mejor los focos de satisfacción e insatisfacción.

8. Aplicación de Técnicas de NLP para Análisis de Sentimiento

Se calcularon puntajes de sentimiento a partir de los comentarios en portugués utilizando dos modelos preentrenados: pysentimiento/bertweet-pt-sentiment (basado en BERTweet) y cardiffnlp/twitter-xlm-roberta-base-sentiment (basado en RoBERTa multilingüe). Ambos modelos fueron implementados mediante la librería transformersde Hugging Face y procesados por lotes para optimizar el uso de memoria. Las salidas de cada modelo asignaron probabilidades a categorías de sentimiento (positivo, negativo, neutro), que luego fueron transformadas en un puntaje continuo mediante la multiplicación del *score* por un valor numérico asociado a la clase (+1, 0, -1). Se evaluó la coherencia entre modelos comparando las etiquetas asignadas y la distribución de los puntajes generados. Finalmente, esta dimensión emocional fue incorporada como variable explicativa en el análisis de las calificaciones otorgadas por los usuarios.

9. Selección de Variables Relevantes

Se aplicó el método SelectKBest con función de información mutua para identificar las variables con mayor relevancia respecto a la variable objetivo. Este proceso permitió reducir la dimensionalidad del conjunto de datos y mejorar la interpretabilidad del modelo, al conservar únicamente las variables con mayor capacidad predictiva.

10. Construcción de Modelos Predictivos

Se entrenaron modelos de clasificación para predecir reseñas negativas, incluyendo Random Forest, XGBoost, AdaBoost y GradientBoosting, así como modelos basados en redes neuronales recurrentes (RNN) y LSTM. Para abordar el desbalance de clases, se aplicó la técnica SMOTE antes de la partición de los

datos, generando un conjunto balanceado de entrenamiento y prueba. La selección de hiperparámetros se realizó mediante GridSearchCV con validación cruzada de 3.

Posteriormente, los modelos con la mejor configuración de parámetros se evaluaron en un conjunto de prueba independiente para estimar su desempeño real, reportando métricas como precisión, F1-macro y F1-weighted.

11. Evaluación de Importancia de Variables con SHAP e Importancia de Características

Se calcularon valores SHAP para interpretar los modelos entrenados y analizar la importancia de las características en las predicciones. Dado que la arquitectura de los modelos varía, se utilizaron implementaciones específicas: TreeExplainerpara modelos de árboles como Random Forest, Gradient Boosting y AdaBoost; y DeepExplainer para redes neuronales como LSTM y RNN. Debido al alto costo computacional de este análisis, los valores SHAP se estimaron sobre una muestra representativa del conjunto de validación (X_sample) en lugar de utilizar la totalidad de los datos. Esta aproximación permitió obtener resultados interpretables y eficientes, identificando el impacto individual de cada variable sobre las predicciones de manera transparente.

12. Análisis Final y Propuesta de Valor

Se integraron los resultados del modelado y del análisis semántico para identificar factores críticos que afectan la experiencia del cliente. A partir de ello, se propusieron estrategias orientadas a la mejora del servicio.

CAPÍTULO 6: MARCO TEÓRICO

El análisis de sentimientos se ha consolidado como una herramienta fundamental en el contexto digital, especialmente en plataformas de comercio electrónico, donde las reseñas de productos reflejan las opiniones y emociones de los consumidores. Este enfoque permite identificar patrones emocionales, clasificar textos como positivos, negativos o neutros, y extraer información valiosa que guía la toma de decisiones estratégicas (Souza et al., 2022). Las aplicaciones del análisis de sentimientos en el comercio electrónico son diversas, abarcando desde la evaluación de la satisfacción del cliente y la mejora de estrategias de marketing hasta la detección de problemas recurrentes en productos o envíos y la personalización de la experiencia de compra (Roumeliotis et al., 2024; Chatterjee et al., 2021). Su relevancia radica en la capacidad de generar información que resulta esencial para que las empresas respondan con agilidad en mercados altamente competitivos y orienten sus decisiones hacia una experiencia de cliente más eficaz y personalizada.

Entre las técnicas utilizadas en análisis de sentimientos, las basadas en *machine learning* han sido ampliamente adoptadas debido a su simplicidad y efectividad en tareas de clasificación de texto. Algoritmos como *Support Vector Machines* (SVM), Naive Bayes y *K-Nearest Neighbors* (KNN) han demostrado ser herramientas robustas para analizar grandes volúmenes de datos textuales (Palanivinayagam et al., 2023; Areshey & Mathkour, 2023). Estos modelos utilizan representaciones matemáticas de las palabras, como *Term Frequency-Inverse Document Frequency* (TF-IDF) y *Bag of Words*, para identificar patrones y relaciones, logrando clasificaciones precisas con menor dependencia de datos etiquetados (Başarslan & Kayaalp, 2022). Sin embargo, su eficacia puede verse limitada al abordar problemas más complejos, como emociones sutiles, abreviaciones y detección del sarcasmo (Galal et al., 2024; Chatterjee et al., 2021).

En el análisis de Palanivinayagam et al. (2023), que incluye una revisión de 224 artículos científicos, se concluye que algoritmos como SVM, Naive Bayes y KNN configuran el núcleo metodológico del análisis de sentimientos en la actualidad. Entre ellos, destaca SVM, que ha alcanzado una precisión del 98.88%, posicionándose como una de las técnicas más eficaces. Sin embargo, este éxito se enfrenta a desafíos significativos, como la selección adecuada de características, el procesamiento de textos heterogéneos y el manejo de datos desbalanceados, aspectos críticos que aún requieren mejoras metodológicas.

Para abordar estas limitaciones y mejorar la precisión de los modelos de clasificación en el comercio electrónico, los modelos de ensamblaje han surgido como una alternativa eficaz. Estos combinan múltiples algoritmos de *machine learning* para mejorar la generalización y robustez de las predicciones. Métodos como *Bagging, Boosting* (como AdaBoost o XGBoost) y *Stacking* han sido ampliamente utilizados para integrar diferentes enfoques y reducir errores de clasificación (Chatterjee et al., 2021; Bello et al., 2024). En particular, su efectividad ha sido destacada en entornos donde los datos presentan alta variabilidad, ruido textual y clases desbalanceadas, condiciones comunes en reseñas de comercio electrónico (Galal et al., 2024; Bellar et al., 2024).

En el contexto del análisis de sentimientos en el comercio electrónico, los modelos de ensamblaje pueden ofrecer ventajas significativas. Por ejemplo, un modelo basado en *Bagging* con *Random Forest* puede mejorar la estabilidad de las predicciones al reducir la varianza y evitar el sobreajuste, como lo señalan Chatterjee et al. (2021) en su estudio sobre reseñas en plataformas de salud y bienestar. Por otro lado, técnicas de *Boosting*, como XGBoost o LightGBM, han demostrado ser efectivas para captar relaciones complejas dentro de los datos textuales, especialmente cuando se requiere asignar mayor peso a observaciones difíciles de clasificar (Palanivinayagam et al., 2023). Finalmente, el método de *Stacking* permite combinar de forma estratégica las predicciones de varios modelos base, aprovechando las fortalezas individuales de cada algoritmo para mejorar el rendimiento general del sistema, como se observa en los trabajos de Bellar et al. (2024) y Galal et al. (2024), quienes lograron resultados superiores combinando redes neuronales profundas y modelos de ensamblaje en grandes volúmenes de reseñas.

El estudio de Chatterjee et al. (2021) aplicó modelos de regresión y algoritmos de *machine learning* como XGBoost y *Random Forest* para analizar 186.057 reseñas de productos en el sector de salud y bienestar dentro del comercio electrónico. Los resultados evidenciaron que las emociones extraídas de los textos —como alegría, ira y miedo— son predictores clave de la satisfacción del cliente. Emociones positivas como la alegría tienen un impacto significativo en las calificaciones, mientras que emociones negativas como la ira y el miedo se asocian fuertemente a reseñas desfavorables. Asimismo, variables como el producto, el servicio de entrega y el tiempo de llegada emergen como factores determinantes en la evaluación del cliente, con diferencias relevantes según la subcategoría analizada. Estos hallazgos coinciden con lo reportado por Ravula (2022), quien demuestra que los retrasos en la entrega afectan negativamente las valoraciones de los clientes y su satisfacción general en plataformas digitales.

Por otro lado, las técnicas de aprendizaje profundo han revolucionado el análisis de sentimientos al introducir modelos capaces de capturar relaciones más complejas y contextuales en los textos. Modelos como las redes neuronales de memoria a largo plazo (LSTM) y los transformadores pre-entrenados, como BERT, han elevado los estándares en el procesamiento de lenguaje natural. Estas arquitecturas no solo comprenden el significado individual de las palabras, sino también su interacción dentro de las oraciones, permitiendo análisis más precisos y contextualmente informados. Este enfoque es especialmente útil para identificar matices lingüísticos en reseñas y comentarios de los usuarios, como lo demuestran estudios como los de Areshey y Mathkour (2023), quienes muestran que un modelo BERT *fine-tuned* supera sistemáticamente a algoritmos tradicionales como SVM, KNN y Naive Bayes en precisión de clasificación de sentimientos. Asimismo, Bellar et al. (2024) confirman que el uso combinado de *embeddings* contextuales y arquitecturas profundas mejora sustancialmente el rendimiento predictivo en bases de datos grandes y heterogéneos.

Los *Large Language Models* (LLM) ofrecen variantes como las versiones base y large, que difieren en complejidad, número de parámetros y capas de atención. Las versiones base son más livianas, eficientes en términos de recursos computacionales y

adecuadas para tareas menos exigentes, mientras que las versiones large tienden a ofrecer mayor precisión, especialmente en tareas complejas de análisis semántico (Souza & Souza Filho, 2023). Modelos como BERT, GPT, T5 (Text-To-Text Transfer Transformer) y RoBERTa incluyen estas configuraciones, lo que permite ajustarlos según las necesidades del problema. Según Roumeliotis et al. (2024), este tipo de configuraciones ha permitido adaptaciones efectivas de *Large Language Models* a dominios específicos como el comercio electrónico, alcanzando mejoras de rendimiento significativas con técnicas de *fine-tuning* en diferentes idiomas y contextos comerciales.

BERT representa un avance significativo en el campo del procesamiento de lenguaje natural al introducir un modelo de lenguaje bidireccional pre-entrenado que ha establecido nuevos estándares de rendimiento en tareas como clasificación de texto, análisis de sentimientos y respuesta a preguntas. Una de sus innovaciones clave es la capacidad de preentrenamiento profundo bidireccional, lograda mediante dos tareas fundamentales: el *Masked Language Model*(MLM) y la *Next Sentence Prediction* (NSP), que permiten al modelo desarrollar una comprensión más rica del contexto (Devlin et al., 2019; Areshey & Mathkour, 2023). Este enfoque ha demostrado superar modelos unidireccionales en múltiples indicadores de referencia de NLP. Además, el tamaño del modelo también influye significativamente en su rendimiento, con BERTLARGE superando consistentemente a BERTBASE, incluso en conjuntos de datos pequeños (Souza & Souza Filho, 2023). En cuanto a su aplicabilidad práctica, BERT ha mostrado ser eficaz tanto en enfoques de *fine-tuning* como en arquitecturas *feature-based*, y puede entrenarse en poco tiempo utilizando recursos moderados, lo que lo convierte en una solución viable para aplicaciones comerciales y académicas. La eficacia de modelos como BERT se debe a su arquitectura bidireccional y su entrenamiento previo en grandes corpus, lo cual permite capturar relaciones semánticas complejas con alta precisión (Devlin et al., 2019).

Una pieza central en estas técnicas es el concepto de *embeddings* de texto, que transforma palabras, frases o documentos en representaciones vectoriales que capturan su significado semántico. Métodos como Word2Vec y GloVe introdujeron esta idea, permitiendo que los algoritmos comprendieran similitudes entre palabras en función de su contexto de uso. Sin embargo, los avances recientes han sido liderados por modelos como BERT, cuyos *embeddings* contextualizados permiten representar dinámicamente el significado de las palabras según su entorno dentro de la oración (Areshey & Mathkour, 2023; Zhang et al., 2023). Estos vectores contextuales han demostrado mejorar significativamente la precisión en tareas de clasificación de sentimientos, especialmente en dominios como el comercio electrónico donde el lenguaje informal, las abreviaciones y la subjetividad son frecuentes. Estudios como el de Souza y Souza Filho (2023) confirman que los *embeddings* generados por modelos pre-entrenados específicos para el portugués, como BERTimbau y PTT5, ofrecen desempeños superiores en reseñas de productos en Brasil, con métricas de hasta 99.4% de AUC y 97% de precisión en distintos conjuntos de datos.

Dentro de este marco, los modelos de lenguaje pre-entrenados como BERT, GPT y LLaMA han emergido como pilares del procesamiento de texto. Estos modelos están diseñados para capturar el conocimiento lingüístico a gran escala mediante el entrenamiento en enormes volúmenes de datos, lo que les permite generalizar patrones

del lenguaje y adaptarse a tareas específicas mediante técnicas de *fine-tuning*. Su capacidad para ajustarse a contextos particulares, como el comercio electrónico, ha sido ampliamente demostrada en investigaciones recientes. Por ejemplo, Huang, Asemi y Mustafa (2023) destacan que estos modelos han impulsado el desarrollo de herramientas automatizadas más precisas y flexibles, capaces de adaptarse a flujos de datos en tiempo real en entornos comerciales digitales. Esta capacidad de transferencia y especialización convierte a los modelos pre-entrenados en herramientas esenciales para abordar desafíos complejos como la detección de sarcasmo, la clasificación multilingüe y la interpretación emocional en textos generados por usuarios.

El análisis de sentimientos ha cobrado gran relevancia en el ámbito del comercio electrónico, no solo por su utilidad práctica en la interpretación de opiniones de los clientes, sino también por el creciente desarrollo de técnicas especializadas. En su revisión, Huang, Asemi y Mustafa (2023) muestran que la investigación en este campo ha evolucionado equilibradamente entre enfoques de machine learning (48%) y *deep learning* (44%), destacando a Amazon y Twitter como principales fuentes de datos. Este panorama evidencia el interés académico por perfeccionar herramientas que permitan capturar de forma automatizada las percepciones del consumidor y generar valor en contextos comerciales digitales.

Las innovaciones recientes han abierto nuevas posibilidades para el análisis de sentimientos en el comercio electrónico. Galal et al. (2024) desarrollaron arquitecturas basadas en BERT que introducen estrategias de congelamiento de capas durante el entrenamiento, logrando optimizar tanto el rendimiento como la eficiencia computacional . En su estudio, se alcanzó un F1-score de 64.41 en la detección de sarcasmo y un FPN (promedio de F1-score para clases positivas y negativas) de 75.26 en clasificación de sentimientos, estableciendo nuevos estándares frente a técnicas tradicionales de *fine-tuning.* Este enfoque demuestra que es posible reducir considerablemente los tiempos de entrenamiento sin sacrificar precisión, especialmente en contextos donde los matices del lenguaje, como la ironía o las emociones ambiguas, son críticos. Así, las estrategias de optimización aplicadas a modelos como BERT no solo mejoran los resultados, sino que también contribuyen a su viabilidad en entornos productivos de análisis masivo de reseñas.

En esta misma línea, Bellar et al. (2024) exploraron el potencial de los modelos de ensamblaje al combinar arquitecturas como CNN, RNN y Bi-LSTM para analizar un extenso conjunto de 22.641 reseñas de productos en comercio electrónico. Esta combinación permitió capturar tanto patrones locales como dependencias a largo plazo en el texto, logrando una precisión del 96.2%. Los resultados demuestran que los modelos híbridos pueden superar a enfoques individuales, especialmente cuando se trabaja con grandes volúmenes de datos y alta variabilidad lingüística. Además, esta investigación refuerza la idea de que los modelos de ensamblaje no solo son eficaces en clasificación de sentimientos, sino también adaptables a diferentes dominios dentro del comercio electrónico, desde productos electrónicos hasta artículos de consumo masivo.

El procesamiento multilingüe también está experimentando avances significativos. Lopes et al. (2022) utilizaron BERTimbau, un modelo optimizado para portugués,

logrando métricas destacadas en análisis de reseñas. Por su parte, Roumeliotis et al. (2024) exploraron el potencial de *Large Language Models* como GPT-3.5 y LLaMA-2, que con técnicas de *fine-tuning* lograron incrementos de precisión de hasta un 10.99%. Estos modelos también demuestran ser herramientas versátiles para adaptarse a contextos multilingües y dominios específicos.

El trabajo de Souza y Souza Filho (2023) ofrece una contribución fundamental al análisis en portugués brasileño, investigando diferentes estrategias de generación de *embeddings* para clasificación de texto. Su estudio reportó métricas excepcionales, como un ROC-AUC de 99.4% en la base de datos B2W y un 98.4% en Olist. Los Modelos de Lenguaje Basados en Transformers (TLM), especialmente aquellos entrenados en cuerpos de texto específicos como BERTimbau y PTT5, demostraron ser altamente efectivos, alcanzando precisiones superiores al 97% en varios conjuntos de datos. La investigación también reveló que las versiones *Large* de los modelos superan a las versiones *Base*, destacando la importancia de la complejidad del modelo en el rendimiento predictivo.

Sin embargo, a pesar de los avances significativos, los desafíos persisten. Entre ellos, el desarrollo de modelos universales que puedan aplicarse a múltiples idiomas y dominios. La detección de sarcasmo, una tarea compleja, requiere técnicas más avanzadas para identificar patrones sutiles en los textos. Además, la clasificación emocional con mayor granularidad —es decir, distinguir entre emociones específicas como frustración, decepción o entusiasmo— es un campo aún en desarrollo, con escaso consenso sobre las métricas más adecuadas para evaluarlo (Ravula, 2022). Otro reto crítico es la optimización de recursos computacionales, especialmente al trabajar con modelos de gran tamaño como BERT *Large* o LLaMA, que requieren infraestructuras potentes para su entrenamiento y despliegue. Si bien técnicas como el congelamiento de capas o la destilación han ayudado a mitigar estas exigencias, aún se requieren soluciones más accesibles para su implementación en contextos comerciales o institucionales con recursos limitados (Areshey & Mathkour, 2023; Zhang et al., 2023).
Al integrar algoritmos supervisados tradicionales, arquitecturas neuronales complejas y modelos de lenguaje pre-entrenados multilingües, los investigadores y profesionales están logrando una comprensión más profunda del comportamiento del consumidor. Esta evolución no solo amplía el horizonte de lo técnicamente posible, sino que también abre nuevas oportunidades para la personalización masiva, la mejora de servicios y la toma de decisiones estratégicas fundamentadas en datos emocionales y contextuales.

CAPÍTULO 7: DESCRIPCIÓN DEL ENFOQUE DE SOLUCIÓN

El enfoque propuesto tiene como objetivo predecir el 'review_score' (de 1 a 5) mediante modelos de clasificación binaria (calificaciones positivas y negativas), utilizando técnicas avanzadas como *machine learning*, redes neuronales y BERT. Este trabajo explora múltiples factores que pueden influir en las calificaciones negativas, incluyendo características del producto, métodos de pago, precio del producto y el contenido emocional de los comentarios de los clientes. Para ello, se calcula un puntaje del sentimiento utilizando BERT y otros modelos, permitiendo un análisis más profundo de las opiniones expresadas en las reseñas.

Debido a la falta de información sobre la ubicación exacta de las bodegas y los nombres específicos de los productos, las variables geoespaciales y las recomendaciones personalizadas de productos quedan fuera del alcance de este estudio. No obstante, el análisis se centra en variables clave como el tiempo de entrega, la categoría del producto y el sentimiento extraído de los comentarios, proporcionando una visión integral de los factores que aumentan o reducen las calificaciones negativas.

CAPÍTULO 8: RESULTADOS

En este capítulo se presentan los principales hallazgos obtenidos a partir del análisis exploratorio de datos (EDA) y la modelación predictiva. El objetivo es identificar patrones relevantes en el comportamiento de los clientes y evaluar los factores que inciden en las calificaciones recibidas por los productos en la plataforma digital para compradores y vendedores Olist.

Exploración de Datos (EDA)

A continuación, se realiza un análisis exploratorio de los datos, utilizando gráficos y métricas descriptivas para identificar tendencias, correlaciones y posibles anomalías en las órdenes, entregas y evaluaciones de los clientes:

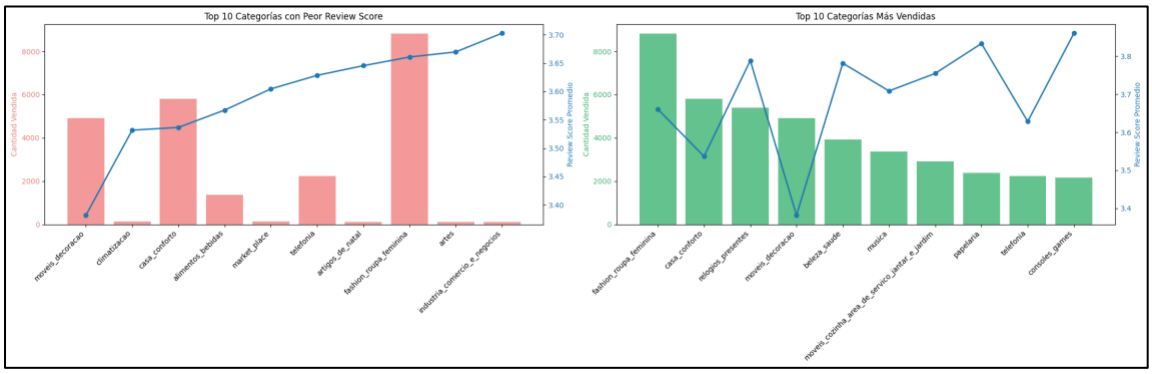

Imagen 1: Gráfico de categorías de producto con peor 'review score' y gráfico con las diez categorías de producto con más ventas y sus respectivos 'review score'.

De acuerdo con la Imagen 1, se observa que, dentro de las categorías con peor puntaje de reseñas, muchas concentran proporciones pequeñas del total de la base de datos. Las más relevantes son *moveis_decoracao*, *casa_conforto*, *telefonia* y *fashion_roupa_feminina*, las cuales presentan calificaciones más bajas entre todas las categorías de productos de la plataforma Olist. Dentro de ellas, la categoría *moveis_decoracao* destaca por tener la peor puntuación de todas.

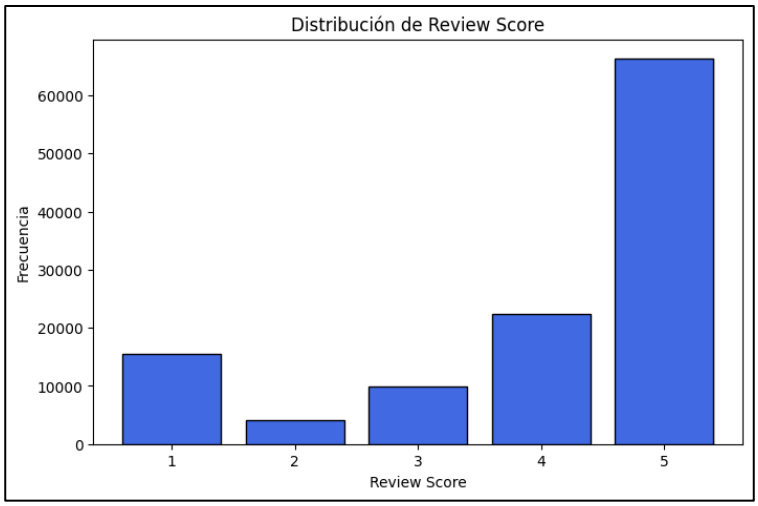

Imagen 2: Distribución del puntaje de reseñas.

De acuerdo con la Imagen 2, se distingue que la mayoría de las órdenes realizadas en la plataforma de comercio electrónico cuentan con una calificación igual o superior a 3, y que gran parte de las calificaciones alcanzan la nota máxima de 5.

En la Imagen 3, se observa que la mayoría de los valores de 'delivery_difference_days' son negativos, lo que indica que la mayoría de los pedidos se entregan antes de la fecha estimada. Se destaca que una proporción importante de pedidos se entrega con varios días de anticipación respecto a lo programado. Además, el segundo gráfico muestra, a través de un gráfico de cajas, que los puntaje de reseña altos (buenas calificaciones) presentan una media y una mediana de 'delivery_difference_days' más negativa. Esto podría sugerir que los adelantos en las entregas favorecen mejores calificaciones.
Se puede consultar el Anexo D y F para ver gráficos adicionales y código usado, respectivamente.

