Escuela de Negocios
Tipo de documento: Tesis de maestría

Master in Management + Analytics

Implementar un modelo de predicción del máximo segmento a alcanzar por un usuario en aplicación de q-commerce

Autoría: Oliveri, Guido Luca
Año: 2025

Universidad Torcuato Di Tella

Master in Management+Analytics Tesis

Implementar un modelo de predicción del máximo segmento a alcanzar por un usuario en aplicación de q-commerce

---

por

Guido Luca Oliveri

Tutor: Damián Ilkow, MIM

Marzo, 2025

Implementar un modelo de predicción del máximo segmento a alcanzar por un usuario en aplicación de q-commerce

Master in Management+Analytics Tesis

Guido Luca Oliveri

Resumen

En este trabajo de investigación se desarrolló un modelo predictivo para identificar el máximo segmento de ciclo de vida que un usuario puede alcanzar dentro de una plataforma digital, utilizando datos históricos de transacciones y comportamientos. El proceso comenzó con la extracción del conjunto de datos, el cual contenía información clave sobre las primeras compras de los usuarios, verticales de preferencia, días de la semana de compra y otros factores relacionados.

Una vez obtenidos los datos, estos fueron procesados y se realizaron diversas etapas: limpieza de valores nulos y valores atípicos, ingeniería de características para optimizar el poder predictivo, y análisis exploratorio de datos para identificar patrones y relaciones significativas. Variables como la vertical de la primera compra y las características temporales mostraron una influencia destacada en la predicción del máximo segmento alcanzado.

Posteriormente, se implementaron técnicas de aprendizaje automático, incluyendo vecinos más cercanos, Random Forest y XGBoost, para predecir el máximo segmento de los usuarios. Se aplicaron métodos de ajuste de hiper parámetros y validación cruzada para maximizar la precisión del modelo.

Finalmente, se analizaron los resultados para identificar factores clave en el comportamiento de los usuarios y evaluar el desempeño de las predicciones realizadas.

Este trabajo incluye e integra un flujo completo de análisis de datos, desde la extracción y limpieza hasta la generación de predicciones, proporcionando un marco útil para entender el comportamiento de los usuarios y diseñar estrategias basadas en los segmentos de ciclo de vida identificados.

Implementar un modelo de predicción del máximo segmento a alcanzar por un usuario en aplicación de q-commerce

Master in Management+Analytics Tesis

Guido Luca Oliveri

Abstract

This thesis presents the development of a predictive model to determine the maximum lifecycle segment that a user can reach within a digital platform. Based on historical transaction and behavioral data, the study analyzes key factors such as purchase verticals, temporal patterns, and user behaviors.

Understanding the maximum segment reached by a user is essential for optimizing the use of incentives by the Marketing department, reducing costs associated with ineffective strategies that could negatively impact the company. Additionally, this knowledge enables the retention of top customers and maximizes the value that can be derived from them. This work also aims to lay the foundation for granting preferential value to certain customer segments in the future, thereby contributing to the company's profitability.

Data preparation was carried out, addressing tasks such as handling missing values, detecting outliers, and feature engineering to maximize the predictive power of the dataset. An exploratory analysis allowed the identification of significant relationships between features such as the vertical of the first purchase and temporal attributes with the user's lifecycle.

For model development, machine learning techniques were implemented, where XGBoost with Bayesian optimization for hyperparameter tuning achieved the best performance.

In conclusion, this thesis not only provides a technical approach to predictive analysis but also offers practical results to support strategic decision-making on the digital platform, where the main objective is to optimize Marketing costs based on user segmentation patterns.

Para Patricia e Iván, por su apoyo incondicional.

Índice

Índice de tablas

Índice de gráficos

Capítulo 1 Introducción

En los últimos años, con la proliferación de los smartphones, las aplicaciones de delivery han experimentado un crecimiento exponencial, convirtiéndose en una opción cada vez más popular para los usuarios. Su objetivo es lograr ser la mejor alternativa para las personas que utilizan las aplicaciones.

Entender cómo eficientizar los gastos de marketing para poder explotar el potencial de los usuarios es uno de los grandes desafíos que se presenta. De aquí es que surge la necesidad de predecir el máximo segmento del usuario, entendido como la categoría más alta dentro de su ciclo de vida en la plataforma, en función de su actividad y patrones de compra.

Para lograr que el gasto de marketing sea más eficiente, vamos a incentivar a las personas de una manera inteligente. Una manera de realizar esto es a través de la asignación de vouchers para que sean utilizados dentro de la aplicación. De esta forma, si nosotros entregamos los incentivos correctos, estaríamos cumpliendo con el objetivo de marketing mencionado.

En este contexto, resulta clave comprender la segmentación de usuarios dentro de la plataforma, ya que esto permite desarrollar estrategias personalizadas y mejorar la efectividad de las campañas de marketing. A continuación, se describen los diferentes segmentos de usuarios y el concepto de intermitencia, una variable fundamental en la clasificación:

- Prospect: Persona que nunca hizo una orden. Este segmento no es un valor posible en la predicción.
- New: Persona que tiene entre 1 y 4 órdenes confirmadas. Este segmento tampoco es un valor posible en la predicción.
- Onboarded: Persona que tiene 5 o más órdenes confirmadas. A partir de este segmento de usuario se empieza a mirar la intermitencia (ver la definición más abajo).
- Stable: Persona que tiene 5 o más órdenes confirmadas y su intermitencia es menor o igual a 30 días.
- Heavy: Persona que tiene 5 o más órdenes confirmadas y su intermitencia es menor o igual a 6 días.
- Super Heavy: Persona que tiene 5 o más órdenes confirmadas y su intermitencia es menor o igual a 3 días.

Por su parte, la intermitencia es la frecuencia con la que compra el usuario.

En la actualidad, una aplicación de delivery genera gran cantidad de información a partir de la actividad de los usuarios. Cada vez que una persona ingresa a la aplicación, clickea en componentes, ingresa su dirección de domicilio, número de tarjeta, teléfono, realiza una orden, se genera información valiosa que nos servirá como punto de partida para realizar este trabajo de investigación.

Nuestro objetivo es que con la predicción del máximo segmento a alcanzar por un usuario, ayudaremos al negocio a tomar la mejor decisión en términos de rentabilidad de la compañía y fidelización de clientes.

1.1) Contexto

1.1.1) Modelo de negocio basado en plataformas

En el principio del siglo XXI, con el auge de internet y su adopción masiva por parte de la sociedad, comenzaron a proliferar los sitios web, que marcaron el primer paso en la interacción digital de los usuarios. Este desarrollo sentó las bases para la aparición de un modelo de negocio innovador, basado en plataformas digitales. Andrei Hagiu (2015) conceptualizó estas plataformas como “Multi-Sided Platforms” (MSP), por su capacidad de conectar a dos o más grupos interdependientes de clientes con necesidades complementarias. Según Hagiu (2007: 6), el modelo de negocio de las plataformas se fundamenta en dos aspectos clave: i) la reducción de costos asociados a la búsqueda de productos y servicios, y ii) la disminución de los costos de transacción entre las partes involucradas.

Aunque este modelo no es nuevo, ha adquirido un papel predominante en las últimas décadas gracias al avance de tecnologías emergentes y el crecimiento de internet, especialmente después de la crisis de las empresas punto com, mencionado por Hagiu (2007: 3). Las plataformas han sabido capitalizar estos cambios tecnológicos para consolidarse como un modelo de negocio central en la economía digital contemporánea.

"Existe un creciente interés en la economía de las plataformas multi-lado (MSPs), que reúnen a dos o más grupos y permiten interacciones entre ellos (por ejemplo, Airbnb, eBay, Uber, XBox, etc.).” (Hagiu & Wright, 2015: 1) [...] “Se produce un efecto de red cruzado si el beneficio para los usuarios en al menos un grupo (lado A) depende de la cantidad de usuarios en el otro grupo (lado B) que se unan. Un efecto de red indirecto ocurre si hay efectos de red cruzados en ambas direcciones (de A a B y de B a A)." (Hagiu & Wright, 2015: 5).

Por ejemplo, si consideramos una plataforma de aprendizaje en línea, la presencia de una amplia variedad de instructores con cursos en múltiples áreas

temáticas atraerá a más estudiantes interesados en aprender, lo que a su vez fomentará que más instructores ofrezcan nuevos contenidos. En este sentido, las plataformas funcionan como intermediarias, conectando a usuarios con necesidades interrelacionadas.

Fuentes de Valor en Modelos de Negocio de Plataformas

Según Hagiu & Wright (2015: 5) los modelos de negocio basados en plataformas generan valor para sus usuarios a través de dos principales mecanismos: efectos de red directos e indirectos. Además, algunas plataformas pueden proporcionar funcionalidades independientes que aportan valor sin depender directamente de los efectos de red. A continuación se detalla cada mecanismo:

1. Efectos de Red Directos:
   Estos efectos se generan dentro de un mismo grupo de usuarios, a partir de la interacción directa entre ellos. Un ejemplo de esto sería una red social profesional, donde los usuarios se conectan para compartir oportunidades laborales, información o establecer contactos. En estos casos, la utilidad de la plataforma aumenta a medida que más usuarios del mismo grupo se suman y participan activamente.

2. Efectos de Red Indirectos:
   En este caso, el valor se genera por la interacción entre diferentes grupos de usuarios. Por ejemplo, en una plataforma de viajes, un aumento en la cantidad de turistas registrados incentiva a que más guías turísticos o proveedores de servicios se unan a la plataforma, lo que a su vez beneficia a los turistas al ofrecerles más opciones. Este intercambio entre grupos complementarios es el motor de crecimiento de muchas plataformas.

3. Funcionalidades Independientes:
   Algunas plataformas ofrecen funcionalidades útiles por sí mismas, independientemente de la cantidad de usuarios o grupos conectados. Un ejemplo sería un editor de imágenes en línea, que permite a los usuarios acceder a herramientas de diseño sin necesidad de interactuar con otros grupos.

Los efectos de red son un atributo distintivo de las plataformas, comparables a las economías de escala, pero del lado de la demanda. A medida que crecen los grupos de usuarios, los costos asociados a las transacciones y a la intermediación disminuyen, aumentando la eficiencia y sostenibilidad del modelo. En plataformas cuyo valor principal depende de los efectos de red indirectos, la interacción entre distintos grupos de usuarios con necesidades

complementarias potencia el crecimiento de la plataforma. A medida que más usuarios de un grupo se suman, aumenta el atractivo para que el otro grupo participe, generando un círculo virtuoso que acelera la adopción del servicio (Hagiu & Wright, 2015: 5).

En resumen, las plataformas digitales han transformado la dinámica de los negocios al aprovechar estos efectos de red y ofrecer soluciones eficientes a problemas tradicionales de intermediación y acceso. Su capacidad para conectar a múltiples partes y generar valor mutuamente ha sido clave en su expansión y relevancia en la economía actual.

1.1.2) Quick Commerce

El rubro del Quick Commerce viene creciendo día a día. Según Statista (2023)[1], el mercado de Quick Commerce en Argentina alcanzará un ingreso de 426,6 millones de dólares estadounidenses para 2025, con una tasa de crecimiento anual compuesta (CAGR) del 8,79% entre 2020 y 2025.

Es importante entender que hay tres actores clave en el Quick Commerce, ya que cada uno cumple un rol esencial en su funcionamiento. Los clientes generan la demanda y determinan la oferta de productos según sus preferencias y hábitos de compra. Los comercios proveen los productos y su disponibilidad y tiempos de preparación impactan directamente en la experiencia del usuario. Finalmente, los repartidores son el eslabón logístico que garantiza la rapidez en la entrega, un factor diferencial en este modelo de negocio. La interacción eficiente entre estos tres actores es fundamental para la competitividad y sostenibilidad del Quick Commerce.

Según NielsenIQ (2024, septiembre 16) en su artículo “The rise of quick commerce”[2], la pandemia creó un ambiente inusual que favoreció el crecimiento explosivo del quick commerce. Compañías como Uber Eats y Deliveroo pivotaron hacia la entrega de productos de conveniencia durante los cierres, mientras que startups como Gopuff construyeron un modelo desde cero basado en microcentros de cumplimiento urbano.

Actualmente las aplicaciones que se dedican al Quick Commerce en Argentina se dividen entre PedidosYa y Rappi, y buscan generar la mejor experiencia al usuario para que este quiera volver en un futuro y sea un cliente leal dentro de la aplicación.

[1] https://www.statista.com/outlook/emo/online-food-delivery/grocery-delivery/quick-commerce/argentina
[2] https://www.nielseniq.com

El cliente es un actor que a partir de sus actitudes y comportamientos, hace que el foco del Quick Commerce vaya cambiando. En base a su cantidad de órdenes, preferencias, búsquedas, clicks y demás métricas que para la gente de la aplicación son fundamentales, hay que responder con la mejor propuesta posible.

Un desafío que tienen las aplicaciones de este rubro es entender cuál es el mejor producto para ofrecerle a cierto usuario con determinadas características. Con este trabajo de investigación, se busca, entre otros objetivos, poder resolver esta cuestión prediciendo el máximo segmento alcanzado por el usuario.

1.1.3) Caso de estudio similar

Existen diversos estudios que han abordado problemáticas similares a la desarrollada en esta investigación, aplicando técnicas de aprendizaje automático para el análisis del comportamiento de compra en plataformas digitales. Entre estos trabajos, destaca el estudio realizado por Roychowdhury, S., Alareqi, E., & Li, W. (2021), titulado *OPAM: Online Purchasing-behavior Analysis using Machine Learning*. En este estudio, los autores implementaron técnicas de aprendizaje supervisado, no supervisado y semi-supervisado para analizar el comportamiento de compra de los usuarios en plataformas de comercio electrónico, enfocándose en la predicción de eventos de compra y en la segmentación de clientes en distintos clústeres basados en sus patrones de compra.

Para el desarrollo del modelo predictivo, utilizaron algoritmos como XGBoost, árboles de decisión, vecinos más cercanos y memoria a corto y largo plazo, aplicando métodos de selección de características y optimización automatizada de modelos mediante optimización basada en algoritmos evolutivos. Según los resultados obtenidos, el algoritmo XGBoost fue el más efectivo en la clasificación de eventos de compra en la categoría de cosméticos, mientras que el clasificador de árbol de decisión obtuvo los mejores resultados en la categoría de electrónicos.

En términos de desempeño, los autores reportan que: “Para el conjunto de datos de cosméticos, el clasificador XGBoost es identificado como el mejor modelo con los siguientes parámetros: (tasa de aprendizaje = 0.001, profundidad máxima = 9, peso mínimo de hijo = 7, número de estimadores = 100, número de trabajos = 1). Para el conjunto de datos de electrónicos, el mejor modelo es el clasificador de árbol de decisión con (profundidad máxima = 10, mínimo de muestras en hoja = 3, mínimo de muestras para división = 2).” (Roychowdhury, S., Alareqi, E., & Li, W., 2021: 5).

Este estudio refuerza la importancia del uso de modelos avanzados de aprendizaje automático para el análisis del comportamiento de compra, destacando la relevancia de la segmentación de usuarios basada en datos transaccionales y su potencial para la personalización de estrategias de marketing digital.

1.2) Problema

En el competitivo rubro del quick commerce, las aplicaciones enfrentan el reto constante de personalizar la experiencia del usuario para maximizar tanto la satisfacción del cliente como la rentabilidad empresarial. Un desafío crítico radica en identificar qué producto o promoción ofrecer a un usuario en función de su perfil y comportamiento histórico. Esto no solo implica responder a las preferencias actuales del cliente, sino también anticipar su evolución dentro del ciclo de vida en la aplicación.

Entender el máximo segmento de ciclo de vida alcanzable por cada usuario representa una pieza clave en esta estrategia (para saber sobre los distintos segmentos de usuarios, ver *Capítulo 1 Introducción*). Saber con antelación el segmento máximo al que un cliente podría llegar permite a la compañía ajustar su enfoque de marketing, optimizar la asignación de recursos y diseñar promociones más efectivas. Por ejemplo, los usuarios en segmentos de alto potencial podrían beneficiarse de estímulos específicos para incrementar su actividad.

Actualmente, la falta de un modelo predictivo capaz de determinar este máximo segmento limita la capacidad de la empresa para tomar decisiones informadas y proactivas. En este contexto, este trabajo de investigación aborda esta problemática desarrollando un modelo que no solo predice el máximo segmento alcanzado por un usuario, sino que también incorpora patrones conductuales y temporales identificados en los datos históricos. Este enfoque busca eficientizar los costos de marketing, asignando incentivos a cierto segmento de usuarios, evitar otorgar beneficios a clientes no rentables, entre otras acciones posibles.

1.3) Objetivo

El propósito central de esta investigación fue abordar un desafío crucial en las aplicaciones de delivery: eficientizar el gasto del área de Marketing.

Para alcanzar este objetivo, se implementó un modelo predictivo capaz de estimar el segmento máximo que un usuario puede alcanzar dentro de la

aplicación de Quick Commerce, facilitando así la toma de decisiones más estratégicas y eficientes (llámese máximo segmento desde ahora a la variable dependiente del asunto).

Se realizará el entrenamiento del modelo de aquellos usuarios que se hayan registrado y hayan tenido órdenes desde enero hasta abril 2024, y sólo de aquellos que hayan alcanzado la cuarta orden confirmada. Según la definición de segmentación de clientes de la compañía, hasta la cuarta orden se considera a un usuario como nuevo (ver explicación de segmentos de usuarios en *Capítulo 1 Introducción*). A partir de la quinta orden, el usuario es considerado como enrolado. Entonces, y a partir de que el usuario alcanzó su cuarta orden, se hará la predicción del máximo segmento que tendrá el usuario dentro de los próximos 5 meses.

El lapso de tiempo para realizar la predicción del máximo segmento alcanzado (cinco meses a partir de que el usuario realiza su cuarta orden) fue determinado por decisiones de negocio (que no profundizaremos ya que no está dentro del alcance de este trabajo).

Por otro lado, se toman cuatro meses máximo de tiempo para realizar el entrenamiento con el objetivo de tener un historial de cada usuario robusto y además tener diferentes casuísticas que van a ayudar a realizar una predicción más exacta.

Además de esto, disponemos la información por día de los diferentes segmentos de usuario desde enero 2024 en adelante. Este punto representa una limitación para tener un histórico más robusto.

Entonces la pregunta que se busca responder es: ¿Cuál es el máximo segmento que alcanzará un usuario en los próximos cinco meses luego de hacer la cuarta orden confirmada?

1.4) Enfoque

El enfoque para llevar a cabo este proceso analítico se estructura en tres fases fundamentales: análisis descriptivo, predictivo y prescriptivo.

En el libro *Data Mining for Business Analytics* de Shmueli, Patel y Bruce (2017: 28), se destacan estas etapas clave del análisis descriptivo, predictivo y prescriptivo dentro de un marco analítico aplicado a problemas de negocio. Estas fases forman un flujo lógico para extraer valor de los datos y convertirlo en decisiones estratégicas. Los autores explican que el análisis descriptivo aborda el "qué sucedió", utilizando herramientas de visualización y reportes interactivos; el

análisis predictivo emplea modelos estadísticos para anticipar "qué podría suceder"; y el análisis prescriptivo sugiere "qué debería hacerse", integrando predicciones con decisiones óptimas basadas en datos.

En este trabajo, en el análisis descriptivo se busca identificar relaciones entre variables y evaluar su relevancia respecto a la variable dependiente (el máximo segmento de usuario a predecir).

El análisis predictivo toma como insumo los hallazgos del análisis descriptivo para calcular la probabilidad de que un usuario alcance el segmento máximo, proporcionando estimaciones precisas basadas en características históricas.

Finalmente el análisis prescriptivo propone alternativas orientadas a maximizar la rentabilidad por usuario, considerando las probabilidades de segmentación derivadas del modelo predictivo.

Capítulo 2 Datos, Métodos y Procedimientos

Este capítulo se enfoca en describir y analizar el conjunto de datos empleado en este estudio, junto con los métodos y procedimientos implementados para abordar las distintas etapas del procesamiento y análisis. El conjunto de datos contiene información detallada sobre el comportamiento de los usuarios en una aplicación de delivery especializada en quick commerce. Cada fila representa una observación asociada a un usuario o transacción, con características que incluyen historial de compras, frecuencia de pedidos, gasto promedio y preferencias de categorías de productos, entre otros. Todas las variables se encuentran en el *Anexo I - Lista de variables del conjunto de datos*, con sus descripciones específicas.

El análisis de calidad de los datos constituye un paso esencial en cualquier estudio de este tipo. Según Samuel Rodríguez (2023) en el artículo de *Big Data Magazine*[3], destaca que la calidad de los datos es esencial para decisiones precisas y efectivas, y los problemas comunes como valores duplicados, inconsistencias y datos desactualizados deben abordarse con estrategias robustas. En este sentido, enfatiza la importancia del data cleansing como la fase inicial crítica en cualquier estrategia de datos, incluyendo normalización, estandarización y estrategias para la eliminación de duplicados y valores atípicos. Para mitigar estos problemas, en este estudio se implementaron técnicas de limpieza y preprocesamiento, incluyendo imputación de valores faltantes, eliminación de duplicados y manejo de valores extremos mediante métodos robustos.

Además, se llevó a cabo un análisis exploratorio de los datos para comprender relaciones clave entre variables y detectar patrones iniciales útiles para la modelización.

En este trabajo, se incluyeron visualizaciones descriptivas, métricas de resumen y pruebas iniciales de hipótesis, con un enfoque particular en variables clave relacionadas con la predicción del segmento de usuario.

En cuanto a los procedimientos metodológicos, se detalla el flujo de trabajo empleado para preparar los datos antes de su uso en modelos de aprendizaje automático. Esto incluyó normalización y escalado de variables, segmentación en conjuntos de entrenamiento, validación y prueba, y selección de características relevantes.

[3] https://bigdatamagazine.es

2.1) Análisis y exploración de datos

2.1.1) Análisis preliminar de los datos

Con el objetivo de entender qué variables pueden llegar a ser las mejores para realizar la predicción del máximo segmento alcanzado por el usuario, se realizó un estudio preliminar de los datos con gráficos descriptivos que aportan ciertas ideas claves.

Podemos ver cómo se distribuyen las órdenes confirmadas por vertical (categoría de productos o servicios a la que pertenece la orden según la clasificación interna de la compañía, como restaurantes, cafés, supermercados, entre otros) y qué porcentaje representa cada una de la siguiente manera (realizado con enero 2024 de histórico):

Vertical Compra	% de Cantidad Órdenes Confirmadas por Vertical	Cantidad Órdenes Confirmadas
Restaurant	87%	650.560
Market	6%	45.403
Coffee	3%	22.252
Drinks	1%	8.703
Kiosks	1%	7.049
Pharmacy	1%	4.995
Courier	1%	4.705
Shop	0%	1.283
Courier Business	0%	850
Pets	0%	404
Total general	100%	746.204

Tabla 1 Representatividad órdenes

Vemos que restaurantes y mercados se llevan aproximadamente el 93% de las órdenes confirmadas, confirmando que es útil tener una métrica por separado

que sume estas 2 verticales diferenciándose del resto. La inclusión de mercados junto con restaurantes responde a la importancia estratégica de la vertical de Quick Commerce en el negocio. Ambas categorías, en ciertas ocasiones, ofrecen productos similares y se complementan en la experiencia del usuario. Desde una perspectiva operativa y comercial, es relevante analizarlas en conjunto para entender mejor las dinámicas de consumo y optimizar las estrategias de incentivos y promoción.

