Segmentación de clientes en E-Commerce mediante Técnicas de Aprendizaje no Supervisado

TRABAJO FINAL DE MÁSTER

Máster en Data Science y Business Analytics
Alumno: Antonio Vicente Ortega
Tutor: Abel Soriano Vázquez
Fecha: 28/01/2024

Índice

Resumen

El Comercio Electrónico ha experimentado un notable crecimiento y relevancia en el siglo XXI. La automatización de los procesos de compra y el almacenamiento masivo de datos de usuarios y clientes convierten a este sector en un entorno propicio para el desarrollo de la Ciencia de Datos y el Aprendizaje Automático. La eficacia de los métodos y la capacidad para trabajar con información proveniente de miles o millones de datos reales hacen de los Algoritmos de Clustering una herramienta de gran utilidad y eficiencia para generar segmentos de clientes e implementar estrategias adaptadas a cada tipología de consumidor.

En este estudio, se obtendrá y transformará un conjunto de datos procedente de un e-Commerce para posteriormente ejecutar Algoritmos de Clustering, comparando y analizando los resultados obtenidos en cada uno de ellos. Además, se destacarán problemáticas y factores clave a tener en cuenta durante su implementación, siendo cruciales para mejorar la eficiencia de los clústeres generados.

Abstract

E-commerce has experienced significant growth and relevance in the 21st century. The automation of purchasing processes and the massive storage of user and customer data make this sector a conducive environment for the development of Data Science and Machine Learning. The effectiveness of methods and the ability to work with information from thousands or millions of real data make Clustering Algorithms a highly useful and efficient tool for generating customer segments and implementing strategies tailored to each consumer typology.

In this study, a dataset from an e-commerce platform will be obtained and transformed for subsequent execution of Clustering Algorithms, comparing and analyzing the results obtained in each of them. Additionally, issues and key factors to consider during implementation will be highlighted, being crucial for improving the efficiency of the generated clusters.

1. Introducción y Antecedentes

1.1 E-Commerce en la actualidad

El comercio online o e-Commerce ha sido uno de los grandes avances propiciados por la consolidación de Internet en el siglo XXI. Supone la creación de una interacción directa entre consumidores y empresas sin necesidad de presencia física, establecimientos o trabajadores comerciales. Un concepto más amplio nos habla del "intercambio de bienes entre organizaciones independientes y personas usando sistemas de comunicación y una red de infraestructura estandarizada" (Turban, E., King, D., Lee, K. L., Liang, T. P., & Turban, C. T., 2015, p.7). Esta estructura está representada por la página web y todo el proceso de compra está gestionado a través de ella.

Para la empresa ha significado la disminución de ciertos costes asociados a la venta física y el crecimiento de la gestión automatizada. De hecho, aquellas compañías que no han afianzado su apuesta por el comercio electrónico y el marketing digital, terminan sufriendo desventaja competitiva respecto a las que sí desarrollan e impulsan estas áreas (Editorial Elearning, s.f.). Por otro lado, el consumidor se ha acostumbrado a la existencia de esta forma de comercio, extendiendo su uso cada año y beneficiándose de las ventajas que éste presenta.