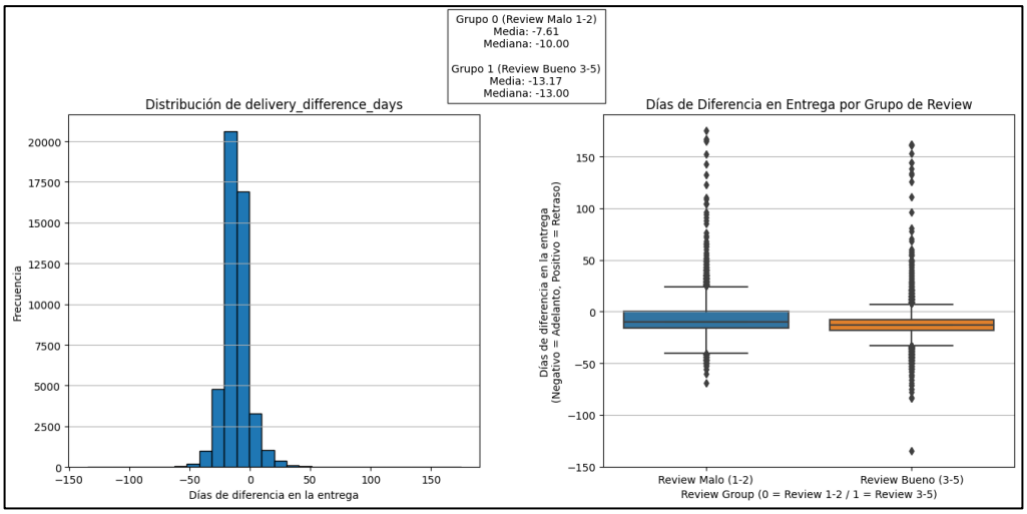

Imagen 3: Distribución de la variable 'delivery_difference_days' y gráfico de caja de 'delivery_difference_days' para reseñas negativos y positivos.

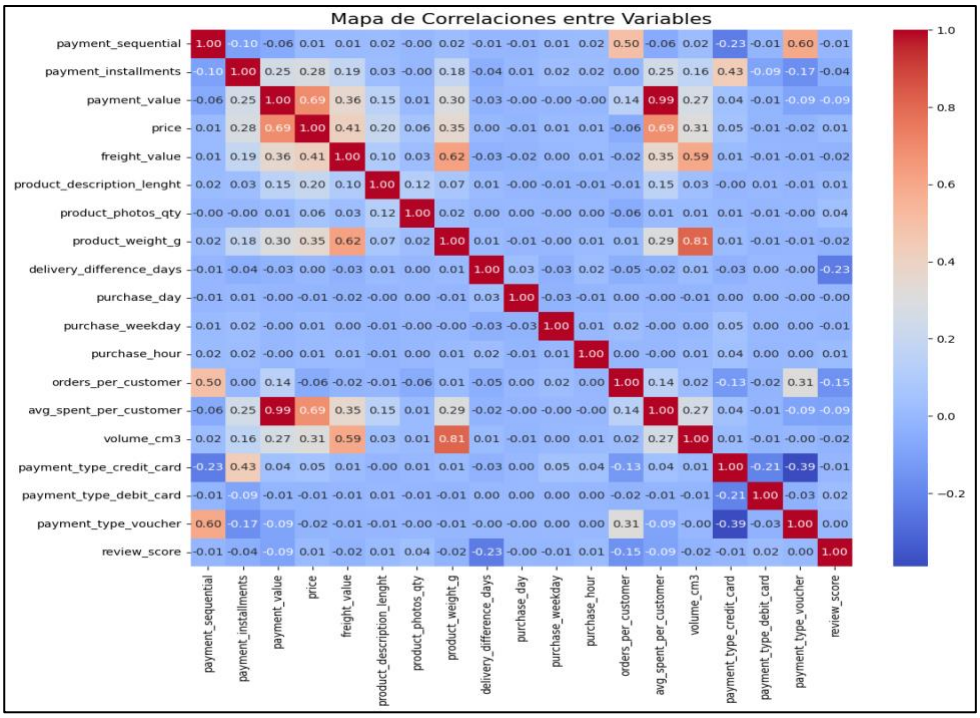

Imagen 4: Mapa de correlaciones entre variables numéricas.

De acuerdo a la Imagen 4, la mayor parte de las variables no tienen correlaciones altas entre sí. Pero variables numéricas como ‘avg_spent_per_customer’ y ‘payment_value’, ‘price’ están altamente correlacionadas. De igual manera, variables como ‘volumen_cm3’, ‘freight_value’ y peso del producto también están muy correlacionadas.

Imagen 5: Nubes de palabras asociados a los comentarios, agrupados para ‘review score’ negativos y positivos.

De acuerdo a las nubes de palabras construidas en base a los comentarios de los clientes y separados por puntaje de reseñas (negativos o positivos), se distinguen palabras en cada grupo. La nube de palabras generada a partir de los comentarios asociados a calificaciones negativas (1 y 2) muestra términos y frases frecuentes como “*não chegou*”, “*defeito*”, “*devolucão”* y “*não gostei*”. En cambio la nube de palabras asociado a puntajes positivos, destaca por expresiones como “*muito bom*”, “*chegou ante*”, “recomendó”, “*bem embalado*”.

Preparación de Datos e Ingeniería de Características

Una vez realizada la integración de las distintas fuentes de información y el análisis exploratorio, se procedió a la limpieza, transformación y selección de variables de la base de datos original. Inicialmente, el conjunto de datos contaba con 119.143 registros y 39 atributos relacionados con clientes, pedidos, productos, pagos, reseñas y vendedores. Se detectaron gran cantidad de valores faltantes en variables como 'review_comment_message', con aproximadamente 69.000 registros nulos. Se puede ver descripción detallada de variables usadas y análisis de datos faltantes inicial en el Anexo A.

Como parte del preprocesamiento:

1. Se consolidaron los comentarios unificando 'review_comment_title' y 'review_comment_message'.

2. Se eliminaron variables irrelevantes, como identificadores únicos y atributos geográficos de baja granularidad.

3. Se creó la variable 'volume_cm3' a partir de las dimensiones de los productos.

4. Se agruparon categorías de productos utilizando la distancia de Levenshtein, reduciendo inconsistencias tipográficas.

5. Se seleccionaron las 10 variables más relevantes mediante el método SelectKBest, basándose en el criterio de información mutua. Las variables seleccionadas fueron: 'payment_value', 'seller_id', 'price', 'freight_value', 'product_description_length', 'delivery_difference_days', 'delivery_status', 'orders_per_customer', 'avg_spent_per_customer' y 'volume_cm3'.

Para asegurar la calidad de las predicciones, se aplicaron técnicas fundamentales de preparación de datos:

Codificación de Variables Categóricas:

Todas las variables categóricas se codificaron mediante *LabelEncoder*, que asigna un valor numérico único (0,1,2,…) a cada categoría distinta. Si bien en teoría se recomienda *one-hot encoding* para variables nominales sin orden (ejemplo: categorías de productos) y LabelEncoder para ordinales (ejemplo: estados de entrega), se optó por este enfoque unificado debido a: la alta cardinalidad de atributos como 'seller_id' que genera matrices dispersas inmanejables con *one-hot encoding*, y la compatibilidad con modelos basados en árboles (Random Forest, XGBoost), que interpretan estas codificaciones como etiquetas discretas sin asumir relaciones ordinales artificiales, manteniendo el rendimiento computacional.

Normalización de Variables Numéricas:

Todas las variables numéricas fueron escaladas al rango [0,1] mediante MinMaxScaler. Esta estandarización es esencial cuando las características tienen escalas dispares, como por ejemplo, 'price' (en miles de pesos brasileños BRL) podría dominar artificialmente sobre 'volume_cm3' (en decenas) en algoritmos sensibles a magnitudes como las redes neuronales.

Balanceo de Variables Numéricas:

La base de datos presentaba un desbalance notorio (calificaciones buenas y malas), lo que podía sesgar los modelos hacia la clase mayoritaria. Para mitigar esto, se aplicó SMOTE, (*Synthetic Minority Over-sampling Technique).* Este método genera muestras sintéticas de la clase minoritaria (reseñas negativas). Esta estrategia equilibró ambas clases a una proporción 1:1, mejorando la capacidad del modelo para predecir correctamente las reseñas negativas sin comprometer su generalización.

Partición de Datos:

La división de la base de datos se realizó en conjuntos de entrenamiento (80%) y prueba (20%) ,conservando la proporción original de clases en ambos subconjuntos. Esta técnica es crítica para garantizar que el conjunto de prueba sea representativo de la distribución real, y evitar escenarios donde alguna clase (ejemplo: reseñas negativas) quede subrepresentada en la fase de validación, lo que distorsionaría las métricas de evaluación. Los fragmentos de código correspondientes a las transformaciones descritas pueden revisarse en el Anexo D.

CAPÍTULO 9: MODELACIÓN Y ANÁLISIS

El objetivo principal de esta etapa es evaluar la capacidad predictiva de distintos modelos de *machine learning* y *deep learning* para identificar reseñas negativas en el contexto del comercio electrónico, utilizando variables cuantitativas relacionadas con el proceso de compra y complementándolas, en algunos casos, con el análisis de sentimientos extraído de los comentarios de los usuarios.

Para llevar a cabo este análisis, se desarrollaron tres enfoques de modelación:

1. **Modelo sin análisis de sentimientos**: se utilizaron principalmente variables cuantitativas como 'payment_value', 'price', 'freight_value', 'delivery_difference_days','delivery_status','orders_per_customer','avg_spent_per_customer', 'volume_cm3'. Además, se consideró la variable nominal 'seller_id'.

2. **Modelo con análisis de sentimientos utilizando Pysentimiento**: se incorporó la variable 'sentiment_score_pysentimiento', obtenida mediante un modelo BERT especializado en portugués, que captura la polaridad y las emociones expresadas en los comentarios.

3. **Modelo con análisis de sentimientos utilizando RoBERTa**: se integró la variable 'sentiment_score_roberta', generada por un modelo de lenguaje avanzado capaz de comprender el contexto y los matices lingüísticos con mayor profundidad.

La inclusión del análisis de sentimientos en los modelos resulta clave para capturar la percepción subjetiva de los clientes, ya que las emociones expresadas en los comentarios enriquecen el proceso predictivo y permiten identificar patrones de insatisfacción que las variables cuantitativas, por sí solas, podrían no reflejar. Esto se traduce en una mejora significativa en la capacidad de los modelos para predecir reseñas negativas, proporcionando una visión más completa y precisa para la toma de decisiones estratégicas.

Elección de mejor modelo para cálculo de puntaje de sentimiento de columna de comentarios

Para el cálculo del puntaje de sentimiento de los comentarios asociados a las calificaciones se consideraron cinco modelos: pysentimiento/bertweet-pt-sentiment (PySentimiento), cardiffnlp/twitter-xlm-roberta-base-sentiment (RoBERTa), nlptown/bertbase-multilingual-uncased-sentiment (BETO), TextBlob y VADER. La elección del modelo a incluir en el cálculo del puntaje de sentimiento se basó tanto en métricas de regresión como de clasificación. Se evaluaron métricas como la correlación de Pearson y Spearman, el error absoluto medio (MAE), el

error cuadrático medio (RMSE), así como la precisión y el F1-score binario. Ver código utilizado para la implementación y evaluación de modelos de sentimiento en el Anexo F.

Las correlaciones de Pearson y Spearman permitieron medir el grado de relación entre los puntajes de sentimiento generados por los modelos y la variable 'review_score'. Por su parte, MAE y RMSE cuantificaron la distancia entre los valores predichos y los reales, proporcionando una medida del error absoluto y del error cuadrático. Finalmente, con la precisión y F1-Score se evaluó la capacidad de los modelos para clasificar correctamente los 'review_score' en dos categorías: positivo (3-5) o negativo (1-2).

De acuerdo con los resultados de la Tabla 1, el modelo PySentimiento mostró la correlación más alta entre los puntajes de sentimiento obtenidos y los 'review_score' de la base de datos. El modelo RoBERTa también entregó resultados similares. Asimismo, los valores de MAE y RMSE de PySentimiento fueron los más bajos del conjunto evaluado. En términos de precisión y F1-score, el modelo que presentó mejor desempeño fue RoBERTa, seguido de BETO. Tanto Vader como TextBlob presentaron un desempeño notoriamente inferior en comparación con los otros modelos analizados.

Modelo	Pearson	Spearman	MAE	RMSE	R^2	Accuracy	F1-Score	Versión del Modelo
PySentimiento	0.627	0.616	0.9	1.275	0.369	0.696	0.745	pysentimiento/bertweet-pt-sentiment
BETO	0.562	0.54	0.996	1.607	-0.004	0.717	0.778	nlptown/bert-base-multilingual-uncased-sentiment
RoBERTa	0.593	0.587	1.033	1.313	0.33	0.773	0.857	cardiffnlp/twitter-xlm-roberta-base-sentiment
VADER	0.151	0.141	1.62	1.756	-0.198	0.335	0.208	SentimentIntensityAnalyzer( )
TextBlob	0.156	0.168	1.665	1.789	-0.243	0.339	0.202	blob.sentiment.polarity

Tabla 1: Cálculo de puntaje de sentimiento a partir de los comentarios mediante distintos modelos y métricas de performance frente a su relación con la característica 'review_score' del dataset.

A continuación, se presentan los resultados de los mejores modelos de clasificación binaria obtenidos mediante la técnica GridSearch para algoritmos basados en árboles y redes neuronales. Las Tablas 2, 3 y 4 muestran los resultados para la modelación sin comentarios, la modelación con el puntaje de sentimiento de PySentimiento y la modelación con el puntaje de sentimiento de RoBERTa, respectivamente. Los fragmentos de código utilizados para la implementación de los modelos y su ajuste de hiperparámetros están disponibles en el Anexo F.

Resultados de Métricas de Modelos de Clasificación Binaria Para Los Tres Escenarios

A continuación, se presentan los resultados obtenidos en los tres enfoques de modelación binaria del 'review_score', utilizando algoritmos basados en árboles y redes

neuronales. Se evaluaron métricas de desempeño como precisión, F1-Macro y F1-Weighted. Los fragmentos de código correspondientes al entrenamiento de modelos, selección de hiperparámetros y evaluación de métricas pueden consultarse en el Anexo F.

Modelo	Mejores Hiperparámetros	Precisión	F1 Macro	F1 Weighted
Random Forest	{'max_depth': None, 'min_samples_split': 2, 'n_estimators': 200}	0.875	0.875	0.875
Gradient Boosting	{'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 200}	0.851	0.851	0.851
AdaBoost	{'learning_rate': 1, 'n_estimators': 100}	0.688	0.686	0.69
XGBoost	{'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 200}	0.820	0.82	0.82
LSTM	LSTM architecture with 3 layers (128, 64, 64) and Adam optimizer	0.771	0.634	0.740
RNN	SimpleRNN with 64 units, Dense(32), Adam optimizer	0.777	0.650	0.75

Tabla 2. Resultados modelos de árboles y redes neuronales en clasificación binaria de 'review_score' sin incluir columna comentarios.

Modelo	Mejores Hiperparámetros	Precisión	F1 Macro	F1 Weighted
Random Forest	{'max_depth': None, 'min_samples_split': 2, 'n_estimators': 200}	0.911	0.911	0.911
Gradient Boosting	{'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 200}	0.889	0.889	0.889
AdaBoost	{'learning_rate': 1, 'n_estimators': 100}	0.836	0.8362	0.836
XGBoost	{'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 200}	0.881	0.8809	0.881
LSTM	LSTM with (128, 64, 64), Adam optimizer, dropout 0.2	0.823	0.768	0.822
RNN	SimpleRNN with 64 units, Dense(32), Adam optimizer	0.823	0.783	0.832

Tabla 3: Resultados modelos de árboles y redes neuronales en clasificación binaria de 'review_score' con puntaje de sentimiento de modelo pysentimiento.

Modelo	Mejores Hiperparámetros	Precisión	F1 Macro	F1 Weighted
Random Forest	{'max_depth': None, 'min_samples_split': 2, 'n_estimators': 200}	0.907	0.907	0.907
Gradient Boosting	{'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 200}	0.883	0.883	0.883
AdaBoost	{'learning_rate': 1, 'n_estimators': 100}	0.829	0.828	0.828
XGBoost	{'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 200}	0.873	0.873	0.873
LSTM	LSTM with (128, 64, 64), Adam optimizer, dropout 0.2	0.807	0.748	0.806
RNN	SimpleRNN with 64 units, Dense(32), Adam optimizer	0.825	0.770	0.823

Tabla 4: Resultados modelos de árboles y redes neuronales en clasificación binaria de 'review_score' con puntaje de sentimiento de modelo RoBERTa.

De acuerdo con los resultados presentados en las Tablas 2, 3 y 4, se observa que la inclusión del puntaje de sentimiento en los modelos mejora significativamente su desempeño, tanto utilizando PySentimiento como RoBERTa.

En particular, el mejor desempeño global se logró al utilizar el puntaje de sentimiento de PySentimiento, que obtuvo mayores niveles de precisión en la mayoría de los casos. Aunque RoBERTa también contribuyó a mejorar el rendimiento en comparación con el escenario sin comentarios, PySentimiento muestra una ligera ventaja en los principales modelos.

El algoritmo Random Forest fue el modelo con mejor rendimiento en todas las configuraciones, destacando especialmente en el escenario que incluye el puntaje de sentimiento de Pysentimiento, donde alcanzó una precisión de 0,911. Tanto Gradient Boosting como XGBoost también experimentaron mejoras importantes al incorporar información proviniendo del análisis de sentimientos. En cuento a AdaBoost, presento el mayor incremento relativo, aumentando su precisión de 0,688 a 0,836 con la incorporación de PySentimiento.

Por otro lado, aunque las redes neuronales (LSTM y RNN) también mejoraron su desempeño al incluir el puntaje de sentimiento, su incremento fue más moderado en comparación con los modelos basados en árboles. Random Forest destaca especialmente por su robustez y su capacidad para manejar datos tabulares con ruido. AdaBoost, en cambio, es más sensible a datos mal clasificados, lo cual explica su menor desempeño. Finalmente, las redes neuronales no logran igualar la capacidad de los modelos de árboles para capturar relaciones complejas en datos tabulares, posiblemente debido a que requieren arquitecturas más sofisticadas para detectar patrones de este tipo de información.

Importancia de Características en Modelos de Árboles sin Puntaje de Sentimiento

Con el objetivo de comprender el impacto relativo de cada variable en la predicción de calificaciones negativas, se calcularon los valores de importancia de características (*feature importance*) para todos los modelos de árboles entrenados en los tres escenarios considerados. Este análisis permite identificar cuáles variables estructuradas y textuales (como el sentimiento del comentario) aportan mayor información al modelo en cada caso, entregando una visión comparativa sobre los factores que más influyen en la experiencia del usuario desde una perspectiva operativa y perceptiva.

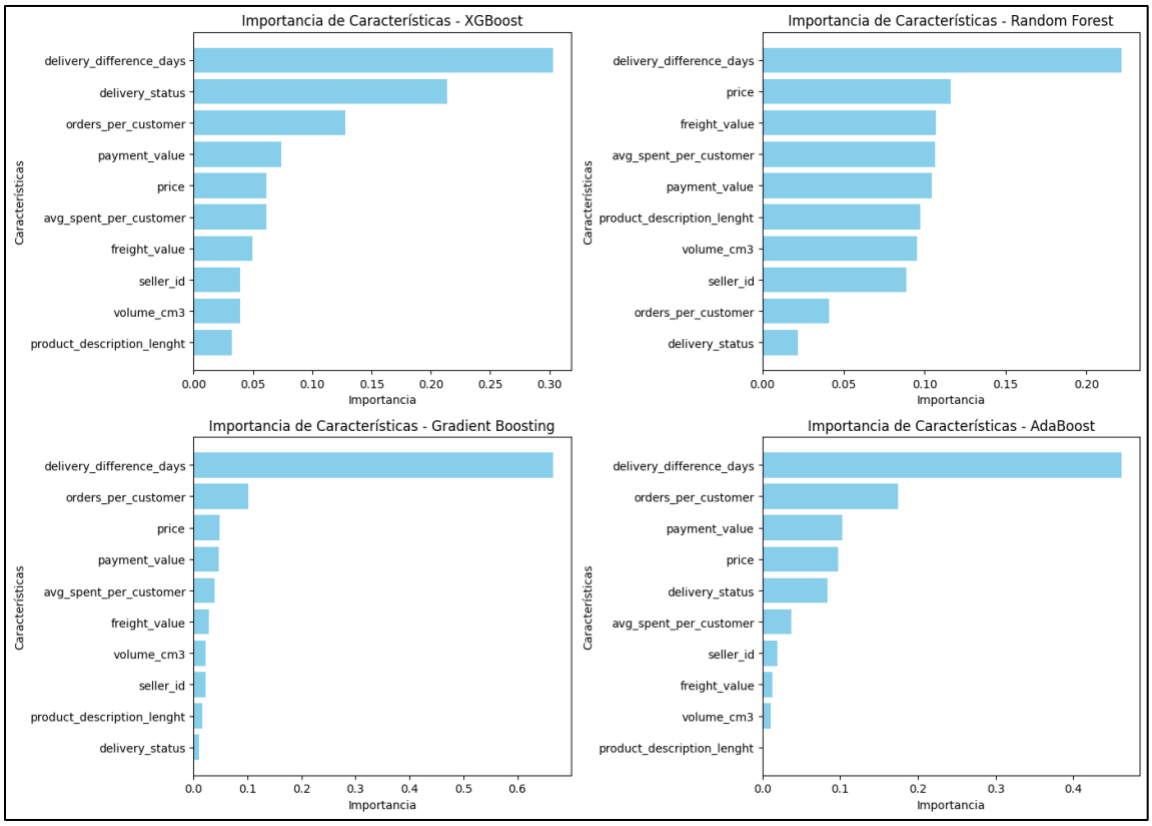

Imagen 6: Importancia de características para modelos de árboles en escenario donde no se incluye el puntaje de sentimiento.

De acuerdo con la Imagen 6, se observa que la característica con mayor influencia en la predicción de los modelos de clasificación basados en árboles fue ‘delivery_difference_days’, especialmente en los modelos Gradient Boosting y AdaBoost. El atributo ‘orders_per_customer’ también tuvo un peso importante en las predicciones.

En el modelo Random Forest, en cambio, la variable más relevante resultó ser ‘price’.

Importancia de Características en Modelos de Árboles con Sentiment_score_pysentimiento

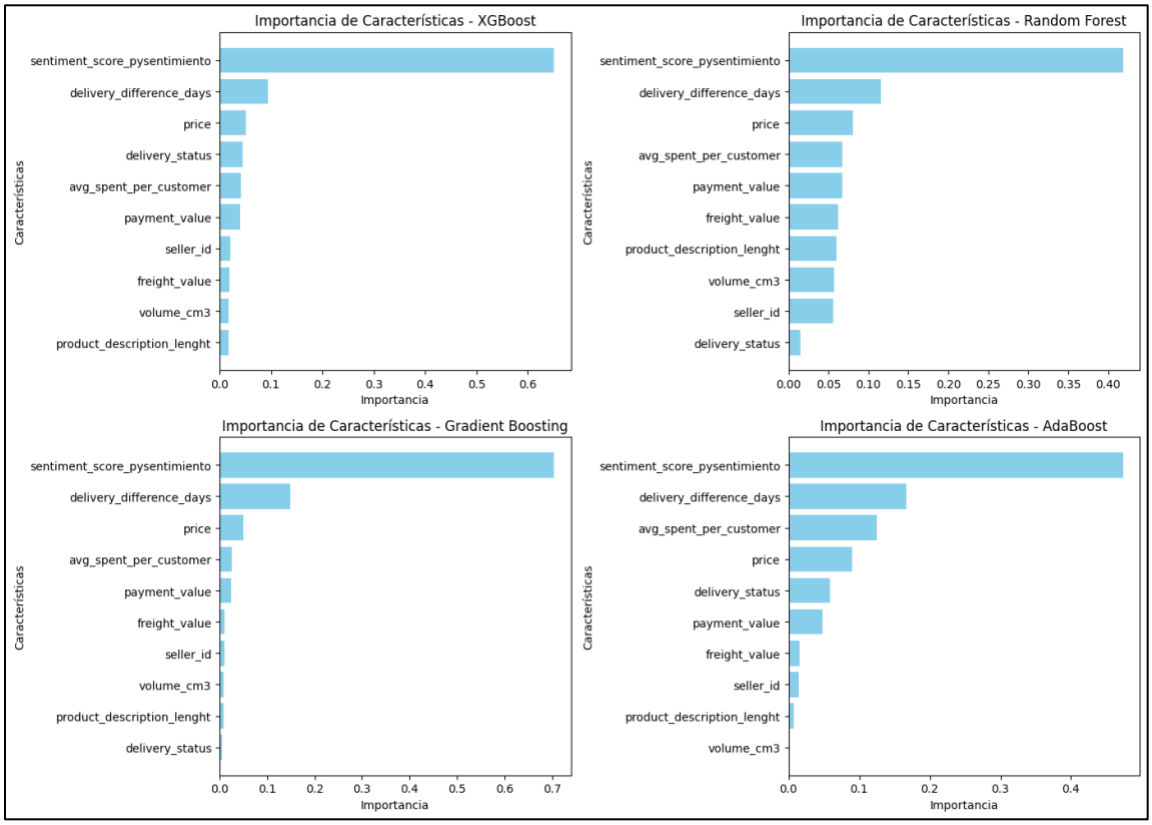

Imagen 7: Importancia de características para modelos de árboles que incluyen puntaje de sentimiento calculado por Pysentimiento.