Haciendo un análisis en profundidad por país para el mismo período, tenemos la siguiente distribución de órdenes confirmadas, tanto en valor absoluto como el porcentaje que representa del total cada país:

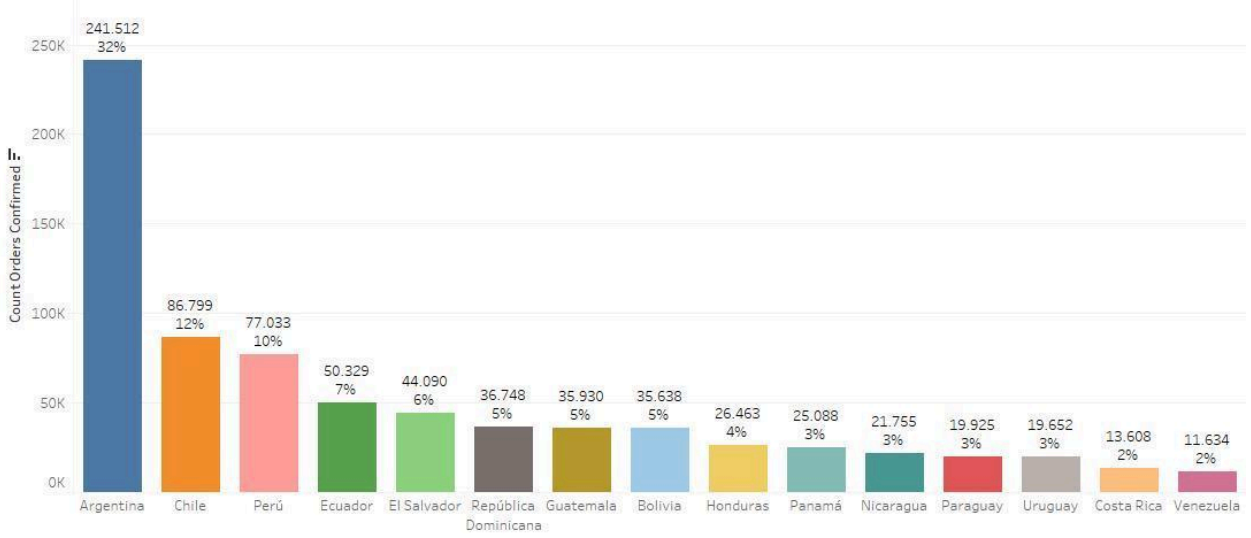

Gráfico 1 Distribución órdenes confirmadas por país

Entendiendo rápidamente que entre Argentina, Chile y Perú ya tenemos más del 50% de órdenes confirmadas.

Otro dato interesante a grandes rasgos es ver por país en dónde se gasta más dinero (estandarizando a euros). La hipótesis es que aquellos países con más órdenes confirmadas serían aquellos en donde se gasta más en promedio:

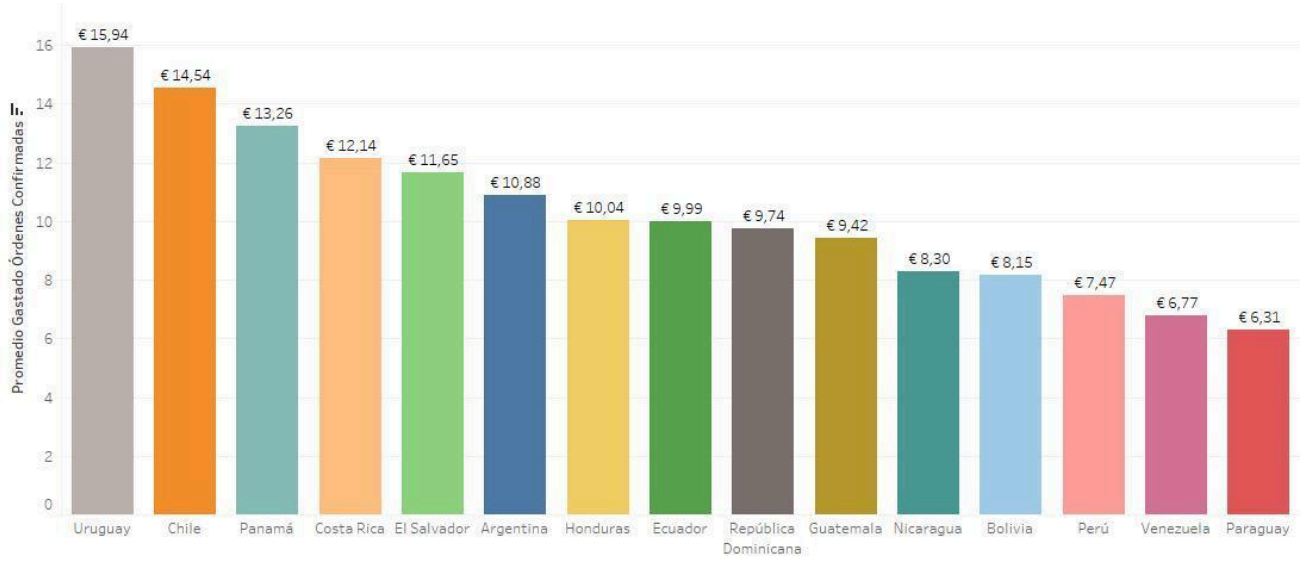

Gráfico 2 Ticket promedio por país (en euros)

Un dato a destacar es que Uruguay siendo uno de los 3 países con menos órdenes confirmadas pasa a ser el país en donde más se gasta dinero en promedio. Con lo cual nuestra hipótesis anterior queda descartada.

2.1.2) Calidad de los datos

- Duplicados: No hay duplicados en los datos, los datos se encuentran agrupados por las dimensiones correspondientes (que están descritas en el *Anexo I - Lista de variables del conjunto de datos*).
- Valores faltantes (NaN):
    - Dado que previamente se realizó una limpieza en la consulta a la base de datos, eliminando las filas correspondientes a usuarios sin compras, no se registran valores nulos en el conjunto de datos.

2.1.3) Valores atípicos y métodos aplicados

- Detección de valores atípicos:
    - Se realizó la limpieza de valores atípicos en aquellas columnas que son numéricas.
    - Para realizar lo mencionado en el primer punto, se utilizó el método de detección y eliminación de valores atípicos basado en el rango intercuartil (IQR). El IQR es la diferencia entre el tercer cuartil (Q3, percentil 75) y el primer cuartil (Q1, percentil 25). Este rango representa la dispersión de la "parte media" de los datos.

      El factor determina cuánta distancia por encima o por debajo del IQR se considerará como límite para identificar valores atípicos. Los valores más allá de X veces la longitud del IQR (hacia abajo o hacia

arriba) serán considerados valores atípicos (donde X es el valor del factor elegido).

Se probaron distintos valores y en base a los resultados obtenidos, elegimos el factor igual a 3. En este caso, los valores que estén más de 3 veces el IQR por debajo de Q1 o por encima de Q3 serán considerados valores atípicos.

Aplicando este método, se realizó una reducción del 11% del conjunto de datos (pasando de 142.530 a 126.866 valores).

Obtenemos los siguientes resultados:

Máximo segmento alcanzado	Cantidad absoluta
stable	47.397
super heavy	46.861
heavy	32.320
onboarded	288

Tabla 2 Distribución en valor absoluto de máximo segmento alcanzado

Entonces, la representatividad en porcentaje por máximo segmento alcanzado es la siguiente:

Máximo segmento alcanzado	Representatividad
stable	37%
super heavy	37%
heavy	25%
onboarded	1%

Tabla 3 Representatividad en porcentaje de máximo segmento alcanzado

2.1.4) Análisis Exploratorio de Datos

El objetivo de esta sección es poder entender por un lado a nivel descripción cómo se encuentran los datos que se van a utilizar para realizar la predicción del máximo segmento alcanzado por un usuario, y por otro la relación que existe entre las distintas características y la variable dependiente. Esto último

mencionado permitirá saber cuáles son las variables que aportan mayor valor a la hora de realizar la predicción.

Primero se decide analizar la diferencia en días entre la cuarta compra de un usuario y el momento en que alcanzó su máximo segmento dentro de la plataforma (más detalle en *Anexo I - Lista de variables del conjunto de datos*), particionado por segmento de usuario. Para eso se decidió utilizar un histograma por cada ciclo de usuario:

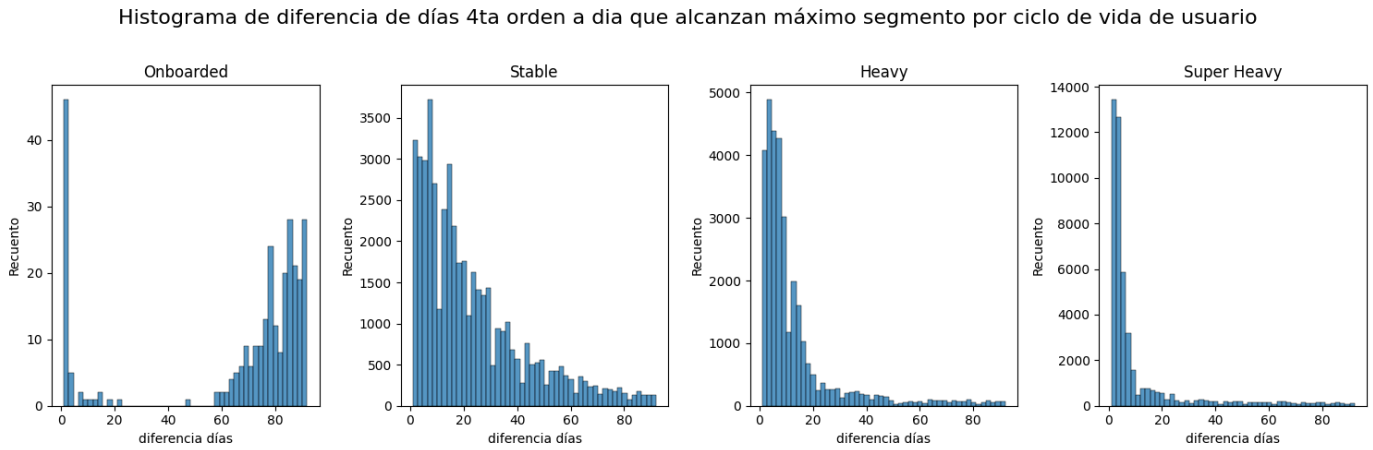

Gráfico 3 Histograma diferencia días 4ta orden a dia que alcanzan máximo segmento

Al observar los histogramas separados por el ciclo de vida del usuario, se pueden extraer los siguientes puntos relevantes:

1. Distribución en la clase Onboarded:
    - Predomina un intervalo alto de días (entre 60 y 80 días) para alcanzar el segmento máximo después de la cuarta orden. Esto podría indicar un comportamiento de usuarios que avanzan lentamente hacia una mayor interacción en la aplicación.
    - Existe un pico en los primeros intervalos de tiempo, esto se debe a que hay usuarios que hacen su cuarta orden, luego pasan a la quinta orden rápidamente (entre uno y cuatro días después de la cuarta orden) y su intermitencia no mejora, por ende su máximo segmento es onboarded.
2. Distribución en la clase Stable:
    - La distribución es más homogénea, con un pico en los primeros 10 días. Esto refleja que los usuarios Stable tienden a alcanzar rápidamente su máximo segmento tras su cuarta orden, con una disminución progresiva a medida que aumenta el intervalo de días.
    - Este patrón podría indicar un comportamiento consistente pero menos intenso comparado con usuarios más activos.

3. Distribución en la clase Heavy:
    - Se observa una distribución muy similar a la de Stable, aunque con una mayor concentración en intervalos más cortos (entre 0 y 10 días).
    - Los usuarios Heavy parecen alcanzar su máximo segmento más rápidamente que los Stable, lo que indica una adopción más activa en la plataforma.
4. Distribución en la clase Super Heavy:
    - La mayor parte de los usuarios alcanzan su máximo segmento inmediatamente después de su cuarta orden (entre 0 y 5 días), lo que sugiere que son los usuarios más activos y comprometidos de la plataforma.
    - Existe una disminución abrupta después del primer intervalo, lo que reafirma la alta frecuencia de interacción y su rápido avance en el ciclo de vida.

Impacto para el negocio:

- Los usuarios Onboarded, ya que tienen los intervalos más largos, es posible que requieran incentivos o mejoras en la experiencia para acelerar su adopción. Esto los ayudaría a alcanzar su máximo segmento en menos tiempo.
- Los usuarios Superheavy confirman ser los más valiosos en términos de compromiso y frecuencia de uso, lo que puede justificar campañas de fidelización específicas para mantenerlos en ese segmento.

La diferencia de días entre el día del máximo segmento alcanzado y el día que realizó la 4ta orden un usuario no se utiliza para entrenar el modelo, ya que sino se podría provocar *data leakage*.

Luego se analiza los valores que oscilan para la diferencia de días entre la 1ra y 4ta orden de un usuario particionado por cada ciclo de vida máximo alcanzado:

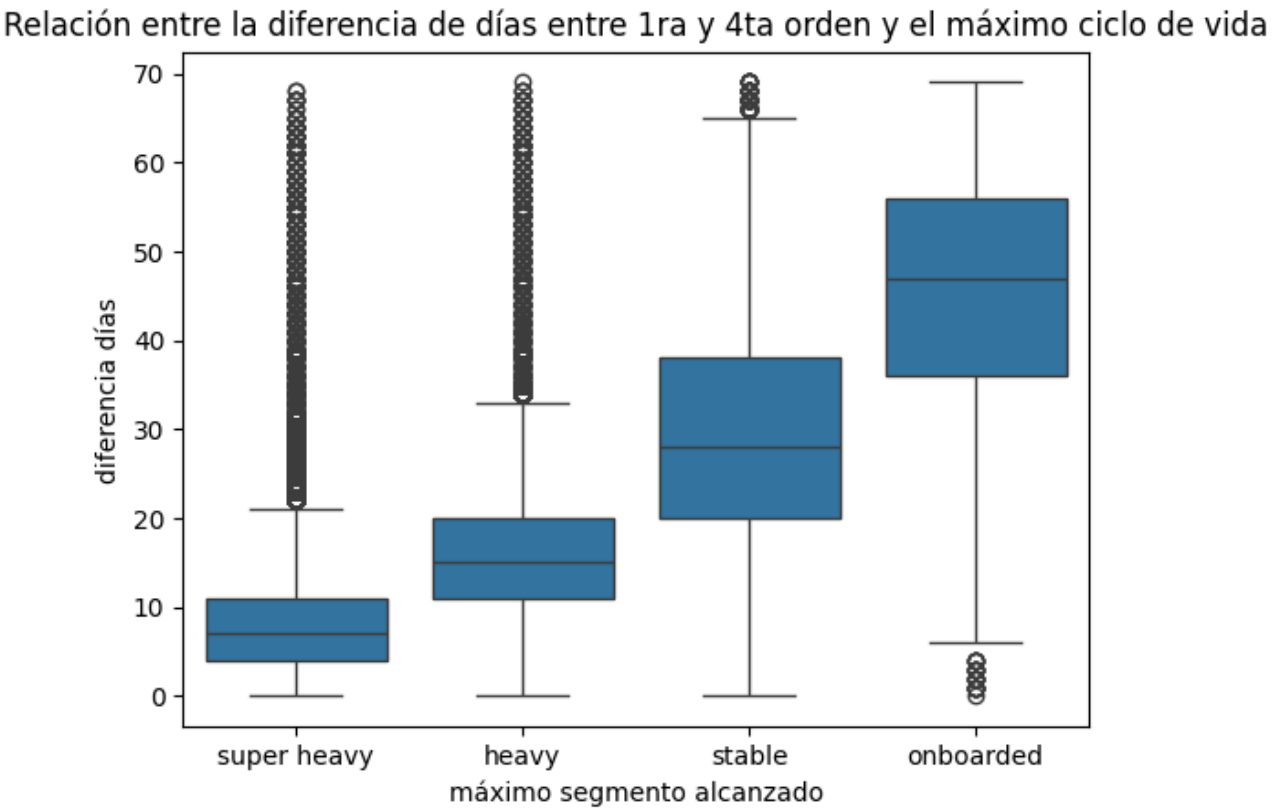

Gráfico 4 Diagrama de cajas diferencia días entre 1ra y 4ta orden

Distribución por máximo segmento alcanzado:

- Superheavy: Los usuarios de este segmento tienen los valores de diferencia de días entre 1ra y 4ta orden más concentrados y más bajos, con una mediana pequeña (~10 días). Esto sugiere que estos usuarios son altamente activos y realizan sus primeras cuatro órdenes en un periodo muy corto.
- Heavy: Presenta una mediana más alta (~15 días) y mayor dispersión en comparación con los Super Heavy. Esto refleja una menor intensidad en la frecuencia de pedidos.
- Stable: Este segmento muestra una mediana aún mayor (~25 días) y una variabilidad más amplia. Los usuarios de este grupo parecen tardar más tiempo en completar las primeras cuatro órdenes, indicando un comportamiento moderado en la interacción con la aplicación.
- Onboarded: Los usuarios de este segmento tienen la mayor dispersión y mediana más alta (entre 40 y 50 días). Esto indica que estos usuarios tienen una interacción mucho menos frecuente con la aplicación, tardando significativamente más en realizar sus primeras cuatro órdenes.

Relación con el segmento:

- Existe una clara relación entre la diferencia de días entre 1ra y 4ta orden y el segmento máximo alcanzado: segmentos más altos (Super Heavy y Heavy) tienen tiempos más cortos entre la primera y cuarta orden, mientras que segmentos más bajos (Onboarded y Stable) muestran tiempos más largos.
- Esto indica que la variable mencionada en el punto anterior tiene un valor predictivo relevante para clasificar a los usuarios en función de su ciclo de vida.

Posteriormente, se analiza la relación entre la vertical de la 1er compra y la cantidad de usuarios por máximo ciclo de vida alcanzado, con el objetivo de encontrar algún hallazgo en los datos:

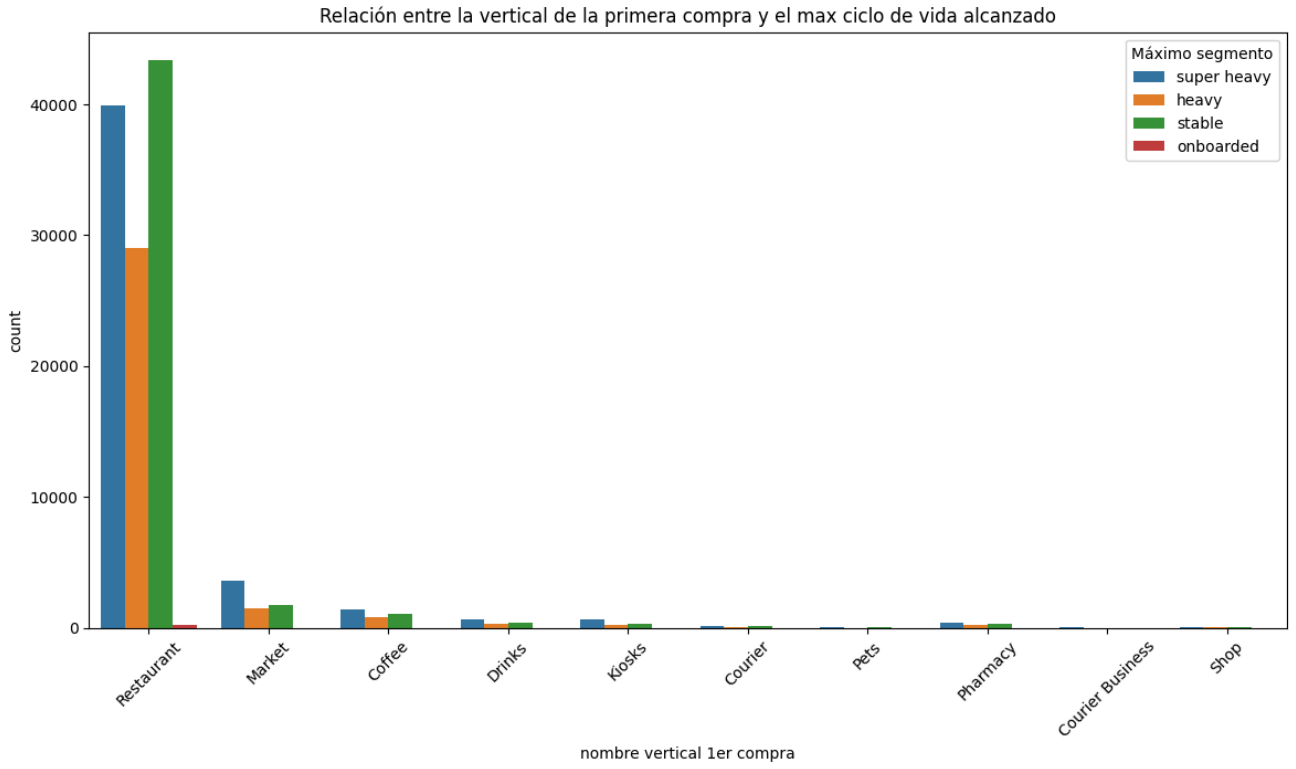

Gráfico 5 Representación en columnas de la 1er compra por vertical

Dominancia de la categoría "Restaurant":

- Dicha vertical domina en todas las clases del ciclo de vida, con la mayor cantidad de usuarios en cada segmento.
- Esto indica que la mayoría de los usuarios realizan su primera compra en esta vertical, probablemente debido a su popularidad o frecuencia en las interacciones iniciales.

Diferencias entre segmentos:

- En los segmentos Super Heavy, Heavy y Stable, "Restaurant" representa la mayor proporción de usuarios, lo que sugiere que esta vertical podría estar asociada con los usuarios más activos en la plataforma.