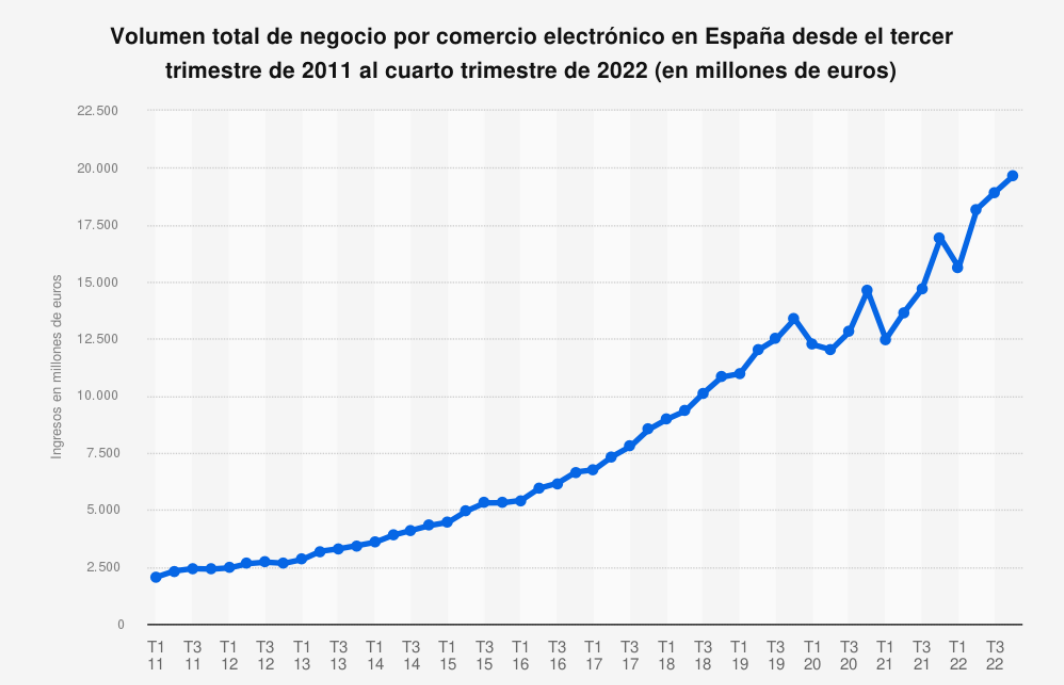


Ilustración 1. Evolución de los ingresos generados por el comercio electrónico desde el primer trimestre de 2011 hasta el cuarto trimestre de 2022. **Fuente: Comisión Nacional de los Mercados y la Competencia, (2023) y recuperada desde https://es.statista.com**

El mercado del comercio online no ha dejado de crecer y, como se observa en la Ilustración 1, en 2022 se alcanzaron unos ingresos cercarnos a los 20.000 millones de euros. De hecho, la integración de esta forma de compra en la sociedad española está completamente asentada. En 2023, el 77% de la población con acceso a internet en España, efectuó alguna compra online. Esta cifra se dispara aún más en la población de entre 25 y 34 años (Rois, 2023).

Este cambio de paradigma entre el comercio tradicional y el electrónico ha propiciado que la relación entre los clientes y las empresas se vea afectada. El cliente del e-Commerce pierde su aspecto físico y ahora es percibido como un conjunto de datos, por lo que su relación con el vendedor tiende a ser mínima. Esta pérdida de interacción con los clientes supone para las tiendas online la necesidad de trabajar para conocer a sus usuarios en profundidad y desarrollar estrategias para mejorar la fidelización, la imagen de marca, el desarrollo del mercado potencial y muchos otros factores como la revisión de precios, sistema de transportes y experiencia poscompra.

Para poder construir estos grupos o perfiles de usuarios, se necesita una gran cantidad de información personal y transaccional de los clientes. En una tienda física sería complicado recoger estos datos sin la utilización de encuestas o formularios que el cliente rellenase voluntariamente. Sin embargo, en el comercio electrónico, toda esta información queda registrada automáticamente mediante los perfiles de usuarios web de la tienda y a través de analíticas de seguimiento como las generadas por Google Analytics. En el caso de estas últimas, su registro viene precedido por la aceptación expresa de las cookies de la web, unos elementos que permiten conectarse con programas externos para volcar la información obtenida (Kütz, 2016).

Omitiendo la posibilidad de utilizar la información de Google Analytics, la empresa ya cuenta con información valiosa de sus usuarios gracias a los registros de compra y a los perfiles anteriormente mencionados, que quedan grabados dentro de la plataforma online. Algunos de estos datos hacen referencia a:

- Información personal: Nombre, apellidos y edad.