De acuerdo a la Imagen 7, se observa que las variables con mayor magnitud en las predicciones obtenidas por los modelos de árboles fueron ‘sentiment_score_pysentimiento’ y ‘delivery_difference_days’. En tercer lugar se ubicó ‘price’, seguido en menor medida por ‘avg_spent_per_customer’. Se destaca que el puntaje de sentimiento calculado por el modelo Pysentimiento tuvo una influencia predominante en las métricas de predicción de los modelos.

Este resultado es coherente dado que el análisis de sentimientos refleja directamente la percepción del cliente sobre su experiencia de compra, lo que lo hace un predictor clave de la calificación final.

Por su parte, el tiempo de entrega impacta significativamente en la satisfacción del usuario, dado que los retrasos o adelantos pueden afectar la percepción del servicio. En cuanto a 'price', aunque influye en la experiencia del cliente, su efecto en la evaluación final puede depender de factores como la relación calidad-precio, las expectativas previas o promociones vigentes.

Importancia de Características en Modelos de Árboles con Sentiment_score_roberta

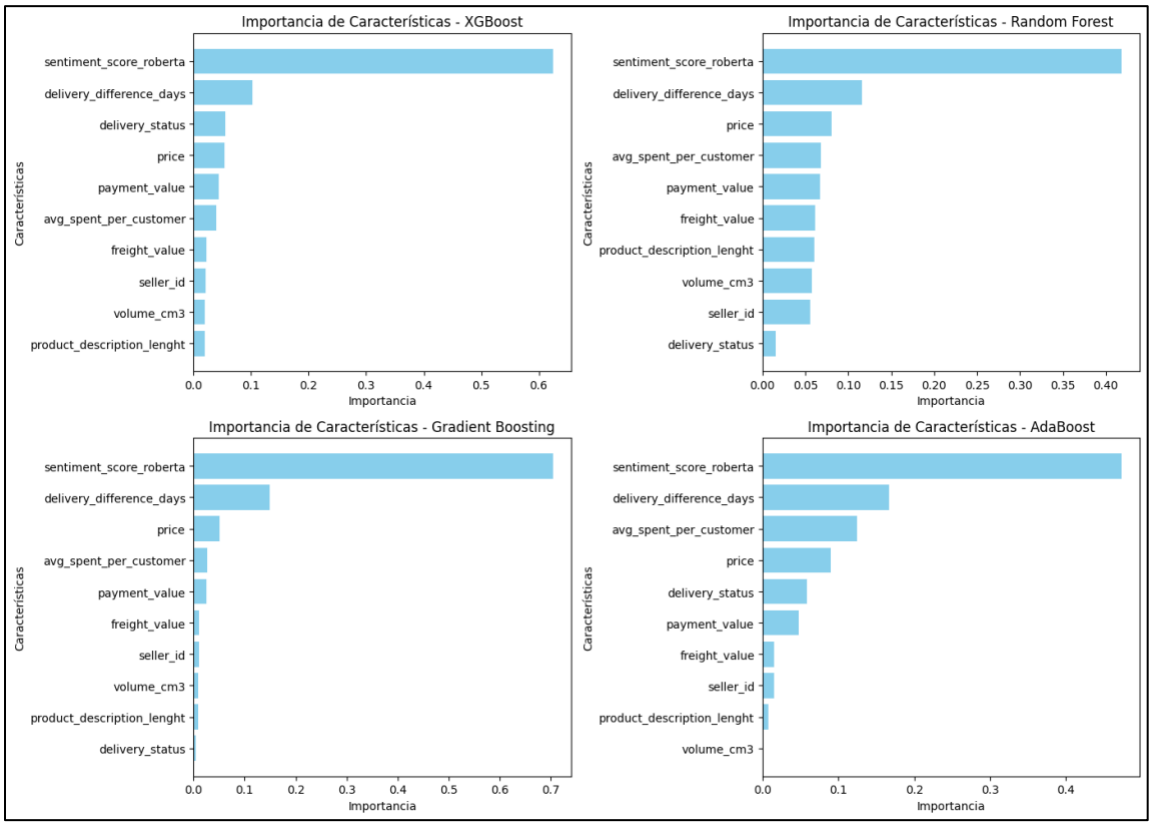

Imagen 8: Importancia de características para modelos de árboles que incluyen puntaje de sentimiento calculado por RoBERTa.

De acuerdo con la Imagen 8, se observa que al calcular la importancia de las características incluyendo 'sentiment_score_roberta', las variables más relevantes en las predicciones fueron 'sentiment_score_roberta' y 'delivery_difference_days'. En tercer lugar, 'price' mantuvo un rol importante, ocupando las primeras posiciones en los modelos Random Forest y Gradient Boosting, y el cuarto lugar en AdaBoost y XGBoost.

Al igual que en el escenario que utiliza el modelo Pysentimiento, el puntaje de sentimiento generado por RoBERTa tuvo un peso considerable en la predicción de los modelos, lo cual es esperable dado que representa la percepción directa del cliente. Por otro lado, 'delivery_difference_days' continuó siendo un factor determinante en la satisfacción del usuario, reafirmando la importancia del cumplimiento de los tiempos de entrega en la evaluación final de la experiencia de compa.

Valores SHAP para los Tres Escenarios en Estudio

Se realizó el cálculo de los valores SHAP para la clase 1, la cual corresponde a las calificaciones negativas, según lo definido en el planteamiento del problema. A diferencia de las métricas tradicionales de importancia de características, los valores SHAP permiten interpretar cómo cada variable influye en una predicción específica, indicando si su efecto aumenta o disminuye la probabilidad de pertenecer a una clase.

Este tipo de análisis no solo ayuda a identificar las variables más relevantes globalmente, sino también a comprender el razonamiento detrás de cada predicción individual del modelo, facilitando la detección de errores, la validación del comportamiento del modelo y la generación de explicaciones comprensibles para usuarios no técnicos.

En este contexto, se calcularon dos métricas: la **magnitud promedio de SHAP** por atributo, que refleja la importancia absoluta de la variable sin importar su dirección, y el **SHAP promedio**, que puede ser positivo o negativo e indica el efecto medio de la variable en la predicción de la clase 1. Las fórmulas utilizadas fueron las siguientes:

$$\text{SHAP Promedio} = \frac{1}{N}\sum_{i=1}^{N} SHAP_i \qquad \text{Magnitud Promedio de SHAP} = \frac{1}{N}\sum_{i=1}^{N} |SHAP_i|$$

Imagen 9: Fórmulas usadas para el cálculo de métricas de los valores SHAP de cada atributo.

A continuación se muestran los resultados obtenidos en los distintos escenarios en estudio:

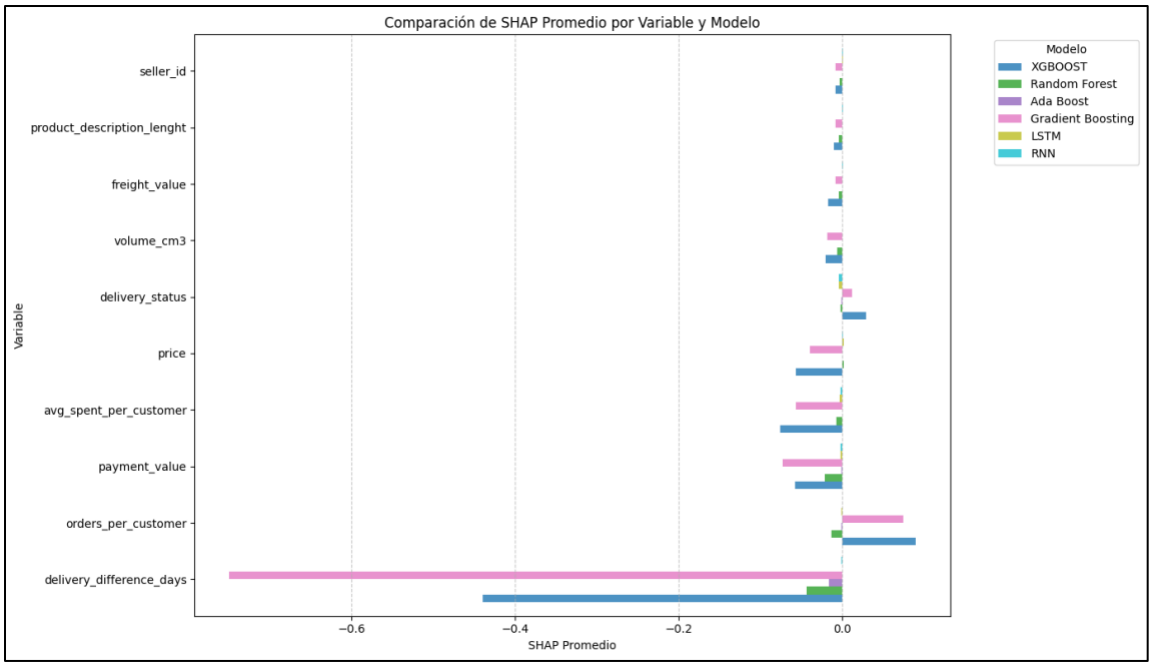

Imagen 10: Valores Shap Promedio de cada caracteristica para modelos entrenados sin puntaje de sentimiento.

De acuerdo con la Imagen 10, se observa que la mayoría de los modelos obtuvieron un valor SHAP negativo para la mayoría de los atributos, excepto en el caso de AdaBoost y XGBoost. En estos dos modelos, las variables 'delivery_status' y 'orders_per_customer' aportaron positivamente a la probabilidad de una calificación negativa.

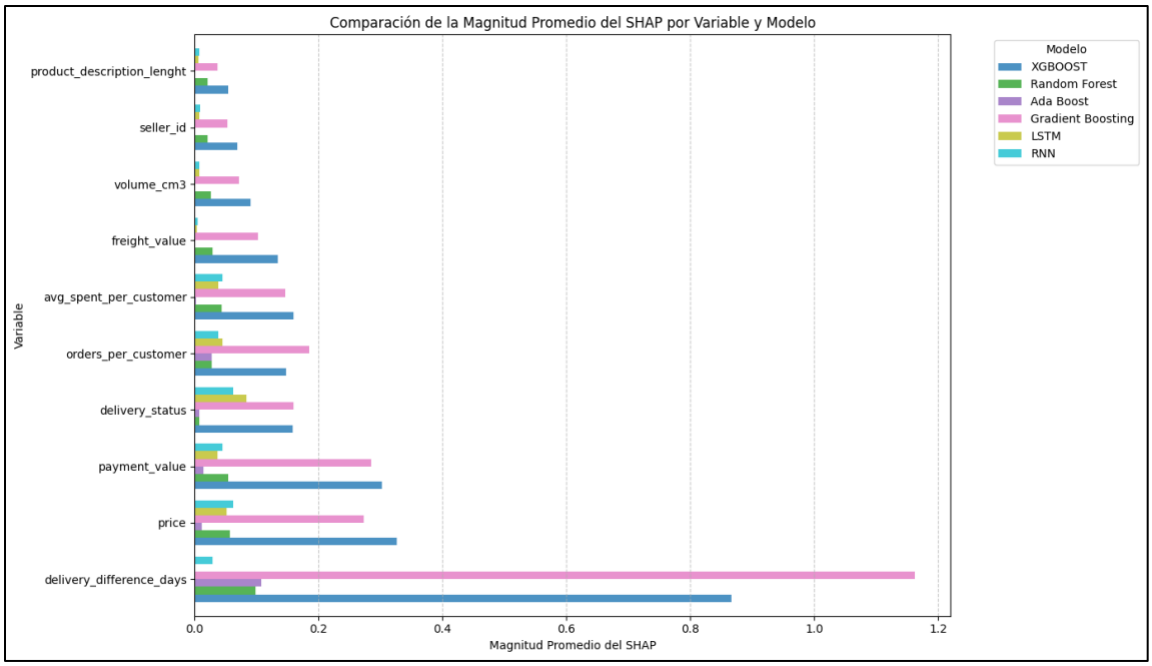

Imagen 11: Magnitud promedio de cada característica para modelos entrenados sin puntaje de sentimiento.

Según la Imagen 11, las características 'delivery_difference_days', 'price' y 'payment_value' corresponden a las variables que más influyeron en las predicciones de los modelos de clasificación binaria. Se destaca que AdaBoost y XGBoost tienden a asignar magnitudes SHAP significativamente más elevadas en comparación con el resto de los modelos evaluados.

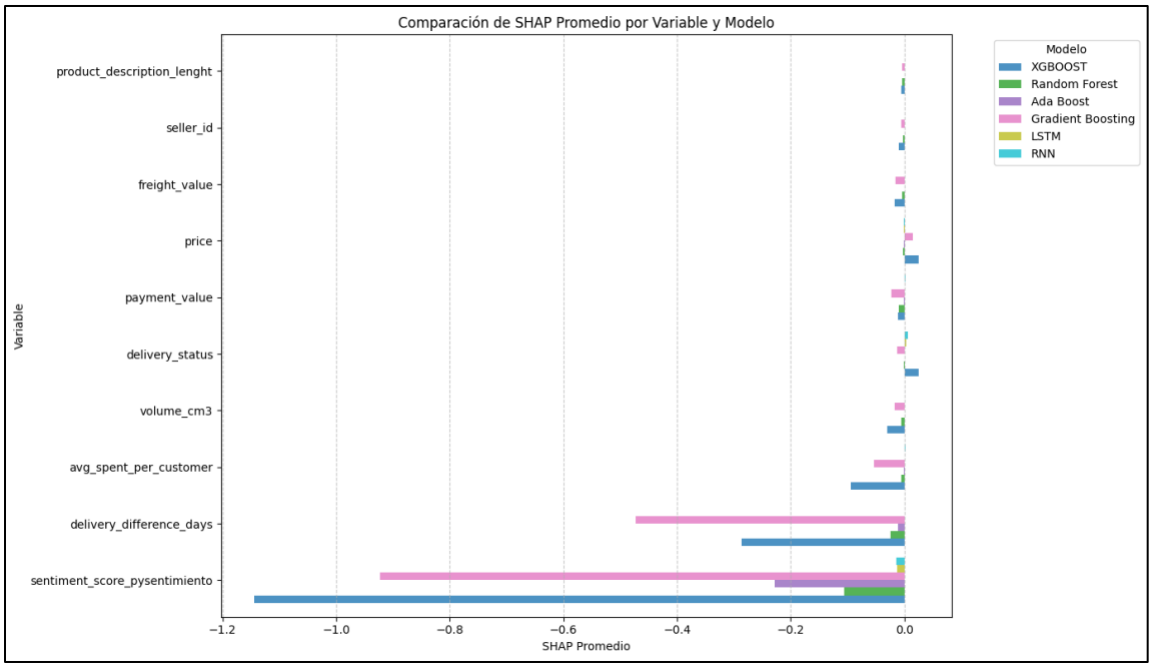

Imagen 12: Valores Shap Promedio de cada característica para modelos entrenados con puntaje de sentimiento de Pysentimiento.

De acuerdo a la Imagen 12, las características 'delivery_status' y 'price' son las únicas que aparecen con un efecto positivo sobre la predicción de la clase 1 ('review_score' negativos). La mayoría de los modelos entregaron SHAP promedio negativos a todos los atributos. El 'sentiment_score_pysentimiento' cuenta con un SHAP promedio notoriamente más elevado que el resto de las características.

La Imagen 13 reafirma que para el escenario con 'sentiment_score_pysentimiento', esta es la característica que más influyó en la predicción de los modelos en estudio. Las características 'delivery_difference_days' y 'price' parecen ser importantes también.

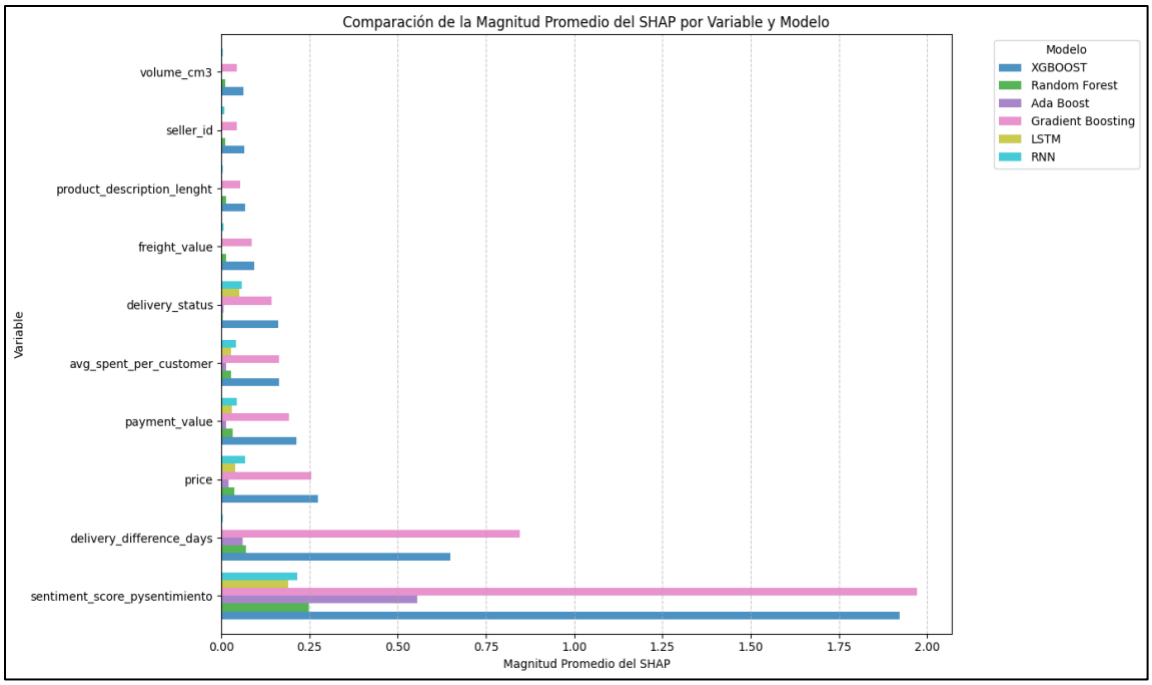

Imagen 13: Magnitud promedio de cada característica para modelos entrenados con puntaje de sentimiento de Pysentimiento.

De acuerdo a las Imágenes 14 y 15, se observó un patrón similar en el escenario que incluye el ‘sentiment_score roberta’. Esta característica es la que más aportó en la predicción de los modelos de clasificación. Las variables ‘delivery_difference_days’ y ‘price’ parecen ser también importantes y esto tiene sentido ya que una está asociada a el cumplimento de los tiempos de entrega y la otra al precio del producto. En este escenario, ‘price’ y ‘delivery_status’ son los únicos atributos que contribuyeron positivamente a la probabilidad de un review score negativo. Para una visualización completa de los valores SHAP, se incluyen las tablas con el detalle numérico por modelo y atributo en el Anexo E.

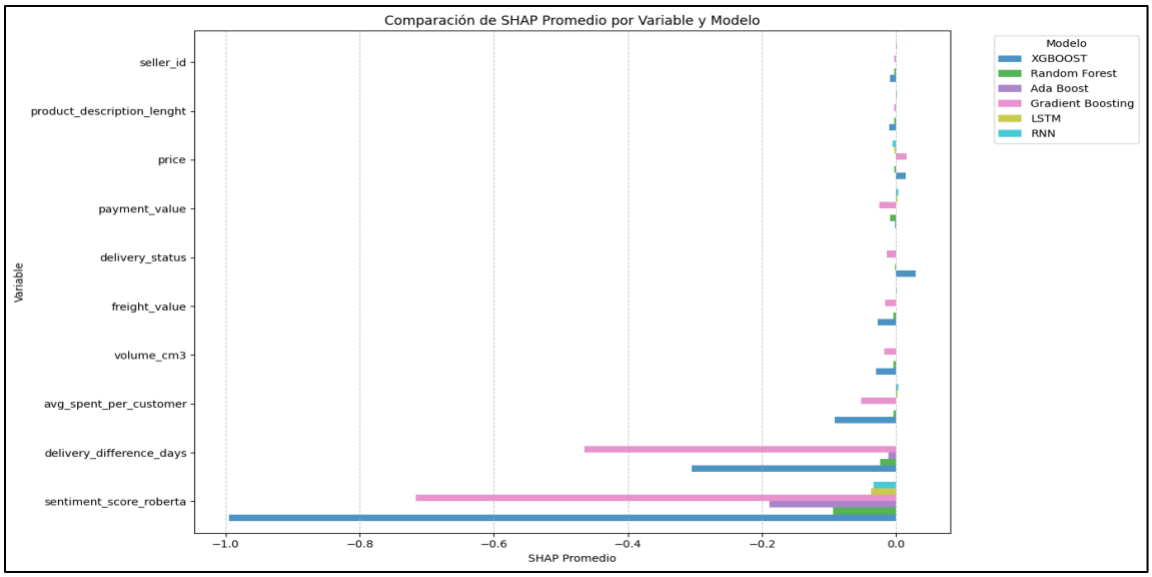

Imagen 14: Valores Shap Promedio de cada caracteristica para modelos entrenados con puntaje de sentimiento de RoBERTa.

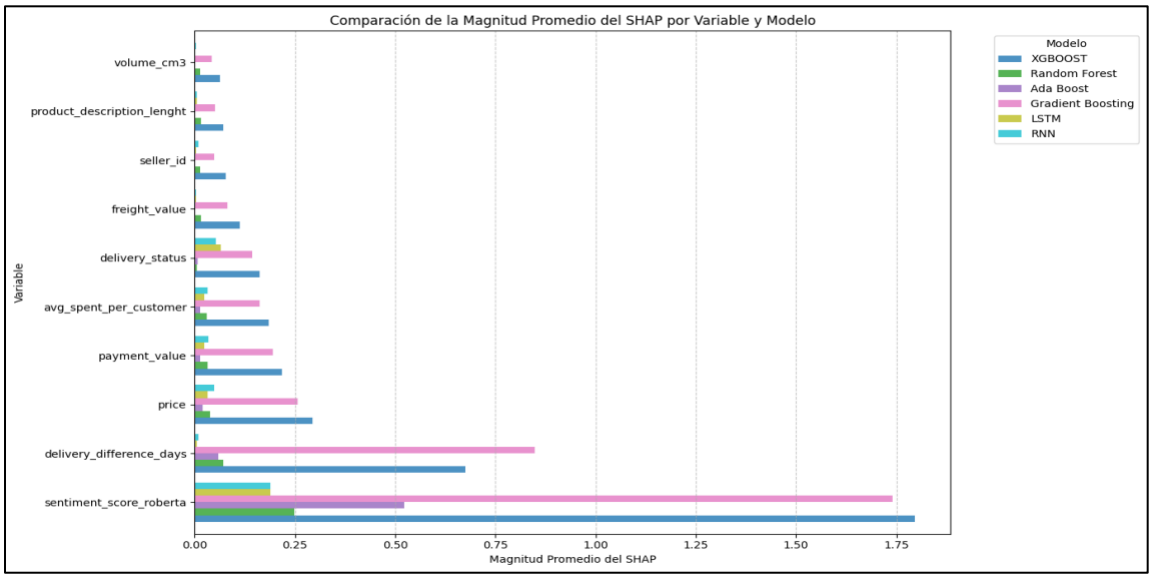

Imagen 15: Magnitud promedio de cada característica para modelos entrenados con puntaje de sentimiento de RoBERTa.