Dado que la vertical de restaurantes presenta una diferencia en volumen con respecto a las demás categorías, se ha optado por analizar de manera independiente el comportamiento de las otras verticales:

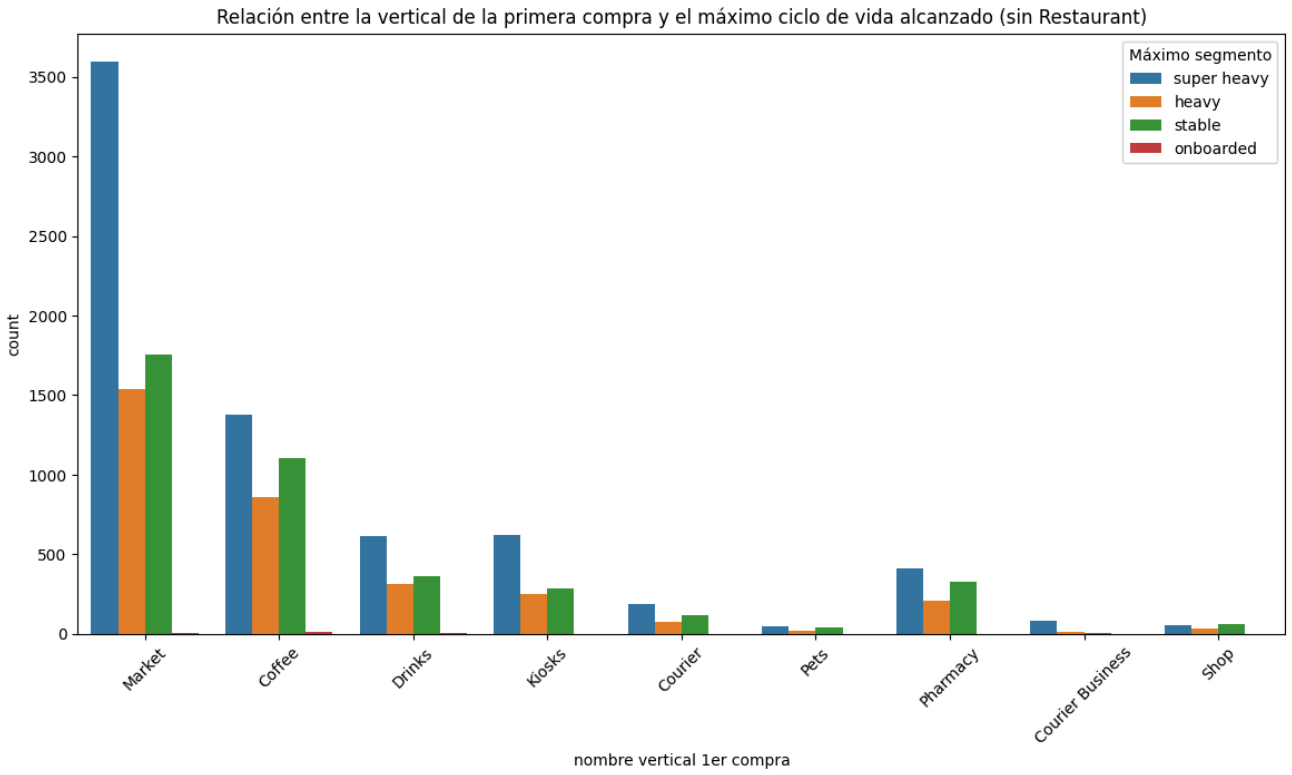

Gráfico 6 Representación en columnas de la 1er compra por vertical (sin Restaurantes)

Puntos clave:

- Market es la vertical con mayor cantidad de usuarios que alcanzan el segmento super heavy, seguida por Coffee.
- En la mayoría de las verticales, la mayor proporción de usuarios pertenece a los segmentos super heavy y stable, lo que sugiere una posible retención en ciertas categorías.
- La categoría Pharmacy muestra una distribución más equilibrada entre super heavy, heavy y stable, lo que podría sugerir distintos patrones de compra según la necesidad de los productos adquiridos.

El nombre de la vertical de la 1er compra podría aportar valor de manera marginal en la predicción del ciclo de vida máximo, pero su capacidad predictiva por sí sola parece ser limitada debido a la falta de diversidad en las verticales utilizadas para la primera compra.

En base a los resultados obtenidos del último análisis, se analiza la relación entre el máximo segmento de usuario alcanzado y si la primer compra fue en restaurantes o mercados (por ser las verticales con mayor volúmen de usuarios):

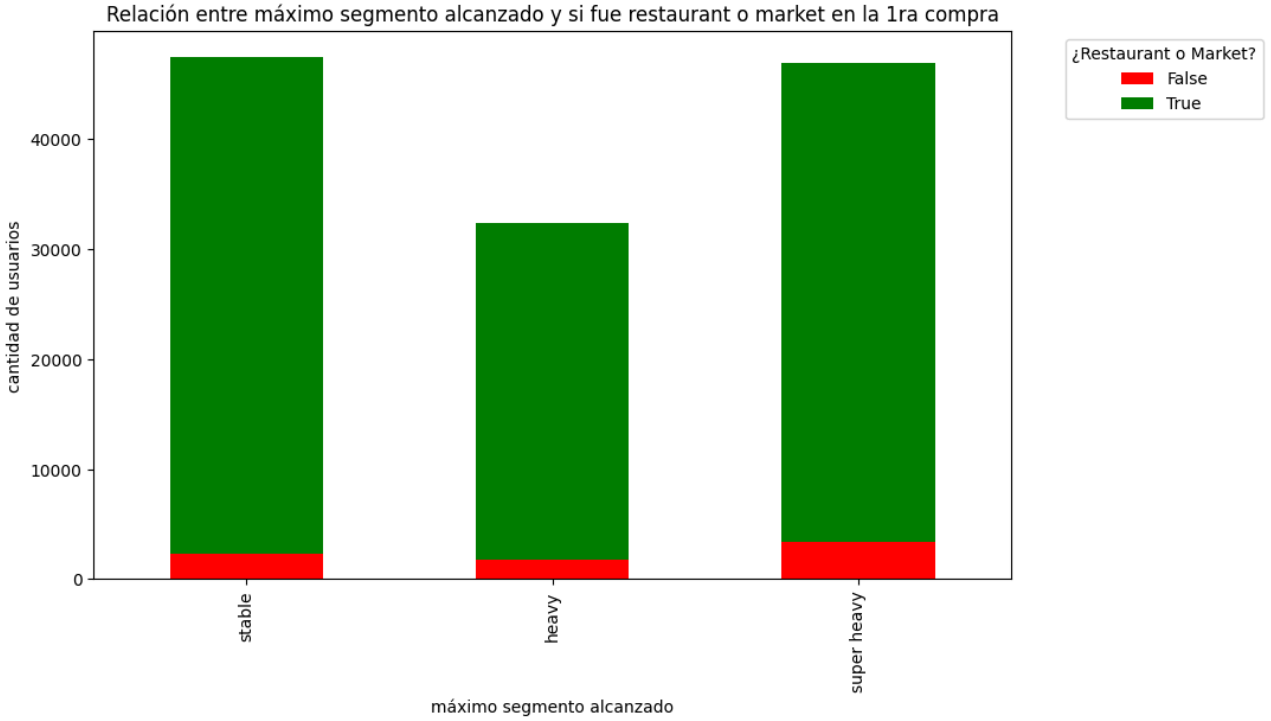

Gráfico 7 Representación en columnas de 1er compra si es restaurantes/mercados ó no

Conclusiones obtenidas:

- Se tomó la decisión de no graficar el caso de Onboarded debido a que hay muy pocas observaciones luego de quitar valores atípicos.
- Para las categorías de ciclo de vida stable, heavy y super heavy la proporción de usuarios que hicieron su primera compra en un restaurante o mercado es mayor en comparación con aquellos que no hicieron. Esto sugiere que el hecho de que un usuario haya hecho su primera compra en un restaurante o mercado parece ser relevante para predecir el ciclo de vida máximo que alcanzarán.

Posteriormente, se analiza la representatividad de usuarios por máximo ciclo de vida alcanzado segmentado según el país de registro en la aplicación:

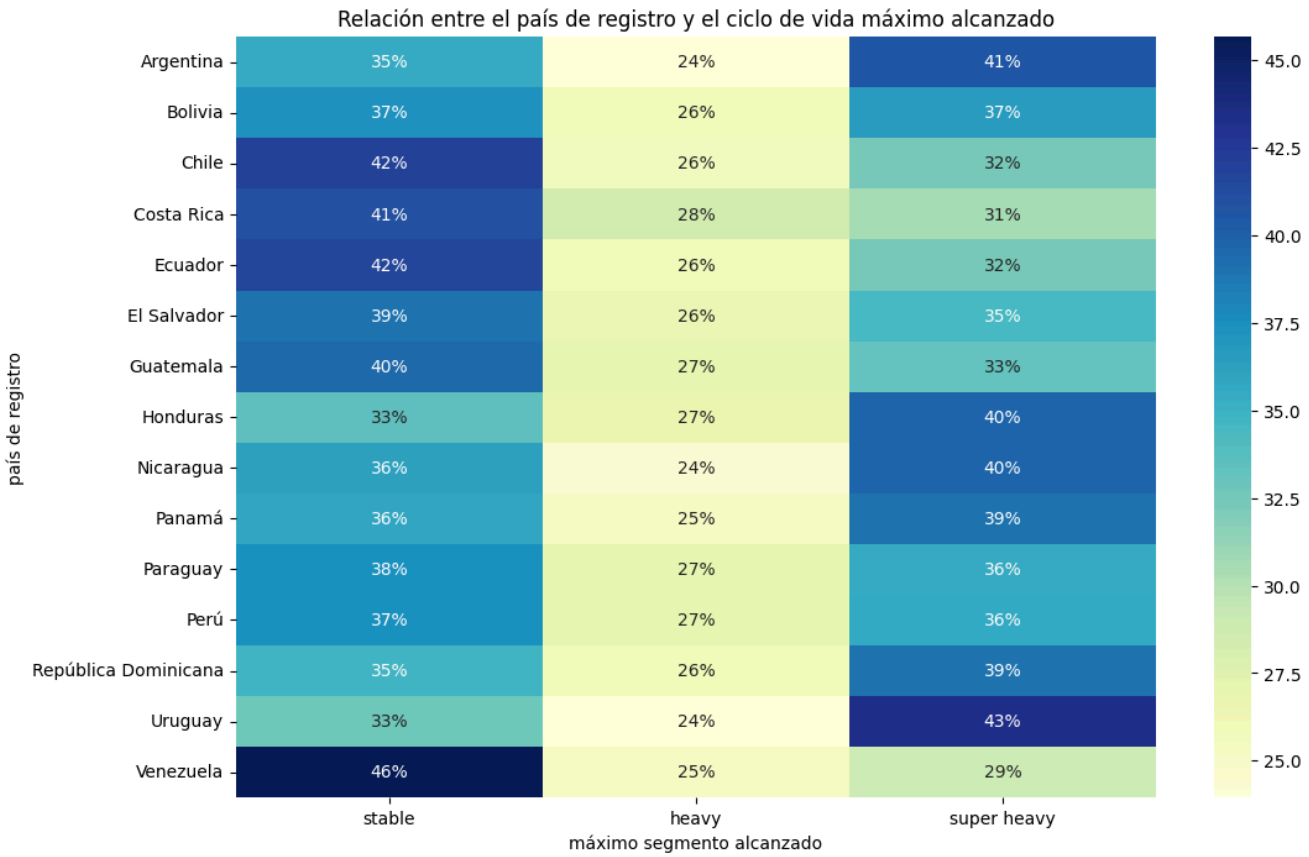

Gráfico 8 Mapa de calor representatividad de máximo segmento alcanzado por país de registro

- No se grafica el segmento Onboarded ya que su cantidad no es representativa por país.
- Dado que la distribución de los usuarios varía por país, esta variable podría contener información sobre el comportamiento de compra o factores regionales que influyen en la capacidad de los usuarios para alcanzar segmentos más altos. Por ejemplo, las políticas locales, la infraestructura de delivery o la popularidad de la plataforma pueden variar por país.
- Las diferencias en los segmentos alcanzados por los usuarios de cada país podrían ser útiles para capturar patrones específicos a nivel regional, lo que podría ayudar a mejorar la precisión del modelo. Países con pocos usuarios en segmentos altos o bajos podrían dar información sobre hábitos de compra o uso de la plataforma.
- El país de registro de un usuario podría ser relevante para el modelo, ya que introduce diferencias geográficas que afectan el comportamiento de los usuarios y su capacidad para alcanzar ciertos segmentos.

A continuación, se analiza la relación entre el día de la semana en que se realizó la 1ra compra y el ciclo de vida máximo alcanzado por los usuarios, con el objetivo de identificar posibles patrones en su comportamiento:

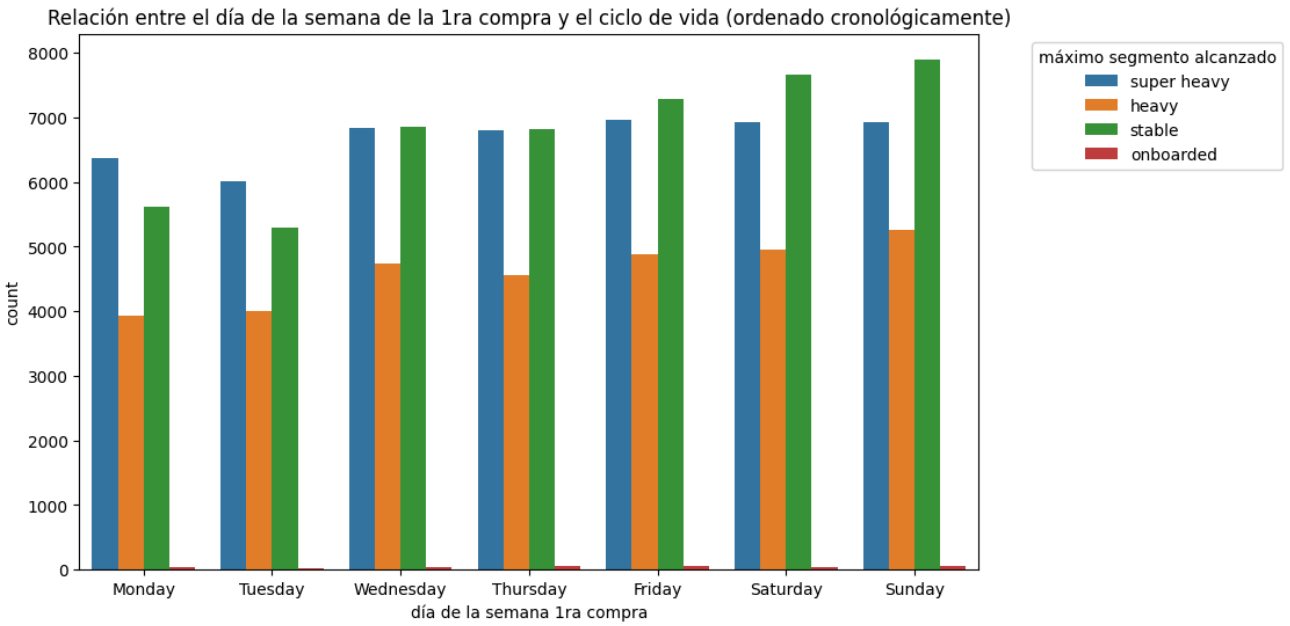

Gráfico 9 Gráfico de columnas por día de 1er compra por máximo segmento alcanzado

- Distribución consistente entre días:
  - La cantidad de usuarios que realizan su primera compra está distribuida de manera relativamente uniforme a lo largo de la semana. Sin embargo:
    - Los días sábado y domingo parecen ser ligeramente más populares para realizar la primera compra, lo cual podría estar relacionado con comportamientos de descanso o hábitos de consumo durante los fines de semana.
    - Los días lunes y martes tienen un menor volumen de primeras compras en comparación con los demás días.
- Distribución por segmentos:
  - En todos los días de la semana menos lunes y martes, el segmento stable es el más representado, seguido de super heavy y heavy.
  - El segmento onboarded tiene una representación consistentemente baja en todos los días, lo que indica que el día de la primera compra no parece estar directamente relacionado con este segmento específico.
- Variaciones menores entre segmentos:
  - Aunque la proporción de segmentos varía ligeramente entre días, no hay un patrón claro que indique que ciertos días favorezcan un ciclo de vida máximo particular. Por ejemplo:
    - Los fines de semana tienen una distribución similar a los días laborales en cuanto a la proporción de usuarios en los segmentos super heavy y heavy.

Respecto al valor predictivo de esta variable, vemos los siguientes puntos a destacar:

- Aporte al modelo:
  - Dado que la distribución de los segmentos es relativamente uniforme entre los días de la semana, esta variable por sí sola podría no aportar un gran valor predictivo al modelo.
  - Sin embargo, podría ser útil en combinación con otras variables (como la vertical de la primera compra, el ticket promedio, entre otras) para capturar patrones específicos de comportamiento que puedan influir en el ciclo de vida.
- Posibles relaciones indirectas: el hecho de que los días de mayor actividad sean los fines de semana podría correlacionarse con otros factores, como el tipo de vertical (por ejemplo, "Restaurant" o "Market") o promociones específicas, que podrían tener más influencia en la predicción del ciclo de vida.
- Limitaciones: por sí sola, el día de la semana de la 1er compra probablemente no sea un diferenciador fuerte para predecir el máximo segmento alcanzado debido a la falta de variabilidad significativa entre los días.

Se analiza la relación entre la preorden en la primera compra (pedido programado fuera del horario del local) y el ciclo de vida máximo alcanzado por los usuarios, con el objetivo de identificar posibles patrones en su comportamiento:

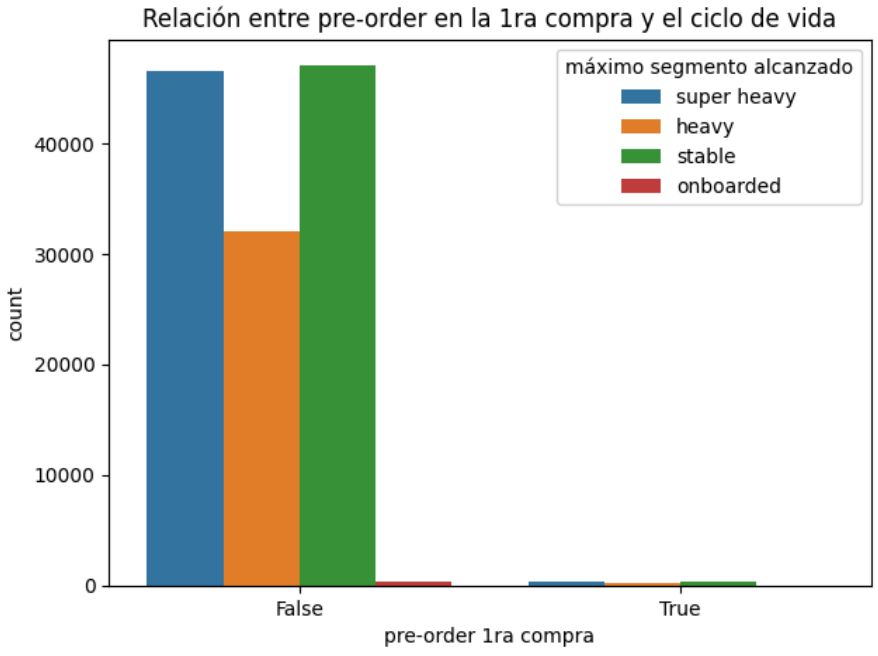

Gráfico 10 Representación en columnas 1er compra es pre-orden ó no

Vemos que la mayoría de los usuarios para su primera compra (sin importar el máximo segmento alcanzado) no hicieron una pre orden. Esto podría indicar que la opción de pre-orden no es común entre los usuarios que hacen su primera compra en la plataforma.

Los usuarios que hicieron una pre-orden en su primera compra tienen una presencia significativamente menor en todos los segmentos, incluidas las categorías más bajas (onboarded y stable).

Esto sugiere que los usuarios que eligen pre-orden en su primera compra podrían tener un comportamiento diferente o ser menos activos en la plataforma en general. Además, la baja cantidad de usuarios que hacen pre-orden podría indicar que estos usuarios no tienen el conocimiento sobre la existencia de la pre-orden en la plataforma.

Con respecto a la predicción del máximo segmento alcanzado, la pre-orden en la primera compra no parecería aportar mucho valor al modelo.

Se analiza la relación entre el tipo de entrega en la primera compra (si es retiro en local o no) y el ciclo de vida máximo alcanzado por los usuarios, con el objetivo de identificar posibles patrones en su comportamiento:

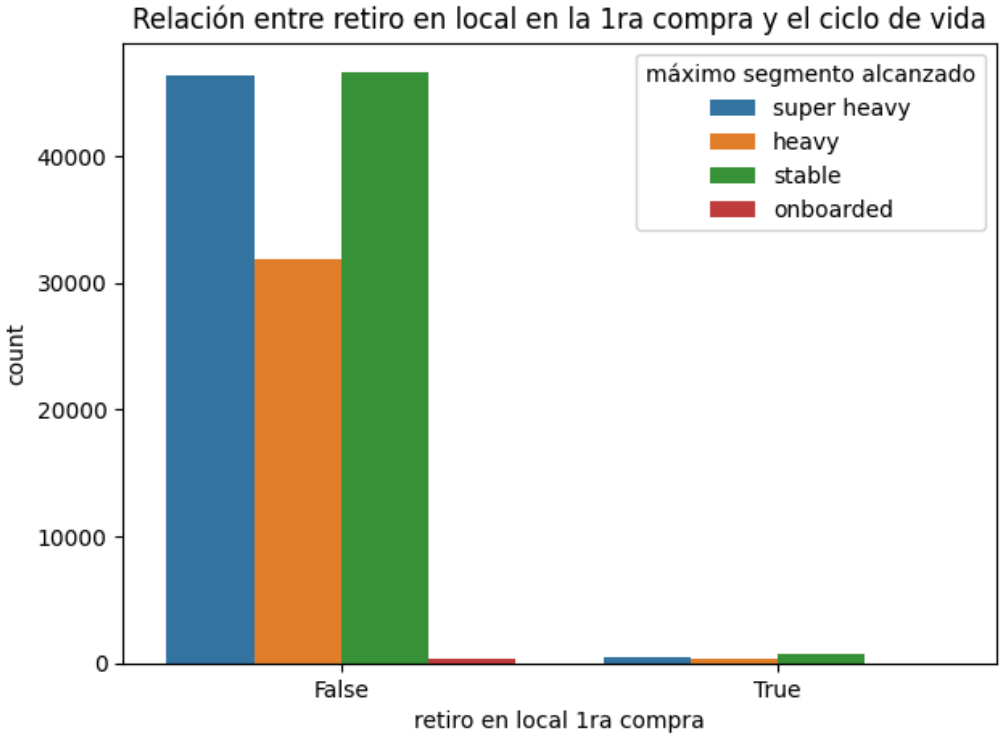

Gráfico 11 Gráfico de columnas 1er compra retiro en local ó no

Se puede observar que la mayoría de los usuarios para su primera compra (sin importar el máximo segmento alcanzado) no hicieron retiro en local.

Esto indica que la opción de retiro en local no es conocida por la gente cuando empieza a utilizar la plataforma, ó que quizás no ofrece ningún incentivo para que la primera compra se haga retirando en local.

No parece aportar mucho valor la variable referida a si fue retiro en local ó no su 1er compra en el modelo, ya que la proporción de usuarios que utilizan retiro en local es muy baja, La mayoría de los usuarios que alcanzan los segmentos más altos (super heavy, heavy) no utilizan esta modalidad.

Luego se analiza el promedio de días entre la primera y cuarta orden para cada ciclo de vida máximo alcanzado, con el objetivo de identificar diferencias en los tiempos de compra según el segmento del usuario:

Máximo segmento alcanzado	Promedio días entre 1ra y 4ta orden
onboarded	14.28
stable	9.72
heavy	5.50
super heavy	2.94

Tabla 4 Promedio de días entre la 1ra y 4ta orden por máximo segmento alcanzado

Se puede observar que a mejor segmento máximo alcanzado, se reduce el promedio de días entre órdenes, lo cual tiene sentido (a mayor segmento de usuario alcanzado, mayor es la frecuencia con la que compra la gente, es decir su intermitencia. Por ende el promedio de días será menor. La definición de intermitencia se encuentra en el *Capítulo 1 Introducción*).