- Localización: Todos los pedidos tienen asociada una dirección de facturación y de envío que puede ser utilizada para situar al cliente geográficamente.

- Importe del pedido e importe medio por pedido en el caso de que sea un cliente recurrente.

- Categoría de producto: En función del tipo de negocio, la tienda online abarcará ramas o tipologías de producto distintas que pueden utilizarse para describir el interés o los gustos de un cliente en concreto.

- Valoraciones: Muchas páginas web permiten valorar y puntuar tanto la opinión general de la tienda como la del pedido realizado, dando mayor profundidad a la información del cliente y

permitiendo tomar acciones de mejora y fidelización. Gran mayoría de los comercios electrónicos acuden a empresas externas que actúan como verificadores de estas reseñas, otorgando mayor transparencia y credibilidad a la marca (Fraguela, 2023).

1.2 Segmentación de mercados

Una de las técnicas más utilizadas para generar grupos de clientes y conocer sus características es la segmentación de mercados. Autores como Banea Graciá definen la segmentación como "el proceso a través del cual el mercado total de un producto o servicio particular es dividido en grupos realmente homogéneos atendiendo a sus características y necesidades particulares" (Banea Graciá, 2012, p.149).

Estamos ante un método que precisa, por un lado, de un grupo de individuos susceptibles a comprar en el comercio y, por otro, de una serie de características previamente definidas con las que se puede identificar a dichos individuos. Los segmentos que se definen con ellos pueden obedecer a una serie de criterios generales o específicos, en función de si están relacionados directamente o no con el proceso de compra y producto adquirido (Santesmases Maestre, 2012). Así pues, en base a estos criterios, obtendremos que la edad, la localidad y el sexo del cliente son criterios generales, mientras que la valoración de la compra y la categoría de producto se enmarcan dentro de los criterios específicos.

En lo referente a los métodos para llevar a cabo la segmentación, Santesmases Maestre (2012) diferencia entre diseños de segmentación "a priori" y diseños de segmentación óptima, identificando los primeros con técnicas estadísticas descriptivas, como el análisis de la varianza y las tabulaciones cruzadas, que tienden a identificar las características inherentes a un grupo o segmento ya creado. Por otro lado, los diseños de segmentación óptima están representadas por técnicas estadísticas multivariantes que dividen a los individuos en distintos grupos diferenciados entre sí.

El beneficio principal de la segmentación de clientes es la información. Sin embargo, lo realmente importante es cómo usa la empresa dicha información. Además de identificar las tipologías de clientes, se pueden realizar estrategias basadas en el comportamiento del consumidor, anticipar cambios en el mercado, descubrir demandas subyacentes o verificar la fuente de ingresos potenciada por las compras de los clientes (Yankelovich y Meer, 2006).

Como se ha mencionado anteriormente, las características intrínsecas a una tienda online y su capacidad de acceder a la información, hacen del e-Commerce un ambiente muy favorable para la segmentación del mercado. Actualmente, ya es una técnica generalizada en grandes empresas como Amazon. Este marketplace se caracteriza por hacer recomendaciones de producto en

función de las compras realizadas previamente, además de tratar de posicionar artículos relacionados con los gustos del consumidor. Haciendo uso de los datos de los usuarios, Amazon afirma poder realizar campañas publicitarias más eficientes, identificar dificultades en el proceso de compra y fidelizar grupos de clientes (Amazon Ads, 2023).

Finalmente, podemos indicar que la segmentación de mercados no es una técnica moderna. Así lo demuestran estudios como el de Smith (1956) que sentaron precedente en su momento. Este autor ya hablaba de la segmentación como una forma de situar al consumidor en el centro de atención, junto a la diferenciación de productos. Sin embargo, desde entonces sí existe una evidente evolución en los métodos utilizados y la potencia de los mismos. La irrupción de la computación y el Big Data proponen soluciones más eficientes y extensivas a millones de datos.