Una vez identificadas las variables estructuradas con mayor impacto en la predicción de reseñas negativas, se profundizó el análisis de la dimensión textual mediante técnicas de procesamiento del lenguaje natural. En particular, se exploraron patrones semánticos y temáticos en los comentarios a través de la clusterización, con el objetivo de complementar los hallazgos previos y obtener una visión más rica sobre las percepciones de los usuarios que pueden no estar reflejadas en las variables numéricas tradicionales.

Clusterización de Comentarios Mediante BERTopic

El análisis de los comentarios de las reseñas se realizó utilizando el modelo BERTopic en combinación con embeddings generados por Sentence-BERT (paraphrase-multilingual-MiniLM-L12-v2) y el algoritmo HDBSCAN para la agrupación de textos. Con el objetivo de identificar patrones y temáticas recurrentes en los comentarios, se evaluó la segmentación en distintos clústeres, variando entre 7 y 11 grupos. El número óptimo de clústeres se determinó utilizando el índice de coherencia c_v, seleccionado el modelo que maximizó dicha métrica. El mejor resultado se obtuvo con 8 clústeres, alcanzando una coherencia de 0.5878. El detalle del código implementado para la aplicación de BERTopic puede consultarse en el Anexo F.

Cluster	Tema_Principal	Total_Reviews	Avg_Score	Low_Score_Percentage
6	acusa_site_nada_porém	14	1.43	85.71
4	pelicula_mosquiteiro_veio_peliculas	35	2.34	60.0
1	nunca_chegando_separado_sempre	62	3.08	45.16
5	algodão_poliester_descrição_percal	31	2.74	35.48
2	tv_show_kit_controle	55	3.4	34.54
-1	produto_prazo_entrega_chegou	15882	3.44	31.99
0	produto_bom_prazo_recomendo	32401	3.79	23.4
3	top_topp_nadadeira_procurava	52	4.9	0.0

Tabla 5: Resultado clusterización por BERTopic con maximización de coeficiente de coherencia.

En la Tabla 5 cada clúster representa un conjunto de comentarios agrupados por similitud semántica, reflejando distintos temas y niveles de satisfacción del usuario. El Clúster -1 agrupa la mayor cantidad de comentarios (15,882) y se centra en aspectos relacionados con entregas y plazos de productos, con una puntuación promedio de 3.44 y un 31.99% de reseñas negativas, lo que sugiere que puede incluir ruido o comentarios demasiado generales. En contraste, el Clúster 0, que es el segundo más grande con 32,401 comentarios, está asociado a productos recomendados y cumplimiento de plazos, obteniendo una mejor evaluación con un promedio de 3.79 y solo 23.40% de reseñas negativas. Por otro lado, el Clúster 3 se distingue por tener la mejor calificación promedio (4.90) y ninguna reseña negativa, lo que indica una percepción altamente positiva entre los usuarios. Finalmente, el Clúster 6 es el que peor evaluación recibe, con una puntuación de 1.43 y un 85.71% de comentarios negativos, lo que sugiere que agrupa principalmente experiencias insatisfactorias con la plataforma.

Tema	Comentarios_Representativos	Traducción_Español
-1	super recomendo parabens grupo stark chegou antes prazo produto qualidade | qualidade controle melhor original acompanha aparelho recomendo | ótima qualidade carteira veio conforme anunciado ótima qualidade recomendo	super recomiendo felicitaciones grupo stark llegó antes de lo previsto producto calidad | calidad control mejor original acompaña dispositivo recomiendo | excelente calidad billetera vino como se anunció excelente calidad recomiendo
0	ótimo produto dentro expectativas | site consta produto entregue porpem recebi gostaria retorno visto pago | infelizmente comprei produto veio trocado consegui solução 2 meses pagamento 1 mês produto errado entregue aviso comprem dessa loja	excelente producto dentro de las expectativas | el sitio indica que el producto fue entregado pero recibí, me gustaría una respuesta dado que pagué | desafortunadamente compré el producto vino cambiado conseguí solución 2 meses pago 1 mes producto equivocado entregado aviso compren de esta tienda
5	lençol pareceachei q percal cem cento algodão dizia profissional imaginei lençóis hotéis | recebi produtos porem veio pedi pedi lençol algodao 170 fios chegou 100 polyester | descrição produto diz 100 algodao enviaram 80algodao 20poliester fechei pedido alteraram data entrega reclamei recebi produto	sábana parece que pensé que era percal 100% algodón decía profesional imaginé sábanas de hoteles | recibí productos pero vino pedí pedí sábana algodón 170 hilos llegó 100 poliéster | descripción del producto dice 100% algodón enviaron 80% algodón 20% poliéster cerré pedido cambiaron fecha de entrega reclamé recibí producto
6	site acusa pedido entregue porém recebi nada | nao compro neste site | todo pedido feito site demorou entregar	el sitio acusa que el pedido fue entregado pero no recibí nada | no compro en este sitio | todo pedido hecho en el sitio tardó en entregarse
3	10 top | top | top 10	10 top | top | top 10
4	tegular to tendo problemaspara ativar ceularnão consigo pta funcionar | mosquiteiro deveria tamanho unico colocar tamanho metragem queria q king nao especificamente produto atende | recebi dentro prazo mim elástico veio elástico diz branco veio amarelado mosquiteiro	regular estoy teniendo problemas para activar el celular no consigo que funcione | mosquitero debería ser de tamaño único poner tamaño metragem quería que king no específicamente el producto atiende | recibí dentro del plazo el elástico vino elástico dice blanco vino amarillento mosquitero
2	chegou produto verificar suporte tv faltou aste ajuste tv | bom dia ja fiz duas reclamações ainda nao respondida recebi dia 1309 2 kits nao entregues 6 kits comprei pedido veio incompleto preciso desses kits ate dia 2509 | show sensacional	llegó el producto verificar soporte de TV faltó asta ajuste TV | buenos días ya hice dos reclamaciones aún no respondida recibí el día 13/09 2 kits no entregados 6 kits compré pedido vino incompleto necesito esos kits hasta el día 25/09 | show sensacional
1	sempre recebo prazo compro sempre nesse site | cliente anos lojas lannister bem parceiras nunca problema algum entregas prazo quanto mp3 gostei simples fácil usar aprovado | sempre comprei loja nunca problemas	siempre recibo a tiempo compro siempre en este sitio | cliente años tiendas lannister bien socias nunca ningún problema entregas a tiempo cuanto mp3 me gustó simple fácil de usar aprobado | siempre compré en la tienda nunca problemas

Tabla 6: Comentarios representativos por clústeres.

De acuerdo a la Tabla 6 y el análisis de clústeres, se distinguieron patrones comunes en los comentarios negativos de los clientes. Los clústeres 6, 4 y 5 presentaron las mayores insatisfacciones, con puntuaciones inferiores a 3 y una alta proporción de reseñas negativas. A continuación, se detalla el análisis de cada uno.

El clúster 6 obtuvo la peor calificación promedio y con un puntaje de 1.43 indica una fuerte insatisfacción. Los principales temas abordados en los comentarios incluyen pedidos no entregados y plazos incumplidos. Comentarios como "El sitio acusa que el pedido fue entregado pero no recibí nada" o "Todo pedido hecho en el sitio tardó en entregarse" reflejan la frustración de los clientes que no recibieron sus compras o

experimentaron largos retrasos. Además, la expresión "No compro en este sitio" sugiere que algunos usuarios han perdido completamente la confianza en la plataforma, lo que puede impactar en la retención de clientes y la reputación de la empresa. La logística y la falta de transparencia en el estado de los pedidos parecen ser las principales causas del malestar de los clientes en este grupo.

El Clúster 4 presentó una calificación promedio de 2.34, lo que lo ubica entre los peores evaluados. En este grupo, los comentarios giran en torno a fallas en el funcionamiento de productos electrónicos y problemas de calidad. Un ejemplo representativo es "Estoy teniendo problemas para activar el celular, no consigo que funcione", lo que sugiere que algunos productos no cumplen con las expectativas de los clientes en términos de operatividad. Otro tema recurrente en este clúster está relacionado con productos defectuosos o diferentes a lo esperado, como se menciona en "Mosquitero debería ser de tamaño único, pero quería el tamaño *king* y no específicamente el producto que me llegó". Además, comentarios sobre cambios en el color y defectos físicos indican que los clientes se sienten decepcionados por recibir artículos que no corresponden a la descripción en la plataforma. Este clúster sugiere que los problemas en la veracidad de la información de los productos y en la calidad percibida generan una experiencia negativa.

Con una calificación promedio de 2.74 y 35.48% de comentarios negativos, el Clúster 5 agrupa quejas relacionadas con productos que no cumplen con las especificaciones prometidas. Los clientes mencionan que la descripción de los productos en la plataforma es engañosa o inexacta, lo que genera expectativas incorrectas. Un comentario ilustrativo es *"Pensé que era percal 100% algodón, pero decía profesional e imaginé sábanas de hoteles"*, lo que indica que la forma en que se presenta el producto en la tienda online puede inducir a errores en la percepción del cliente. Otra queja frecuente es la recepción de artículos con composición de materiales diferente a la indicada, como se menciona en "La descripción del producto dice 100% algodón, pero enviaron 80% algodón y 20% poliéster". Este tipo de discrepancias pueden generar desconfianza en la plataforma y afectar la fidelidad del cliente. Es fundamental mejorar la precisión en la descripción de los productos para evitar estas situaciones y asegurar que los clientes reciban exactamente lo que esperan.

CAPÍTULO 10: CONCLUSIONES

Este trabajo ha desarrollado una metodología analítica para identificar los factores clave que influyen en las calificaciones de productos en plataformas de e-commerce, implementada con la base de datos de la plataforma Olist Brasil. Mediante la aplicación de modelos de *machine learning*, análisis de sentimientos y técnicas explicativas como SHAP, se logró extraer información clave sobre los factores que influyen en la satisfacción del cliente y las razones detrás de las reseñas negativas.

Los resultados obtenidos indican que el tiempo de entrega es la variable más determinante en la percepción de los clientes. La diferencia entre la fecha estimada y la fecha real de entrega ('delivery_difference_days') mostró un fuerte impacto en la probabilidad de recibir una calificación positiva o negativa. Además, el análisis de importancia de características reveló que otras variables operativas y comerciales también desempeñan un papel relevante, como el precio del producto, la cantidad de pedidos realizados por cliente ('orders_per_customer'), y el monto promedio gastado por usuario. Estos factores sugieren que los clientes no solo evalúan el cumplimiento logístico, sino también aspectos económicos y la experiencia general de compra.

El análisis confirmó que los retrasos en la entrega ('delivery_status') afectan negativamente las reseñas de los clientes, aumentando la probabilidad de obtener puntuaciones bajas (1 o 2 estrellas). En contraste, los pedidos entregados antes de lo estimado suelen recibir mejores evaluaciones. Además, las métricas SHAP indicaron que los modelos de clasificación consideran el estado de entrega ('delivery_status') como una variable relevante, lo que refuerza la idea de que la puntualidad es un factor crítico en la experiencia de compra.

La clusterización de comentarios con BERTopic permitió segmentar las opiniones de los clientes en diferentes grupos temáticos. Se identificaron clústeres donde predominaban quejas sobre pedidos no entregados, productos distintos a la descripción y defectos de fabricación, mientras que otros agrupaban experiencias positivas con palabras clave como *"rápido"*, *"excelente"*, *"buena calidad"* y *"recomiendo"*. Estos hallazgos destacan la importancia de mejorar la precisión en la información del catálogo y optimizar los procesos logísticos para reducir la insatisfacción.

Los resultados obtenidos pueden ser utilizados para optimizar la estrategia de servicio al cliente en plataformas de comercio electrónico, desarrollando herramientas predictivas para identificar clientes con alta probabilidad de insatisfacción antes de que realicen una reseña negativa. Además, el uso de modelos de análisis de sentimientos podría integrarse en los sistemas de atención al cliente para detectar patrones de insatisfacción y generar respuestas proactivas.

Para futuras investigaciones, se recomienda incorporar datos geoespaciales que permitan evaluar el impacto de la distancia entre los centros de distribución y los clientes en los tiempos de entrega. Asimismo, explorar el uso de modelos de lenguaje más avanzados, como LLaMA o GPT, podría mejorar la precisión del análisis de comentarios y proporcionar una visión más detallada de la experiencia del usuario en e-commerce.

CAPÍTULO 11: BIBLIOGRAFÍA

- **Huang, H., Asemi, A., & Mustafa, M. B. (2023**). *Sentiment analysis in e-commerce platforms: A review of current techniques and future directions*. *IEEE Access*. https://doi.org/10.1109/ACCESS.2023.3307308

- **Galal, O., Abdel-Gawad, A. H., & Farouk, M. (2024).** *Rethinking of BERT Sentence Embedding for Text Classification*. Preprint en *Research Square*. https://doi.org/10.21203/rs.3.rs-3920665/v1

- **Roumeliotis, K. I., Tselikas, N. D., & Nasiopoulos, D. K.** (2024). LMs in e-commerce: A comparative analysis of GPT and LLaMA models in product review evaluation. *Natural Language Processing Journal, 6*, 100056. https://doi.org/10.1016/j.nlp.2024.100056

- **Palanivinayagam, A., El-Bayeh, C. Z., & Damaševičius, R.** (2023). Twenty Years of Machine-Learning-Based Text Classification: A Systematic Review. *Algorithms, 16*(5), 236. https://doi.org/10.3390/a16050236

- **Deniz, E., Erbay, H., & Coşar, M.** (2022). Multi-Label Classification of E-Commerce Customer Reviews via Machine Learning. *Axioms, 11*(9), 436. https://doi.org/10.3390/axioms11090436

- **Dündar, A. O., & Öztürk, R.** (2020). The Effect of On-Time Delivery on Customer Satisfaction and Loyalty in Channel Integration. *Business and Management Studies: An International Journal, 8*(3), 2675–2693. https://doi.org/10.15295/bmij.v8i3.1520

- **Bellar, O., Baina, A., & Ballafkih, M.** (2024). Sentiment Analysis: Predicting Product Reviews for E-Commerce Recommendations Using Deep Learning and Transformers. *Mathematics, 12*(15), 2403. https://doi.org/10.3390/math12152403

- **Nasrabadi, N., Wicaksono, H., & Fatahi Valilai, O.** (2024). Shopping marketplace analysis based on customer insights using social media analytics. *MethodsX, 13*, 102868. https://doi.org/10.1016/j.mex.2024.102868

- **Areshey, A., & Mathkour, H.** (2023). Transfer Learning for Sentiment Classification Using Bidirectional Encoder Representations from Transformers (BERT) Model. *Sensors, 23*(11), 5232. https://doi.org/10.3390/s23115232

- **Yuan, Y., & Qiao, P.** (2018). A research review on e-commerce logistics delay. Advances in Social Science, Education and Humanities Research, 236, 272-276. https://doi.org/10.2991/meess-18.2018.56

- **Patra, G. K., Kuraku, C., Konkimalla, S., Boddapati, V. N., Sarisa, M., Rajaram, S. K., & Reddy, M. S. (**2023). A sentiment analysis of customer product review based on machine learning techniques in e-commerce. *Journal of Artificial Intelligence & Cloud Computing, 2*(4), 1-4. https://doi.org/10.47363/JAICC/2023(2)389

- **Rashid, M. R. A., Hasan, K. F., Hasan, R., Das, A., Sultana, M., & Hasan, M.** (2024). A comprehensive dataset for sentiment and emotion classification from Bangladesh e-commerce reviews. *Data in Brief, 53*, 110052. https://doi.org/10.1016/j.dib.2024.110052

- **Lopes, É. P., Freitas, L., Gomes, G., & Hammes, L.** (2022). Exploring BERT for aspect-based sentiment analysis in Portuguese language. *The International FLAIRS Conference Proceedings, 35*, 130601. https://doi.org/10.32473/flairs.v35i.130601

- **Ravula, P. (**2022). Impact of delivery performance on online review ratings: The role of temporal distance of ratings. *Journal of Marketing Analytics, 10*(3), 267-286. https://doi.org/10.1057/s41270-022-00168-5

- **Veiga, C. P., Veiga, C. R., Michel, J. S., Di Iorio, L. F., & Su, Z.** (2024). E-commerce in Brazil: An in-depth analysis of digital growth and strategic approaches for online retail. *Journal of Theoretical and Applied Electronic Commerce Research, 19*(2), 1559-1580. https://doi.org/10.3390/jtaer19020076

- **Jain, S., & Roy, P. K.** (2024). E-commerce review sentiment score prediction considering misspelled words: A deep learning approach. *Electronic Commerce Research, 24*, 1737-1761. https://doi.org/10.1007/s10660-022-09582-4

- **Souza, F. D., & Souza Filho, J. B.** (2022). Embedding generation for text classification of Brazilian Portuguese user reviews: From bag-of-words to transformers. *Neural Computing and Applications, 35*, 9393-9406. https://doi.org/10.1007/s00521-022-08068-6

- **Palanivinayagam, A., El-Bayeh, C. Z., & Damaševičius, R.** (2023). Twenty years of machine-learning-based text classification: A systematic review. *Algorithms, 16*(5), 236. https://doi.org/10.3390/a16050236

- **Devlin, J., Chang, M. W., Lee, K., & Toutanova, K.** (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Recuperado de https://github.com/google-research/bert

- **Sayeed, M. S., Mohan, V., & Sonai Muthu, K**. (2023). BERT: A Review of Applications in Sentiment Analysis. *HighTech and Innovation Journal, 4*(2), 453–460. https://doi.org/10.28991/HIJ-2023-04-02-015.

- **Americas Market Intelligence.** (2023). *The online customer journey in Brazil and its impact on logistics*. Recuperado de https://americasmi.com/insights/online-customer-journey-brazil-logistics-impact/.

- **GSMTasks.** (n.d.). *Ecommerce shipping delays: Causes and solutions*. Recuperado el [23-01-2025], de https://gsmtasks.com/ecommerce-shipping-delays/#:~:text=In%20the%20ecommerce%20context%2C%20a,congestion%20or%20extended%20processing%20times.

- **Chatterjee, S., Goyal, D., Prakash, A., & Sharma, J**. (2021). Exploring healthcare/health-product e-commerce satisfaction: A text mining and machine learning application. *Journal of Business Research, 131,* 815–825. https://doi.org/10.1016/j.jbusres.2020.10.043

- **Ahmed, M., Chowdhury, S., & Rana, M. M.** (2023). *BdSentiLLM: A Novel LLM Approach to Sentiment Analysis of Product Reviews*. arXiv. https://arxiv.org/abs/2309.13947

- **Khan, M. A., Khan, A., & Liu, X.** (2023). *Context-Aware Customer Needs Identification by Linguistic Pattern Mining Based on Online Product Reviews*. IEEE Access, 11, 19158–19172. https://doi.org/10.1109/ACCESS.2023.3248613

- **Wang, Y., Ren, C., Sun, H., & Wang, Y.** (2023). *Examining Customer Satisfaction Through Transformer-Based Sentiment Analysis for Improving Bilingual E-Commerce Experiences*. Symmetry, 15(6), 1192. https://doi.org/10.3390/sym15061192

- **Wu, W., Zhang, Y., Lin, Y., Zhang, X., Zhang, Y., & Yang, Y.** (2023). *Developing Data-Driven QFD: A Systemic Approach to Employing Product Manuals and Customer Reviews*. Journal of Intelligent Manufacturing, 34, 1871–1888. https://doi.org/10.1007/s10845-022-02034-0

- **Zhao, Y., Zhao, H., Li, X., & Wang, L.** (2023). *What Factors Influence Online Product Sales? Online Reviews, Review System Curation, Online Promotional Marketing, and Seller Guarantees Analysis*. PLOS ONE, 18(2), e0281731. https://doi.org/10.1371/journal.pone.0281731

- **Ipa, A. S., Roy, P. N., Rony, M. A. T., Raza, A., Fitriyani, N. L., Gu, Y., & Syafrudin, M.** (2024). BdSentiLLM: A novel LLM approach to sentiment analysis of product reviews. *IEEE Access, 12*, 189330–189345. https://doi.org/10.1109/ACCESS.2024.3516826

- **Daza, A., González Rueda, N. D., Aguilar Sánchez, M. S., Robles Espíritu, W. F., & Chauca Quiñones, M. E.** (2024). Sentiment analysis on e-commerce product reviews using machine learning and deep learning algorithms: A bibliometric analysis, systematic literature review, challenges and future

works. *International Journal of Information Management Data Insights, 4*, 100267. https://doi.org/10.1016/j.jjimei.2024.100267

- **Lee, J., Jeong, B., Yoon, J., & Song, C. H.** (2023). Context-aware customer needs identification by linguistic pattern mining based on online product reviews. *IEEE Access, 11*, 71859–71875. https://doi.org/10.1109/ACCESS.2023.3295452

- **Zaghloul, M., Barakat, S., & Rezk, A.** (2024). Predicting e-commerce customer satisfaction: Traditional machine learning vs. deep learning approaches. *Journal of Retailing and Consumer Services, 79*, 103865. https://doi.org/10.1016/j.jretconser.2024.103865

- **Park, G., Kim, S., & Geum, Y.** (2025). Developing data-driven QFD: A systemic approach to employing product manuals and customer reviews. *IEEE Access, 13*, 22380–22395. https://doi.org/10.1109/ACCESS.2025.3532658

- **Eid, M. I.** (2011). Determinants of e-commerce customer satisfaction, trust, and loyalty in Saudi Arabia. *Journal of Electronic Commerce Research, 12*(1), 78–92.

- **Chehal, D., Gupta, P., & Gulati, P.** (2022). Evaluating annotated dataset of customer reviews for aspect-based sentiment analysis. *Journal of Web Engineering, 21*(2), 145–178. https://doi.org/10.13052/jwe1540-9589.2122

- **Shan, S., Sun, J., & Macawile, R. M. C.** (2025). Examining customer satisfaction through transformer-based sentiment analysis for improving bilingual e-commerce experiences. *IEEE Access, 13*, 51027–51045. https://doi.org/10.1109/ACCESS.2025.3551666

- **Elzeheiry, S., Gab-Allah, W. A., Mekky, N., & Elmogy, M.** (2023). Sentiment analysis for e-commerce product reviews: Current trends and future directions. *Preprints*, 2023051649. https://doi.org/10.20944/preprints202305.1649.v1

- **Jim, J. R., Talukder, M. A. R., Malakar, P., Kabir, M. M., Nur, A., & Mridha, M. F.** (2024). Recent advancements and challenges of NLP-based sentiment analysis: A state-of-the-art review. *Natural Language Processing Journal, 6*, 100059. https://doi.org/10.1016/j.nlp.2024.100059

- **Singh, U., Saraswat, A., Azad, H. K., Abhishek, K., & Shitharth, S.** (2022). Towards improving e-commerce customer review analysis for sentiment detection. *Scientific Reports, 12*, 21983. https://doi.org/10.1038/s41598-022-26432-3

- **Zhao, Z., Wang, J., Sun, H., Liu, Y., Fan, Z., & Xuan, F.** (2020). What factors influence online product sales? Online reviews, review system curation, online

promotional marketing and seller guarantees analysis. *IEEE Access, 8*, 219985–219999. https://doi.org/10.1109/ACCESS.2019.2963047

- **IBM. (s.f.).** Machine Learning y conceptos relacionados. Recuperado el 30 de junio de 2025, de https://www.ibm.com/cloud/learn.