Por último, se analiza el mes en el que los usuarios alcanzan su máximo segmento para poder entender en qué momento se concentran la mayor cantidad de usuarios:

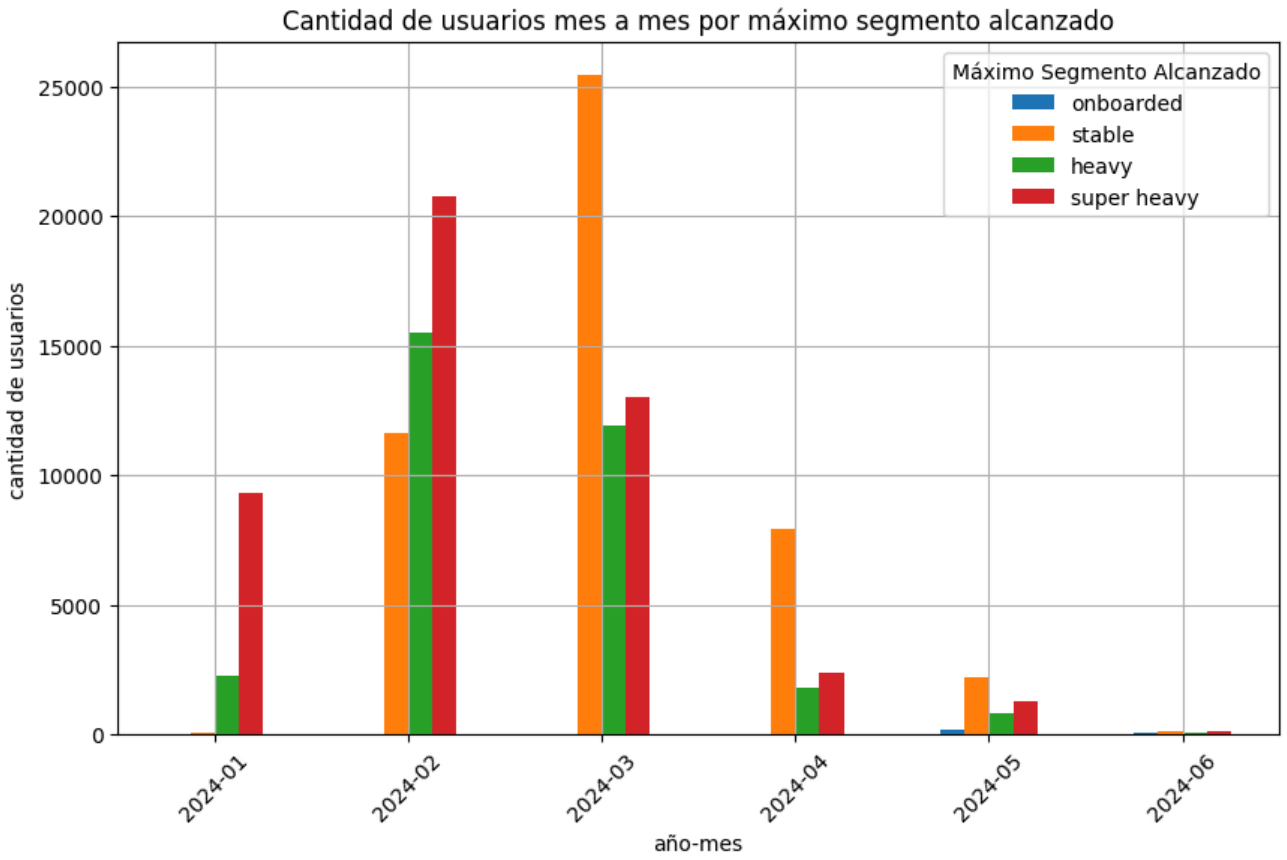

Gráfico 12 Mes en el que los usuarios alcanzan su máximo segmento

Se pueden mencionar algunos puntos interesantes referidos al comportamiento de los usuarios:

Segmento stable (barra naranja):

- Segmento más numeroso en los meses de marzo, abril y mayo de 2024, alcanzando su punto máximo en marzo de 2024 (más de 25.000 usuarios).
- Presenta una caída pronunciada a partir de marzo.

Segmento super heavy (barra roja):

- Segmento más numeroso en enero y febrero 2024.
- Su cantidad de usuarios crece hasta febrero de 2024 (~20.000 usuarios) y disminuye a partir de marzo.
- La caída es menos pronunciada que la de stable, aunque sigue una tendencia descendente.

Segmento heavy (barra verde):

- Tiene un comportamiento similar al segmento super heavy, pero con menos usuarios en general.

- El crecimiento es visible hasta febrero, con una caída más estable en los meses posteriores.

Segmento onboarded (barra azul):

- Es el segmento con menos usuarios durante todo el periodo.
- Se mantiene prácticamente constante en niveles mínimos, indicando que muy pocos usuarios se quedan en este segmento sin avanzar a otros.

Por otro lado, podemos mencionar ciertos hallazgos en relación a lo temporal:

a) Febrero como punto máximo:
    - Heavy y Super Heavy alcanzan su mayor número de usuarios en febrero de 2024.
b) Declive general a partir de abril:
    - A partir de abril de 2024, se observa un descenso sostenido en la cantidad de usuarios de Stable, Super Heavy y Heavy .
    - Esto podría indicar pérdida de usuarios activos, cambios en el mercado o en el comportamiento de consumo.

La cantidad de usuarios por máximo segmento alcanzado no la utilizamos para predecir ya que sino podríamos provocar *data leakage*.

Decisión sobre la clase Onboarded

Se decidió agregar muestras sintéticas para la clase Onboarded (sobremuestreo con el método SMOTE) ya que la cantidad de usuarios que hay es muchísimo menor que la del resto de segmentos de usuario (aproximadamente representa el 1% del total).

SMOTE (técnica de sobremuestreo de minorías sintéticas por sus siglas) es un método de sobremuestreo que genera nuevas instancias sintéticas en la clase minoritaria. En lugar de replicar datos existentes, crea nuevos puntos interpolando entre instancias reales cercanas en el espacio de características.

Este método genera muestras sintéticas interpolando entre observaciones reales, lo que puede aumentar la redundancia en las características de la clase onboarded.

Entonces se optó por insertar hasta 3.000 observaciones de onboarded y no un número balanceado con respecto a los demás segmentos de usuario para no correr el riesgo de realizar sobreajuste.

Como resultado, la distribución del máximo segmento alcanzado quedó de una manera similar a la mencionada anteriormente: stable 37%, super heavy 37%, heavy 25% y onboarded ~1%.

Se realizaron las pruebas correspondientes de los distintos modelos de predicción con la clase de Onboarded incluida.

El modelo que mejor desempeño tuvo (XGBoost con búsqueda aleatoria de hiper parámetros que es distinto al modelo con mejor desempeño sin incluir la clase Onboarded) predijo literalmente 0 para esta clase (primer fila de la clasificación del reporte):

```
Confusion Matrix XGBoost con hiperparámetros RandomizedSearchCV:
 [[   0   49    3   10]
 [   0 7212 1600  759]
 [   0 1658 3173 1595]
 [   0  762 1393 7160]]

Classification Report XGBoost con hiperparámetros RandomizedSearchCV:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        62
           1       0.74      0.75      0.75      9571
           2       0.51      0.49      0.50      6426
           3       0.75      0.77      0.76      9315

    accuracy                           0.69     25374
   macro avg       0.50      0.50      0.50     25374
weighted avg       0.69      0.69      0.69     25374
```

Gráfico 13 Matriz confusión y métricas XGBoost con búsqueda aleatoria de hiper parámetros con clase Onboarded

Es por eso que se decidió quitar esta clase no representativa para realizar las predicciones que veremos más adelante.

2.1.5) Variables utilizadas en el modelo

Para la construcción del modelo predictivo, se han utilizado diversas variables que caracterizan el comportamiento de los usuarios en la plataforma. Debido a la gran cantidad de variables disponibles (160 en total tras la etapa de transformación de datos), estas se agrupan en distintas categorías según su naturaleza y función dentro del análisis.

1. Variables de identificación y registro

Estas variables permiten identificar a los usuarios y su contexto dentro de la plataforma:

- Identificador del usuario: Permite diferenciar a cada usuario en el dataset.
- País de registro: Indica el país donde el usuario se registró en la aplicación.

2. Variables de comportamiento de compra

Estas variables reflejan la evolución del usuario en la plataforma en términos de frecuencia y categorías de compra:

- Fecha de compra (de 1ra a 4ta compra): Indica el día en que el usuario realizó cada compra.
- Día de la semana de compra (de 1ra a 4ta compra): Día de la semana en el que el usuario realizó la compra.
- Categoría de la compra (de 1ra a 4ta compra): Especifica el tipo de producto o servicio adquirido, por ejemplo, restaurante o mercado.

3. Variables de modalidad de compra

Estas variables describen características específicas sobre cómo se realizó cada compra:

- Indicador de restaurante o mercado (de 1ra a 4ta compra): Indica si la compra fue en un restaurante o en un mercado.
- Indicador de pre-orden (de 1ra a 4ta compra): Indica si la compra fue un pedido anticipado.
- Indicador de retiro en local (de 1ra a 4ta compra): Señala si la compra fue retirada físicamente en el local por el usuario.

4. Variables de segmentación y evolución del usuario

Estas variables describen la evolución del usuario en la plataforma y su relación con los segmentos de fidelización:

- Máximo segmento alcanzado: Indica el máximo segmento alcanzado por el usuario en la plataforma.
- Fecha de máximo segmento alcanzado: Indica el día en que el usuario alcanzó su máximo segmento.
- Día de la semana del máximo segmento alcanzado: Día de la semana en que el usuario alcanzó su máximo nivel de fidelización.

5. Variables de intervalos temporales entre compras

Capturan el tiempo transcurrido entre eventos clave en el historial de compras del usuario:

- Diferencia en días entre la primera y la cuarta compra: Mide la cantidad de días entre ambas compras.
- Diferencia en días entre la cuarta compra y la fecha actual: Indica cuánto tiempo ha pasado desde la última compra registrada.
- Diferencia en días entre la cuarta compra y el día del máximo segmento alcanzado: Refleja el intervalo entre la última compra y el momento en que el usuario alcanzó su mayor segmento.
- Diferencia en días entre la primera compra y el día del máximo segmento alcanzado: Permite evaluar cuánto tiempo tardó un usuario en llegar a su mayor nivel de fidelización desde su primera compra.

Las variables mencionadas en el último punto, entre otras, forman parte de la ingeniería de características que se hizo en el trabajo de investigación, y que se explicarán a continuación.

2.1.6) Ingeniería de características y transformaciones

La ingeniería de características posibilita identificar patrones significativos en los datos y mejorar la capacidad predictiva de los algoritmos. Las variables derivadas durante el proceso de ingeniería de características deben estar alineadas con el problema de negocio y evitar sesgos que puedan comprometer la validez de los resultados.

Como mencionan Kuhn y Johnson en su libro *Feature Engineering and Selection: A Practical Approach for Predictive Modeling* (2019)[4], "el proceso de diseño y selección de características es el núcleo de la construcción de modelos predictivos efectivos".

Como consecuencia de lo mencionado, en este trabajo de investigación se puso foco en que las variables creadas intenten aportar valor al modelo, y no en crear una cantidad mínima de variables que no tenga sentido.

Variables Derivadas en el Análisis

1. dif_dias_1ra_4ta_orden

   $$\text{dif\_dias\_1ra\_4ta\_orden} = \text{fecha\_4ta\_orden} - \text{fecha\_1ra\_orden}$$

2. dif_dias_4ta_orden_a_hoy

   $$\text{dif\_dias\_4ta\_orden\_a\_hoy} = \text{hoy} - \text{fecha\_4ta\_orden}$$

3. dif_dias_4ta_orden_a_max_seg

[4] http://www.feat.engineering/feature-selection

dif_dias_4ta_orden_a_max_seg= fecha_max_segmento − fecha_4ta_orden

4. dif_dias_1er_orden_a_max_seg

   dif_dias_1er_orden_a_max_seg = fecha_max_segmento − fecha_1ra_orden

5. avg_days_per_segment
   - Es el promedio de días que un usuario pasa en cada segmento antes de avanzar al siguiente (entre la primera y cuarta orden).
   - Se obtiene dividiendo el total de días por el número de órdenes de segmentos:

     avg_days_per_segment = dias_totales_por_segmento / numero_de_ordenes_por_segmento

6. log_avg_days_between_orders
   - Es el logaritmo del promedio de días entre órdenes (entre la primera y cuarta orden). Se aplica la transformación logarítmica utilizando la función np.log1p() para manejar valores de cero sin errores: log_avg_days_between_orders = log(1+promedio_dias_entre_ordenes)

Exclusión de Variables por posible *data leakage*

Las siguientes variables se crearon con fines descriptivos para analizar el comportamiento de los usuarios durante el análisis exploratorio de datos (EDA), pero no se utilizaron para entrenar el modelo:

- dif_dias_4ta_orden_a_hoy
- dif_dias_4ta_orden_a_max_seg
- dif_dias_1er_orden_a_max_seg

Estas variables dependen directamente del conocimiento del máximo segmento alcanzado, que es el objetivo de la predicción. Incluirlas podría provocar *data leakage*, comprometiendo la validez de los resultados. Por esta razón, fueron excluidas del conjunto de datos utilizado para entrenar el modelo.

Las demás variables creadas (dif_dias_1ra_4ta_orden, avg_days_per_segment, log_avg_days_between_orders) sí se utilizaron para la predicción del máximo segmento, ya que contienen métricas referidas entre la 1ra y 4ta orden, información necesaria para entrenar nuestro modelo.

Transformaciones de datos

Para garantizar la calidad y robustez del modelo, se llevaron a cabo las siguientes transformaciones previas al entrenamiento:

1. Codificación de la variable dependiente:
   Dado que la predicción implica un orden natural entre las categorías, se utilizó LabelEncoder para transformar la variable objetivo en valores numéricos de 1 a 3 (donde 1 es *stable*, 2 es *heavy* y 3 es *super heavy*). Esto asegura que el modelo pueda interpretar correctamente la jerarquía implícita en los segmentos, donde un mayor valor representa una mayor frecuencia de compra y un mayor compromiso con la plataforma.
2. División del conjunto de datos en entrenamiento, validación y prueba:
   El conjunto de datos fue dividido en tres partes: entrenamiento, validación y prueba. Entrenamiento y validación representan el 80% de los datos, y prueba representa el 20%. A diferencia de una simple partición en entrenamiento y prueba, en este trabajo se implementó validación cruzada con 10 folds en el conjunto de entrenamiento, asegurando que los modelos sean evaluados de manera más robusta y reduciendo la varianza en las métricas de desempeño. Finalmente, los modelos se evalúan en el conjunto de prueba solo después de haber sido ajustados con la mejor configuración.
3. Selección y escalado de las variables:
   Se implementó selección de variables utilizando el algoritmo de Boruta con validación cruzada con 5 folds, en lugar de utilizar directamente las 160 variables originales. Este proceso permite identificar las características más relevantes para la predicción, reduciendo la dimensionalidad del problema y mejorando la interpretabilidad del modelo. Como resultado, el número de variables predictoras pasó de 160 a 20, y son las siguientes:

```
Variables seleccionadas por Boruta:
Index(['dif_dias_1ra_4ta_orden', 'dif_1er_2da', 'dif_2da_3ra', 'dif_3ra_4ta',
       'avg_days_between_orders', 'month_1er_compra', 'day_1er_compra',
       'month_2da_compra', 'day_2da_compra', 'month_3ra_compra',
       'day_3ra_compra', 'month_4ta_compra', 'day_4ta_compra',
       'log_avg_days_between_orders', 'vertical_name_1er_compra_Restaurant',
       'vertical_name_2da_compra_Market',
       'vertical_name_2da_compra_Restaurant',
       'vertical_name_3ra_compra_Market',
       'vertical_name_3ra_compra_Restaurant',
       'vertical_name_4ta_compra_Restaurant'],
      dtype='object')
Número de variables seleccionadas por Boruta con CV: 20
```

Gráfico 14 Selección de variables con algoritmo de Boruta

Luego de la selección de características, se estandarizaron las variables predictoras en ambas particiones, entrenamiento y prueba. Este paso es esencial para garantizar que las características numéricas sean comparables y que los algoritmos utilizados puedan converger de manera eficiente.

4. Evaluación de los modelos y métricas de desempeño:

Para evaluar los modelos, se aplicó validación cruzada con 10 folds en el conjunto de entrenamiento. Esta estrategia se utilizó en todos los algoritmos evaluados, asegurando una evaluación más robusta de su desempeño antes de ser probados en el conjunto de prueba. Para los dos mejores modelos seleccionados en función de su desempeño (XGBoost con búsqueda en cuadrícula y XGBoost con optimización bayesiana), además de la validación cruzada con 10 folds, se calcularon intervalos de confianza para todas las métricas (accuracy, precisión, recall y f1-score). Esto permite verificar si las diferencias en el desempeño entre modelos son estadísticamente significativas o no (observando si los intervalos de confianza se solapan).

5. Tamaño del conjunto de datos:

Después de los pasos de preparación, el conjunto de entrenamiento y validación contiene 101.262 observaciones y 20 variables seleccionadas, mientras que el conjunto de prueba está compuesto por 25.316 observaciones con las mismas 20 variables. Estas particiones permiten entrenar y validar los modelos con un volumen suficiente de datos, asegurando una adecuada capacidad de generalización.

Capítulo 3 Análisis de Resultados

En este capítulo se describen los algoritmos utilizados para generar las predicciones del máximo segmento que un usuario puede alcanzar, basándonos en los datos obtenidos y procesados previamente. Además, se presentan sus respectivos resultados. A continuación, se presenta el razonamiento detrás de su selección progresiva.

3.1) Razonamiento detrás de la selección

La selección de algoritmos en este proyecto siguió una estrategia progresiva, comenzando con vecinos más cercanos y regresión logística multinomial sin hiper parámetros. Esta elección como modelos base se debe a su simplicidad y utilidad para establecer una línea de referencia inicial.

Vecinos más cercanos evalúa patrones en los datos de forma no paramétrica, permitiendo determinar si las características y la estructura del conjunto de datos son adecuadas para modelos más complejos. Su sensibilidad a la calidad de los datos lo convierte en un buen punto de partida para explorar su preparación para el modelado avanzado.

La regresión logística multinomial aporta interpretabilidad y rapidez en la implementación, permitiendo identificar de manera inicial el impacto de las variables explicativas sobre la predicción del máximo segmento.

Estos algoritmos sirven como referencia para medir el valor agregado de enfoques más sofisticados, asegurando que la complejidad adicional se justifique por una mejora significativa en el rendimiento.

Posteriormente, se incorporaron algoritmos más avanzados como regresión logística multinomial con hiper parámetros, Random Forest y XGBoost, estos últimos dos con y sin hiper parámetros respectivamente.

Por un lado, según Synced (2017), en el artículo "How Random Forest Algorithm Works in Machine Learning"[5], "Random Forest reduce el riesgo de sobreajuste y mejora la precisión de las predicciones al agregar los resultados de múltiples árboles de decisión".

Por el otro, según Badola, S., Mishra, V. N., & Parkash, S. (2023: 14), "en la etapa de evaluación, XGBoost optimizado es identificado como el algoritmo más eficiente para clasificar áreas susceptibles a deslizamientos utilizando las 15 variables de características del conjunto de datos utilizado".

[5] https://synced.medium.com/how-random-forest-algorithm-works-in-machine-learning-3c0fe15b6674

Tanto Random Forest como XGBoost resultan apropiados para predecir el máximo segmento de un usuario debido a su capacidad para manejar relaciones no lineales y captar interacciones complejas entre variables, como las fechas de compras y patrones temporales. Además, ambos algoritmos son robustos frente a conjuntos de datos con baja variabilidad y datos heterogéneos, lo que permite extraer información valiosa incluso cuando las características presentan estructuras no lineales o correlaciones sutiles. Estas fortalezas los convierten en herramientas ideales para optimizar la precisión predictiva y tomar decisiones estratégicas en escenarios de segmentación de usuarios.

Adicionalmente, la decisión de incorporar hiper parámetros permitió ajustar variantes de cada modelo y comprender mejor los factores que influyen en los resultados.

Este enfoque progresivo posibilitó establecer una base sólida con modelos simples y luego escalar a algoritmos más robustos y precisos, equilibrando simplicidad y rendimiento en el desarrollo del modelo predictivo.

De manera complementaria, se optó por un único modelo que abarque todas las verticales con el objetivo de mantener una visión integral del comportamiento de los usuarios, dado que una misma persona puede realizar compras en distintas verticales a lo largo del período analizado. Esta aproximación no solo permite capturar patrones de consumo más completos, sino que también enriquece los patrones obtenidos, proporcionando un entendimiento más holístico y representativo del ecosistema de compra en este estudio.

3.2) Multicolinealidad

Se emplearon dos técnicas para evaluar la multicolinealidad en las variables:

3.2.1) Factor de Inflación de Varianza (VIF)

En términos generales, un VIF superior a 10 puede ser indicativo de multicolinealidad preocupante. Sin embargo, al intentar calcular el VIF, el procesamiento no pudo completarse debido a la gran cantidad de datos y complejidad computacional involucrada.

3.2.2) Matriz de correlación

La matriz de correlación revela distintos patrones según el tipo de variable analizada. Dado el gran número de variables evaluadas, se adjunta la matriz completa en *Anexo III - Matriz de correlación* para referencia.

Las variables temporales, como los días entre compras y el promedio de días entre órdenes, presentan correlaciones moderadas entre sí, lo que indica una relación en la frecuencia de compra de los usuarios.

Los indicadores binarios de la categoría de compra (restaurantes, mercados, etc.) muestran baja correlación entre sí, lo que sugiere que las elecciones de los usuarios en cuanto a tipo de compra son relativamente independientes.

En cuanto a las variables relacionadas con el país de registro, se observa que tienen una baja correlación con las demás, lo que indica que las diferencias en el comportamiento de compra según el país no son significativas en estas dimensiones.

No se identificaron correlaciones altas entre variables que podrían generar colinealidad en el modelo, lo que sugiere que las características utilizadas son en su mayoría independientes, favoreciendo la estabilidad y la interpretación del modelo.

En este análisis, no se detectaron correlaciones fuertes (ni positivas ni negativas), lo que indica que no existen relaciones lineales significativas entre las variables incluidas. Esto respalda la confiabilidad del modelo predictivo y minimiza la preocupación por la multicolinealidad.

3.3) Modelos y resultados obtenidos

Con las etapas previas completas mencionadas en el *2.1.6) Ingeniería de características y transformaciones*, se procedió a entrenar los modelos predictivos, cuyos resultados serán detallados en las secciones siguientes.

Para lograr entender la predicción de cada clase, es importante saber cómo se encuentran definidas: la clase 0 corresponde al segmento 'Stable', la clase 1 a 'Heavy' y la clase 2 a 'Super Heavy'. Esta información será útil a la hora de interpretar cada gráfico con la información de cada modelo entrenado.