Sobre la utilización de metodologías modernas, ya existe literatura acerca del uso de la segmentación dentro del comercio electrónico. Por ejemplo, Bohan Zhao (2022) hace ejecuta de técnicas de clustering para generar grupos de clientes y realizar recomendaciones de producto en base a las características de cada segmento.

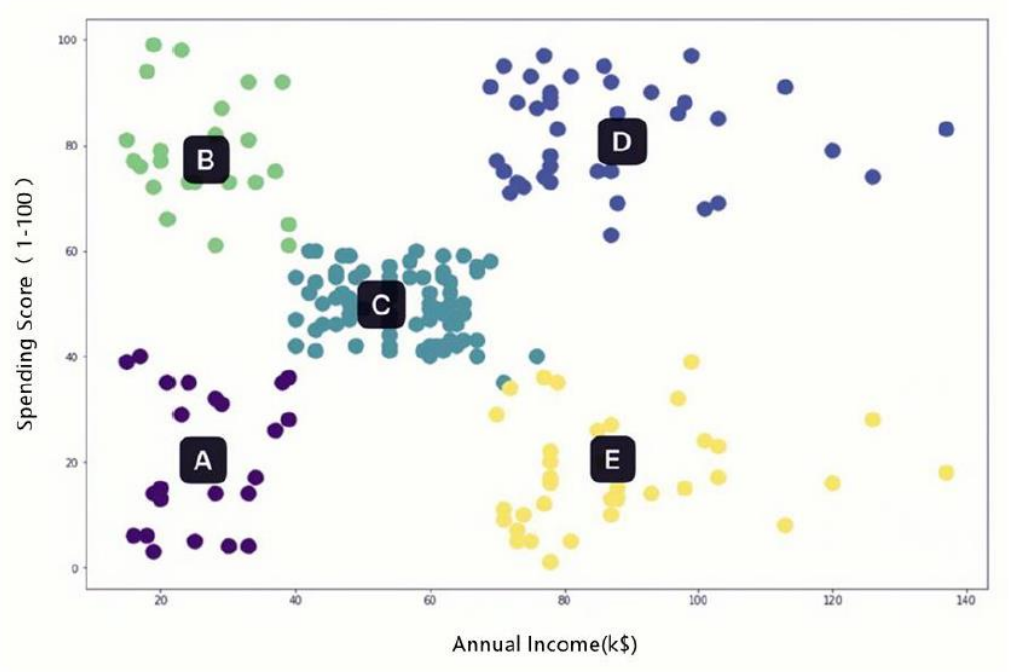


Ilustración 2. Bohan Zhao utiliza técnicas de clustering, identificando cinco grupos de clientes con diferentes ingresos anuales (anual income) y potencial de compra (spending score). Destaca el grupo D, con altos ingresos y mayor puntuación de potencial de compra que otros grupos. En base a estos segmentos, la empresa podrá tomar decisiones sobre qué productos mostrar y recomendar a cada usuario. **Fuente: Zhao, Bohan. (2022, p.3020). Recuperado en: https://www.atlantis-press.com/article/125971546.pdf.**

Como se observa en la Ilustración 2, el autor hace uso de un algoritmo de clustering, una técnica de Machine Learning de aprendizaje no supervisado. En el próximo epígrafe, se detallará cómo se utilizan las técnicas de aprendizaje no supervisado para generar grupos con características

similares y evidenciar así la importancia de la segmentación dentro de un sector como el comercio electrónico.

1.3 Machine Learning: Técnicas de aprendizaje no supervisado para la segmentación de mercados

El desarrollo de la ciencia computacional ha llevado al ser humano a investigar formas de automatizar tareas, crear procesos más eficientes y aumentar tanto el alcance como la potencia de las soluciones propuestas. El Machine Learning o Aprendizaje Automático se establece como "el uso de programas informáticos que mejoran su rendimiento mediante la experiencia" (Mitchell, 1997, p.2). Sin embargo, no podemos olvidar que el fin del Aprendizaje Automático debería ser obtener información que no sería posible extraer por otros medios, cuya utilización suponga el uso de excesivos recursos y cuya solución no resulte tan óptima como la propuesta por el algoritmo.