- **DataCamp. (s.f.).** Tutorials de ciencia de datos y machine learning. Recuperado de https://www.datacamp.com/tutorial

- **Wikipedia. (s.f.).** Distancia de Levenshtein. Recuperado de https://es.wikipedia.org/wiki/Distancia_de_Levenshtein

- **Scikit-learn. (s.f.).** User Guide en español. Recuperado de https://scikit-learn.org/stable/user_guide.html

- **Grootendorst, M. (s.f.).** BERTopic Documentation. Recuperado de https://maartengr.github.io/BERTopic/

- **Microsoft. (2025).** Machine Learning Overview. Recuperado de https://learn.microsoft.com/en-us/azure/machine-learning/

- **KeepCoding. (s.f.).** Glosario de NLP. Recuperado de https://keepcoding.io/blog/glosario-nlp/

- **The Machine Learners. (s.f.).** Precision, Recall y F1-Score. Recuperado de https://themachinelearners.com/metrics

- **The Black Box Lab. (s.f.).** Guía de modelos supervisados. Recuperado de https://theblackboxlab.com/guias

- **Efimov, D. (2023).** Sentence-BERT in Practice. Recuperado de https://efimov.ai/sentence-bert

- **DataScientest. (s.f.).** Glosario de Inteligencia Artificial. Recuperado de https://datascientest.com/es/glosario

- **AWS. (s.f.).** Natural Language Processing (NLP). Recuperado de https://aws.amazon.com/nlp/

- **Pysentimiento. (s.f.).** Documentación oficial. Recuperado de https://github.com/pysentimiento/pysentimiento

- **Hugging Face. (s.f.).** Model Card: RoBERTa. Recuperado de https://huggingface.co/roberta-base

- **Briggs, J. (2021).** Understanding MLM and NSP in BERT. Recuperado de https://medium.com/@jbriggs/bert-training-tasks

CAPÍTULO 12: ANEXOS

A. Glosario de Términos

- **AdaBoost**: algoritmo de ensamblado que combina clasificadores débiles secuencialmente, ajustando el peso de los errores para mejorar la precisión. En cada iteración, se enfoca en los casos mal clasificados por los modelos anteriores. [IBM, s.f.]
- **Análisis de sentimientos**: Técnica de NLP para identificar la opinión o emoción expresada en un texto. [IBM, s.f.]
- **Atributo**: Variable utilizada como entrada para un modelo de machine learning.
- ***Bag of Words***: Modelo de representación de texto que cuenta la frecuencia de palabras en un documento. [IBM, s.f.]
- **BERT**: Modelo de lenguaje basado en transformers que entiende el contexto de las palabras bidireccionalmente. [DataCamp, s.f.]
- **BERTopic**: Técnica basada en BERT y HDBSCAN para agrupar textos por temas latentes. [Grootendorst, s.f.]
- ***Deep Learning***: Subcampo del aprendizaje automático que usa redes neuronales profundas para aprender representaciones complejas. [IBM, s.f.]
- **Distancia de Levenshtein**: Medida de diferencia entre dos cadenas según el número mínimo de operaciones necesarias para transformarlas. [Wikipedia, s.f.]
- ***Embedding***: Representación vectorial densa y continua de datos categóricos o de texto. [DataCamp, s.f.]
- **F1-Score**: Media armónica entre precisión y recall; útil en clases desbalanceadas. [The Machine Learners, s.f.]
- ***Feature Engineering* (Ingeniería de Características):** Proceso de crear o transformar variables para mejorar el modelo.
- ***Finetunning:*** Ajuste fino de un modelo previamente entrenado en una nueva tarea o conjunto de datos. [DataCamp, s.f.]
- **Gradient Boosting**: Algoritmo de ensamblado que entrena modelos secuencialmente para corregir errores anteriores. [IBM, s.f.]
- **HDBSCAN**: Algoritmo de clustering jerárquico basado en densidad, útil para datos con ruido. [IBM, s.f.]
- **Importancia de características**: Medida del impacto de cada variable sobre las predicciones de un modelo. [Qlik, s.f.]
- **Información Mutua**: Métrica que evalúa la dependencia entre una variable y la clase objetivo. [Duque, s.f.]
- ***Label Encoding***: Conversión de categorías en valores numéricos consecutivos. [scikit-learn documentación en español, s.f.]

- ***Large Language Model* (LLM):** Modelo de lenguaje de gran tamaño entrenado con enormes volúmenes de texto. [DataCamp, s.f.]
- **Lematización**: Transformación de una palabra a su forma base o raíz. [IBM, s.f.]
- ***Long Short-Term Memory* (LSTM)**: Tipo de red neuronal recurrente que captura dependencias de largo plazo. [DataScientest, s.f.]
- ***Machine Learning***: Área de la IA que crea modelos que aprenden patrones a partir de datos. [IBM, s.f.]
- **MinMaxScaler**: Técnica de normalización que escala los datos entre un mínimo y máximo definidos. [scikit-learn documentación en español, s.f.]
- **Modelo**: Representación matemática entrenada para hacer predicciones basadas en datos. [Microsoft Learn, 2025]
- **Modelos de clasificación**: Algoritmos de ML que asignan una clase o categoría a cada ejemplo. [The Black Box Lab, s.f.]
- **Modelos de regresión**: Modelos de ML que predicen un valor numérico continuo. [The Black Box Lab, s.f.]
- ***Masked Language Modeling* (MLM)**: Tarea de preentrenamiento donde algunas palabras del texto son ocultadas y predichas. [Briggs, 2021]
- ***Natural Language Processing* (NLP)**: tecnología de machine learning que brinda a las computadoras la capacidad de interpretar, manipular y comprender el lenguaje humano. [AWS, s.f.]
- ***Next Sentence Prediction* (NSP)**: Tarea de preentrenamiento en la que se predice si una oración sigue a otra. [Briggs, 2021]
- **One-Hot Encoding**: Transformación de variables categóricas en vectores binarios. [DataCamp, s.f.]
- ***Overfitting* (Sobreajuste)**: Cuando un modelo aprende demasiado los datos de entrenamiento y pierde capacidad de generalización. [IBM, s.f.]
- **Precisión**: Proporción de verdaderos positivos sobre todos los predichos como positivos. [The Machine Learners, s.f.]
- **Pysentimiento:** Modelo preentrenado basado en BERTweet, optimizado para análisis de sentimientos en portugués, disponible a través de la librería Pysentimiento y alojado en Hugging Face. [Hugging Face, s.f.]
- **Random Forest**: Algoritmo de clasificación que combina múltiples árboles de decisión entrenados aleatoriamente. [IBM, s.f.]
- ***Recall* (Sensibilidad)**: Proporción de verdaderos positivos correctamente identificados. [The Machine Learners, s.f.]
- **Redes Neuronales**: Modelos inspirados en el cerebro humano que aprenden representaciones complejas. [IBM, s.f.]
- **RoBERTa**: Modelo preentrenado basado en RoBERTa, ajustado para análisis de sentimientos en textos de Twitter, disponible en Hugging Face. [Hugging Face, s.f.]
- **Red Neuronal Recurrente (RNN)**: Red neuronal que procesa secuencias de datos recordando información previa. [IBM, s.f.]
- **Select-KBest**: Método de selección de variables que elige las k con mejor puntuación según un criterio. [scikit-learn documentación en español, s.f.]
- **Sentence-BERT**: Modelo que genera vectores semánticos de frases para tareas como similitud o clustering. [Efimov, 2023]
- **SHAP (Valores SHAP)**: Método que explica la contribución de cada variable en la predicción de un modelo. [DataCamp, s.f.]

- **SMOTE**: Técnica de sobremuestreo que genera ejemplos sintéticos para clases minoritarias. [Microsoft Learn, s.f.]
- **Stopwords**: Palabras frecuentes sin contenido relevante que suelen eliminarse en NLP. [KeepCoding, s.f.]
- **TF-IDF**: Técnica que pondera palabras según su frecuencia en un documento y en todo el corpus. [KeepCoding, s.f.]
- **Tokenización**: Proceso de dividir un texto en palabras, frases u otras unidades significativas. [IBM, s.f.]
- ***Underfitting* (Subajuste):** Cuando un modelo es demasiado simple y no aprende bien los patrones de los datos. [IBM, s.f.]
- **Validación cruzada**: Técnica para evaluar un modelo dividiendo los datos en múltiples particiones de entrenamiento y prueba. [DataScientest, s.f.]
- **XGBoost**: algoritmo de *gradient boosting* que construye de forma secuencial múltiples árboles de decisión, optimizando una función de pérdida con regularización y utilizando aproximaciones de segundo orden para lograr alta precisión y eficiencia computacional. [The Machine Learners, s.f.]

B. Descripción de Variables Utilizadas

A continuación, se presentan las principales variables empleadas en el análisis, indicando su nombre, descripción y tabla de origen en el dataset de e-commerce de Olist:

1. **avg_spent_per_customer**: gasto promedio por cliente.
2. **delivery_difference_days**: diferencia en días entre fecha final de entrega menos fecha estimada de entrega.
3. **delivery_status**: indicador de adelanto o retraso de un pedido.
4. **freight_value**: valor del envío del artículo. Si un pedido tiene más de un artículo, el valor del envío se divide entre ellos.
5. **orders_per_customer**: número de órdenes por cliente.
6. **payment_value**: valor total de la transacción del pedido.
7. Price: precio del artículo basado en el ID del producto.
8. **producto_description_length**: número de caracteres en la descripción del producto.
9. **review_score**: variable objetivo que se define como la calificación del cliente con una escala de 1 a 5.
10. **seller_id**: identificador único del ID del vendedor.
11. **sentiment_score_pysentimiento**: puntaje de sentimiento obtenido a partir del modelo Pysentimiento sobre la variable string ‘review_comment_message’.
12. **sentiment_score_roberta**: puntaje de sentimiento obtenido a partir del modelo RoBERTa sobre la variable string ‘review_comment_message’.
13. **volumen_cm3**: volumen del producto en centímetros cúbicos, obtenido como el producto entre el largo, alto y ancho del paquete.

C. Análisis de Datos Faltantes

En la siguiente tabla se presenta la cantidad de registros nulos encontrados en cada variable de la base de datos integrada:

Variable	Cantidad de Datos Nulos
customer_id	0
customer_unique_id	0
customer_zip_code_prefix	0
customer_city	0
customer_state	0
order_id	0
order_status	0
order_purchase_timestamp	0
order_approved_at	177
order_delivered_carrier_date	2086
order_delivered_customer_date	3421
order_estimated_delivery_date	0
payment_sequential	3
payment_type	3
payment_installments	3
payment_value	3
review_id	997
review_score	997
review_comment_title	105154
review_comment_message	68898
review_creation_date	997
review_answer_timestamp	997
order_item_id	833
product_id	833
seller_id	833
shipping_limit_date	833
price	833
freight_value	833
product_category_name	2542
product_name_lenght	2542
product_description_lenght	2542
product_photos_qty	2542
product_weight_g	853
product_length_cm	853
product_height_cm	853
product_width_cm	853
seller_zip_code_prefix	833
seller_city	833
seller_state	833

Tabla 7: Cantidad de datos nulos para la base de datos integrada con las 7 tablas del e-commerce Olist.

D. Gráficos Adicionales del Análisis Exploratorio de Datos (EDA)

En esta sección se presentan gráficos complementarios obtenidos durante el análisis exploratorio de datos (EDA), que permiten visualizar patrones relevantes en las variables del conjunto de datos, y que apoyan el entendimiento de la información utilizada para el modelado.

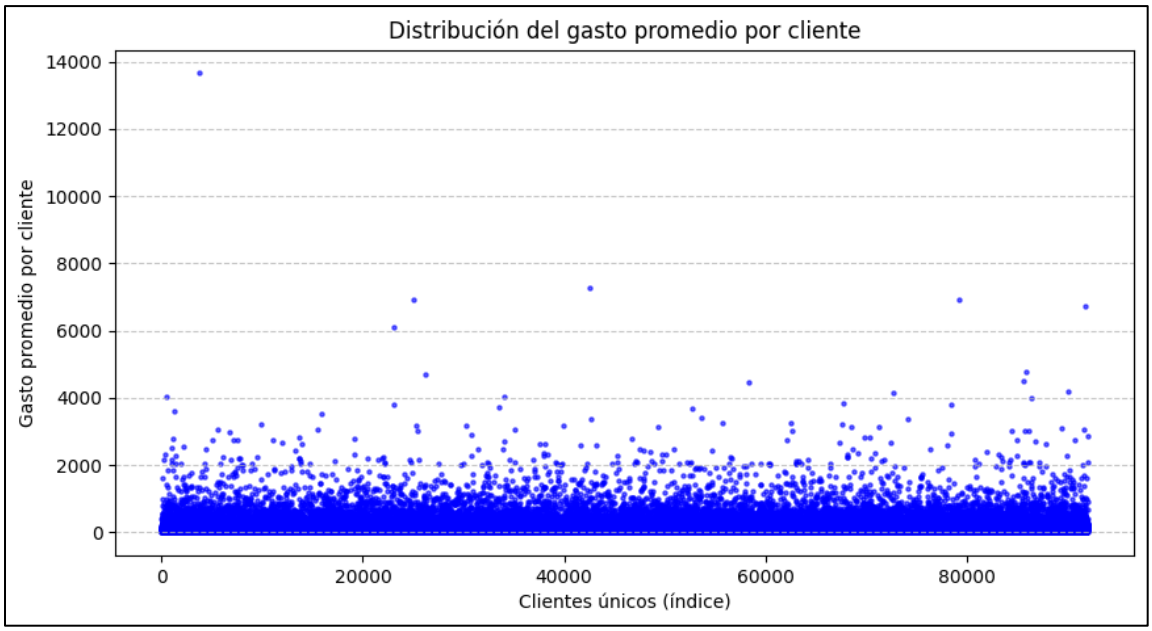

Imagen 16: Gráfico de dispersión de gasto promedio por cliente

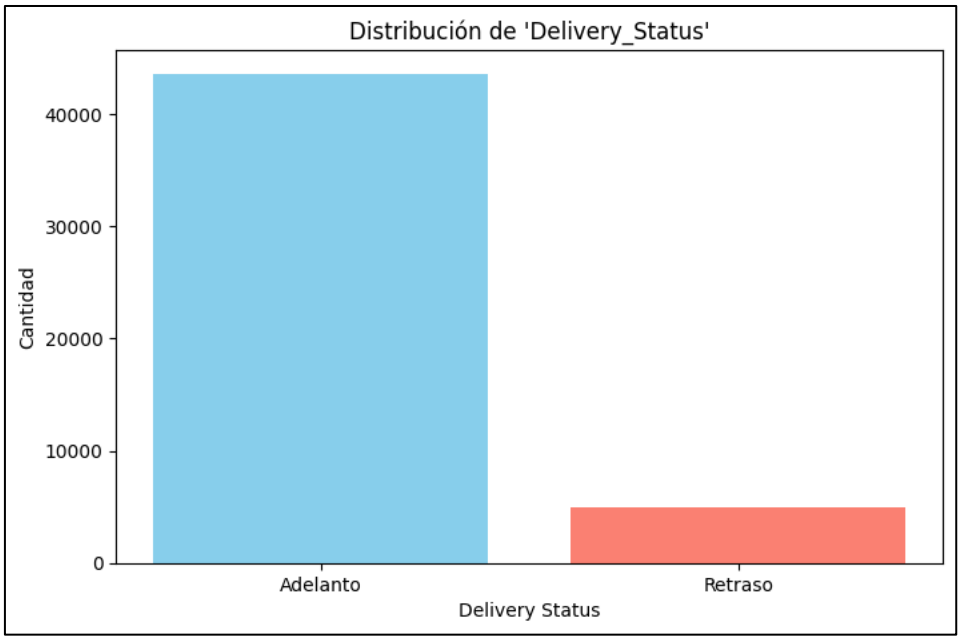

Imagen 17: Tabla comparativa entre estados de adelanto y retraso de pedidos.

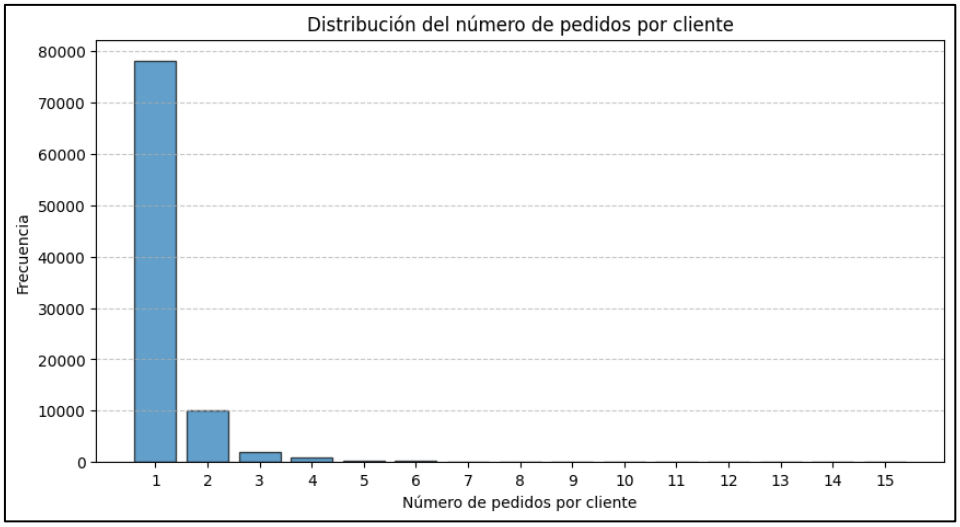

Imagen 18: Cantidad de pedidos por cliente en una misma orden.

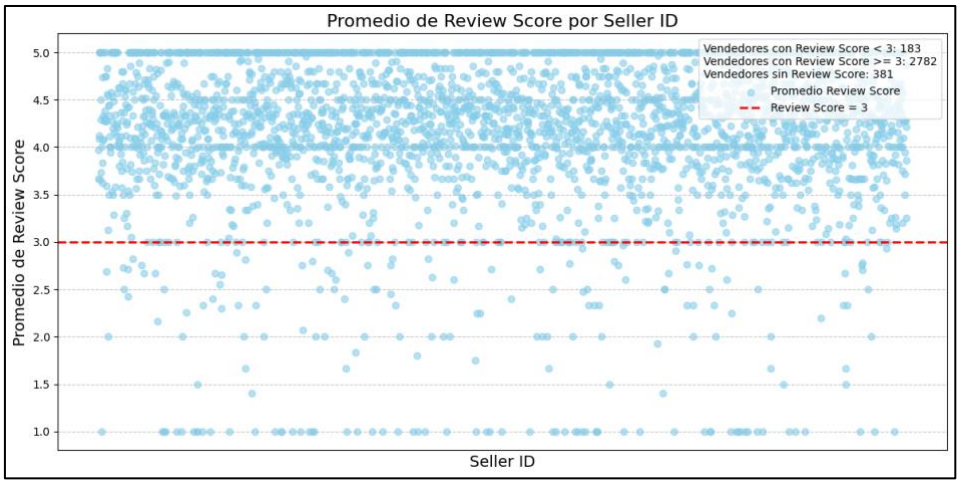

Imagen 19: Gráfico de dispersión del 'review_score' promedio de cada 'seller ID' presente en el e-commerce Olist.

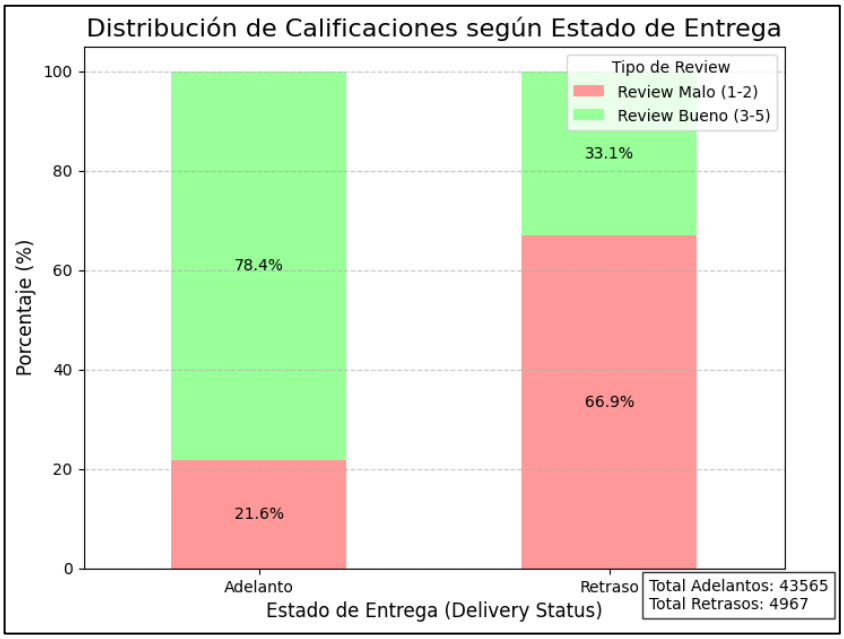

Imagen 20: Distribución porcentual de calificaciones negativos y positivos para ordenes adelantadas y retrasadas.

E. Tablas Detalladas de Valores SHAP

E.1 SHAP promedio por característica – Modelos sin sentiment score

En esta sección se presentan los resultados detallados de los valores SHAP promedio y magnitud promedio, calculados para cada modelo de clasificación binaria utilizado en los tres escenarios analizados:

- **Escenario 1:** Modelos sin *puntaje de sentimiento*.
- **Escenario 2:** Modelos con *puntaje de sentimiento* de modelo *PySentimiento*.
- **Escenario 3:** Modelos con *puntaje de sentimiento* de modelo *RoBERTa*.

Para cada escenario, se presentan los resultados obtenidos con los seis modelos evaluados: Random Forest, Gradient Boosting, XGBoost, AdaBoost, LSTM y RNN.

	SHAP Promedio	Magnitud Promedio	Desviación Estándar	Porcentaje Positivo	Porcentaje Negativo	Cantidad de Positivos	Cantidad de Negativos	Efecto Global (Positivo/Negativo)
delivery_difference_days	-0.4398	0.8660	0.9083	10.25%	89.75%	205	1795	Negativo
price	-0.0571	0.3261	0.4073	52.65%	47.35%	1053	947	Negativo
payment_value	-0.0579	0.3020	0.3494	43.75%	56.25%	875	1125	Negativo
avg_spent_per_customer	-0.0762	0.1600	0.2270	27.65%	72.35%	553	1447	Negativo
delivery_status	0.0288	0.1577	0.2554	89.80%	10.20%	1796	204	Positivo
orders_per_customer	0.0903	0.1476	0.2402	59.70%	40.30%	1194	806	Positivo
freight_value	-0.0174	0.1338	0.1831	51.35%	48.65%	1027	973	Negativo
volume_cm3	-0.0205	0.0902	0.1208	40.05%	59.95%	801	1199	Negativo
seller_id	-0.0085	0.0693	0.1163	52.15%	47.85%	1043	957	Negativo
product_description_lenght	-0.0101	0.0539	0.0737	41.20%	58.80%	824	1176	Negativo

Imagen 21: Valores SHAP modelo Random Forest escenario sin puntaje de sentimiento.

	SHAP Promedio	Magnitud Promedio	Desviación Estándar	Porcentaje Positivo	Porcentaje Negativo	Cantidad de Positivos	Cantidad de Negativos	Efecto Global
delivery_difference_days	-0.0442	0.0988	0.1173	14.45%	85.55%	289	1711	Negativo
price	0.0003	0.0574	0.0713	54.65%	45.35%	1093	907	Positivo
payment_value	-0.0208	0.0543	0.0630	35.10%	64.90%	702	1298	Negativo
avg_spent_per_customer	-0.0068	0.0427	0.0546	41.50%	58.50%	830	1170	Negativo
freight_value	-0.0040	0.0282	0.0392	51.70%	48.30%	1034	966	Negativo
orders_per_customer	-0.0116	0.0268	0.0367	32.80%	67.20%	656	1344	Negativo
volume_cm3	-0.0058	0.0257	0.0338	39.60%	60.40%	792	1208	Negativo
seller_id	-0.0034	0.0207	0.0289	43.95%	56.05%	879	1121	Negativo
product_description_lenght	-0.0043	0.0201	0.0264	38.60%	61.40%	772	1228	Negativo
delivery_status	-0.0019	0.0076	0.0227	49.60%	50.40%	992	1008	Negativo

Imagen 22: Valores SHAP modelo Random Forest escenario sin puntaje de sentimiento.

	SHAP Promedio	Magnitud Promedio	Desviación Estándar	Porcentaje Positivo	Porcentaje Negativo	Cantidad de Positivos	Cantidad de Negativos	Efecto Global
delivery_difference_days	-0.0290	0.2408	0.3740	10.20%	89.80%	204	1796	Negativo
delivery_status	-0.0118	0.1388	0.2449	89.80%	10.20%	1796	204	Negativo
price	-0.0043	0.1119	0.1388	59.80%	40.20%	1196	804	Negativo
payment_value	-0.0042	0.0902	0.1073	43.10%	56.90%	862	1138	Negativo
orders_per_customer	-0.0020	0.0592	0.0794	36.30%	63.70%	726	1274	Negativo
avg_spent_per_customer	-0.0036	0.0491	0.0775	57.75%	42.25%	1155	845	Negativo
freight_value	-0.0024	0.0230	0.0328	77.30%	22.70%	1546	454	Negativo
volume_cm3	-0.0004	0.0201	0.0209	40.65%	59.35%	813	1187	Negativo
seller_id	-0.0012	0.0137	0.0271	90.55%	9.45%	1811	189	Negativo
product_description_lenght	0.0000	0.0000	0.0000	0.00%	0.00%	0	0	Negativo

Imagen 23: Valores SHAP modelo AdaBoost escenario sin puntaje de sentimiento.