Para establecer una línea de base que justifique el despliegue del modelo en producción, se considera un punto de referencia teórico basado en una asignación aleatoria de clases. Si cada usuario tuviera la misma probabilidad de pertenecer a cualquiera de los cuatro segmentos, un clasificador que asigne etiquetas al azar tendría un accuracy esperado del 25%. Para asegurar que el modelo aporta valor y mejora significativamente sobre una decisión aleatoria, se establece como criterio mínimo de desempeño que el accuracy obtenido sea al menos el doble de esta referencia, es decir, mayor o igual al 50%. Esta línea de base se adopta como umbral fijo para evaluar la viabilidad del modelo en producción.

3.3.0) Métricas de evaluación de modelos

Para comparar los modelos de manera efectiva, se han seleccionado métricas que evalúan su desempeño en la predicción del máximo segmento alcanzado por un usuario. La elección de estas métricas responde tanto a la distribución de las clases en los datos como a la necesidad del negocio de tomar decisiones estratégicas basadas en la segmentación correcta de usuarios.

Dado el contexto del negocio, se priorizan las siguientes métricas:

- Recall: Es fundamental para identificar correctamente a los usuarios de alto valor (heavy y super heavy), evitando su subestimación y asegurando que las estrategias comerciales lleguen a los clientes adecuados.
- Precisión: Minimiza la asignación errónea de usuarios a segmentos superiores (por ejemplo, etiquetar como super heavy a un usuario stable), evitando sesgos en estrategias de fidelización e incentivos.
- F1-score: Se utiliza como métrica complementaria debido al desbalance de clases, ya que balancea precisión y recall.
- Accuracy: Es la métrica clave para determinar el mejor modelo, ya que mide el porcentaje total de predicciones correctas.

Si bien en problemas con clases desbalanceadas suele complementarse accuracy con recall y precisión, en este caso accuracy es suficiente para comparar modelos porque el objetivo del negocio es clasificar correctamente a los usuarios en su segmento final, sin importar cuál sea. Si bien los falsos positivos y falsos negativos pueden tener impactos diferentes en términos de estrategia comercial, el objetivo del modelo es maximizar la correcta clasificación global de los usuarios. Accuracy sigue siendo la métrica más adecuada, ya que evalúa el desempeño general del modelo sin favorecer una clase específica, asegurando una comparación objetiva entre enfoques.

Si bien precisión y recall permiten analizar el comportamiento del modelo en cada segmento, accuracy resume el desempeño global y se alinea con la necesidad del negocio de optimizar estrategias de marketing y fidelización.

Además de lo mencionado, aunque se analizan otras métricas para entender el comportamiento del modelo, la selección del mejor modelo se basa exclusivamente en accuracy porque:

- El problema es cerrado: Los usuarios solo pueden pertenecer a una de cuatro categorías definidas.
- El impacto se mide en el agregado: La estrategia se aplicará a grandes volúmenes de usuarios, por lo que un modelo con mayor accuracy garantiza

mejores decisiones sin necesidad de ajustes en precisión o recall por segmento.

3.3.1) Métodos de optimización de hiper parámetros

El proceso de optimización de hiper parámetros es clave para mejorar el desempeño de los modelos predictivos. Se emplearon distintas estrategias para encontrar la configuración óptima de parámetros, equilibrando rendimiento y eficiencia computacional. A continuación, se presentan las técnicas utilizadas:

- Búsqueda en cuadrícula: Explora de manera exhaustiva todas las combinaciones posibles de hiper parámetros dentro de un espacio predefinido, garantizando la identificación de la mejor configuración dentro de ese rango.
- Búsqueda aleatoria de hiper parámetros: En lugar de evaluar todas las combinaciones, selecciona un subconjunto aleatorio, reduciendo el costo computacional sin comprometer significativamente la calidad de la optimización.
- Optimización bayesiana: Modela la función de rendimiento de los hiper parámetros como un proceso gaussiano, permitiendo seleccionar valores de manera más eficiente que las búsquedas exhaustivas. En algunos casos, incorpora validación cruzada para mejorar la estimación del desempeño del modelo.
- Optimización basada en algoritmos evolutivos: Implementada a través de TPOT, utiliza técnicas de evolución genética para explorar combinaciones de hiper parámetros y modelos, seleccionando automáticamente la mejor configuración.
- Optimización de ajuste adaptativo de recursos: Método que asigna dinámicamente los recursos computacionales a distintas configuraciones de hiper parámetros, descartando iteraciones menos prometedoras y concentrando el procesamiento en las opciones con mejor desempeño.
- AutoML: Estrategia automatizada que combina múltiples métodos de optimización y selección de modelos, reduciendo la intervención manual en la configuración de los algoritmos.

Este conjunto de técnicas permitió evaluar distintas configuraciones de manera eficiente, optimizando los modelos para su aplicación en el problema de segmentación de usuarios.

3.3.2) Vecinos más cercanos

Para determinar el número óptimo de vecinos, se realizaron pruebas utilizando diferentes valores de k. Luego de observar la precisión y el número de

comparaciones que el algoritmo realiza durante la clasificación, se optó por utilizar k=7.

El modelo obtuvo un resultado en la predicción de 0.6540. Este resultado refleja un desempeño aceptable para un modelo básico, pero deja margen para explorar algoritmos más sofisticados que puedan capturar mejor las relaciones no lineales o complejas en los datos.

```
Cross-validation scores: [0.65063691 0.66001777 0.65929291 0.65494766 0.65326881 0.65475015
 0.65277503 0.66215682 0.6539601  0.65435513]
Mean CV Accuracy: 0.6556161302124679
Accuracy en Test KNN: 0.6539737715278875

Confusion Matrix KNN:
 [[7037 1551  892]
 [2076 2708 1680]
 [1023 1538 6811]]

Classification Report KNN:
              precision    recall  f1-score   support

           0       0.69      0.74      0.72      9480
           1       0.47      0.42      0.44      6464
           2       0.73      0.73      0.73      9372

    accuracy                           0.65     25316
   macro avg       0.63      0.63      0.63     25316
weighted avg       0.65      0.65      0.65     25316
```

Gráfico 15 Matriz confusión y métricas KNN

3.3.3) Regresión logística multinomial

Dado que la segmentación de usuarios presenta una estructura con niveles progresivos de recurrencia (desde onboarded hasta super heavy), inicialmente se consideró que la regresión logística ordinal sería la opción más adecuada para capturar esta jerarquía en la clasificación. Sin embargo, al evaluar su desempeño, la regresión logística ordinal sin ajuste de hiper parámetros obtuvo un accuracy de 0.5901, mientras que la regresión logística multinomial alcanzó 0.6894.

Asimismo, se probó la regularización con Lasso, obteniendo 0.3702 de accuracy para regresión logística ordinal, donde las clases 0 y 1 tuvieron precision, recall y F1-score de 0, lo que no aporta valor al modelo:

```
Cross-validation scores RL: [0.37444455 0.37444455 0.37023504 0.37023504 0.37448153 0.37448153
 0.37023504 0.37438278 0.37023504 0.37448153]
Mean CV Accuracy RL: 0.3727656637457207
Hora de finalización: 2025-03-15 15:36:07.707486
Tiempo de ejecución: 0:00:24.938704
Mejores hiperparámetros encontrados RL: {'alpha': 1, 'loss': 'log_loss', 'max_iter': 1000, 'penalty': 'l1', 'random_state': 11}
Accuracy en Test RL: 0.37020066361194504

Confusion Matrix RL:
 [[   0    0 9480]
 [   0    0 6464]
 [   0    0 9372]]

Classification Report RL:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00      9480
           1       0.00      0.00      0.00      6464
           2       0.37      1.00      0.54      9372

    accuracy                           0.37     25316
   macro avg       0.12      0.33      0.18     25316
weighted avg       0.14      0.37      0.20     25316

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

Gráfico 16 Regresión logística ordinal Lasso

Por ende vemos que ciertos modelos revelaron que la regresión logística ordinal tiene dificultades para captar ciertos patrones de segmentación presentes en los datos, lo que afectó su rendimiento.

Dado que el principal objetivo del modelo es clasificar correctamente a los usuarios en su segmento final sin imponer un orden específico, se optó por la regresión logística multinomial, que demostró un mejor desempeño general y es más adecuada para la estructura de los datos.

Tal como se mencionó, la regresión logística multinomial sin ajuste de hiper parámetros obtuvo un resultado de 0.6894. Este desempeño inicial sugiere que tiene una capacidad razonable para predecir la clase objetivo en base a las variables explicativas proporcionadas.

```
Cross-validation scores RL: [0.68756789 0.69181396 0.695635   0.68980841 0.68882086 0.69316611
 0.69405491 0.69365989 0.69425242 0.69701758]
Mean CV Accuracy RL: 0.6925797019370876
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.py:1247: FutureWarning: 'multi_class' was deprecated in version 1.5 and will be removed in 1.7. From then on, it will always use 'multinomial'. Leave it to its de
  warnings.warn(
Accuracy en Test RL: 0.6893664085953547

Confusion Matrix RL:
 [[7227 1433  820]
 [1818 3098 1548]
 [ 860 1385 7127]]

Classification Report RL:
              precision    recall  f1-score   support

           0       0.73      0.76      0.75      9480
           1       0.52      0.48      0.50      6464
           2       0.75      0.76      0.76      9372

    accuracy                           0.69     25316
   macro avg       0.67      0.67      0.67     25316
weighted avg       0.68      0.69      0.69     25316
```

Gráfico 17 Matriz confusión y métricas regresión logística multinomial

Optimización mediante hiper parámetros

Tras evaluar los resultados iniciales, se exploraron técnicas de optimización ajustando los hiper parámetros de la regresión logística. Para más detalle, revisar *Anexo II - Hiper Parámetros Utilizados*.

3.3.3.1) Regularización Ridge

Se detallan los experimentos realizados para evaluar el desempeño de la regresión logística multinomial utilizando regularización Ridge (penalización l2) con diferentes combinaciones de hiper parámetros. Se buscó determinar si el ajuste de hiper parámetros mejora la performance en comparación con la regresión logística multinomial sin ajustes.

Caso 1: Regularización Ridge con valores iniciales de hiper parámetros

Hipótesis: Ridge con los hiper parámetros iniciales tendría un mejor desempeño que la regresión logística multinomial sin ajustar por hiper parámetros.

```
Cross-validation scores RL: [0.68292683 0.68944406 0.68832708 0.68635197 0.68645072 0.68704325
 0.69020344 0.69365989 0.6860557  0.69128975]
Mean CV Accuracy RL: 0.6881752679153693
Hora de finalización: 2025-03-15 16:57:30.550684
Tiempo de ejecución: 0:00:29.114747
Mejores hiperparámetros encontrados RL: {'C': 0.01, 'max_iter': 10, 'penalty': 'l2', 'solver': 'lbfgs'}
Accuracy en Test RL: 0.6857323431821772

Confusion Matrix RL:
 [[7254 1242  984]
 [1827 2648 1989]
 [ 856 1058 7458]]

Classification Report RL:
              precision    recall  f1-score   support

           0       0.73      0.77      0.75      9480
           1       0.54      0.41      0.46      6464
           2       0.71      0.80      0.75      9372

    accuracy                           0.69     25316
   macro avg       0.66      0.66      0.65     25316
weighted avg       0.67      0.69      0.68     25316
```

Gráfico 18 Matriz confusión y métricas regresión logística multinomial caso 1 Ridge

- La mejor combinación encontrada fue:
  - C=0.01
  - max_iter = 10
- Esta configuración obtuvo un *accuracy* de ~0.6857, lo que es inferior al desempeño de la regresión logística multinomial sin hiper parámetros (~0.6894). Por ende nuestra hipótesis queda descartada.

Caso 2: Regularización Ridge con mayor rango de hiper parámetros

Hipótesis: Aumentar el valor de C y el número de iteraciones mejoraría el desempeño del caso 1 y de la regresión logística multinomial sin hiper parámetros.

```
Cross-validation scores RL: [0.68687667 0.69299891 0.69603002 0.68921588 0.68852459 0.69277108
 0.69395615 0.69375864 0.69375864 0.69662256]
Mean CV Accuracy RL: 0.6924513146792404
Hora de finalización: 2025-03-15 16:55:43.586051
Tiempo de ejecución: 0:00:44.131189
Mejores hiperparámetros encontrados RL: {'C': 0.1, 'max_iter': 15, 'penalty': 'l2', 'solver': 'lbfgs'}
Accuracy en Test RL: 0.6902354242376363

Confusion Matrix RL:
 [[7163 1492  825]
 [1735 3170 1559]
 [ 833 1398 7141]]

Classification Report RL:
              precision    recall  f1-score   support

           0       0.74      0.76      0.75      9480
           1       0.52      0.49      0.51      6464
           2       0.75      0.76      0.76      9372

    accuracy                           0.69     25316
   macro avg       0.67      0.67      0.67     25316
weighted avg       0.69      0.69      0.69     25316
```

Gráfico 19 Matriz confusión y métricas regresión logística multinomial caso 2 Ridge

- La mejor combinación encontrada fue:
  - C=0.1
  - max_iter = 15
- Este modelo obtuvo un resultado de ~0.6902, lo que representa una mejora en comparación con los hiper parámetros probados en el Caso 1 (~0.6857).
- El desempeño del modelo ajustado fue mayor que regresión logística multinomial sin ajuste.

3.3.3.2) Regularización Lasso

Caso 1: Regularización Lasso con valores iniciales de hiper parámetros en regresión logística multinomial

Hipótesis: La regularización Lasso podría mejorar el *accuracy* del modelo Ridge caso 2 al introducir una técnica de selección de características implícita, eliminando aquellas que no aportan significativamente al modelo.

Se probaron distintas combinaciones de hiper parámetros:

La primera combinación utilizó valores de C de 0.001, 0.01 y 0.1, y las iteraciones de 5, 10 y 15.

```
Cross-validation scores RL: [0.68746914 0.69240644 0.69603002 0.69030219 0.68842583 0.69365989
 0.69444993 0.69316611 0.69425242 0.6965238 ]
Mean CV Accuracy RL: 0.6926685771718335
Hora de finalización: 2025-03-15 16:53:34.800813
Tiempo de ejecución: 0:02:13.906296
Mejores hiperparámetros encontrados RL: {'C': 0.1, 'max_iter': 15, 'penalty': 'l1', 'solver': 'saga'}
Accuracy en Test RL: 0.6895639121504187

Confusion Matrix RL:
 [[7234 1423  823]
 [1823 3089 1552]
 [ 860 1378 7134]]

Classification Report RL:
              precision    recall  f1-score   support

           0       0.73      0.76      0.75      9480
           1       0.52      0.48      0.50      6464
           2       0.75      0.76      0.76      9372

    accuracy                           0.69     25316
   macro avg       0.67      0.67      0.67     25316
weighted avg       0.68      0.69      0.69     25316

/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_sag.py:349: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn(
```

Gráfico 20 Matriz confusión y métricas regresión logística multinomial Caso 1 Lasso

El resultado obtenido fue de ~0.6896. Por ende no superó al caso 2 de Ridge (~0.6902).

Caso 2: Regularización Lasso con incremento de valores de hiper parámetros en regresión logística multinomial

Para la segunda configuración se incrementaron los valores de C (0.01 y 1), dado que valores más pequeños pueden dificultar el ajuste del modelo, y se incrementó el número de iteraciones (50, 100 y 150).

```
Cross-validation scores RL: [0.68756789 0.69290017 0.6965238  0.69000593 0.68891961 0.69365989
 0.69336362 0.69415366 0.69306735 0.69761011]
Mean CV Accuracy RL: 0.6927772025670976
Hora de finalización: 2025-03-15 16:49:22.068821
Tiempo de ejecución: 0:11:25.220160
Mejores hiperparámetros encontrados RL: {'C': 1, 'max_iter': 150, 'penalty': 'l1', 'solver': 'saga'}
Accuracy en Test RL: 0.6895244114394059

Confusion Matrix RL:
 [[7225 1433  822]
 [1809 3106 1549]
 [ 855 1392 7125]]

Classification Report RL:
              precision    recall  f1-score   support

           0       0.73      0.76      0.75      9480
           1       0.52      0.48      0.50      6464
           2       0.75      0.76      0.76      9372

    accuracy                           0.69     25316
   macro avg       0.67      0.67      0.67     25316
weighted avg       0.69      0.69      0.69     25316

/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_sag.py:349: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn(
```

Gráfico 21 Matriz confusión y métricas regresión logística multinomial Caso 2 Lasso

El resultado obtenido fue de ~0.6895. Por ende no superó al caso 2 de Ridge (~0.6902).

Caso 3: Regularización Lasso con reducción de complejidad de valores de hiper parámetros en regresión logística multinomial

Para la tercera configuración se eligió un sólo valor de C igual a 1 y un sólo valor de cantidad de iteraciones igual a 1000.

```
Cross-validation scores RL: [0.68756789 0.69280142 0.6965238  0.69000593 0.68891961 0.69356113
 0.69326486 0.69415366 0.69306735 0.69770887]
Mean CV Accuracy RL: 0.6927574524065795
Hora de finalización: 2025-03-15 16:33:16.838578
Tiempo de ejecución: 0:07:58.637647
Mejores hiperparámetros encontrados RL: {'C': 1, 'max_iter': 1000, 'penalty': 'l1', 'solver': 'saga'}
Accuracy en Test RL: 0.6895244114394059

Confusion Matrix RL:
 [[7225 1433  822]
 [1809 3106 1549]
 [ 855 1392 7125]]

Classification Report RL:
              precision    recall  f1-score   support

           0       0.73      0.76      0.75      9480
           1       0.52      0.48      0.50      6464
           2       0.75      0.76      0.76      9372

    accuracy                           0.69     25316
   macro avg       0.67      0.67      0.67     25316
weighted avg       0.69      0.69      0.69     25316
```

Gráfico 22 Matriz confusión y métricas regresión logística multinomial Caso 3 Lasso

Con esta simplificación, el resultado fue de ~0.6895. Por ende superó al caso base de regresión logística (~0.6894) pero no al caso 2 de Ridge (~0.6902).

Conclusión General

La regresión logística multinomial con Ridge caso 2 obtuvo el mejor accuracy (~0.6902) en comparación con la versión sin hiper parámetros y con Lasso, ya que la regularización L2 de Ridge ayuda a controlar la varianza del modelo sin forzar la eliminación de características, lo que permite capturar mejor los patrones en los datos sin comprometer la estabilidad del entrenamiento.

3.3.4) Random Forest

El primer modelo se ejecutó sin ajustar hiper parámetros y obtuvo un resultado de 0.6484.

```
Cross-validation scores RF: [0.64757579 0.65192061 0.65257752 0.64833103 0.6512937  0.65060241
 0.65079992 0.65702153 0.65336757 0.6504049 ]
Mean CV Accuracy RF: 0.6513894973691904
Accuracy en Test RF: 0.648404171275083

Confusion Matrix RF:
 [[6893 1664  923]
 [1976 2792 1696]
 [1106 1536 6730]]

Classification Report RF:
              precision    recall  f1-score   support

           0       0.69      0.73      0.71      9480
           1       0.47      0.43      0.45      6464
           2       0.72      0.72      0.72      9372

    accuracy                           0.65     25316
   macro avg       0.63      0.63      0.63     25316
weighted avg       0.64      0.65      0.65     25316
```

Gráfico 23 Matriz confusión y métricas Random Forest

Este modelo nos dió inferior a regresión logística multinomial con hiper parámetros del caso 2 Ridge (~0.6902), lo que nos lleva a utilizar hiper parámetros aquí para ver si podemos obtener un mejor resultado.

Optimización de hiper parámetros

A) Búsqueda en cuadrícula

Se realizaron tres configuraciones de búsqueda en cuadrícula para optimizar los hiper parámetros.

Hipótesis: probar distintas configuraciones de búsqueda en cuadrícula para RandomForest mejoraría el desempeño de la regresión logística multinomial con hiper parámetros del caso 2 Ridge (~0.6902).

Para el primer caso, los mejores hiper parámetros son los siguientes:

{'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 10}

Resultado: 0.6353

Se obtiene un resultado que es peor que el caso de RandomForest sin hiper parámetros (~0.6484).

```
Mejores parámetros para Random Forest: {'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 10}
Cross-validation scores RF: [0.63622001 0.63503505 0.64052933 0.63065376 0.63806044 0.63776417
 0.63559155 0.63806044 0.63983804 0.63717164]
Mean CV Accuracy RF: 0.636892442671729
Accuracy RF con hiperparametros GridSearch: 0.6353294359298467

Confusion Matrix RF con hiperparametros GridSearch:
 [[6888 1631  961]
 [2112 2671 1681]
 [1217 1630 6525]]

Classification Report RF con hiperparametros GridSearch:
              precision    recall  f1-score   support

           0       0.67      0.73      0.70      9480
           1       0.45      0.41      0.43      6464
           2       0.71      0.70      0.70      9372

    accuracy                           0.64     25316
   macro avg       0.61      0.61      0.61     25316
weighted avg       0.63      0.64      0.63     25316
```

Gráfico 24 Matriz confusión y métricas Random Forest Caso 1 búsqueda en cuadrícula

Para el segundo caso, se cambió de la grilla que la variable de profundidad máxima posea los valores de 10 y 20. Por otro lado, para la cantidad máxima de variables a evaluar en cada split del árbol se probó con dos configuraciones:

- 'sqrt', que selecciona en cada split la raíz cuadrada del número total de características del dataset.
- 0.5, que selecciona aleatoriamente el 50% de las características disponibles en cada split del árbol.

Los mejores hiper parámetros son:

{'max_depth': 10, 'max_features': sqrt, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 10}

Resultado: 0.6899

Se obtiene un mejor resultado que el caso de RandomForest sin hiper parámetros (~0.6484) y un peor resultado que la regresión logística multinomial con hiper parámetros del caso 2 Ridge (~0.6902).

```
Mejores parámetros para Random Forest: {'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 10}
Cross-validation scores RF: [0.68598795 0.69438136 0.69444993 0.69267233 0.69030219 0.6938574
 0.69217855 0.69761011 0.69119099 0.69464744]
Mean CV Accuracy RF: 0.692727825703044
Accuracy RF con hiperparametros GridSearch: 0.6898799178385211

Confusion Matrix RF con hiperparametros GridSearch:
 [[7065 1584  831]
 [1651 3225 1588]
 [ 810 1387 7175]]

Classification Report RF con hiperparametros GridSearch:
              precision    recall  f1-score   support

           0       0.74      0.75      0.74      9480
           1       0.52      0.50      0.51      6464
           2       0.75      0.77      0.76      9372

    accuracy                           0.69     25316
   macro avg       0.67      0.67      0.67     25316
weighted avg       0.69      0.69      0.69     25316
```

Gráfico 25 Matriz confusión y métricas Random Forest Caso 2 búsqueda en cuadrícula

Se prueba un tercer caso aumentando ciertos valores de los hiper parámetros. El n_estimators ahora es 30, los valores de max_depth son 50, 100 y 150, el min_samples_split es igual a 5, el min_samples_leaf es igual a 3. Por último, para el número máximo de características utilizadas en cada división se probaron dos configuraciones:

- 'sqrt' (mencionado anteriormente su explicación)
- 0.8, que selecciona aleatoriamente el 80% de las características disponibles en cada split del árbol.

Los mejores hiper parámetros son:

{'max_depth': 50, 'max_features': 'sqrt', 'min_samples_leaf': 3, 'min_samples_split': 5, 'n_estimators': 30}

Resultado: 0.6766

Se consigue un mejor resultado que el caso de RandomForest sin hiper parámetros (~0.6484), pero un peor resultado que RandomForest con búsqueda en cuadrícula caso 2 (~0.6899).