El Aprendizaje Automático está influenciado por distintas ramas como la Inteligencia Artificial, la Estadística o la informática, haciendo uso de todas ellas para llevar a cabo su funcionamiento, por lo que muchos autores hablan también de "Aprendizaje Estadístico" y "Analítica Predictiva" (Müller y Guido, 2017). El desempeño del aprendizaje se basa, por lo tanto, en la utilización de algoritmos o programas informáticos capaces de encontrar una solución a un problema propuesto y mejorando su respuesta con cada ejecución, al ser capaces de optimizar su ejecución y evaluar la eficiencia mediante indicadores estadísticos del rendimiento.

La literatura hace referencia dos formas distintas de aprendizaje: El aprendizaje supervisado y el aprendizaje no supervisado. El primero requiere la introducción de una variable objetivo independiente, en base a la cual el algoritmo aprende y ejecuta su respuesta en los datos proporcionados. Por el contrario, el aprendizaje no supervisado se centra en las características del conjunto de los datos y no requiere de una variable externa o modelo alguno para su funcionamiento (Hastie, T., et Al., 2001). Este método se basa en la transformación de los datos proporcionados para poder obtener información subyacente y difícil de conseguir con su formato inicial.

Centrando la atención en la segmentación de mercados para un negocio como el comercio electrónico, hacíamos referencia anteriormente al uso del clustering. Este se considera un método de aprendizaje automático no supervisado ya que utiliza algoritmos que dividen el conjunto de datos inicial en distintos grupos con características representativas y distintas entre sí. Utiliza, por lo tanto, las variables proporcionadas para encontrar rasgos comunes a varios sujetos que puedan formar un segmento único. Cuando se introducen nuevos datos, estos algoritmos utilizan la

experiencia y características generadas en ejecuciones anteriores para clasificar los valores, siendo esta la forma con la que el programa aprende (Mahesh, 2020).

Dentro de las técnicas de clustering, encontramos diferentes algoritmos cuyo funcionamiento y eficiencia difiere en función del objetivo propuesto, la calidad y tipología de los datos proporcionados. Los algoritmos más utilizados son los Combinacionales, K-Medias, Cuantización de Vectores, K-Medianas, Clustering Jerárquico Aglomerativo o Clustering Divisivo (Hastie, T., et Al., 2001).

En este estudio, planteamos la incógnita de cuáles de los algoritmos mencionados anteriormente pueden resultar más eficientes para generar segmentos, basándonos en la base de datos de un comercio electrónico en la cual se han registrado pedidos con características similares en cuanto a precio, tipo de producto, región o valoración entre otros. La utilidad principal de estos objetivos nos permitirá participar en la toma de decisiones estratégicas de la empresa, al contar con diferentes grupos objetivos a los que atacar mediante acciones de marketing.

2. Objetivos

El presente estudio de "Segmentación de clientes en E-commerce mediante técnicas de Aprendizaje no supervisado" tiene como fin los siguientes objetivos:

1. Modelar y explorar una base de datos de pedidos de un comercio electrónico con el propósito de generar un dataset que contenga variables concretas, relevantes y susceptibles de ser utilizadas en la clasificación de clientes en grupos independientes.
2. Analizar la eficacia de algoritmos de aprendizaje no supervisado para la creación de clústeres o segmentos de clientes, evaluando su idoneidad y rendimiento en el contexto del comercio electrónico. Se utilizarán, para ello, cuatro algoritmos diferentes: K-Means, K-Medoides, Jerárquico Aglomerativo y DBSCAN.
3. Fomentar la integración de tecnologías de Big Data y, específicamente, técnicas de Aprendizaje Automático en el ámbito del Comercio Electrónico, destacando la importancia de la accesibilidad y el aprovechamiento de los datos por parte de la empresa.
4. Evaluar los factores que influyen a la hora de ejecutar algoritmos de clustering así como las soluciones implementadas para incrementar su eficacia.
5. Se detallarán las dificultades específicas encontradas al trabajar con datos reales de comercio electrónico, como la variabilidad en los comportamientos de compra y la presencia de valores atípicos. Además, se propondrán estrategias específicas para abordar estas dificultades y optimizar el proceso de aprendizaje automático.