	SHAP Promedio	Magnitud Promedio	Desviación Estándar	Porcentaje Positivo	Porcentaje Negativo	Cantidad de Positivos	Cantidad de Negativos	Efecto Global
delivery_difference_days	-0.7501	1.1619	0.9858	10.20%	89.80%	204	1796	Negativo
payment_value	-0.0731	0.2844	0.3204	42.65%	57.35%	853	1147	Negativo
price	-0.0399	0.2733	0.3508	56.30%	43.70%	1126	874	Negativo
orders_per_customer	0.0746	0.1844	0.2532	41.80%	58.20%	836	1164	Positivo
delivery_status	0.0119	0.1590	0.2768	89.80%	10.20%	1796	204	Positivo
avg_spent_per_customer	-0.0564	0.1464	0.2148	20.05%	79.95%	401	1599	Negativo
freight_value	-0.0083	0.1023	0.1467	56.80%	43.20%	1136	864	Negativo
volume_cm3	-0.0182	0.0712	0.1065	40.35%	59.65%	807	1193	Negativo
seller_id	-0.0081	0.0527	0.0929	58.20%	41.80%	1164	836	Negativo
product_description_lenght	-0.0084	0.0362	0.0574	38.45%	61.55%	769	1231	Negativo

Imagen 24: Valores SHAP modelo GradientBoosting escenario sin puntaje de sentimiento.

	Característica	SHAP Promedio	Magnitud Promedio	Desviación Estándar	Porcentaje Positivo	Porcentaje Negativo	Cantidad de Positivos	Cantidad de Negativos	Efecto Global
6	delivery_status	-0.0064	0.0835	0.1308	10.40%	89.60%	208	1792	Negativo
2	price	0.0049	0.0523	0.0880	64.20%	35.80%	1284	716	Positivo
7	orders_per_customer	0.0009	0.0415	0.0499	32.35%	67.65%	647	1353	Positivo
8	avg_spent_per_customer	-0.0038	0.0370	0.0541	30.60%	69.40%	612	1388	Negativo
0	payment_value	-0.0035	0.0367	0.0537	30.95%	69.05%	619	1381	Negativo
1	seller_id	0.0002	0.0058	0.0091	59.50%	40.50%	1190	810	Positivo
4	product_description_lenght	0.0000	0.0040	0.0061	61.80%	38.20%	1236	764	Positivo
9	volume_cm3	0.0002	0.0040	0.0066	68.70%	31.30%	1374	626	Positivo
5	delivery_difference_days	0.0001	0.0021	0.0032	45.40%	54.60%	908	1092	Positivo
3	freight_value	-0.0001	0.0012	0.0047	46.90%	53.10%	938	1062	Negativo

Imagen 25: Valores SHAP modelo LSTM escenario sin puntaje de sentimiento.

	Característica	SHAP Promedio	Magnitud Promedio	Desviación Estándar	Porcentaje Positivo	Porcentaje Negativo	Cantidad de Positivos	Cantidad de Negativos	Efecto Global
6	delivery_status	-0.0054	0.0835	0.1308	10.40%	89.60%	208	1792	Negativo
2	price	0.0060	0.0523	0.0880	64.20%	35.80%	1284	716	Positivo
7	orders_per_customer	-0.0001	0.0415	0.0499	32.35%	67.65%	647	1353	Negativo
8	avg_spent_per_customer	-0.0034	0.0370	0.0541	30.60%	69.40%	612	1388	Negativo
0	payment_value	-0.0031	0.0367	0.0537	30.95%	69.05%	619	1381	Negativo
1	seller_id	-0.0003	0.0058	0.0091	59.50%	40.50%	1190	810	Negativo
4	product_description_lenght	0.0002	0.0040	0.0061	61.80%	38.20%	1236	764	Positivo
9	volume_cm3	0.0000	0.0040	0.0066	68.70%	31.30%	1374	626	Positivo
5	delivery_difference_days	-0.0012	0.0021	0.0032	45.40%	54.60%	908	1092	Negativo
3	freight_value	-0.0005	0.0012	0.0047	46.90%	53.10%	938	1062	Negativo

Imagen 26: Valores SHAP modelo RNN escenario sin puntaje de sentimiento.

E.2 SHAP promedio por característica – Modelos Pysentimiento sentiment score

	SHAP Promedio	Magnitud Promedio	Desviación Estándar	Porcentaje Positivo	Porcentaje Negativo	Cantidad de Positivos	Cantidad de Negativos	Efecto Global (Positivo/Negativo)
sentiment_score_pysentimiento	-1.144073	1.921415	2.211161	54.700000	45.300000	1094	906	Negativo
delivery_difference_days	-0.287118	0.648866	0.734126	10.650000	89.350000	213	1787	Negativo
price	0.025125	0.273991	0.354518	58.900000	41.100000	1178	822	Positivo
payment_value	-0.011715	0.212628	0.277850	40.600000	59.400000	812	1188	Negativo
avg_spent_per_customer	-0.093717	0.163615	0.224087	31.650000	68.350000	633	1367	Negativo
delivery_status	0.025181	0.161828	0.254338	89.800000	10.200000	1796	204	Positivo
freight_value	-0.016806	0.093822	0.133357	54.650000	45.350000	1093	907	Negativo
product_description_lenght	-0.005697	0.067315	0.093442	39.950000	60.050000	799	1201	Negativo
seller_id	-0.010343	0.065680	0.105892	50.600000	49.400000	1012	988	Negativo
volume_cm3	-0.030667	0.063936	0.098878	40.700000	59.300000	814	1186	Negativo

Imagen 27: Valores SHAP modelo XGBoost escenario con 'sentiment_score_pysentimiento' .

	SHAP Promedio	Magnitud Promedio	Desviación Estándar	Porcentaje Positivo	Porcentaje Negativo	Cantidad de Positivos	Cantidad de Negativos	Efecto Global
sentiment_score_pysentimiento	-0.0996	0.2445	0.2736	54.50%	45.50%	1090	910	Negativo
delivery_difference_days	-0.0217	0.0666	0.0903	12.05%	87.95%	241	1759	Negativo
price	-0.0031	0.0369	0.0522	52.15%	47.85%	1043	957	Negativo
payment_value	-0.0087	0.0316	0.0418	37.15%	62.85%	743	1257	Negativo
avg_spent_per_customer	-0.0039	0.0291	0.0407	42.65%	57.35%	853	1147	Negativo
freight_value	-0.0028	0.0146	0.0225	45.00%	55.00%	900	1100	Negativo
product_description_lenght	-0.0027	0.0143	0.0222	40.75%	59.25%	815	1185	Negativo
volume_cm3	-0.0037	0.0137	0.0204	34.80%	65.20%	696	1304	Negativo
seller_id	-0.0023	0.0131	0.0215	40.35%	59.65%	807	1193	Negativo
delivery_status	-0.0022	0.0060	0.0167	40.25%	59.75%	805	1195	Negativo

Imagen 28: Valores SHAP modelo Random Forest escenario con 'sentiment_score_pysentimiento'.

	SHAP Promedio	Magnitud Promedio	Desviación Estándar	Porcentaje Positivo	Porcentaje Negativo	Cantidad de Positivos	Cantidad de Negativos	Efecto Global
avg_spent_per_customer	0.2943	0.2943	0.0348	100.00%	0.00%	2000	0	Positivo
delivery_difference_days	-0.0166	0.1079	0.1710	10.20%	89.80%	204	1796	Negativo
payment_value	-0.0006	0.0141	0.0165	43.10%	56.90%	862	1138	Negativo
price	-0.0005	0.0118	0.0136	59.80%	40.20%	1196	804	Negativo
volume_cm3	0.0113	0.0113	0.0014	100.00%	0.00%	2000	0	Positivo
delivery_status	-0.0007	0.0074	0.0132	89.80%	10.20%	1796	204	Negativo
sentiment_score_pysentimiento	0.0015	0.0015	0.0000	100.00%	0.00%	2000	0	Positivo
freight_value	-0.0001	0.0009	0.0014	86.80%	13.20%	1736	264	Negativo
seller_id	-0.0001	0.0009	0.0026	96.60%	3.40%	1932	68	Negativo
product_description_lenght	0.0000	0.0000	0.0000	0.00%	0.00%	0	0	Negativo

Imagen 29: Valores SHAP modelo AdaBoost escenario con 'sentiment_score_pysentimiento'.

	SHAP Promedio	Magnitud Promedio	Desviación Estándar	Porcentaje Positivo	Porcentaje Negativo	Cantidad de Positivos	Cantidad de Negativos	Efecto Global
delivery_difference_days	-0.6642	0.9417	0.7450	10.20%	89.80%	204	1796	Negativo
avg_spent_per_customer	0.6695	0.6816	0.4281	95.35%	4.65%	1907	93	Positivo
sentiment_score_pysentimiento	-0.2704	0.2735	0.1330	2.50%	97.50%	50	1950	Negativo
volume_cm3	-0.0538	0.1926	0.2509	42.05%	57.95%	841	1159	Negativo
price	-0.0279	0.1696	0.2306	51.95%	48.05%	1039	961	Negativo
delivery_status	0.0159	0.1259	0.2063	89.80%	10.20%	1796	204	Positivo
payment_value	-0.0168	0.1222	0.1626	44.70%	55.30%	894	1106	Negativo
seller_id	-0.0200	0.0966	0.1496	59.25%	40.75%	1185	815	Negativo
freight_value	-0.0244	0.0954	0.1474	47.35%	52.65%	947	1053	Negativo
product_description_lenght	-0.0180	0.0661	0.0947	49.85%	50.15%	997	1003	Negativo

Imagen 30: Valores SHAP modelo Gradient Boosting escenario con 'sentiment_score_pysentimiento'.

Característica	SHAP Promedio	Magnitud Promedio	Desviación Estándar	Porcentaje Positivo	Porcentaje Negativo	Cantidad de Positivos	Cantidad de Negativos	Efecto Global
sentiment_score_pysentimiento	-0.0138	0.1885	0.2231	52.20%	47.80%	1044	956	Negativo
delivery_status	0.0045	0.0553	0.1001	10.40%	89.60%	208	1792	Positivo
price	-0.0002	0.0390	0.0716	67.20%	32.80%	1344	656	Negativo
payment_value	0.0002	0.0297	0.0494	30.10%	69.90%	602	1398	Positivo
avg_spent_per_customer	-0.0000	0.0291	0.0482	29.85%	70.15%	597	1403	Negativo
product_description_lenght	-0.0002	0.0039	0.0054	54.45%	45.55%	1089	911	Negativo
freight_value	0.0000	0.0029	0.0067	70.45%	29.55%	1409	591	Positivo
delivery_difference_days	0.0001	0.0021	0.0037	36.25%	63.75%	725	1275	Positivo
seller_id	0.0000	0.0011	0.0018	67.80%	32.20%	1356	644	Positivo
volume_cm3	-0.0000	0.0005	0.0014	50.15%	49.85%	1003	997	Negativo

Imagen 31: Valores SHAP modelo LSTM escenario con 'sentiment_score_pysentimiento'.

Característica	SHAP Promedio	Magnitud Promedio	Desviación Estándar	Porcentaje Positivo	Porcentaje Negativo	Cantidad de Positivos	Cantidad de Negativos	Efecto Global
sentiment_score_pysentimiento	-0.0133	0.2037	0.2349	52.70%	47.30%	1054	946	Negativo
price	-0.0002	0.0579	0.0998	60.05%	39.95%	1201	799	Negativo
delivery_status	0.0041	0.0471	0.0851	10.40%	89.60%	208	1792	Positivo
payment_value	0.0009	0.0382	0.0611	37.00%	63.00%	740	1260	Positivo
avg_spent_per_customer	0.0009	0.0347	0.0553	37.20%	62.80%	744	1256	Positivo
seller_id	0.0004	0.0084	0.0132	64.55%	35.45%	1291	709	Positivo
freight_value	0.0004	0.0055	0.0119	41.55%	58.45%	831	1169	Positivo
product_description_lenght	-0.0001	0.0050	0.0081	45.20%	54.80%	904	1096	Negativo
delivery_difference_days	0.0003	0.0046	0.0086	31.55%	68.45%	631	1369	Positivo
volume_cm3	0.0000	0.0037	0.0076	61.85%	38.15%	1237	763	Positivo

Imagen 32: Valores SHAP modelo RNN escenario con 'sentiment_score_pysentimiento'.

E.3 SHAP promedio por característica – Modelos con RoBERTa sentiment score

	SHAP Promedio	Magnitud Promedio	Desviación Estándar	Porcentaje Positivo	Porcentaje Negativo	Cantidad de Positivos	Cantidad de Negativos	Efecto Global (Positivo/Negativo)
sentiment_score_roberta	-0.994727	1.795219	1.973830	53.100000	46.900000	1062	938	Negativo
delivery_difference_days	-0.305030	0.674570	0.756847	10.400000	89.600000	208	1792	Negativo
price	0.014520	0.293843	0.375292	59.150000	40.850000	1183	817	Positivo
payment_value	-0.001854	0.217457	0.274068	44.700000	55.300000	894	1106	Negativo
avg_spent_per_customer	-0.091510	0.185077	0.238954	27.500000	72.500000	550	1450	Negativo
delivery_status	0.029909	0.161557	0.256011	89.800000	10.200000	1796	204	Positivo
freight_value	-0.027348	0.112746	0.159349	51.550000	48.450000	1031	969	Negativo
seller_id	-0.008467	0.076505	0.116703	50.300000	49.700000	1006	994	Negativo
product_description_lenght	-0.010626	0.071861	0.104116	39.950000	60.050000	799	1201	Negativo
volume_cm3	-0.029385	0.062424	0.087359	36.000000	64.000000	720	1280	Negativo

Imagen 33: Valores SHAP modelo XGBoost escenario con 'sentiment_score_roberta'.

	SHAP Promedio	Magnitud Promedio	Desviación Estándar	Porcentaje Positivo	Porcentaje Negativo	Cantidad de Positivos	Cantidad de Negativos	Efecto Global
sentiment_score_roberta	-0.0945	0.2485	0.2683	53.00%	47.00%	1060	940	Negativo
delivery_difference_days	-0.0234	0.0707	0.0951	12.35%	87.65%	247	1753	Negativo
price	-0.0026	0.0374	0.0520	56.20%	43.80%	1124	876	Negativo
payment_value	-0.0089	0.0326	0.0425	38.15%	61.85%	763	1237	Negativo
avg_spent_per_customer	-0.0046	0.0291	0.0402	41.00%	59.00%	820	1180	Negativo
freight_value	-0.0034	0.0151	0.0227	43.10%	56.90%	862	1138	Negativo
product_description_lenght	-0.0027	0.0148	0.0224	41.25%	58.75%	825	1175	Negativo
seller_id	-0.0023	0.0131	0.0211	41.70%	58.30%	834	1166	Negativo
volume_cm3	-0.0040	0.0130	0.0190	33.90%	66.10%	678	1322	Negativo
delivery_status	-0.0014	0.0060	0.0170	56.15%	43.85%	1123	877	Negativo

Imagen 34: Valores SHAP modelo Random Forest escenario con 'sentiment_score_roberta'.

	SHAP Promedio	Magnitud Promedio	Desviación Estándar	Porcentaje Positivo	Porcentaje Negativo	Cantidad de Positivos	Cantidad de Negativos	Efecto Global
avg_spent_per_customer	0.2943	0.2943	0.0348	100.00%	0.00%	2000	0	Positivo
delivery_difference_days	-0.0166	0.1079	0.1710	10.20%	89.80%	204	1796	Negativo
payment_value	-0.0006	0.0141	0.0165	43.10%	56.90%	862	1138	Negativo
price	-0.0005	0.0118	0.0136	59.80%	40.20%	1196	804	Negativo
volume_cm3	0.0113	0.0113	0.0014	100.00%	0.00%	2000	0	Positivo
delivery_status	-0.0007	0.0074	0.0132	89.80%	10.20%	1796	204	Negativo
sentiment_score_roberta	0.0015	0.0015	0.0000	100.00%	0.00%	2000	0	Positivo
freight_value	-0.0001	0.0009	0.0014	86.80%	13.20%	1736	264	Negativo
seller_id	-0.0001	0.0009	0.0026	96.60%	3.40%	1932	68	Negativo
product_description_lenght	0.0000	0.0000	0.0000	0.00%	0.00%	0	0	Negativo

Imagen 35: Valores SHAP modelo AdaBoost escenario con 'sentiment_score_roberta'.

	SHAP Promedio	Magnitud Promedio	Desviación Estándar	Porcentaje Positivo	Porcentaje Negativo	Cantidad de Positivos	Cantidad de Negativos	Efecto Global
delivery_difference_days	-0.6642	0.9417	0.7450	10.20%	89.80%	204	1796	Negativo
avg_spent_per_customer	0.6695	0.6816	0.4281	95.35%	4.65%	1907	93	Positivo
sentiment_score_roberta	-0.2704	0.2735	0.1330	2.50%	97.50%	50	1950	Negativo
volume_cm3	-0.0538	0.1926	0.2509	42.05%	57.95%	841	1159	Negativo
price	-0.0279	0.1696	0.2306	51.95%	48.05%	1039	961	Negativo
delivery_status	0.0159	0.1259	0.2063	89.80%	10.20%	1796	204	Positivo
payment_value	-0.0168	0.1222	0.1626	44.70%	55.30%	894	1106	Negativo
seller_id	-0.0200	0.0966	0.1496	59.25%	40.75%	1185	815	Negativo
freight_value	-0.0244	0.0954	0.1474	47.35%	52.65%	947	1053	Negativo
product_description_lenght	-0.0180	0.0661	0.0947	49.85%	50.15%	997	1003	Negativo

Imagen 36: Valores SHAP modelo Gradient Boosting escenario con 'sentiment_score_roberta'.

	Característica	SHAP Promedio	Magnitud Promedio	Desviación Estándar	Porcentaje Positivo	Porcentaje Negativo	Cantidad de Positivos	Cantidad de Negativos	Efecto Global
9	sentiment_score_roberta	-0.0393	0.1977	0.2166	49.60%	50.40%	992	1008	Negativo
6	delivery_status	-0.0006	0.0494	0.0822	10.40%	89.60%	208	1792	Negativo
2	price	-0.0030	0.0349	0.0679	64.30%	35.70%	1286	714	Negativo
0	payment_value	0.0028	0.0246	0.0429	34.75%	65.25%	695	1305	Positivo
7	avg_spent_per_customer	0.0028	0.0240	0.0421	35.20%	64.80%	704	1296	Positivo
4	product_description_lenght	0.0003	0.0058	0.0081	60.25%	39.75%	1205	795	Positivo
5	delivery_difference_days	0.0002	0.0054	0.0091	40.10%	59.90%	802	1198	Positivo
1	seller_id	-0.0002	0.0032	0.0041	53.55%	46.45%	1071	929	Negativo
3	freight_value	-0.0004	0.0019	0.0044	62.35%	37.65%	1247	753	Negativo
8	volume_cm3	-0.0001	0.0012	0.0024	67.45%	32.55%	1349	651	Negativo

Imagen 37: Valores SHAP modelo LSTM escenario con 'sentiment_score_roberta'.

	Característica	SHAP Promedio	Magnitud Promedio	Desviación Estándar	Porcentaje Positivo	Porcentaje Negativo	Cantidad de Positivos	Cantidad de Negativos	Efecto Global
9	sentiment_score_roberta	-0.0316	0.1902	0.2159	49.55%	50.45%	991	1009	Negativo
6	delivery_status	-0.0011	0.0567	0.0939	10.40%	89.60%	208	1792	Negativo
2	price	-0.0052	0.0484	0.0801	56.80%	43.20%	1136	864	Negativo
0	payment_value	0.0036	0.0325	0.0515	37.20%	62.80%	744	1256	Positivo
7	avg_spent_per_customer	0.0038	0.0313	0.0493	38.00%	62.00%	760	1240	Positivo
5	delivery_difference_days	-0.0002	0.0087	0.0163	24.85%	75.15%	497	1503	Negativo
1	seller_id	0.0010	0.0077	0.0102	54.75%	45.25%	1095	905	Positivo
4	product_description_lenght	0.0007	0.0048	0.0074	55.70%	44.30%	1114	886	Positivo
3	freight_value	0.0004	0.0042	0.0133	45.65%	54.35%	913	1087	Positivo
8	volume_cm3	0.0002	0.0026	0.0073	63.45%	36.55%	1269	731	Positivo

Imagen 38: Valores SHAP modelo RNN escenario con 'sentiment_score_roberta'.