```
Mejores parámetros para Random Forest: {'max_depth': 50, 'max_features': 'sqrt', 'min_samples_leaf': 3, 'min_samples_split': 5, 'n_estimators': 30}
Cross-validation scores RF: [0.67522465 0.67798953 0.68299427 0.67855027 0.68032787 0.68487063
 0.68240174 0.68210547 0.68121667 0.68309303]
Mean CV Accuracy RF: 0.6808774124611898
Accuracy RF con hiperparametros GridSearch: 0.676568178227208

Confusion Matrix RF con hiperparametros GridSearch:
 [[7012 1615  853]
 [1794 3022 1648]
 [ 892 1386 7094]]

Classification Report RF con hiperparametros GridSearch:
              precision    recall  f1-score   support

           0       0.72      0.74      0.73      9480
           1       0.50      0.47      0.48      6464
           2       0.74      0.76      0.75      9372

    accuracy                           0.68     25316
   macro avg       0.65      0.65      0.65     25316
weighted avg       0.67      0.68      0.67     25316
```

Gráfico 26 Matriz confusión y métricas Random Forest Caso 3 búsqueda en cuadrícula

Como consecuencia de los resultados, para el caso de RandomForest con hiper parámetros con búsqueda en cuadrícula, la hipótesis no se cumple ya que no supera el desempeño de la regresión logística multinomial con hiper parámetros del caso 2 Ridge (~0.6902) .

B) Búsqueda aleatoria de hiper parámetros

Hipótesis: el uso de búsqueda aleatoria de hiper parámetros mejoraría el desempeño que tuvimos en regresión logística multinomial caso 2 Ridge (~0.6902).

- Mejores hiperparámetros: {'n_estimators': 50, 'min_samples_split': 2, min_samples_leaf': 1, 'max_depth': 10, 'bootstrap': True}
- Resultado: 0.6914

```
Mejores hiperparámetros RF RandomizedSearchCV: {'n_estimators': 50, 'min_samples_split': 2, 'min_samples_leaf': 1, 'max_depth': 10, 'bootstrap': True}
Cross-validation scores RF RandomizedSearchCV: [0.68756789 0.69467759 0.6974126  0.68970966 0.6904997  0.69731385
 0.695635   0.69612878 0.69365989 0.69632629]
Mean CV Accuracy RF RandomizedSearchCV: 0.693893124180502
Accuracy RF con hiperparámetros RandomizedSearchCV: 0.6914599462790331

Confusion Matrix RF con hiperparámetros RandomizedSearchCV:
 [[7063 1582  835]
 [1638 3249 1577]
 [ 805 1374 7193]]

Classification Report RF con hiperparámetros RandomizedSearchCV:
              precision    recall  f1-score   support

           0       0.74      0.75      0.74      9480
           1       0.52      0.50      0.51      6464
           2       0.75      0.77      0.76      9372

    accuracy                           0.69     25316
   macro avg       0.67      0.67      0.67     25316
weighted avg       0.69      0.69      0.69     25316
```

Gráfico 27 Matriz confusión y métricas Random Forest búsqueda aleatoria de hiper parámetros

Nuestra hipótesis se comprueba y hasta ahora el mejor modelo es el Random Forest con el método de hiper parámetros con búsqueda aleatoria de hiper parámetros (0.6914).

C) Optimización bayesiana

Hipótesis: utilizar Optimización bayesiana (que tiene la característica de realizar una búsqueda más precisa) mejoraría el rendimiento obtenido con Random Forest con el método de búsqueda aleatoria de hiper parámetros (0.6914).

- Mejores hiperparámetros: {'bootstrap': False, 'max_depth': 9, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 50}
- Resultado: 0.6906

```
Mejores hiperparámetros RF Bayesian Optimization: OrderedDict([('bootstrap', False), ('max_depth', 9), ('min_samples_leaf', 1), ('min_samples_split', 2), ('n_estimators', 50)])
Cross-validation scores RF Bayesian Optimization: [0.68697541 0.6944801  0.6991902  0.69020344 0.69010468 0.6947462
 0.69454869 0.69543749 0.69454869 0.69642504]
Mean CV Accuracy RF Bayesian Optimization: 0.6936659939214401
Accuracy RF con Bayesian Optimization: 0.6905909306367515

Confusion Matrix RF con Bayesian Optimization:
 [[7062 1589  829]
 [1641 3229 1594]
 [ 807 1373 7192]]

Classification Report RF con Bayesian Optimization:
              precision    recall  f1-score   support

           0       0.74      0.74      0.74      9480
           1       0.52      0.50      0.51      6464
           2       0.75      0.77      0.76      9372

    accuracy                           0.69     25316
   macro avg       0.67      0.67      0.67     25316
weighted avg       0.69      0.69      0.69     25316
```

Gráfico 28 Matriz confusión y métricas Random Forest Optimización bayesiana

Nuestra hipótesis no se comprueba, ya que el resultado obtenido es menor que Random Forest con el método de búsqueda aleatoria de hiper parámetros.

D) Optimización basada en algoritmos evolutivos

Hipótesis: la optimización basada en algoritmos evolutivos mejoraría el rendimiento obtenido con Random Forest con búsqueda aleatoria de hiper parámetros (0.6914).

- Mejores hiper parámetros obtenidos de los resultados {bootstrap=False, class_weight='balanced', criterion='entropy', max_features~0.038, min_samples_leaf=13, min_samples_split=15, n_estimators=128, n_jobs=1, random_state=11}
- *Accuracy*: 0.6842

```
Cross-validation scores RF TPOT: [0.67937198 0.6860867  0.69128975 0.68329054 0.68477187 0.68477187
 0.68566068 0.68990717 0.6887221  0.69119099]
Mean CV Accuracy RF TPOT: 0.6865063652115857
Mejor modelo RF TPOT: Pipeline(steps=[('passthrough-1', Passthrough()),
                ('passthrough-2', Passthrough()),
                ('featureunion-1',
                 FeatureUnion(transformer_list=[('featureunion',
                                                 FeatureUnion(transformer_list=[('zerocount',
                                                                                 ZeroCount())])),
                                                ('passthrough',
                                                 Passthrough())])),
                ('featureunion-2',
                 FeatureUnion(transformer_list=[('skiptransformer',
                                                 SkipTransformer()),
                                                ('passthrough',
                                                 Passthrough())])),
                ('randomforestclassifier',
                 RandomForestClassifier(bootstrap=False,
                                        class_weight='balanced',
                                        criterion='entropy',
                                        max_features=0.0381342225786,
                                        min_samples_leaf=13,
                                        min_samples_split=15, n_estimators=128,
                                        n_jobs=1, random_state=11))])
Accuracy RF con TPOT: 0.6841918154526782

Confusion Matrix RF con TPOT:
 [[6519 2228  733]
 [1222 3952 1290]
 [ 689 1833 6850]]

Classification Report RF con TPOT:
              precision    recall  f1-score   support

           0       0.77      0.69      0.73      9480
           1       0.49      0.61      0.55      6464
           2       0.77      0.73      0.75      9372

    accuracy                           0.68     25316
   macro avg       0.68      0.68      0.67     25316
weighted avg       0.70      0.68      0.69     25316
```

Gráfico 29 Matriz confusión y métricas Random Forest algoritmos evolutivos

Nuestra hipótesis queda descartada ya que obtuvimos un peor resultado.

E) Optimización de ajuste adaptativo de recursos

Hipótesis: el uso de optimización de ajuste adaptativo de recursos mejoraría el rendimiento obtenido por Random Forest con búsqueda aleatoria de hiper parámetros (0.6914).

- Mejores hiperparámetros: {'bootstrap': True, 'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 50}
- *Accuracy*: 0.6914

```
Mejores hiperparámetros RF Hyperband: {'bootstrap': True, 'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 50}
Cross-validation scores RF Hyperband: [0.68756789 0.69467759 0.6974126  0.68970966 0.6904997  0.69731385
 0.695635   0.69612878 0.69365989 0.69632629]
Mean CV Accuracy RF Hyperband: 0.693893124180502
Accuracy RF con Hyperband: 0.6914599462790331

Confusion Matrix RF con Hyperband:
 [[7063 1582  835]
 [1638 3249 1577]
 [ 805 1374 7193]]

Classification Report RF con Hyperband:
              precision    recall  f1-score   support

           0       0.74      0.75      0.74      9480
           1       0.52      0.50      0.51      6464
           2       0.75      0.77      0.76      9372

    accuracy                           0.69     25316
   macro avg       0.67      0.67      0.67     25316
weighted avg       0.69      0.69      0.69     25316
```

Gráfico 30 Matriz confusión y métricas Random Forest Optimización de ajuste adaptativo de recursos

En conclusión, el mejor resultado se alcanzó con Random Forest con la optimización de ajuste adaptativo de recursos y búsqueda aleatoria de hiper parámetros (0.6914).

3.3.5) XGBoost

El primer modelo se ejecutó sin ajustar hiper parámetros y obtuvo un resultado de 0.6881.

```
Cross-validation scores XGB: [0.6860867  0.69517132 0.69494371 0.6896109  0.68931464 0.69425242
 0.69444993 0.69523998 0.69198104 0.69435118]
Mean CV Accuracy XGB: 0.6925401811374371
Accuracy XGB: 0.6881023858429451

Confusion Matrix XGB:
 [[7041 1589  850]
 [1665 3194 1605]
 [ 814 1373 7185]]

Classification Report XGB:
              precision    recall  f1-score   support

           0       0.74      0.74      0.74      9480
           1       0.52      0.49      0.51      6464
           2       0.75      0.77      0.76      9372

    accuracy                           0.69     25316
   macro avg       0.67      0.67      0.67     25316
weighted avg       0.69      0.69      0.69     25316
```

Gráfico 31 Matriz confusión y métricas XGBoost

Dado que no obtiene un mejor rendimiento que Random Forest con optimización de ajuste adaptativo de recursos y búsqueda aleatoria de hiper parámetros (0.6914), se realizarán ajustes de hiper parámetros empleando diversas técnicas de optimización:

A) Búsqueda en cuadrícula

Hipótesis: el uso de búsqueda en cuadrícula en XGBoost mejoraría el rendimiento obtenido en Random Forest con optimización de ajuste adaptativo de recursos y búsqueda aleatoria de hiper parámetros (0.6914).

- Mejores hiperparámetros: {'colsample_bytree': 0.6, 'learning_rate': 0.01, 'max_depth': 6, 'n_estimators': 50, 'subsample': 0.6}
- Resultado: 0.6917

```
Mejores hiperparámetros XGBoost GridSearchCV: {'colsample_bytree': 0.6, 'learning_rate': 0.01, 'max_depth': 6, 'n_estimators': 50, 'subsample': 0.6}
Cross-validation scores XGB GridSearchCV (accuracy): [0.68776538 0.69655377 0.6974126  0.69099348 0.69168477 0.69464744
 0.69306735 0.69444993 0.69632629 0.69405491]
Mean accuracy XGB GridSearchCV: 0.6937 (95% CI: [0.6917, 0.6957])
Cross-validation scores XGB GridSearchCV (precision_macro): [0.66458616 0.67699884 0.67726227 0.67071763 0.6712607  0.67462338
 0.67022869 0.67233149 0.67615755 0.67086009]
Mean precision_macro XGB GridSearchCV: 0.6725 (95% CI: [0.6699, 0.6751])
Cross-validation scores XGB GridSearchCV (recall_macro): [0.66444448 0.67676833 0.67679142 0.67039038 0.67129231 0.67457204
 0.66949843 0.67146887 0.6756685  0.67007403]
Mean recall_macro XGB GridSearchCV: 0.6721 (95% CI: [0.6695, 0.6747])
Cross-validation scores XGB GridSearchCV (f1_macro): [0.6640148  0.67674975 0.67674209 0.67033498 0.67122074 0.67453424
 0.66903018 0.67115354 0.67565319 0.66950163]
Mean f1_macro XGB GridSearchCV: 0.6719 (95% CI: [0.6691, 0.6746])
Accuracy XGBoost con hiperparámetros GridSearchCV: 0.6916969505451098
Precision XGBoost con hiperparámetros GridSearchCV: 0.6711146054487904
Recall XGBoost con hiperparámetros GridSearchCV: 0.6707886802564192
F1 Score XGBoost con hiperparámetros GridSearchCV: 0.6706962181545473

Confusion Matrix XGBoost con hiperparámetros GridSearchCV:
 [[7133 1513  834]
 [1700 3179 1585]
 [ 822 1351 7199]]

Classification Report XGBoost con hiperparámetros GridSearchCV:
              precision    recall  f1-score   support

           0       0.74      0.75      0.75      9480
           1       0.53      0.49      0.51      6464
           2       0.75      0.77      0.76      9372

    accuracy                           0.69     25316
   macro avg       0.67      0.67      0.67     25316
weighted avg       0.69      0.69      0.69     25316
```

Gráfico 32 Matriz confusión y métricas XGBoost con búsqueda en cuadrícula

Como podemos ver por el resultado obtenido, nuestra hipótesis queda comprobada.

B) Búsqueda aleatoria de hiper parámetros

En esta ocasión, además de cambiar de método de hiper parámetros, se decidió ampliar el rango de los mismos para incluir más combinaciones (n_estimators con 50 y 75, learning_rate con 0.01 y 0.1, subsample y colsample_bytree con 0.6 y 0.8)

Hipótesis: el uso de búsqueda aleatoria de hiper parámetros para XGBoost con mayor cantidad de valores mejoraría el rendimiento obtenido con XGBoost con búsqueda en cuadrícula (~0.6917).

- Mejores hiperparámetros: {'subsample': 0.6, 'n_estimators': 75, 'max_depth': 6, 'learning_rate': 0.1, 'colsample_bytree': 0.6}
- Resultado: 0.6901

```
Cross-validation scores XGBoost RandomizedSearchCV: [0.68687667 0.69586255 0.69859767 0.69257357 0.69178353 0.69464744
 0.69603002 0.69514122 0.69356113 0.69721509]
Mean CV Accuracy XGBoost RandomizedSearchCV: 0.6942288886113765
Mejores hiperparámetros XGBoost RandomizedSearchCV: {'subsample': 0.6, 'n_estimators': 75, 'max_depth': 6, 'learning_rate': 0.1, 'colsample_bytree': 0.6}
Accuracy XGBoost con hiperparámetros RandomizedSearchCV: 0.6901564228156106

Confusion Matrix XGBoost con hiperparámetros RandomizedSearchCV:
 [[7082 1560  838]
 [1667 3211 1586]
 [ 815 1378 7179]]

Classification Report XGBoost con hiperparámetros RandomizedSearchCV:
              precision    recall  f1-score   support

           0       0.74      0.75      0.74      9480
           1       0.52      0.50      0.51      6464
           2       0.75      0.77      0.76      9372

    accuracy                           0.69     25316
   macro avg       0.67      0.67      0.67     25316
weighted avg       0.69      0.69      0.69     25316
```

Gráfico 33 Matriz confusión y métricas XGBoost con búsqueda aleatoria de hiper parámetros

Nuestra hipótesis queda descartada por los resultados obtenidos.

C) Optimización bayesiana

Hipótesis: el uso de la búsqueda bayesiana para XGBoost mejoraría el rendimiento obtenido con XGBoost con búsqueda en cuadrícula (~0.6917).

- Mejores hiperparámetros: {'colsample_bytree': 0.6226, 'learning_rate': 0.0730, 'max_depth': 4, 'n_estimators': 50, 'subsample': 0.6464}
- Resultado: 0.6922

```
Mejores hiperparámetros XGBoost BayesSearchCV: OrderedDict([('colsample_bytree', 0.6226195310523449), ('learning_rate', 0.07298973122882445), ('max_depth', 4), ('n_estimators', 50), ('subsample', 0.6464214925868602)])
Cross-validation scores XGB BayesSearchCV (accuracy): [0.68895033 0.69655377 0.6983014  0.69346237 0.69079597 0.695635
 0.69405491 0.69405491 0.69682007 0.69642504]
Mean accuracy XGB BayesSearchCV: 0.6945 (95% CI: [0.6925, 0.6965])
Cross-validation scores XGB BayesSearchCV (precision_macro): [0.66674568 0.67778734 0.67846633 0.67457501 0.67067752 0.6764555
 0.67182006 0.67229499 0.67690072 0.67448767]
Mean precision_macro XGB BayesSearchCV: 0.6740 (95% CI: [0.6715, 0.6765])
Cross-validation scores XGB BayesSearchCV (recall_macro): [0.66668214 0.67770113 0.67811715 0.67441988 0.67073735 0.67643111
 0.67128105 0.67163994 0.67657164 0.67394032]
Mean recall_macro XGB BayesSearchCV: 0.6738 (95% CI: [0.6713, 0.6762])
Cross-validation scores XGB BayesSearchCV (f1_macro): [0.66641992 0.67769489 0.67807944 0.67443833 0.67068433 0.67642635
 0.67092651 0.67139775 0.67654772 0.67362872]
Mean f1_macro XGB BayesSearchCV: 0.6736 (95% CI: [0.6711, 0.6762])
Accuracy XGBoost con hiperparámetros BayesSearchCV: 0.692249960499289
Precision XGBoost con hiperparámetros BayesSearchCV: 0.6724903540295012
Recall XGBoost con hiperparámetros BayesSearchCV: 0.672253954034644
F1 Score XGBoost con hiperparámetros BayesSearchCV: 0.6722311152325756

Confusion Matrix XGBoost con hiperparámetros BayesSearchCV:
 [[7115 1532  833]
 [1680 3239 1545]
 [ 822 1379 7171]]

Classification Report XGBoost con hiperparámetros BayesSearchCV:
              precision    recall  f1-score   support

           0       0.74      0.75      0.75      9480
           1       0.53      0.50      0.51      6464
           2       0.75      0.77      0.76      9372

    accuracy                           0.69     25316
   macro avg       0.67      0.67      0.67     25316
weighted avg       0.69      0.69      0.69     25316
```

Gráfico 34 Matriz confusión y métricas XGBoost con Optimización bayesiana

Con la evidencia del resultado, nuestra hipótesis queda comprobada.

D) Optimización basada en algoritmos evolutivos

Hipótesis: la optimización basada en algoritmos evolutivos para XGBoost mejoraría el rendimiento obtenido por XGBoost con búsqueda bayesiana (~0.6922).

- Modelo: TPOTClassifier
- Hiperparámetros: generations=2, population_size=3, random_state=11,
- Resultado: 0.6855

```
Cross-validation scores XGB TPOT: [0.67868075 0.68964155 0.69267233 0.68941339 0.68941339 0.68477187
 0.6896109  0.69257357 0.6913885  0.69553624]
Mean CV Accuracy XGB TPOT: 0.6893702516066887
Accuracy XGBoost con TPOT: 0.6855348396271133

Confusion Matrix XGBoost con TPOT:
 [[6582 2125  773]
 [1276 3793 1395]
 [ 713 1679 6980]]

Classification Report XGBoost con TPOT:
              precision    recall  f1-score   support

           0       0.77      0.69      0.73      9480
           1       0.50      0.59      0.54      6464
           2       0.76      0.74      0.75      9372

    accuracy                           0.69     25316
   macro avg       0.68      0.68      0.67     25316
weighted avg       0.70      0.69      0.69     25316
```

Gráfico 35 Matriz confusión y métricas XGBoost con algoritmos evolutivos

Como podemos ver en los resultados, nuestra hipótesis tiene que ser descartada.

E) AutoML

Hipótesis: Dado que AutoML en XGBoost permite explorar configuraciones óptimas de manera rápida y eficiente, mejoraría el rendimiento obtenido de XGBoost con búsqueda bayesiana (~0.6922).

- Resultado: 0.6913

```
No path specified. Models will be saved in: "AutogluonModels/ag-20250309_125905"
Preset alias specified: 'medium_quality_faster_train' maps to 'medium_quality'.
AutoGluon infers your prediction problem is: 'multiclass' (because dtype of label-column == int, but few unique label-values observed).
        If 'multiclass' is not the correct problem_type, please manually specify the problem_type parameter during Predictor init (You may specify problem_type as one of: ['binary', 'multiclass', 'regression', 'quantile'])
        Warning: Exception caused CatBoost to fail during training (ImportError)... Skipping this model.
                `import catboost` failed. A quick tip is to install via `pip install autogluon.tabular[catboost]==1.2`.
Accuracy XGBoost con AutoGluon: 0.6913414441459946

Confusion Matrix XGBoost con AutoGluon:
 [[6997 1662  821]
 [1560 3374 1530]
 [ 784 1457 7131]]

Classification Report XGBoost con AutoGluon:
              precision    recall  f1-score   support