3. Material y Metodología

3.1 Fuente de datos

Hay diversos factores a tener en cuenta a la hora de escoger la fuente de datos sobre la que realizar un estudio de segmentación de clientes:

- Accesibilidad de los datos: Como se destacó en la sección de Introducción y Antecedentes, el comercio electrónico dispone de diversas fuentes susceptibles de análisis. Una opción comúnmente utilizada es Google Analytics, una plataforma de analítica web proporcionada por el gigante estadounidense Google. Esta herramienta, conectada directamente a la página web a través de un script de seguimiento, recopila datos en tiempo real y genera informes detallados sobre visualizaciones, interacciones, visitas y compras. Sin embargo, es importante señalar que los informes generados por Google Analytics en formato CSV presentan limitaciones en cuanto a filas y carecen de elementos comunes que faciliten la fusión de tablas. La forma más eficiente de acceder a la totalidad de los datos implica la conexión de la cuenta de Google Analytics a otros servicios de Google, como Google Cloud y BigQuery, que sirven como base de almacenamiento y herramienta de consulta y extracción, respectivamente. El acceso a estos servicios conlleva el pago de tarifas mensuales y requiere disponer de una cuenta de Google Analytics con un volumen suficientemente amplio para generar un dataset robusto.

  En este contexto, se optó por utilizar una fuente de datos pública y ampliamente reconocida en el ámbito del Big Data, como Kaggle. Esta plataforma alberga miles de datasets, tanto reales como ficticios, diseñados para prácticas y estudios mediante minería de datos y aprendizaje automático. Entre las opciones disponibles, se seleccionó el conjunto de datos "Brazilian E-Commerce Public Dataset by Olist", accesible en https://www.kaggle.com/datasets/olistbr/brazilian-ecommerce. Este conjunto consta de varios archivos CSV con un formato muy similar al que se obtendría de un comercio electrónico real. Es relevante recordar que la mayoría de las tiendas en línea están desarrolladas en sistemas de gestión de contenidos (CMS) orientados al comercio electrónico. Uno de los más potentes y utilizados es PrestaShop, cuyo lenguaje es PHP y utiliza MySQL para la gestión de datos. Esto implica que se pueden realizar consultas SQL de manera sencilla para obtener información sobre pedidos, productos o categorías, permitiendo la fusión y modelado de datos para obtener un conocimiento más profundo.