F. Fragmentos de Código Utilizado en el Proyecto

```
# ============================
# Carga y Unificación de Datasets Olist
# ============================

Import pandas as pd

reviews_order = pd.read_csv('/Users/ignacioaraneda/Desktop/Magister Analitica Negocios
FEN/2024_Primavera/TallerAFE/Informe_2/archive/olist_order_reviews_dataset.csv')  # Contiene review_id y order_id

order_items = pd.read_csv('/Users/ignacioaraneda/Desktop/Magister Analitica Negocios FEN/2024_Primavera/Taller
AFE/Informe_2/archive/olist_order_items_dataset.csv')      # Contiene order_id y product_id

products = pd.read_csv('/Users/ignacioaraneda/Desktop/Magister Analitica Negocios FEN/2024_Primavera/Taller AFE/Informe_2/archive/olist_products_dataset.csv')
# Contiene product_id y detalles de productos

customers=pd.read_csv('/Users/ignacioaraneda/Desktop/Magister Analitica Negocios FEN/2024_Primavera/Taller AFE/Informe_2/archive/olist_customers_dataset.csv')
# Contiene product_id y detalles de productos

geolocation=pd.read_csv('/Users/ignacioaraneda/Desktop/Magister Analitica Negocios FEN/2024_Primavera/Taller
AFE/Informe_2/archive/olist_geolocation_dataset.csv')            # Contiene product_id y detalles de productos

order_payments=pd.read_csv('/Users/ignacioaraneda/Desktop/Magister Analitica Negocios FEN/2024_Primavera/Taller
AFE/Informe_2/archive/olist_order_payments_dataset.csv')

orders=pd.read_csv('/Users/ignacioaraneda/Desktop/Magister Analitica Negocios FEN/2024_Primavera/Taller AFE/Informe_2/archive/olist_orders_dataset.csv')

sellers=pd.read_csv('/Users/ignacioaraneda/Desktop/Magister Analitica Negocios FEN/2024_Primavera/Taller AFE/Informe_2/archive/olist_sellers_dataset.csv')

df_todo = pd.merge(customers, orders, on='customer_id', how='left')
df_todo = pd.merge(df_todo, order_payments, on='order_id', how='left')
df_todo = pd.merge(df_todo, reviews_order, on='order_id', how='left')
df_todo = pd.merge(df_todo, order_items, on='order_id', how='left')
df_todo = pd.merge(df_todo, products, on='product_id', how='left')
df_todo = pd.merge(df_todo, sellers, on='seller_id', how='left')

# Verificar el resultado

print(f"Número de filas en df_todo: {len(df_todo)}")
display(df_todo.head())

df_todo.shape
df_todo.isnull().sum()
df_todo.columns
```

```
# ==========================================
# Visualización de Review Scores y Retrasos por Estado
# ==========================================

import matplotlib.pyplot as plt

# Contar la frecuencia de cada review_score
review_counts = df_todo['review_score'].value_counts().sort_index()

# Crear el gráfico de barras
plt.figure(figsize=(8,5))
plt.bar(review_counts.index, review_counts.values, color='royalblue', edgecolor='black')

# Configurar etiquetas y título
plt.xlabel('Review Score')
plt.ylabel('Frecuencia')
plt.title('Distribución de Review Score')
plt.xticks([1, 2, 3, 4, 5])  # Asegurar que solo aparezcan los valores 1-5 en el eje X

# Mostrar el gráfico
plt.show()

# Crear nueva columna agrupada: 0 = mala reseña, 1 = buena reseña
df_todo['review_group'] = df_todo['review_score'].apply(lambda x: 0 if x < 3 else 1)

# Calcular medias y medianas
mean_diff = df_todo.groupby('review_group')['delivery_difference_days'].mean()
median_diff = df_todo.groupby('review_group')['delivery_difference_days'].median()

# Crear los subplots
fig, axes = plt.subplots(1, 2, figsize=(16, 6))
plt.subplots_adjust(wspace=0.3)  # espacio entre gráficos

# === Gráfico 1: Histograma ===
axes[0].hist(df_todo['delivery_difference_days'], bins=30, edgecolor='black')
axes[0].set_title('Distribución de delivery_difference_days')
axes[0].set_xlabel('Días de diferencia en la entrega')
axes[0].set_ylabel('Frecuencia')
axes[0].grid(axis='y')

# === Gráfico 2: Boxplot ===
sns.boxplot(x='review_group', y='delivery_difference_days', data=df_todo, ax=axes[1])

# Dibujar medianas
for i, group in enumerate(median_diff.index):
    axes[1].hlines(y=median_diff[group], xmin=i-0.2, xmax=i+0.2, colors='green', linestyles='solid')

# Texto para la leyenda
legend_text = (
    f"Grupo 0 (Review Malo 1-2)\n"
    f"Media: {mean_diff[0]:.2f}\n"
    f"Mediana: {median_diff[0]:.2f}\n\n"
    f"Grupo 1 (Review Bueno 3-5)\n"
    f"Media: {mean_diff[1]:.2f}\n"
    f"Mediana: {median_diff[1]:.2f}"
)

# Agregar texto dentro del segundo gráfico
fig.text(0.5, 0.92, legend_text, fontsize=10, ha='center', bbox=dict(facecolor='white', alpha=0.8))

axes[1].set_title('Días de Diferencia en Entrega por Grupo de Review')
axes[1].set_xlabel('Review Group (0 = Review 1-2 / 1 = Review 3-5)')
axes[1].set_ylabel('Días de diferencia en la entrega\n(Negativo = Adelanto, Positivo = Retraso)')
axes[1].set_xticks([0,1])
axes[1].set_xticklabels(['Review Malo (1-2)', 'Review Bueno (3-5)'])
axes[1].grid(axis='y')

plt.show()

# Eliminar columna auxiliar
df_todo = df_todo.drop(columns=['review_group'])
```

```
# ==============================================
# Análisis de Categorías de Producto y Visualización de Resultados
# ==============================================

# Agrupar: promedio de review + cantidad de productos vendidos
categoria_stats = df_todo.groupby('product_category_name') \
    .agg(
        avg_score=('review_score', 'mean'),
        cantidad_vendida=('product_category_name', 'count')  # contamos filas
    ) \
    .reset_index()
# Filtrar categorías con al menos 50 ventas
categoria_stats = categoria_stats[categoria_stats['cantidad_vendida'] >= 50]
# Top 10 con peor puntaje promedio
top_10_bajas = categoria_stats.sort_values(by='avg_score', ascending=True).head(10)
# Top 10 más vendidas
top_10_vendidas = categoria_stats.sort_values(by='cantidad_vendida', ascending=False).head(10)
# Gráficos
fig, axes = plt.subplots(1, 2, figsize=(22, 7))
plt.subplots_adjust(wspace=0.3)  # espacio entre gráficos
# === GRÁFICO 1: Peores categorías ===
ax1 = axes[0]
color_barras = 'lightcoral'
color_linea = 'tab:blue'
ax1.set_title('Top 10 Categorías con Peor Review Score')
bars1 = ax1.bar(range(len(top_10_bajas)), top_10_bajas['cantidad_vendida'], color=color_barras, alpha=0.8)
ax1.set_ylabel('Cantidad Vendida', color=color_barras)
ax1.tick_params(axis='y', labelcolor=color_barras)
# Etiquetas en X
ax1.set_xticks(range(len(top_10_bajas)))
ax1.set_xticklabels(top_10_bajas['product_category_name'], rotation=45, ha='right')

# Segundo eje: review score promedio (línea)
ax1b = ax1.twinx()
ax1b.plot(range(len(top_10_bajas)), top_10_bajas['avg_score'], color=color_linea, marker='o', linewidth=2)
ax1b.set_ylabel('Review Score Promedio', color=color_linea)
ax1b.tick_params(axis='y', labelcolor=color_linea)
# === GRÁFICO 2: Más vendidas ===

ax2 = axes[1]
color_barras2 = 'mediumseagreen'
color_linea2 = 'tab:blue'
ax2.set_title('Top 10 Categorías Más Vendidas')
bars2 = ax2.bar(range(len(top_10_vendidas)), top_10_vendidas['cantidad_vendida'], color=color_barras2, alpha=0.8)
ax2.set_ylabel('Cantidad Vendida', color=color_barras2)
ax2.tick_params(axis='y', labelcolor=color_barras2)
# Etiquetas en X

ax2.set_xticks(range(len(top_10_vendidas)))
ax2.set_xticklabels(top_10_vendidas['product_category_name'], rotation=45, ha='right')

# Segundo eje: review score promedio (línea)
ax2b = ax2.twinx()
ax2b.plot(range(len(top_10_vendidas)), top_10_vendidas['avg_score'], color=color_linea2, marker='o', linewidth=2)
ax2b.set_ylabel('Review Score Promedio', color=color_linea2)
ax2b.tick_params(axis='y', labelcolor=color_linea2)

# Ajustar todo
plt.tight_layout()
plt.show()
```

```
# ==================================================
# Verificación de Reglas Temporales y Cálculo de 'delivery_status'
# ==================================================

# Función para verificar reglas específicas
def verificar_reglas(df):
    inconsistencias = []

    # Regla 1: 'order_approved_at' no puede ser mayor a 'order_purchase_timestamp'
    regla1 = df[df['order_approved_at'] < df['order_purchase_timestamp']]
    for idx, row in regla1.iterrows():
        inconsistencias.append({
            'index': idx,
            'error': "'order_approved_at' < 'order_purchase_timestamp'",
            'valor_1': row['order_approved_at'],
            'valor_2': row['order_purchase_timestamp']
        })

    # Regla 2: 'order_delivered_carrier_date' no puede ser mayor a 'order_delivered_customer_date'
    regla2 = df[df['order_delivered_carrier_date'] > df['order_delivered_customer_date']]
    for idx, row in regla2.iterrows():
        inconsistencias.append({
            'index': idx,
            'error': "'order_delivered_carrier_date' > 'order_delivered_customer_date'",
            'valor_1': row['order_delivered_carrier_date'],
            'valor_2': row['order_delivered_customer_date']
        })

    # Regla 3: 'review_answer_timestamp' no puede ser mayor a 'review_creation_date'
    regla3 = df[df['review_answer_timestamp'] < df['review_creation_date']]
    for idx, row in regla3.iterrows():
        inconsistencias.append({
            'index': idx,
            'error': "'review_answer_timestamp' < 'review_creation_date'",
            'valor_1': row['review_answer_timestamp'],
            'valor_2': row['review_creation_date']
        })

    # Regla 4: 'order_delivered_carrier_date' y 'order_delivered_customer_date' no pueden ser menores a 'order_approved_at'
    regla4 = df[(df['order_delivered_carrier_date'] < df['order_approved_at']) |
                (df['order_delivered_customer_date'] < df['order_approved_at'])]
    for idx, row in regla4.iterrows():
        inconsistencias.append({
            'index': idx,
            'error': "'order_delivered_carrier_date' o 'order_delivered_customer_date' < 'order_approved_at'",
            'valor_1': row['order_delivered_carrier_date'],
            'valor_2': row['order_delivered_customer_date'],
            'order_approved_at': row['order_approved_at']
        })

    return pd.DataFrame(inconsistencias)

# Aplicar las verificaciones a df_todo
inconsistencias_df = verificar_reglas(df_todo)

# Mostrar resultados
if inconsistencias_df.empty:
    print("No se encontraron inconsistencias en df_todo.")
else:
    print("Se encontraron las siguientes inconsistencias en df_todo:")
    print(inconsistencias_df)

if inconsistencias_df.empty:
    print("No se encontraron inconsistencias en df_todo.")
else:

    # Eliminar las filas inconsistentes usando los índices
    indices_inconsistentes = inconsistencias_df['index'].unique()
    df_todo = df_todo.drop(index=indices_inconsistentes)

# Calcular el delay (diferencia en días)
df_todo['delivery_delay'] = (df_todo['order_delivered_customer_date'] - df_todo['order_estimated_delivery_date']).dt.total_seconds()

# Categorizar el resultado de delivery_delay
df_todo['delivery_status'] = df_todo['delivery_delay'].apply(
    lambda x: 'Retraso' if x > 0 else ('Adelanto' if x < 0 else 'Exacto')
)
```

```
# ==================================================
# Ingeniería de Características y Limpieza Final del Dataset
# ==================================================
# Calcular la diferencia en días entre la fecha de entrega y la fecha estimada
df_todo['delivery_difference_days'] = (df_todo['order_delivered_customer_date'] - df_todo['order_estimated_delivery_date']).dt.days


# Extraer componentes temporales
df_todo['purchase_day'] = df_todo['order_purchase_timestamp'].dt.day
df_todo['purchase_month'] = df_todo['order_purchase_timestamp'].dt.month
df_todo['purchase_year'] = df_todo['order_purchase_timestamp'].dt.year
df_todo['purchase_weekday'] = df_todo['order_purchase_timestamp'].dt.weekday
df_todo['purchase_hour'] = df_todo['order_purchase_timestamp'].dt.hour

df_todo[['purchase_day', 'purchase_weekday', 'review_score','purchase_month','purchase_year','purchase_hour','order_purchase_timestamp']].corr()

df_todo = df_todo.drop(columns=['purchase_month', 'purchase_year'])



# Combinar 'review_comment_title' y 'review_comment_message'
df_todo['review_comment_message'] = df_todo.apply(
    lambda row: f"{row['review_comment_title']} - {row['review_comment_message']}"
    if pd.notnull(row['review_comment_title']) and pd.notnull(row['review_comment_message'])
    else row['review_comment_title'] if pd.notnull(row['review_comment_title'])
    else row['review_comment_message'],
    axis=1
)

# Eliminar la columna 'review_comment_title'
df_todo = df_todo.drop(columns=['review_comment_title'])

# Verificar el resultado
print(df_todo[['review_comment_message']].head())

# Calcular el número de pedidos por cliente único
df_todo['orders_per_customer'] = df_todo.groupby('customer_unique_id')['order_id'].transform('count')

# Calcular el gasto promedio por cliente único
df_todo['avg_spent_per_customer'] = df_todo.groupby('customer_unique_id')['payment_value'].transform('mean')

drop_columns = ['product_id','order_id','customer_unique_id','customer_id', 'customer_city','shipping_limit_date','review_id',
    'seller_city', 'seller_state',
    'customer_zip_code_prefix', 'seller_zip_code_prefix', 'delayed_count', 'order_item_id',
    'product_name_lenght', 'approval_delay','order_approved_at', 'review_creation_date', 'review_answer_timestamp','order_purchase_timestamp',
      'order_delivered_carrier_date', 'order_delivered_customer_date',
      'order_estimated_delivery_date', 'order_status'
]
df_todo = df_todo.drop(columns=drop_columns)

# Crear la columna 'volume_cm3' a partir de las dimensiones
df_todo['volume_cm3'] = (
    df_todo['product_width_cm'] *
    df_todo['product_length_cm'] *
    df_todo['product_height_cm']
)
# Eliminar las columnas originales si ya no son necesarias
df_todo = df_todo.drop(columns=['product_width_cm', 'product_length_cm', 'product_height_cm'])
# Filtrar solo las órdenes con estado 'delivered'
df_todo = df_todo[df_todo['order_status'] == 'delivered']
```

```
# ==================================================
# Agrupación de Categorías Similares y Codificación de Variables
# ==================================================

from Levenshtein import ratio
import pandas as pd

# Lista de categorías únicas
categories = df_todo['product_category_name'].unique()

# Umbral para considerar categorías como similares
threshold = 0.5  # Ajusta según el nivel de similitud deseado

# Crear un diccionario para agrupar categorías similares
category_mapping = {}

for i, category in enumerate(categories):
    for j, similar_category in enumerate(categories):
        if i != j:
            similarity = ratio(category, similar_category)
            if similarity >= threshold:  # Si la similitud es mayor al umbral
                if category not in category_mapping:
                    category_mapping[category] = category  # Grupo base
                category_mapping[similar_category] = category_mapping[category]

# Completar categorías que no fueron agrupadas
for category in categories:
    if category not in category_mapping:
        category_mapping[category] = category

# Aplicar el mapeo al DataFrame
df_todo['product_category_name'] = df_todo['product_category_name'].apply(lambda x: category_mapping.get(x, x))

drop_columns = ['product_id','order_id','customer_unique_id','customer_id', 'customer_city','shipping_limit_date','review_id',
    'seller_city', 'seller_state',
    'customer_zip_code_prefix', 'seller_zip_code_prefix', 'delayed_count', 'order_item_id',
    'product_name_lenght', 'approval_delay','order_approved_at', 'review_creation_date', 'review_answer_timestamp','order_purchase_timestamp',
      'order_delivered_carrier_date', 'order_delivered_customer_date',
      'order_estimated_delivery_date', 'order_status'
]

df_todo = df_todo.drop(columns=drop_columns)



# Convertir columnas categóricas nominales en variables dummy
df_todo = pd.get_dummies(df_todo, columns=['payment_type'], drop_first=True)

from sklearn.preprocessing import LabelEncoder

# LabelEncoder para columnas ordinales (con orden implícito)
label_encoder = LabelEncoder()
df_todo['delivery_status'] = label_encoder.fit_transform(df_todo['delivery_status'])
df_todo['seller_id'] = label_encoder.fit_transform(df_todo['seller_id'])
df_todo['product_category_name'] = label_encoder.fit_transform(df_todo['product_category_name'])
df_todo['customer_state'] = label_encoder.fit_transfor_]()_


# Aplicar el mapeo al DataFrame
df_todo['product_category_name'] = df_todo['product_category_name'].apply(lambda x: category_mapping.get(x, x))

# Convertir columnas categóricas nominales en variables dummy
df_todo = pd.get_dummies(df_todo, columns=['payment_type'], drop_first=True)

from sklearn.preprocessing import LabelEncoder

# LabelEncoder para columnas ordinales (con orden implícito)
label_encoder = LabelEncoder()
df_todo['delivery_status'] = label_encoder.fit_transform(df_todo['delivery_status'])
df_todo['seller_id'] = label_encoder.fit_transform(df_todo['seller_id'])
df_todo['product_category_name']=label_encoder.fit_transform(df_todo['product_category_name'])
df_todo['customer_state']=label_encoder.fit_transform(df_todo['customer_state'])
```

```
# ==================================================
# Análisis de Sentimiento con Modelos Pysentimiento y VADER
# ==================================================
df_review_comments = df_todo[['review_comment_message','review_score']]
# Rellenar valores nulos con 'desconocido'
from nltk.corpus import stopwords
stop_words = stopwords.words('portuguese')

# Función para eliminar stop words
df_todo['review_comment_message'] = df_todo['review_comment_message'].apply(
    lambda x: ' '.join([word for word in x.split() if word not in stop_words]) if x != 'desconocido' else x
)
df_review_comments['review_comment_message'] = df_review_comments['review_comment_message'].fillna('desconocido')

# Convertir todos los valores a cadenas
df_review_comments['review_comment_message'] = df_review_comments['review_comment_message'].astype(str)

# Función para limpiar el texto
def clean_text(text):
    text = re.sub(r'[^a-zA-Záéíóúãõçüñ\s]', ' ', text)  # Reemplazar caracteres especiales y números por espacios
    text = re.sub(r'\s+', ' ', text)  # Reemplazar múltiples espacios por uno solo
    return text.lower().strip()  # Convertir a minúsculas y eliminar espacios extra

# Aplicar limpieza al texto
df_review_comments['clean_text'] = df_review_comments['review_comment_message'].apply(clean_text)

# Configurar GPU si está disponible
device = 0 if torch.cuda.is_available() else -1

# Cargar el modelo optimizado para análisis de sentimiento en portugués
sentiment_pipeline = pipeline("sentiment-analysis", model="pysentimiento/bertweet-pt-sentiment", device=device)

# Procesar en lotes para evitar problemas de memoria
batch_size = 500  # Ajusta el tamaño según la RAM disponible
num_batches = len(df_review_comments) // batch_size + 1
all_scores = []

for i in range(num_batches):
    batch = df_review_comments['clean_text'][i * batch_size:(i + 1) * batch_size].tolist()
    try:
        sentiments = sentiment_pipeline(batch)
        scores = [
            {'NEG': -1, 'NEU': 0, 'POS': 1}.get(sent['label'], 0) * sent['score']
            for sent in sentiments
        ]
        all_scores.extend(scores)
    except Exception as e:
        print(f" Error procesando batch {i}: {e}")
        all_scores.extend([0] * len(batch))  # Si falla, asigna 0s para evitar errores

# Asignar los resultados a la columna
df_review_comments['sentiment_score_pysentimiento'] = all_scores

from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

# Inicializar el analizador de VADER
analyzer = SentimentIntensityAnalyzer()

# Función para calcular sentimiento con VADER
def vader_sentiment(text):
    if text == 'desconocido' or len(text.strip()) == 0:
        return 0  # Asignar 0 si el texto es vacío
    try:
        score = analyzer.polarity_scores(text)['compound']  # Obtener el score compuesto
        return score  # Devuelve un valor entre -1 (negativo) y 1 (positivo)
    except Exception as e:
        print(f"Error procesando el texto con VADER: {text}, Error: {e}")
        return 0  # Asignar 0 en caso de error

# Aplicar el análisis de sentimiento con VADER
df_review_comments['sentiment_score_vader'] = df_review_comments['review_comment_message'].apply(vader_sentiment)
```

```
# ==============================================
# Análisis de Sentimiento con Modelos Roberta, BETO y TextBlob
# ==============================================


# Rellenar valores nulos con 'desconocido'
df_review_comments['review_comment_message'] = df_review_comments['review_comment_message'].fillna('desconocido')

# Convertir todos los valores a cadenas
df_review_comments['review_comment_message'] = df_review_comments['review_comment_message'].astype(str)

# Reemplazar símbolos y números por espacios
def clean_text(text):
    text = re.sub(r'[^a-zA-Z\s]', ' ', text)  # Reemplazar números y símbolos por espacios
    text = re.sub(r'\s+', ' ', text)  # Reemplazar múltiples espacios por uno solo
    text = text.lower().strip()  # Convertir a minúsculas y eliminar espacios extra
    return text
df_review_comments['review_comment_message'] = df_review_comments['review_comment_message'].apply(clean_text)
# Cargar el modelo multilingüe optimizado para sentimientos
sentiment_pipeline = pipeline("sentiment-analysis", model="cardiffnlp/twitter-xlm-roberta-base-sentiment")
# Calcular sentimiento
def calculate_sentiment(text):
    if text == 'desconocido' or len(text.strip()) == 0:
        return 0  # Asignar 0 si el texto es 'desconocido'
    try:
        sentiment = sentiment_pipeline(text)[0]
        # Convertir etiquetas a valores numéricos
        label_to_score = {
            'negative': -1,  # Sentimiento negativo
            'neutral': 0,    # Sentimiento neutro
            'positive': 1    # Sentimiento positivo
        }
        return label_to_score.get(sentiment['label'], 0) * sentiment['score']  # Escalar el puntaje por el score
    except Exception as e:
        print(f"Error procesando el texto: {text}, Error: {e}")
        return 0  # Asignar 0 en caso de error
df_review_comments['sentiment_score_bert'] = df_review_comments['review_comment_message'].apply(calculate_sentiment)

# Verificar resultados
print(df_review_comments[['review_comment_message', 'sentiment_score_bert']].head())
# Guardar resultados en un archivo CSV
df_review_comments.to_csv('/content/review_comment_message_sentiment_roberta.csv', index=False)
print("Archivo 'review_comment_message_sentiment.csv' generado exitosamente.")
# Cargar el modelo BETO multilingüe para análisis de sentimiento
sentiment_pipeline_beto = pipeline("sentiment-analysis", model="nlptown/bert-base-multilingual-uncased-sentiment")
# Función para calcular sentimiento con BETO
def beto_sentiment(text):
    if text == 'desconocido' or len(text.strip()) == 0:
        return 0  # Asignar 0 si el texto es vacío
    try:
        sentiment = sentiment_pipeline_beto(text)[0]
        score = int(sentiment['label'][0])  # Extraer el número de la etiqueta (1-5)
        return score - 3  # Normalizar: [-2, -1, 0, 1, 2]
    except Exception as e:
        print(f"Error procesando el texto con BETO: {text}, Error: {e}")
        return 0  # Asignar 0 en caso de error
# Aplicar el análisis de sentimiento con BETO
df_review_comments['sentiment_score_beto'] = df_review_comments['review_comment_message'].apply(beto_sentiment)
# Guardar resultados en CSV
df_review_comments.to_csv('/content/review_comment_message_sentiment_beto.csv', index=False)
print(" Archivo 'review_comment_message_sentiment_beto.csv' generado exitosamente.")
# Rellenar valores nulos con 'desconocido'
df_review_comments['review_comment_message'] = df_review_comments['review_comment_message'].fillna('desconocido')

# Convertir los valores a cadenas
df_review_comments['review_comment_message'] = df_review_comments['review_comment_message'].astype(str)

# Función para analizar sentimiento con TextBlob
def calculate_sentiment_textblob(text):
    if text == 'desconocido' or len(text.strip()) == 0:
        return 0  # Sentimiento neutral si está vacío o desconocido
    try:
        blob = TextBlob(text)
        return blob.sentiment.polarity  # Devolver polaridad (-1 a 1)
    except Exception as e:
        print(f"Error procesando el texto: {text}, Error: {e}")
        return 0

# Aplicar la función al DataFrame
df_review_comments['sentiment_score_textblob'] = df_review_comments['review_comment_message'].apply(calculate_sentiment_textblob)
```

```
# ==============================================
# Comparación de Modelos de Análisis de Sentimiento vs Review Score
# ==============================================


df = pd.read_csv('/content/review_comment_message_sentiment_roberta.csv')  # CSV con todas las columnas
df_beto = pd.read_csv('/content/review_comment_message_sentiment_beto.csv')
df_vader = pd.read_csv('/content/review_comment_message_sentiment_vader.csv')
df_textblob = pd.read_csv('/content/review_comment_message_sentiment_textblob.csv')
df_pysentimiento= pd.read_csv('/content/review_comment_message_sentiment_pysentimiento.csv')
# Asegurar que las columnas de sentimiento estén en el mismo DataFrame
df['sentiment_score_beto'] = df_beto['sentiment_score_beto']
df['sentiment_score_vader'] = df_vader['sentiment_score_vader']
df['sentiment_score_textblob'] = df_textblob['sentiment_score_textblob']
df['sentiment_score_pysentimiento'] = df_pysentimiento['sentiment_score_pysentimiento']
# Normalizar el review_score a un rango de -2 a 2 para que sea comparable con BETO
df['review_score_normalized'] = df['review_score'] - 3

# Normalizar los sentiment scores a la misma escala
scaler = MinMaxScaler(feature_range=(-2, 2))
df[['sentiment_score_pysentimiento','sentiment_score_bert', 'sentiment_score_beto', 'sentiment_score_vader', 'sentiment_score_textblob']] = scaler.fit_transform(