           0       0.75      0.74      0.74      9480
           1       0.52      0.52      0.52      6464
           2       0.75      0.76      0.76      9372

    accuracy                           0.69     25316
   macro avg       0.67      0.67      0.67     25316
weighted avg       0.69      0.69      0.69     25316
```

Gráfico 36 Matriz confusión y métricas XGBoost con AutoML

Con la evidencia de este resultado, nuestra hipótesis queda descartada.

XGBoost con búsqueda bayesiana demostró ser robusto, obteniendo el mejor resultado (~0.6922).

3.3.6) Cuadro de resultados

El cuadro de resultados sobre el conjunto de test para cada uno de los modelos utilizados es el siguiente:

Modelo	**Hiper Parámetros / Configuración**	***Accuracy***	**Precisión (Ponderado)**	**Recall (Ponderado)**	**F1-Score (Ponderado)**
Vecinos Más Cercanos	k=7	0.6540	0.6483	0.6546	0.6506
Regresión Logística	-	0.6894	0.6848	0.6894	0.6867
Regresión Logística Ridge 1	C: 0.01, max_iter: 10, penalty: 'l2', solver: 'lbfgs'	0.6857	0.6747	0.6857	0.6771
Regresión Logística Ridge 2	C: 0.1, max_iter: 15, penalty: 'l2', solver: 'lbfgs'}	0.6902	0.6868	0.6902	0.6883
Regresión Logística Lasso 1	C=0.1, Max_iter=15, Penalty: 'l1', Solver: 'saga’	0.6896	0.6848	0.6896	0.6868
Regresión Logística Lasso 2	C=1, Max_iter=150, Penalty: 'l1', Solver: 'saga'	0.6895	0.6851	0.6895	0.6869
Regresión Logística Lasso 3	C=1, Max_iter=1000, Penalty: 'l1', Solver: 'saga'	0.6895	0.6851	0.6895	0.6869
Random Forest	-	0.6484	0.6415	0.6488	0.6452
Random Forest búsqueda en cuadrícula Caso 1	max_depth=None, max_features='sqrt', min_samples_leaf=1, min_samples_split=2, n_estimators=10	0.6353	0.6340	0.6353	0.6342

Random Forest búsqueda en cuadrícula Caso 2	max_depth=10, max_features='sqrt', min_samples_leaf=1, min_samples_split=2, n_estimators=10	0.6899	0.6886	0.6899	0.6888
Random Forest búsqueda en cuadrícula Caso 3	max_depth=50, max_features='sqrt', min_samples_leaf=3, min_samples_split=5, n_estimators=30	0.6766	0.6739	0.6766	0.6751
Random Forest búsqueda aleatoria de hiper parámetros	n_estimators=50, min_samples_split=2, min_samples_leaf=1, max_depth=10, bootstrap=True	0.6914	0.6889	0.6915	0.6898
Random Forest Optimización bayesiana	bootstrap=False, max_depth=9, min_samples_leaf=1, min_samples_split=2, n_estimators=50	0.6906	0.6890	0.6909	0.6895
Random Forest Algoritmos Evolutivos	bootstrap=False, class_weight='balanced', criterion='entropy', max_features=0.0381, min_samples_leaf=13, min_samples_split=15, n_estimators=128	0.6842	0.6819	0.6849	0.6824

Random Forest Optimización de ajuste adaptativo de recursos	bootstrap=True, max_depth=10, min_samples_leaf=1, min_samples_split=2, n_estimators=50	0.6914	0.6889	0.6915	0.6898
XGBoost	-	0.6881	0.6887	0.6881	0.6884
XGBoost búsqueda en cuadrícula	colsample_bytree=0.6, learning_rate=0.01, max_depth=6, n_estimators=50, subsample=0.6	0.6917	0.6811	0.6788	0.6761
XGBoost búsqueda aleatoria de hiper parámetros	subsample=0.6, n_estimators=75, max_depth=6, learning_rate=0.1, colsample_bytree=0.6	0.6901	0.6890	0.6902	0.6895
XGBoost Optimización bayesiana	colsample_bytree=0.6226, learning_rate=0.0729, max_depth=4, n_estimators=50, subsample=0.6464	0.6922	0.6895	0.6922	0.6901
XGBoost Algoritmos Evolutivos	-	0.6855	0.6987	0.6855	0.6901
XGBoost AutoML	-	0.6913	0.6890	0.6913	0.6896

Tabla 5 Resultados de las predicciones por modelo

Tras comparar los modelos evaluados, se observa que los mejores desempeños globales en términos de *accuracy*, precisión, *recall* y *F1-score* corresponden a XG Boost optimización bayesiana (*accuracy* ~0.6922) y luego a XG Boost con búsqueda en cuadrícula que obtuvo 0.6917 en *accuracy* y 0.6761 en *F1-score* respectivamente, y además con un buen equilibrio entre precisión y *recall*.

En contraste, el modelo de RF con búsqueda en cuadrícula caso 1 (max_depth=None, max_features='sqrt', min_samples_leaf=1,

min_samples_split=2, n_estimators=10) obtuvo el peor rendimiento (0.6353 en *accuracy*), mientras que Random Forest sin hiper parámetros también mostró resultados más bajos en comparación con las demás configuraciones.

Dentro de los métodos de regresión logística multinomial, las versión de Ridge 2 fue la más competitiva, aunque sin superar en *accuracy* a los mejores modelos basados en XG Boost.

3.3.7) Estimación de la variabilidad de las métricas

Dado que la mayoría de los algoritmos utilizados en esta investigación poseen un componente aleatorio (como la inicialización de Random Forest o el descenso de gradiente en la regresión logística multinomial) y que el proceso de división en conjuntos de entrenamiento y prueba también es aleatorio, las métricas de evaluación puntuales pueden no ser completamente estables.

Por ello, se calculan intervalos de confianza al 95% sobre las métricas de los dos mejores modelos con el objetivo de obtener una estimación más robusta de su rendimiento y evaluar si existen diferencias estadísticamente significativas.

Se aplicó validación cruzada de 10 folds sobre el conjunto de entrenamiento, permitiendo obtener valores esperados de cada métrica con sus respectivos intervalos de confianza.

Los resultados muestran que el modelo XGBoost con búsqueda de grilla obtuvo un accuracy promedio de 0.6937 con un intervalo de confianza del 95% ([0.6917, 0.6957]), mientras que el modelo XGBoost con optimización bayesiana presentó un accuracy promedio de 0.6945, con un intervalo de [0.6925, 0.6965]. Dado que los intervalos de confianza de ambos modelos se solapan, no se puede afirmar con certeza que uno sea significativamente superior al otro en términos de accuracy. Este mismo patrón se observa en las demás métricas:

- Precisión macro: GridSearchCV (0.6725, IC: [0.6699, 0.6751]) vs. BayesSearchCV (0.6740, IC: [0.6715, 0.6765])
- Recall macro: GridSearchCV (0.6721, IC: [0.6695, 0.6747]) vs. BayesSearchCV (0.6738, IC: [0.6713, 0.6762])
- F1-score macro: GridSearchCV (0.6719, IC: [0.6691, 0.6746]) vs. BayesSearchCV (0.6736, IC: [0.6711, 0.6762])

Dado que los intervalos de confianza de todas las métricas analizadas se solapan, se concluye que no hay diferencias estadísticamente significativas entre los dos mejores modelos al 95% de confianza, lo que sugiere que ambos presentan un rendimiento similar en validación cruzada.

Capítulo 4: Análisis Prescriptivo

En esta sección, se realiza un análisis detallado del modelo con mejor resultado y los modelos base utilizados en el estudio, comparando sus desempeños y explorando el impacto práctico de las decisiones basadas en sus resultados. Además, se identifican las decisiones que se podrían tomar desde el equipo de negocio.

4.1) Comparación entre el mejor modelo y modelos base

Los resultados muestran una superioridad del modelo XG Boost optimización bayesiana que alcanzó una *accuracy* de 0.6922, en comparación con los modelos base: vecinos más cercanos que obtuvo 0.6540 y regresión logística multinomial sin hiper parámetros que dió como resultado 0.6894.

- Comparado con vecinos más cercanos, la mejora es aproximadamente del 5,84% y de 0,0382 en valor absoluto.
- Para el caso de regresión logística multinomial, la mejora es de 0,41% y de 0,0028 en valor absoluto.

Para el caso de vecinos más cercanos, su naturaleza dependiente de la distancia entre puntos y la falta de optimización directa en características específicas del problema lo coloca por debajo de modelos como Random Forest con hiper parámetros.

La mejora de XG Boost optimizado con búsqueda bayesiana sobre la regresión logística multinomial puede explicarse por la capacidad de XG Boost para manejar relaciones no lineales, interacciones entre variables y distribución de datos desbalanceados de manera más efectiva. Mientras que la regresión logística multinomial asume relaciones lineales entre las características y la variable objetivo, XG Boost utiliza árboles de decisión en boosting, lo que le permite modelar patrones complejos y capturar dependencias no evidentes en los datos. Además, al aplicar búsqueda bayesiana en la optimización de hiper parámetros, se encuentra una configuración más adecuada para mejorar el rendimiento predictivo, evitando sobreajuste y maximizando la generalización del modelo.

La superioridad del modelo XG Boost optimizado con búsqueda bayesiana en validación cruzada sobre el resto de los algoritmos evaluados radica en su capacidad para aprender de los errores de iteraciones previas mediante boosting, ajustando progresivamente los pesos de las observaciones difíciles de clasificar. En comparación con otros modelos como Random Forest o incluso versiones de XG Boost con optimización estándar, la búsqueda bayesiana permite encontrar de

manera eficiente los hiper parámetros óptimos, reduciendo la necesidad de exploración exhaustiva y mejorando la convergencia del modelo. Esto se traduce en una mayor precisión y estabilidad en el desempeño del algoritmo, lo que justifica su posición como el mejor modelo en términos de accuracy en el conjunto de test.

4.2) Toma de decisiones del negocio

Utilizando como punto de partida el resultado de los diferentes modelos que se entrenaron, el negocio podría tomar ciertas decisiones para que la compañía pueda maximizar sus oportunidades y minimizar sus riesgos.

Estas decisiones pueden ajustarse según el resultado obtenido, evaluando el riesgo asociado a las estrategias implementadas:

A) Campañas de marketing dirigidas:

Con la predicción del máximo segmento alcanzado por un usuario, el equipo de marketing puede diseñar campañas de notificaciones *push* o asignación de vouchers dirigidas a segmentos específicos. Según Calle García, A. J., Quimis Vera, M. C., Piguave Vargas, M. T., & Zambrano Luzardo, J. S. (2024: 200), "La aplicación de algoritmos de aprendizaje automático ha permitido a las empresas personalizar estrategias de marketing de manera más efectiva, anticipando las preferencias del consumidor y adaptando campañas publicitarias de manera más precisa. La mejora promedio del 20-30% en diversas métricas clave destaca la contribución significativa de la IA a este aspecto crucial."

Primero se podrían implementar campañas limitadas en alcance (a segmentos de usuarios más bajos como onboarded ó stable) para validar su efectividad antes de una implementación a gran escala. Luego se podrían diseñar promociones estratégicas dirigidas para usuarios de segmentos altos (heavy ó super heavy) con potencial de crecimiento, maximizando el retorno de la inversión.

B) Establecer prioridades en las acciones de Marketing:

Priorizar acciones de bajo costo, como emails o notificaciones *push* genéricas, a ciertos segmentos de usuario. Luego implementar estrategias agresivas mediante descuentos exclusivos para segmentos con mayor probabilidad de contribuir al valor bruto de ventas (segmentos más altos como super heavy). Estas acciones se realizan con el objetivo de eficientizar el costo por parte del área de marketing.

C) Pruebas A/B en la aplicación:

El uso de A/B testing permite realizar cambios controlados y minimizar riesgos, garantizando que solo se implementen modificaciones que realmente resuenen con los usuarios y generen resultados medibles.

Se podrían analizar métricas como el valor bruto de ventas, la cantidad de órdenes y las verticales de consumo para evaluar el impacto de las promociones. Además, utilizando un test de hipótesis adecuado, se podría determinar si las diferencias entre los grupos de control y variación son estadísticamente significativas, optimizando así las estrategias de promoción futuras.

Entonces se podrían ofrecer promociones diferenciadas a usuarios de distintos segmentos. Por ejemplo, lanzar cierta promoción agresiva para el 50% de usuarios super heavy versus no lanzarla para el restante 50% de los usuarios super heavy, y así entender el impacto de esta acción.

4.2.1) Recomendaciones explícitas

El recall es una métrica clave para evaluar el desempeño del modelo, ya que mide cuántas instancias de cada clase fueron correctamente identificadas. Específicamente, permite responder la pregunta: de todos los elementos que realmente pertenecen a una clase (0: Stable, 1: Heavy o 2: Super Heavy), ¿cuántos fueron correctamente clasificados?

Una clase con un recall bajo indica que el modelo no está capturando adecuadamente a los usuarios de ese segmento, lo que puede impactar negativamente en las estrategias dirigidas a este grupo. Por el contrario, una clase con un recall alto permite implementar estrategias con mayor confianza, asegurando que los usuarios clasificados en ese grupo han sido correctamente identificados.

Los modelos que presentaron mejor desempeño en términos de recall para cada clase fueron:

- Clase 0: Regresión Logística Ridge 1 Multinomial (0.77)
- Clase 1: Random Forest con algoritmos evolutivos (0.61)
- Clase 2: Regresión Logística Ridge 1 Multinomial (0.80)

Por otro lado, el recall por clase de XGBoost con optimización bayesiana es:

- Clase 0: 0.75 (0.02 puntos porcentuales menos que el mejor modelo para esta clase).
- Clase 1: 0.50 (0.11 puntos porcentuales menos que el mejor modelo para esta clase).

- Clase 2: 0.77 (0.03 puntos porcentuales menos que el mejor modelo para esta clase).

Dado el desempeño del modelo XGBoost con optimización bayesiana, se puede asumir su clasificación para las clases 0 y 2 de manera agresiva, con la consideración de que presenta una ligera diferencia con respecto al mejor modelo en cada caso.

A partir de estos resultados, se pueden definir estrategias específicas para cada segmento de clientes.

Para la clase Stable se podrían diseñar estrategias de promociones exclusivas basadas en el historial de compra. Además, se podría implementar un test A/B/C en el que un 33% de los usuarios reciba una notificación push con un descuento en cierto producto, otro 33% reciba un email con el mismo descuento y el 33% restante vea el descuento directamente en la aplicación sin notificación previa. De esta manera, se podría evaluar la efectividad de cada canal de comunicación y optimizar futuras estrategias.

Para la clase Super Heavy, una estrategia efectiva sería implementar programas de fidelización VIP, ofreciendo beneficios exclusivos en lugar de descuentos masivos. Esto podría incluir acceso anticipado a lanzamientos de productos o promociones, personalización en la experiencia del cliente con atención prioritaria y envíos rápidos, entre otros incentivos. Por ejemplo, si un cliente es frecuente en la aplicación, podría recibir un estatus premium con envíos gratis o atención prioritaria. Además, se podría ofrecer acceso anticipado a nuevos productos o servicios como una estrategia para reforzar su lealtad y mantenerlos enganchados con beneficios exclusivos. Un ejemplo de esto podría ser invitar a clientes de este segmento a probar nuevos restaurantes antes de su incorporación a la plataforma.

En cuanto a la clase Heavy, se ha determinado que el mejor modelo es el Random Forest con algoritmos evolutivos, ya que maximiza la recuperación de esta clase con un recall de 0.61. Para estos clientes, la estrategia debería enfocarse en potenciar su engagement y gasto en lugar de simplemente atraerlos nuevamente. Una opción sería implementar estrategias de upselling y cross-selling inteligente, recomendando productos personalizados según su historial de compras, ofreciendo descuentos en productos complementarios o promoviendo estrategias como "frecuentemente comprados juntos". Por ejemplo, si un cliente suele pedir sushi en la aplicación, podría recibir un descuento en bebidas premium o postres relacionados. Otra estrategia podría ser la gamificación y creación de experiencias, diseñando recompensas por frecuencia de compra y personalizando la comunicación para que los clientes se sientan

parte de una comunidad. Un sistema de recompensas acumulativas podría incentivar a estos usuarios a seguir comprando para alcanzar beneficios exclusivos.

Finalmente, considerando que el recall de la clase 1 no es tan alto (0.61), podría ser beneficioso diseñar una estrategia de retargeting inteligente para los falsos positivos, es decir, clientes que en realidad no pertenecían a este segmento. Si un cliente no responde a una primera promoción, se podría ofrecer un incentivo mayor en una segunda fase para confirmar si realmente tiene interés en la oferta.

La implementación de estrategias diferenciadas según la clasificación de cada grupo permitirá optimizar la efectividad de las campañas de marketing y maximizar la rentabilidad de cada segmento.

Capítulo 5: Conclusiones

En este capítulo se resumen los objetivos cumplidos, y se proponen mejoras si hubiera más tiempo para continuar con el trabajo de investigación.

5.1) Objetivos cumplidos

El objetivo principal de este trabajo fue resolver una problemática clave en las aplicaciones de delivery: identificar qué producto o promoción ofrecer a un usuario en función de su perfil y comportamiento histórico, de manera de eficientizar los costos de Marketing, ofreciendo incentivos a cierto segmento de usuarios y evitando otorgar los mismos a clientes de bajo potencial y no rentables, entre las acciones más destacadas.

Para ello, se desarrolló un modelo predictivo que predice el máximo segmento alcanzado por un usuario, permitiendo decisiones más estratégicas y efectivas.

El modelo cumple exitosamente con este objetivo, estableciendo una base sólida para llevar a cabo acciones específicas (mencionadas en la sección de recomendaciones explícitas del capítulo 4).

5.2) Posibles mejoras

A pesar de los resultados obtenidos, existen áreas claras para futuras mejoras que podrían optimizar el desempeño del modelo y la robustez de las conclusiones:

1. Mejora en el *accuracy* y exploración de modelos adicionales:
   Aunque se aplicaron técnicas como regresión logística multinomial, random forest y XGBoost con hiper parámetros optimizados, el *accuracy* podría mejorar. Una línea de trabajo futura sería probar modelos más avanzados como redes neuronales o máquinas de soporte vectorial (SVM). Esto podría complementarse con un mayor enfoque en la ingeniería de características, creando nuevas variables derivadas de las existentes que puedan capturar relaciones más complejas. Además, iterar con una gama más amplia de hiper parámetros contribuiría a un desempeño superior, superando el *benchmark* establecido en este trabajo.

2. Ampliación del histórico de datos y enriquecimiento de información:
   El análisis estuvo limitado a datos de segmento de usuario disponibles desde enero de 2024. Tener un histórico más extenso, idealmente de meses o años, permitiría ampliar el conjunto de datos, generando conclusiones

más robustas y un modelo más preciso. Además, se observó una alta proporción de valores nulos en variables clave como el género y la fecha de nacimiento. Incentivar a los usuarios a completar esta información podría mejorar significativamente la calidad de los datos, permitiendo un entendimiento más granular y predicciones más precisas.

3. Análisis y predicciones a nivel país:
   En este trabajo se tomaron todos los datos sin discriminar por país para entrenar a los modelos y realizar las predicciones, pero un punto interesante podría ser ver los análisis por país y realizar predicciones con esta granularidad. Podría derivar en decisiones más asertivas y precisas por parte del negocio. Incluso realizar pruebas A/B por país. Argentina, que es el mercado más representativo a nivel cantidad de usuarios y volumen de ventas, será el primer país elegido para probar los experimentos, y luego iremos realizando pruebas en los demás países.
   Hay ciertos productos que se consumen más en determinados países, y con esta perspectiva lograríamos identificar este valioso punto.
   Además, hay ciertos productos que se ofrecen en un país, y en otro no se ofrecen. Por ejemplo, en Chile existe la vertical llamada "Gas", mientras que dicha vertical en Argentina no se encuentra disponible.

4. Segmentación Personalizada:
   Se podría realizar una segmentación más precisa para ofrecer experiencias más diferenciadas, confiando en la estabilidad del modelo. Por ejemplo, en vez de tener la segmentación de usuarios nuevos (entre 1 y 4 órdenes) en base a los resultados, analizar si quizás es conveniente tener de 1 a 3 órdenes la categoría nuevos, y luego a partir de 4 ya tener la segmentación de la categoría onboarded (y no a partir de 5 como está definida por negocio).

Anexo I - Lista de variables del conjunto de datos

Para cada fila que tenemos disponible, las siguientes variables se encuentran disponibles:

user_id: Identificador único del usuario.

pais_registro_user: País donde se registró el usuario.

dia_max_segmento_alcanzado: Fecha en la que el usuario alcanzó su máximo segmento.

dia_semana_max_segmento_alcanzado: Día de la semana cuando se alcanzó el máximo segmento.

max_lifecycle: Segmento máximo alcanzado por el usuario.

dia_1er_compra: Fecha de la primera compra.

dia_semana_1er_compra: Día de la semana de la primera compra.

vertical_name_1er_compra: Categoría de la primera compra (e.g., restaurante o mercado).

is_restaurant_or_market_1ra: Indicador de si la primera compra fue en un restaurante o mercado.

is_pre_order_1er_compra: Indicador de si la primera compra fue un pedido anticipado.

is_pickup_1er_compra: Indicador de si la primera compra fue retirada en el local por el usuario.

country_1er_compra: País de la primera compra.

dia_2da_compra: Fecha de la segunda compra.

dia_semana_2da_compra: Día de la semana de la segunda compra.

vertical_name_2da_compra: Categoría de la segunda compra.

is_restaurant_or_market_2da: Indicador de si la segunda compra fue en un restaurante o mercado.

is_pre_order_2da_compra: Indicador de si la segunda compra fue un pedido anticipado.

is_pickup_2da_compra: Indicador de si la segunda compra fue retirada en el local por el usuario.

country_2da_compra: País de la segunda compra.

dia_3ra_compra: Fecha de la tercera compra.

dia_semana_3ra_compra: Día de la semana de la tercera compra.

vertical_name_3ra_compra: Categoría de tercera compra.

is_restaurant_or_market_3ra: Indicador de si la tercera compra fue en un restaurante o mercado.

is_pre_order_3ra_compra: Indicador de si la tercera compra fue un pedido anticipado.

is_pickup_3ra_compra: Indicador de si la tercera compra fue retirada en el local por el usuario.

country_3ra_compra: País de la tercera compra.

dia_4ta_compra: Fecha de la cuarta compra.

dia_semana_4ta_compra: Día de la semana de la cuarta compra.

vertical_name_4ta_compra: Categoría de la cuarta compra.

is_restaurant_or_market_4ta: Indicador de si la cuarta compra fue en un restaurante o mercado.

is_pre_order_4ta_compra: Indicador de si la cuarta compra fue un pedido anticipado.

is_pickup_4ta_compra: Indicador de si la cuarta compra fue retirada en el local por el usuario.

country_4ta_compra: País de la cuarta compra.

dif_dias_1ra_4ta_orden: Diferencia en días entre la primera y la cuarta compra.

dif_dias_4ta_orden_a_hoy: Diferencia en días entre la cuarta compra y la fecha actual.

dif_dias_4ta_orden_a_max_seg: Diferencia en días entre la cuarta compra y el día del máximo segmento alcanzado.

dif_dias_1er_orden_a_max_seg: Diferencia en días entre la primera compra y el día del máximo segmento alcanzado.

Anexo II - Hiper Parámetros Utilizados

1. **n_estimators**: Define el número de árboles generados o iteraciones de *boosting*. Un número alto puede mejorar la predicción, pero incrementa el riesgo de sobreajuste.
2. **learning_rate (o eta)**: Controla la magnitud con la que cada árbol contribuye a la predicción. Tasas bajas pueden requerir más árboles, pero suelen generalizar mejor.
3. **max_depth**: Establece la profundidad máxima de los árboles. Profundidades mayores capturan patrones complejos, aunque aumentan el riesgo de sobreajuste.
4. **min_child_weight**: Especifica el peso mínimo en los nodos hijos para permitir divisiones. Un valor alto previene sobre ajustes al requerir divisiones más significativas.
5. **subsample**: Controla la fracción de datos utilizada para construir cada árbol, ayudando a reducir el sobreajuste.
6. **colsample_bytree**: Define la proporción de características seleccionadas aleatoriamente en cada árbol.
7. **gamma**: Representa la reducción mínima en la función de pérdida necesaria para dividir un nodo. Un valor alto prioriza divisiones relevantes.
8. **regularización (lambda y alpha)**: Se utilizaron términos de regularización L2 (lambda) y L1 (alpha) para penalizar modelos complejos y prevenir el sobreajuste.
9. **scale_pos_weight**: Ajusta el peso de las clases en problemas de clasificación desbalanceados, mejorando la predicción en la clase minoritaria.
10. **objective**: Define la función objetivo, como 'binary:logistic' para clasificación binaria o 'multi:softmax' para tareas de múltiples clases.
11. **booster**: Especifica el algoritmo base: gbtree (árboles de decisión), gblinear (regresión lineal) o dart (árboles con *dropout*).
12. **tree_method**: Permite optimizar la construcción de árboles, destacando el método gpu_hist para aceleración en GPU, útil en grandes volúmenes de datos.
13. **objective**: En modelos de regresión logística, define la función de pérdida utilizada en la optimización. Puede ser 'binary' para clasificación binaria o 'multiclass' para clasificación de múltiples categorías.
14. **C**: Parámetro de regularización en regresión logística que controla la penalización aplicada a coeficientes grandes. Valores más pequeños aumentan la regularización.
15. **solver**: Algoritmo utilizado para la optimización en la regresión logística ('liblinear', 'saga', 'lbfgs').

Anexo III - Matriz de correlación

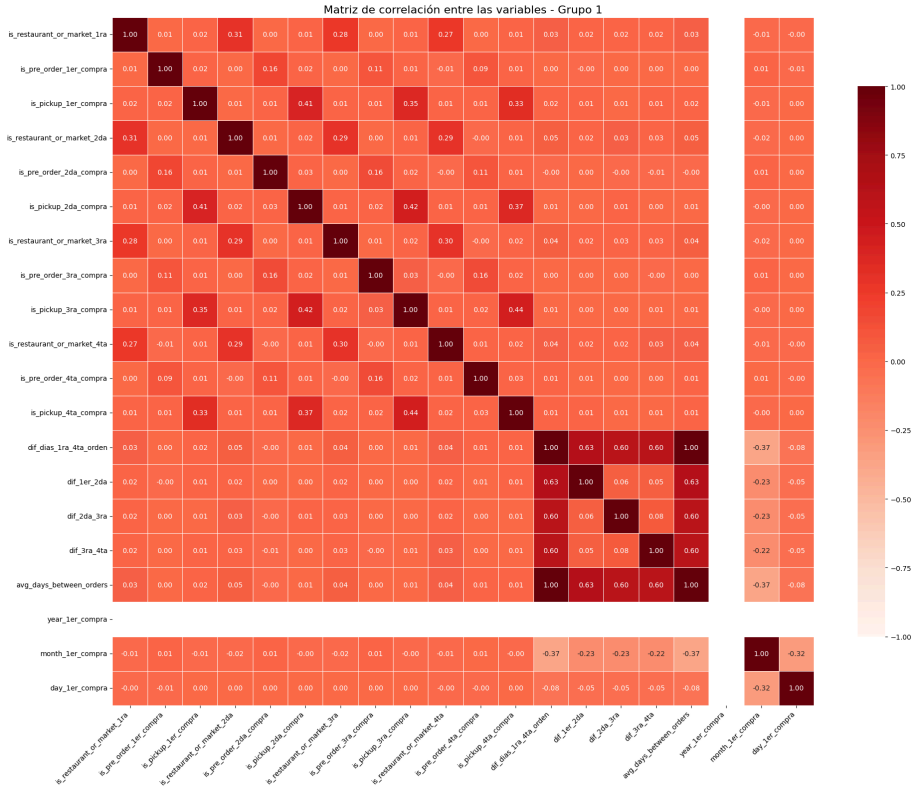

Gráfico 37 Matriz de correlación Grupo 1

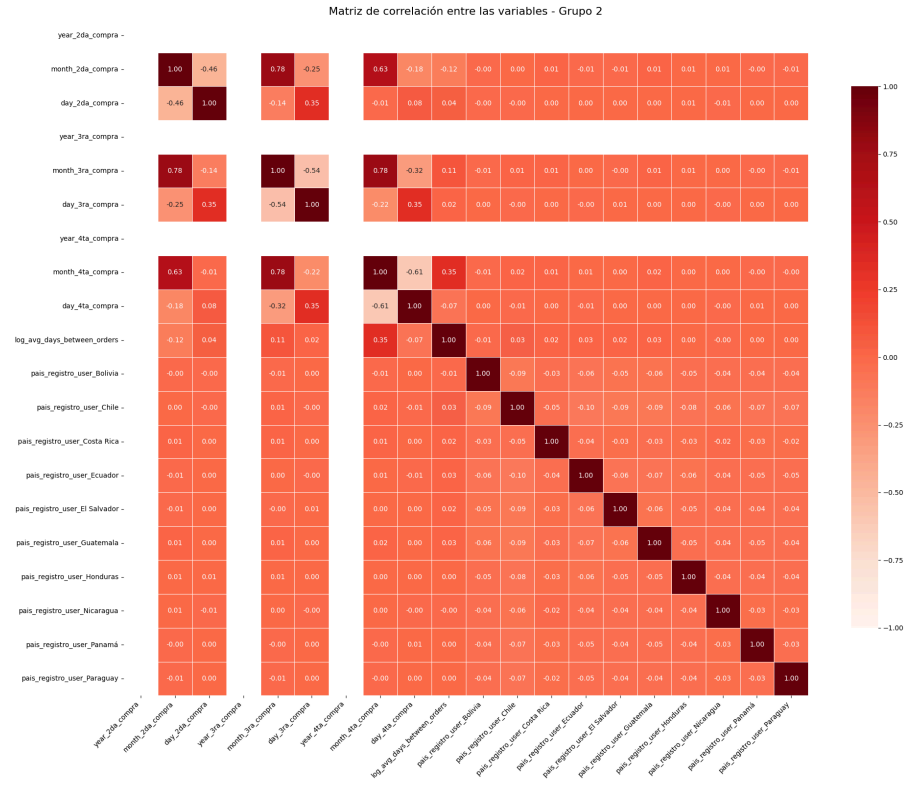

Gráfico 38 Matriz de correlación Grupo 2

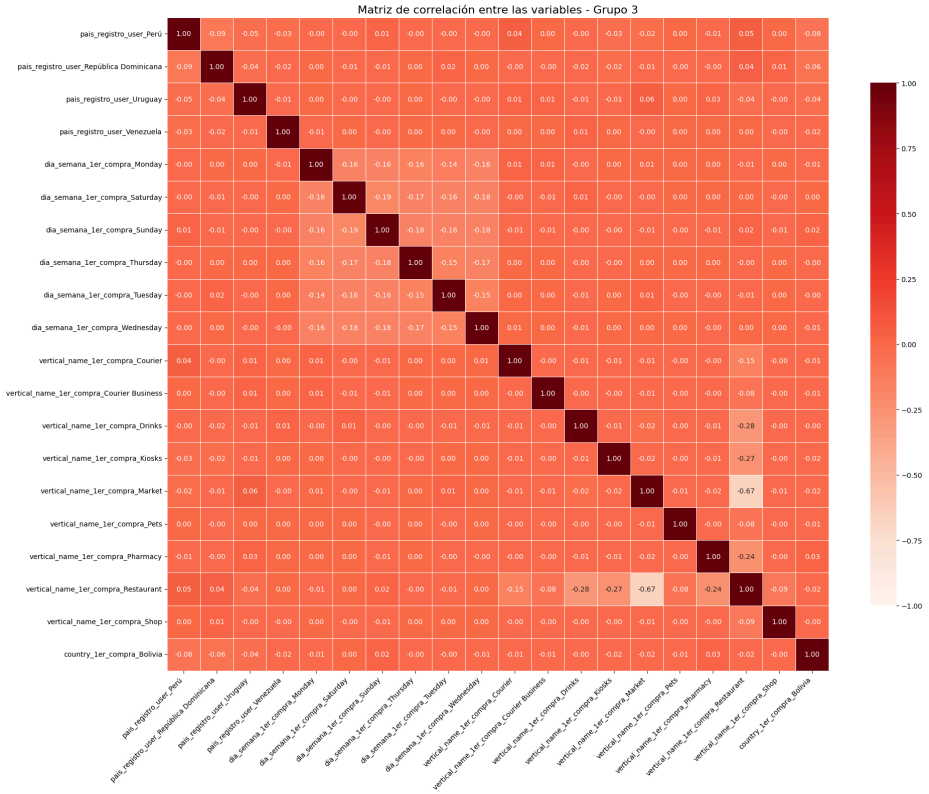

Gráfico 39 Matriz de correlación Grupo 3

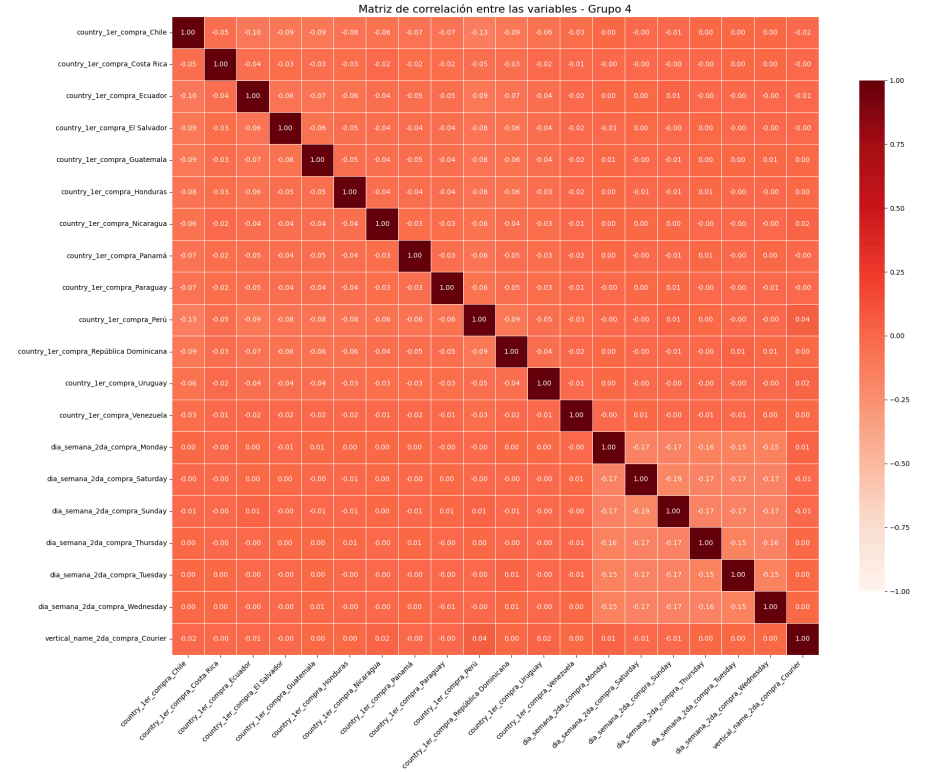

Gráfico 40 Matriz de correlación Grupo 4

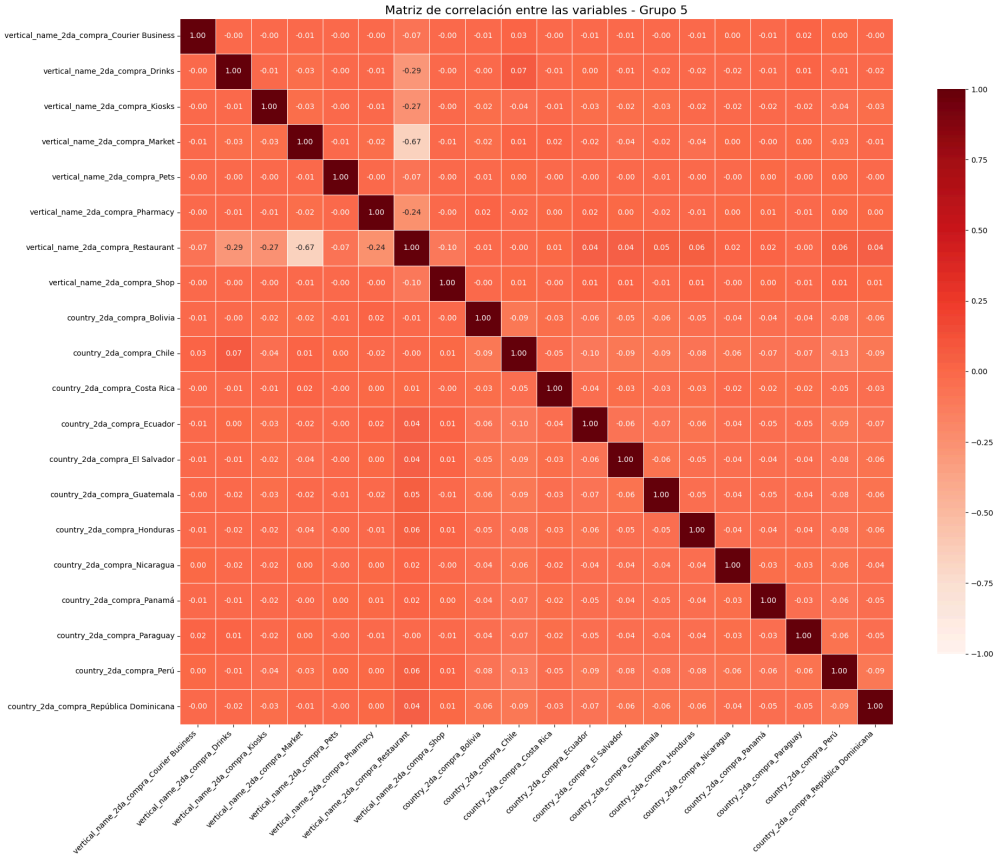

Gráfico 41 Matriz de correlación Grupo 5

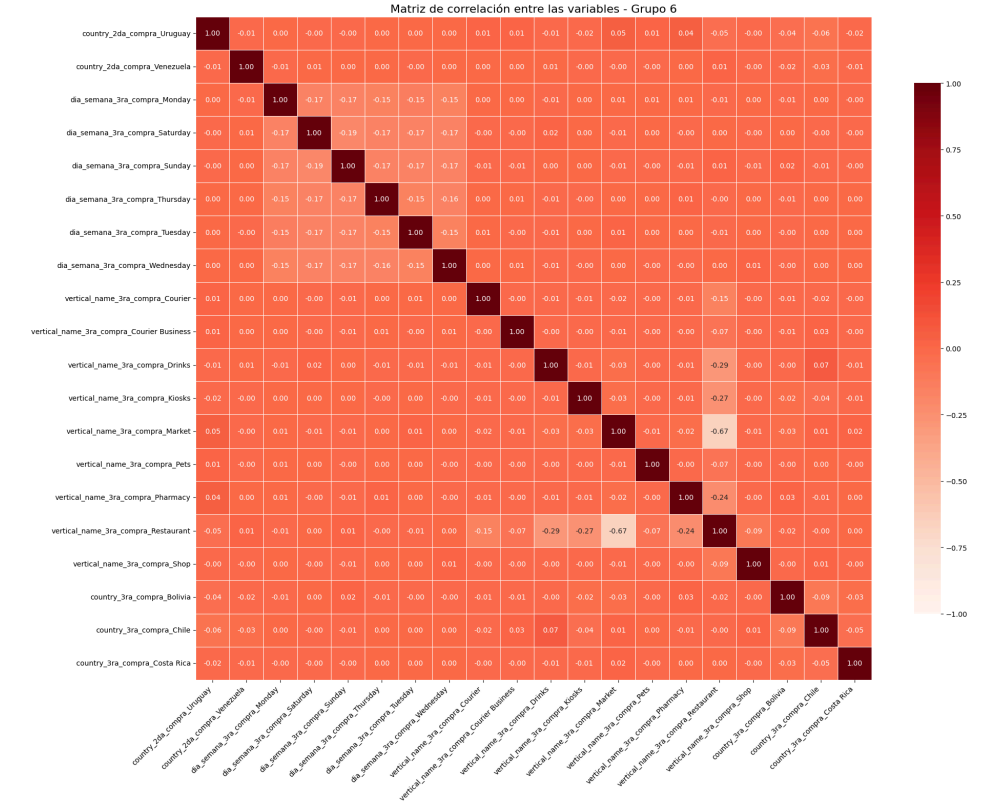

Gráfico 42 Matriz de correlación Grupo 6

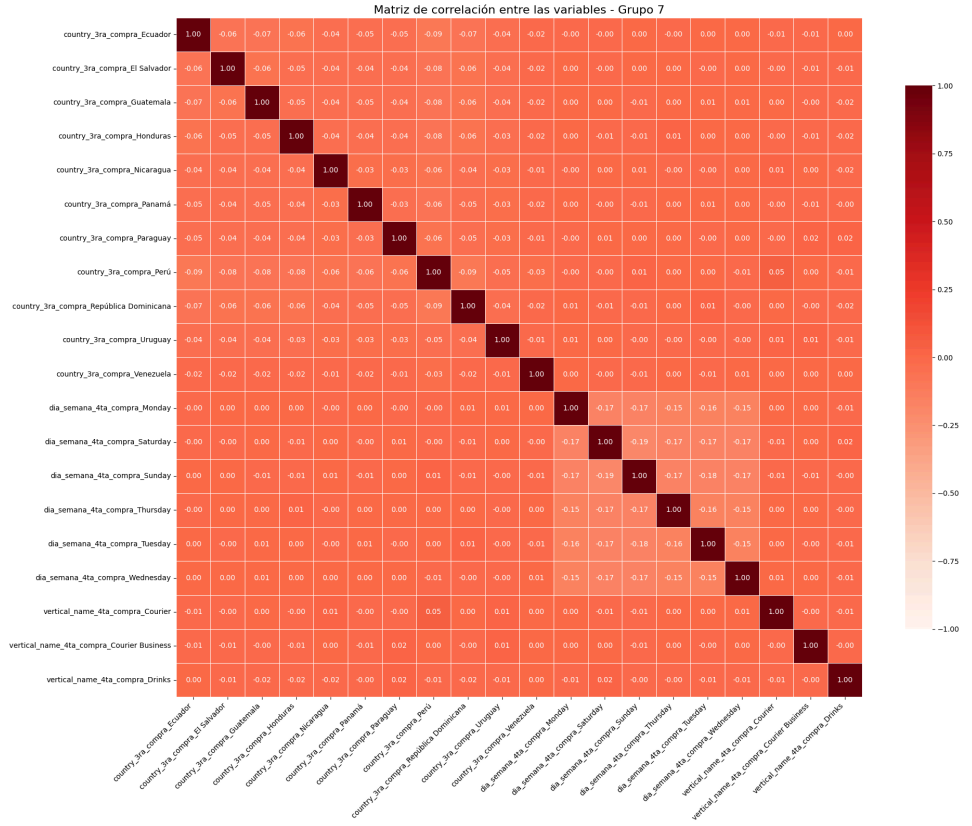

Gráfico 43 Matriz de correlación Grupo 7

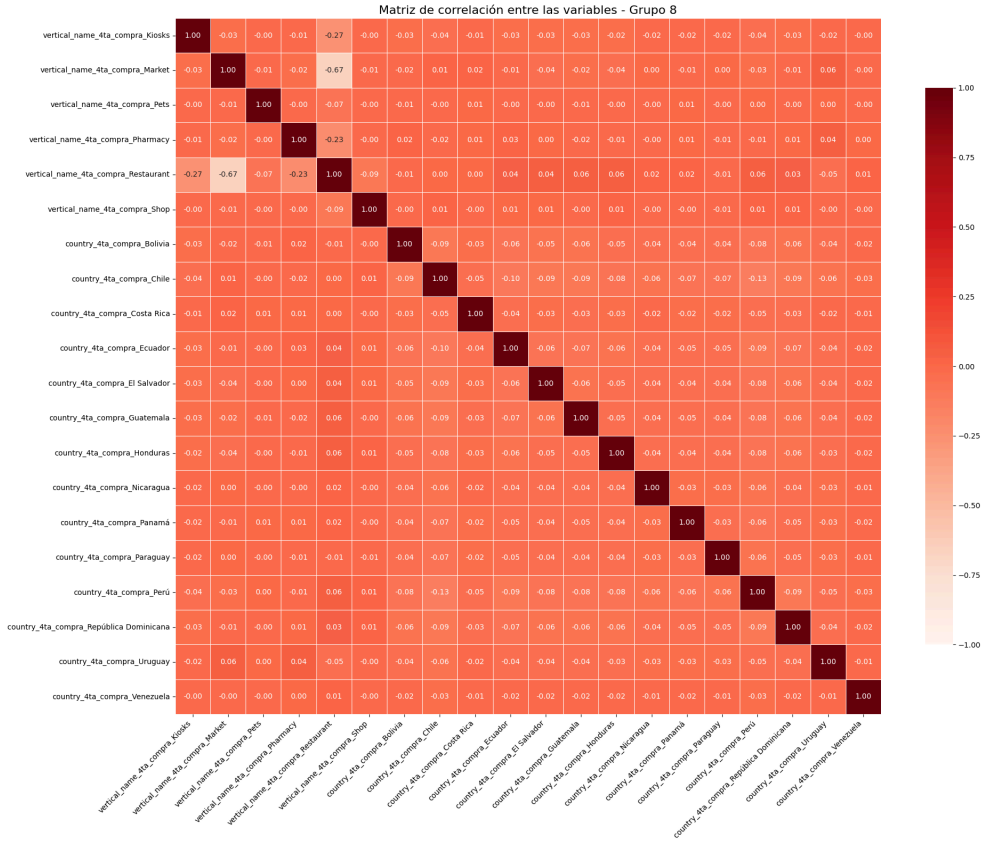

Gráfico 44 Matriz de correlación Grupo 8

Tal como podemos observar en cada matriz de correlación (está dividida en grupos debido a la cantidad de variables que se maneja), no existe una relación fuerte en ningún caso. Por ende no habría indicios de tener multicolinealidad.

Anexo IV - Código Python

Github con código de Python

Bibliografía

1. Hagiu, A. (2007) “Multi-sided platforms: From microfoundations to design and expansion strategies,” SSRN Electronic Journal [Preprint]. Available at: https://doi.org/10.2139/ssrn.955584.
2. Hagiu, A., & Wright, J. (2015). *Multi-sided platforms*. Harvard Business School Working Paper, No. 15-037. Disponible en: https://www.hbs.edu/ris/Publication%20Files/15-037_30391b66-46c5-4d02-8251-0cb2893cc5e2.pdf
3. Statista. (2023). *Quick Commerce - Argentina | Statista Market Forecast*. Recuperado de https://www.statista.com/outlook/emo/online-food-delivery/grocery-delivery/quick-commerce/argentina
4. Roychowdhury, S., Alareqi, E., & Li, W. (2021). OPAM: Online Purchasing-behavior Analysis using Machine Learning. *arXiv preprint arXiv:2102.01625.* Recuperado de https://arxiv.org/abs/2102.01625
5. NielsenIQ. (2024, September 16). *The rise of quick commerce*. Retrieved from https://www.nielseniq.com.
6. Rodríguez, S. (2023). *La fase 0 de cualquier estrategia de datos: El data cleansing*. Big Data Magazine. Recuperado de https://bigdatamagazine.es.
7. Shmueli, G., Bruce, P. C., Stephens, M. L., & Patel, N. R. (2017). *Data Mining for Business Analytics: Concepts, Techniques, and Applications with JMP Pro*. Wiley.
8. Kuhn, M., & Johnson, K. (2019). “Feature Engineering and Selection: A Practical Approach for Predictive Modeling”. CRC Press. Recuperado de http://www.feat.engineering/feature-selection
9. Synced. (2017, October 24). *How Random Forest Algorithm Works in Machine Learning*. Medium. Recuperado de https://synced.medium.com/how-random-forest-algorithm-works-in-machine-learning-3c0fe15b6674
10. Badola, S., Mishra, V. N., & Parkash, S. (2023). *Landslide susceptibility mapping using XGBoost machine learning method.* IEEE International Conference on Machine Intelligence for GeoAnalytics and Remote Sensing (MIGARS). https://doi.org/10.1109/MIGARS57353.2023.10064496.
11. Calle García, A. J., Quimis Vera, M. C., Piguave Vargas, M. T., & Zambrano Luzardo, J. S. (2024). La inteligencia artificial como herramienta en la

segmentación de mercado. Ciencia y Desarrollo, 27(1), 194-202. Recuperado de http://revistas.uap.edu.pe/ojs/index.php/CYD/index