    df[['sentiment_score_pysentimiento','sentiment_score_bert', 'sentiment_score_beto', 'sentiment_score_vader', 'sentiment_score_textblob']]
)
# Definir los modelos de sentimiento a evaluar
models = {
    "RoBERTa": df['sentiment_score_bert'],
    "TextBlob": df['sentiment_score_textblob'],
    "BETO": df['sentiment_score_beto'],
    "VADER": df['sentiment_score_vader'],
    "PySentimiento": df['sentiment_score_pysentimiento']
}
# Calcular métricas: Pearson, Spearman, MAE, RMSE y R²
metrics = []
for model, predictions in models.items():
    pearson_corr, _ = pearsonr(df['review_score_normalized'], predictions)
    spearman_corr, _ = spearmanr(df['review_score_normalized'], predictions)
    mae = mean_absolute_error(df['review_score_normalized'], predictions)
    rmse = np.sqrt(mean_squared_error(df['review_score_normalized'], predictions))
    r2 = r2_score(df['review_score_normalized'], predictions)

    metrics.append([model, pearson_corr, spearman_corr, mae, rmse, r2])

# Crear DataFrame con los resultados ordenados por menor MAE
df_metrics = pd.DataFrame(metrics, columns=['Modelo', 'Pearson', 'Spearman', 'MAE', 'RMSE', 'R²'])
df_metrics = df_metrics.sort_values(by='MAE', ascending=True)

# Mostrar tabla de resultados
from IPython.display import display
display(df_metrics)

# Visualizar Correlaciones Pearson y Spearman
plt.figure(figsize=(10,5))
df_melted = df_metrics.melt(id_vars="Modelo", value_vars=["Pearson", "Spearman"], var_name="Métrica", value_name="Valor")
sns.barplot(data=df_melted, x="Modelo", y="Valor", hue="Métrica", palette="viridis")
plt.title("Comparación de Correlaciones: Pearson vs Spearman")
plt.ylabel("Correlación")
plt.xlabel("Modelo")
plt.show()

#  Visualizar Error (MAE y RMSE)
plt.figure(figsize=(10,5))
df_melted = df_metrics.melt(id_vars="Modelo", value_vars=["MAE", "RMSE"], var_name="Métrica", value_name="Valor")
sns.barplot(data=df_melted, x="Modelo", y="Valor", hue="Métrica", palette="coolwarm")
plt.title("Comparación de Errores: MAE vs RMSE")
plt.ylabel("Error")
plt.xlabel("Modelo")
plt.show()

# Graficar relación entre cada modelo y el review_score_normalized con línea de tendencia
plt.figure(figsize=(12, 8))
for i, (model, predictions) in enumerate(models.items(), 1):
    plt.subplot(3, 2, i)
    sns.scatterplot(x=df['review_score_normalized'], y=predictions, alpha=0.5)
    sns.regplot(x=df['review_score_normalized'], y=predictions, scatter=False, color='red')  # Línea de tendencia
    plt.title(f"Comparación {model} vs Review Score")
    plt.xlabel("Review Score Normalizado")
    plt.ylabel("Sentiment Score")
plt.tight_layout()
plt.show()
```

```
# ==============================================
# Modelado de Tópicos con BERTopic y Evaluación de Coherencia
# ==============================================

from bertopic import BERTopic
import pandas as pd
from hdbscan import HDBSCAN
from gensim.models.coherencemodel import CoherenceModel
from gensim.corpora import Dictionary
from nltk.stem import WordNetLemmatizer
import nltk

from sentence_transformers import SentenceTransformer
# Fijar semillas para reproducibilidad
SEED = 42
random.seed(SEED)
np.random.seed(SEED)
torch.manual_seed(SEED)
if torch.cuda.is_available():
    torch.cuda.manual_seed_all(SEED)
# Descargar recursos de NLTK
nltk.download('wordnet')
# Rellenar valores nulos con 'desconocido'
df_review_comments['review_comment_message'] = df_review_comments['review_comment_message'].fillna('desconocido')

# Convertir todos los valores a cadenas
df_review_comments['review_comment_message'] = df_review_comments['review_comment_message'].astype(str)

# Función para limpiar y lematizar el texto
lemmatizer = WordNetLemmatizer()

def clean_text(text):
    text = re.sub(r'[^a-zA-Záéíóúãõçüñ\s]', ' ', text)  # Eliminar números y símbolos
    text = re.sub(r'\s+', ' ', text)  # Reemplazar múltiples espacios por uno solo
    text = text.lower().strip()  # Convertir a minúsculas y eliminar espacios extra
    # Lematización
    text = ' '.join([lemmatizer.lemmatize(word) for word in text.split()])
    return text

df_review_comments['clean_text'] = df_review_comments['review_comment_message'].apply(clean_text)
# Configurar HDBSCAN
hdbscan_model = HDBSCAN(min_cluster_size=10, metric='euclidean', cluster_selection_method='eom', prediction_data=True)

# Usar Sentence-BERT para generar embeddings
sentence_model = SentenceTransformer("paraphrase-multilingual-MiniLM-L12-v2")
embeddings = sentence_model.encode(df_review_comments['clean_text'])
# Evaluar diferentes valores de nr_topics (clusters)
best_coherence = -1
best_nr_topics = None
best_topic_model = None
for nr_topics in range(7, 12):  # De 7 a 11 clusters
    print(f"🔄 Probando BERTopic con {nr_topics} clusters...")

    # Crear y entrenar el modelo BERTopic con el número de clusters actual
    topic_model = BERTopic(
        language="portuguese",
        hdbscan_model=hdbscan_model,
        embedding_model=sentence_model,
        nr_topics=nr_topics
    )
    topics, probs = topic_model.fit_transform(df_review_comments['clean_text'], embeddings)
    # Obtener las palabras clave de cada tema generado por BERTopic (excluyendo el tema -1)
    topics_dict = topic_model.get_topics()
    topic_words = [[word for word, _ in topics_dict[topic_id]] for topic_id in topics_dict if topic_id != -1]
    # Tokenizar los textos originales para la evaluación de coherencia
    texts = [text.split() for text in df_review_comments['clean_text']]

    # Crear el diccionario de palabras con Gensim
    dictionary = Dictionary(texts)
    corpus = [dictionary.doc2bow(text) for text in texts]
    # Calcular la coherencia del modelo BERTopic con la métrica c_v
    if topic_words:
        coherence_model = CoherenceModel(topics=topic_words, texts=texts, dictionary=dictionary, coherence='c_v')
        coherence_score = coherence_model.get_coherence()
        print(f"📊 Coherencia para {nr_topics} clusters: {coherence_score:.4f}")

        # Guardar el mejor modelo basado en coherencia
        if coherence_score > best_coherence:
            best_coherence = coherence_score
            best_nr_topics = nr_topics
            best_topic_model = topic_model
    else:
        print(f" No se encontraron temas válidos para {nr_topics} clusters.")
```

```
# ==========================================
# Asignación de Tópicos Óptimos y Exportación de Resultados
# ==========================================

# Confirmar el mejor modelo encontrado
if best_topic_model:
    print(f"\n Mejor modelo encontrado con {best_nr_topics} clusters (coherencia {best_coherence:.4f})")

    # Asignar los temas detectados al DataFrame
    df_review_comments['topic'] = best_topic_model.fit_transform(df_review_comments['clean_text'], embeddings)[0]
    # Guardar los resultados en un CSV
df_review_comments.to_csv('/content/review_comment_message_topics_best.csv', index=False)
    print(" Archivo 'review_comment_message_topics_best.csv' generado exitosamente.")

    # Mostrar los temas principales
    print(best_topic_model.get_topic_info())
else:
    print("No se pudo encontrar un modelo válido con los valores de clusters evaluados.")
```

```
# ==============================================
# Transformación de la Variable Objetivo y Selección de Características
# ==============================================

import pandas as pd

df_todo['review_score'] = df_todo['review_score'].apply(lambda x: 'malo' if x in [1, 2]  else ('bueno'))

# **Paso 1: Dividir características (X) y variable objetivo (y)**
X = df_todo.drop(['review_score'], axis=1)  # Eliminar la variable target
encoder = LabelEncoder()
y = encoder.fit_transform(df_todo['review_score'])  # Variable objetivo

#  Verificar cómo fueron asignadas las etiquetas
print("Clases asignadas por LabelEncoder:", list(encoder.classes_))

#  Verificar ejemplos en y
print("Valores únicos en y:", set(y))

from sklearn.feature_selection import SelectKBest, mutual_info_classif, RFE
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import LabelEncoder, StandardScaler
import numpy as np

np.random.seed(42)

# **Paso 2: Normalización (Recomendado para modelos como RFE)**
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# **Paso 3: Selección inicial con SelectKBest**
k_initial = 10  # Se eligen más características de las necesarias para luego refinar con RFE
selector_kbest = SelectKBest(score_func=mutual_info_classif, k=k_initial)
X_kbest = selector_kbest.fit_transform(X_scaled, y)

# Obtener las columnas seleccionadas por SelectKBest
selected_kbest = X.columns[selector_kbest.get_support()]


# **Paso 5: Resultados finales**
print("Características seleccionadas después de SelectKBest:", list(selected_kbest))
```

```
# ==============================================
# Preprocesamiento, Escalado de Variables y Balanceo con SMOTE
# ==============================================


# Preprocesamiento de datos
from sklearn.preprocessing import MinMaxScaler
from imblearn.over_sampling import SMOTE

# Lista de variables numéricas continuas a escalar con MinMaxScaler
variables_numericas = [
    'price',                    # Precio del producto
    'product_weight_g',         # Peso del producto en gramos
    'freight_value',            # Costo del envío
    'payment_value',            # Valor total del pago
    'avg_spent_per_customer',   # Gasto promedio por cliente
    'volume_cm3',               # Volumen del producto en cm3
    'product_description_lenght',# Longitud de la descripción del producto
    'product_photos_qty',       # Cantidad de fotos del producto
    'delivery_difference_days',  # Diferencia de días en la entrega
    'orders_per_customer'       # Número de pedidos por cliente
]
# scaler = MinMaxScaler()
# X_scaled = scaler.fit_transform(X_selected)
# # Crear una copia del DataFrame original para mantener los datos intactos
# df_scaled = df_todo.copy()

# Inicializar MinMaxScaler
scaler = MinMaxScaler()

# Escalar directamente sobre df_todo
df_todo[variables_numericas] = scaler.fit_transform(df_todo[variables_numericas])
# Escalado
X_selected = X[selected_kbest]

# Balanceo con SMOTE
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_selected, y)

# Dividir en entrenamiento y prueba
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled, test_size=0.2, random_state=42, stratify=y_resampled)

print("Preprocesamiento completo. Datos listos para entrenar.")

#Para los shap

# Volver a dividir los datos originales antes de SMOTE
X_train_real, X_test_real, y_train_real, y_test_real = train_test_split(X_selected, y,
                                                                        test_size=0.2,
                                                                        stratify=y,
                                                                        random_state=42)
X_sample, _, y_sample, _ = train_test_split(X_test_real, y_test_real,
                                            train_size=2000,
                                            stratify=y_test_real,
                                            random_state=42)
```

```
# ==============================================
# Entrenamiento, Optimización y Evaluación de Modelos de Clasificación
# ==============================================


import joblib
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier, AdaBoostClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report, accuracy_score, f1_score
from xgboost import XGBClassifier

# Calcular el ratio entre clases para XGBoost (usado en scale_pos_weight)
ratio = (y_train == 0).sum() / (y_train == 1).sum()

# Definir los modelos con sus respectivos hiperparámetros para GridSearchCV
param_grids = {
    "Random Forest": {
        "model": RandomForestClassifier(random_state=42, class_weight='balanced'),  # ✅ Ajusta el peso de las clases
        "params": {
            "n_estimators": [100, 200],
            "max_depth": [None, 10, 20],
            "min_samples_split": [2, 5]
        }
    },
    "Gradient Boosting": {
        "model": GradientBoostingClassifier(random_state=42),  # No soporta class_weight directamente
        "params": {
            "n_estimators": [100, 200],
            "learning_rate": [0.01, 0.1],
            "max_depth": [3, 5]
        }
    },
    "AdaBoost": {
        "model": AdaBoostClassifier(random_state=42),  # AdaBoost no soporta class_weight directamente
        "params": {
            "n_estimators": [50, 100],
            "learning_rate": [0.01, 0.1, 1]
        }
    },
    "XGBoost": {
        "model": XGBClassifier(random_state=42, scale_pos_weight=ratio, use_label_encoder=False, eval_metric='logloss'),  # ✅ Ajuste de peso
        "params": {
            "n_estimators": [100, 200],
            "learning_rate": [0.01, 0.1],
            "max_depth": [3, 5]
        }
    }
}

# Entrenar, realizar GridSearch y guardar los mejores modelos
for name, config in param_grids.items():
    print(f"\nRealizando GridSearch para {name}...")

    # GridSearchCV con F1 Macro para optimizar el equilibrio entre clases
    grid_search = GridSearchCV(estimator=config["model"], param_grid=config["params"],
                               scoring="f1_macro", cv=3, n_jobs=-1, verbose=1)
    grid_search.fit(X_train, y_train)

    # Mejor modelo y sus hiperparámetros
    best_model = grid_search.best_estimator_
    best_params = grid_search.best_params_
    print(f"Mejores hiperparámetros para {name}: {best_params}")

    # Evaluar el modelo en los datos de prueba
    y_pred = best_model.predict(X_test)
    print(f"\nResultados para {name}:")
    print(classification_report(y_test, y_pred))
    print("Accuracy:", accuracy_score(y_test, y_pred))
    print("F1 Macro:", f1_score(y_test, y_pred, average='macro'))
    print("F1 Weighted:", f1_score(y_test, y_pred, average='weighted'))

    # Guardar el modelo entrenado con un nombre basado en el algoritmo
    model_filename = f"{name.lower().replace(' ', '_')}_best_model_pysentimiento.pkl"
    joblib.dump(best_model, model_filename)
    print(f"Modelo {name} guardado como {model_filename}")
```

```
# ==================================================
# Interpretabilidad del Modelo con Importancia y SHAP para XGBoost
# ==============================================


import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import pandas as pd


# Obtener importancia de características
feature_importance = xgb_model.feature_importances_
importance_df = pd.DataFrame({"Feature": selected_kbest, "Importance": feature_importance})

# Ordenar por importancia descendente
importance_df = importance_df.sort_values(by="Importance", ascending=False)

# Mostrar importancia de características
print("Importancia de Características en XGBoost:\n", importance_df)

# Gráfico de barras
plt.figure(figsize=(10, 5))
plt.barh(importance_df["Feature"], importance_df["Importance"])
plt.xlabel("Importancia")
plt.ylabel("Características")
plt.title("Importancia de Características - XGBoost")
plt.gca().invert_yaxis()  # Invertir para que las más importantes estén arriba
plt.show()
import shap

# Crear el TreeExplainer para XGBoost
explainer = shap.TreeExplainer(xgb_model)

# Calcular valores SHAP
shap_values_xgb = explainer.shap_values(X_sample)

# Verificar la forma de shap_values
print("Forma de los valores SHAP:", shap_values_xgb.shape)

# Si hay tres dimensiones, seleccionar solo la clase positiva (1)
if len(shap_values_xgb.shape) == 3:
    shap_values_xgb = shap_values_xgb[:, :, 1]  # Extraer valores SHAP para la clase 1

# Convertir a DataFrame
shap_df = pd.DataFrame(shap_values_xgb, columns=selected_kbest)

# Generar el gráfico de resumen
shap.summary_plot(shap_values_xgb, X_sample, feature_names=selected_kbest)


# Crear el DataFrame con los valores SHAP
shap_df = pd.DataFrame(shap_values_xgb, columns=selected_kbest)

# Calcular métricas extendidas de los valores SHAP
shap_metrics_extended = pd.DataFrame({
    "SHAP Promedio": shap_df.mean(),  # Media de SHAP (dirección del efecto)
    "Magnitud Promedio": np.abs(shap_df).mean(),  # Magnitud media de SHAP
    "Desviación Estándar": shap_df.std(),  # Variabilidad de SHAP
    "Porcentaje Positivo": (shap_df > 0).mean() * 100,  # % de veces que la variable tiene SHAP positivo
    "Porcentaje Negativo": (shap_df < 0).mean() * 100,  # % de veces que la variable tiene SHAP negativo
    "Cantidad de Positivos": (shap_df > 0).sum(),  # Cantidad de valores SHAP positivos
    "Cantidad de Negativos": (shap_df < 0).sum(),  # Cantidad de valores SHAP negativos
    "Efecto Global (Positivo/Negativo)": np.where(shap_df.mean() > 0, "Positivo", "Negativo")  # Si la media es positiva o negativa
}).sort_values(by="Magnitud Promedio", ascending=False)  # Ordenar por importancia


# Mostrar las métricas
display("Valores SHAP para XGBoost:")
display(shap_metrics_extended)
```

```
# ==================================================
# Preparación de Datos para RNN, LSTM y Análisis SHAP
# ==================================================
import numpy as np
from tensorflow.keras.utils import to_categorical
# ==========================
# 1️⃣ ESCALAR TODO EL DATAFRAME
# ==========================
# Aplica MinMaxScaler a todo X_selected (para redes neuronales)
scaler = MinMaxScaler()
X_scaled_nn = scaler.fit_transform(X_selected)  # Escalado para redes neuronales

# ============================
# 2️⃣ DIVIDIR EN ENTRENAMIENTO/TEST
# ============================
# Usa los datos escalados para la red neuronal
X_train, X_test, y_train, y_test = train_test_split(X_scaled_nn, y,
                                                    test_size=0.2,
                                                    stratify=y,
                                                    random_state=42)
# ==============================
# 3️⃣ TRANSFORMAR EN TENSORES 3D
# ==============================
# Para Redes Neuronales (LSTM, GRU, etc.)
X_train_3d = np.expand_dims(X_train, axis=1)  # Añade la dimensión de "timesteps"
X_test_3d = np.expand_dims(X_test, axis=1)
# ==============================
# 4️⃣ ETIQUETAS EN CATEGÓRICOS
# ==============================
y_train_cat = to_categorical(y_train)
y_test_cat = to_categorical(y_test)
# ============================
# 5️⃣ PARA SHAP (ANTES DE SMOTE)
# ============================
# Volver a dividir los datos originales antes de SMOTE
X_train_real, X_test_real, y_train_real, y_test_real = train_test_split(X_scaled_nn, y,
                                                                        test_size=0.2,
                                                                        stratify=y,
                                                                        random_state=42)

X_train_3d_real = np.expand_dims(X_train_real, axis=1)
X_test_3d_real = np.expand_dims(X_test_real, axis=1)
y_train_cat_real = to_categorical(y_train_real)
y_test_cat_real = to_categorical(y_test_real)

# ============================
# 6️⃣ SAMPLES Y BACKGROUND PARA SHAP
# ============================

# Sample para SHAP (1000 muestras)
X_sample, _, y_sample, _ = train_test_split(X_test_real, y_test_real,
                                            train_size=2000,
                                            stratify=y_test_real,
                                            random_state=42)

# Background SHAP para entrenamiento
background = X_train_3d_real[np.random.choice(X_train_3d_real.shape[0], 2000, replace=False)]

# Samples para SHAP
samples, _, y_samples, _ = train_test_split(X_test_3d_real, y_test_cat_real,
                                            test_size=0.3,
                                            stratify=y_test_cat_real,
                                            random_state=42)

# Recortar si es necesario
background = background[:2000]
samples = samples[:2000]

# ============================
# ✅ VERIFICACIONES
# ============================
print(f"Shape X_train_3d: {X_train_3d.shape}")
print(f"Shape y_train_cat: {y_train_cat.shape}")
print(f"Shape background: {background.shape}")
print(f"Shape samples: {samples.shape}")
```

```
# ==================================================
# Entrenamiento y Evaluación de Modelos RNN y LSTM
# ==================================================

from tensorflow.keras.layers import SimpleRNN
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from sklearn.metrics import f1_score
from tensorflow.keras.callbacks import EarlyStopping

# Definir el modelo RNN
model_rnn = Sequential([
    SimpleRNN(64, input_shape=(X_train_3d.shape[1], X_train_3d.shape[2]), activation='tanh'),
    Dense(32, activation='relu'),
    Dense(y_train_cat.shape[1], activation='softmax')
])

# Compilar el modelo
model_rnn.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Entrenar el modelo
model_rnn.fit(X_train_3d, y_train_cat, epochs=50, batch_size=32, validation_split=0.2, verbose=1)

# Evaluar el modelo
loss, accuracy = model_rnn.evaluate(X_test_3d, y_test_cat, verbose=0)
print(f"RNN Accuracy: {accuracy:.4f}")

# 📌 Obtener predicciones del modelo (probabilidades)
y_pred_proba = model_rnn.predict(X_test_3d)

# 📌 Convertir probabilidades en etiquetas de clase
y_pred = np.argmax(y_pred_proba, axis=1)
y_true = np.argmax(y_test_cat, axis=1)  # Convertir one-hot a etiquetas reales

# 📌 Calcular F1-Score weighted y macro
f1_weighted = f1_score(y_true, y_pred, average='weighted')
f1_macro = f1_score(y_true, y_pred, average='macro')

# 📌 Mostrar resultados
print(f"RNN Accuracy: {accuracy:.4f}")
print(f"RNN F1-Score (Weighted): {f1_weighted:.4f}")
print(f"RNN F1-Score (Macro): {f1_macro:.4f}")

model_lstm = Sequential([
    LSTM(128, input_shape=(X_train_3d.shape[1], X_train_3d.shape[2]), activation='tanh', return_sequences=True),
    Dropout(0.2),
    LSTM(64, activation='tanh', return_sequences=True),
    Dropout(0.2),
    LSTM(64, activation='tanh'),
    Dropout(0.2),
    Dense(64, activation='relu'),
    Dense(y_train_cat.shape[1], activation='softmax')
])

# Compilar el modelo
model_lstm.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Entrenar el modelo (usando solo 1 época para pruebas)
early_stop = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
model_lstm.fit(X_train_3d, y_train_cat, epochs=50, batch_size=64, validation_split=0.2, verbose=1, callbacks=[early_stop])

# Evaluar el modelo
loss, accuracy = model_lstm.evaluate(X_test_3d, y_test_cat, verbose=0)
print(f"LSTM Accuracy: {accuracy:.4f}")

from sklearn.metrics import f1_score

# Obtener predicciones del modelo (probabilidades)
y_pred_proba = model_lstm.predict(X_test_3d)

# Convertir probabilidades en etiquetas de clase
y_pred = np.argmax(y_pred_proba, axis=1)
y_true = np.argmax(y_test_cat, axis=1)  # Convertir one-hot a etiquetas reales

# Calcular F1-Score weighted y macro
f1_weighted = f1_score(y_true, y_pred, average='weighted')
f1_macro = f1_score(y_true, y_pred, average='macro')


# Mostrar resultados
print(f"LSTM Accuracy: {accuracy:.4f}")
print(f"LSTM F1-Score (Weighted): {f1_weighted:.4f}")
print(f"LSTM F1-Score (Macro): {f1_macro:.4f}")
```

```
# ==============================================
# Interpretación del Modelo LSTM con SHAP Values
# ==============================================

# **Crear el explicador SHAP**
explainer = shap.GradientExplainer(model_lstm, background)

# **Calcular SHAP values sin verificar la aditividad**
shap_values = explainer.shap_values(samples)
shap_values_lstm = np.squeeze(shap_values, axis=1)

# ========================================
# 3. Ajustar la forma de shap_values para evitar errores
# ========================================
# Seleccionar la clase correcta (por ejemplo, clase 0)
shap_values_selected = shap_values_lstm[..., 1]  # Seleccionar la clase deseada (shape: (100, 15))

# No es necesario agregar SHAP values si ya tienen la forma correcta
shap_values_lstm = shap_values_selected

# Agregar muestras a través de la dimensión de timesteps
samples_aggregated = np.mean(samples, axis=1)

# ========================================
# 4. Visualización con nombres correctos
# ========================================
shap.initjs()

# Summary plot global
print("\n📊 Importancia global de características en LSTM:")
shap.summary_plot(
    shap_values_lstm,
    samples_aggregated,
    feature_names=selected_features,  # Usar los nombres correctos
    plot_type="bar",
    max_display=15
)

# Convertir los SHAP values a un array de numpy si no lo es
shap_values_array = np.array(shap_values_lstm)  # (100, 15)

# Crear un DataFrame con métricas por característica
metrics_df = pd.DataFrame({
    "Característica": selected_features,
    "SHAP Promedio": np.mean(shap_values_array, axis=0),
    "Magnitud Promedio": np.mean(np.abs(shap_values_array), axis=0),
    "Desviación Estándar": np.std(shap_values_array, axis=0),
    "Porcentaje Positivo": np.mean(shap_values_array > 0, axis=0) * 100,
    "Porcentaje Negativo": np.mean(shap_values_array < 0, axis=0) * 100,
    "Cantidad de Positivos": np.sum(shap_values_array > 0, axis=0),
    "Cantidad de Negativos": np.sum(shap_values_array < 0, axis=0)
})

# Añadir la columna "Efecto Global"
metrics_df["Efecto Global"] = np.where(metrics_df["SHAP Promedio"] > 0, "Positivo", "Negativo")

# Ordenar por magnitud promedio de impacto
metrics_df = metrics_df.sort_values(by="Magnitud Promedio", ascending=False)

# Mostrar resultados en Jupyter Notebook o en terminal
print("\n🔍 Métricas SHAP - LSTM:")
display(metrics_df.style.format({
    "SHAP Promedio": "{:.4f}",
    "Magnitud Promedio": "{:.4f}",
    "Desviación Estándar": "{:.4f}",
    "Porcentaje Positivo": "{:.2f}%",
    "Porcentaje Negativo": "{:.2f}%",
    "Cantidad de Positivos": "{:.0f}",
    "Cantidad de Negativos": "{:.0f}"
}).background_gradient(subset=["Magnitud Promedio"], cmap='viridis'))
```