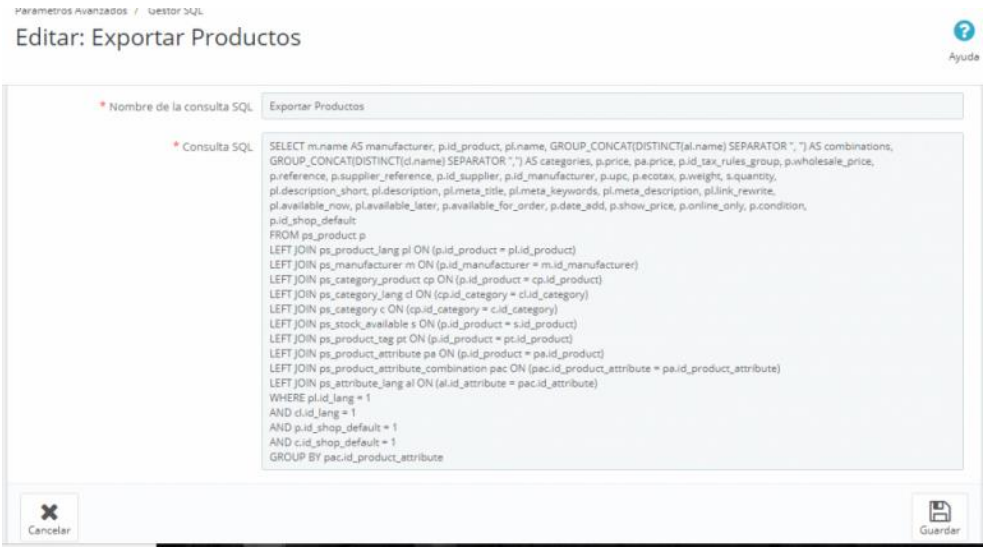


Ilustración 3. Consulta SQL para exportar productos y atributos de producto en PrestaShop 1.7. Imagen extraída del blog de Víctor Ródenas, uno de los grandes divulgadores de PrestaShop en España. **Fuente: Ródenas Gascó, V. J. (2015). Recuperado de https://victor-rodenas.com/2015/12/20/consulta-sql-para-exportar-productos-con-atributos-en-prestashop/**

Al igual que en el caso de varias consultas SQL, el conjunto de datos público 'Brazilian E-Commerce Public Dataset by Olist' se compone de diversos archivos relacionados con pedidos, clientes, pagos, valoraciones y productos. Cada uno de estos archivos incluye identificadores únicos (ID) que posibilitan la fusión y modelado de datos para crear un dataset integral. En el siguiente subepígrafe, se detallarán estos archivos y el contexto del conjunto.

- Volumen y significancia de los datos: Para llevar a cabo el estudio de segmentación de clientes, necesitamos tener acceso a un registro amplio que permita dar consistencia a los resultados obtenidos y sustentar así las recomendaciones propuestas en base a los mismos. Como se ha mencionado anteriormente, la descarga de informes de Google Analytics se encontraría limitada a cierto número de registros insuficientes para el estudio e inconexos entre sí. La elección de Kaggle permite hacer uso de una base de datos muy amplia (más de 100.000 filas) y amigable al uso.

  Por otro lado, dicha elección se encuentra también sustentada por la tipología del dato y la calidad de la información que nos proporcionan, encontrando valores muy interesantes, como los relativos a la puntuación del producto, el nombre de la categoría del artículo, el importe del pedido o la localización del cliente.

3.2 Contexto y descripción del conjunto de datos

El conjunto de datos escogido está suministrado por Olist, una plataforma que conecta el inventario de negocios brasileños, tanto en línea como en plataformas de gestión de recursos, con marketplaces de renombre como Amazon y Carrefour, facilitando la venta directa al cliente final. La información recopilada corresponde a pedidos anonimizados realizados en cada una de estas plataformas de venta.

El conjunto de datos consta de 9 archivos en formato CSV, que abarcan un total de 52 columnas distintas y más de 100,000 filas. Estos archivos están interrelacionados mediante identificadores (ID), siguiendo la estructura detallada a continuación:

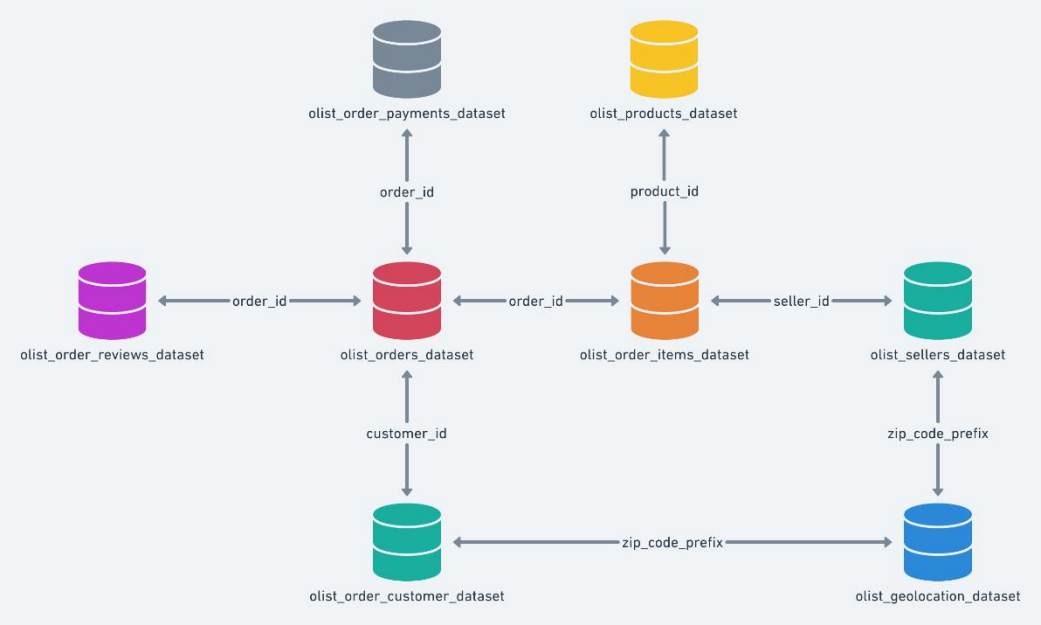


Ilustración 4. Esquema y relación de los datos proporcionados. **Fuente: Sionek, A. (2017). Recuperado desde: https://www.kaggle.com/datasets/olistbr/brazilian-ecommerce**

- Category_name_translation: Este archivo opera como un diccionario para la traducción de los nombres de las categorías de productos, estableciendo la relación entre cada nombre en portugués y su equivalente en inglés.
- Customers: Relaciona el ID del cliente con datos personales, como la ciudad, código postal y estado o provincia del cliente.
- Geolocation: Establece la relación entre códigos postales, su correspondiente latitud, longitud y estado o provincia.
- Order_items: Asocia la ID del pedido con la ID del artículo, ID del vendedor, ID del producto, fecha límite de envío, precio e impuestos soportados.
- Order_payments: Relación entre el ID del pedido, los importes pagados, los métodos de pago utilizados y la secuencia de pagos.

- Order_reviews: Este archivo vincula la ID del pedido con su respectiva valoración (puntuada del 1 al 5), el comentario del pedido y la fecha de la valoración.
- Orders: Representa cada uno de los pedidos, identificados por un ID, relacionándolos con su respectivo ID de cliente y otros datos relevantes, como el estado del pedido, fecha de compra, fecha de aprobación, fecha de entrega y fecha estimada de entrega.
- Products: Relaciona la ID del producto con datos descriptivos, como el nombre de la categoría, longitud del nombre del artículo, número de imágenes, longitud de la descripción y medidas del producto (peso, longitud, altura y profundidad).
- Sellers: Este archivo ofrece información sobre cada vendedor en los marketplaces, identificados por ID, incluyendo el código postal, ciudad y estado o provincia desde el cual operan.

En las secciones posteriores, se detallará la fusión de los archivos mencionados, la creación del dataframe objetivo, así como la descripción y análisis de los datos recopilados en el mismo.

3.3 Python y librerías para Ciencia de Datos

Python es, sin duda, uno de los lenguajes más utilizados en Ciencia de Datos, gracias a su fácil aprendizaje, al respaldo de una comunidad extensa y a la gratuidad de sus potentes librerías, capaces de abordar problemas de gran complejidad, como los planteados por el Machine Learning. (Terra, 2023).

Por todas estas razones, Python se presenta como el lenguaje ideal para poner a prueba diferentes algoritmos de aprendizaje automático, especialmente en la segmentación de clientes. Además, cuenta con librerías diseñadas específicamente para este propósito, como es el caso de Scikit-Learn. Esta librería de código abierto ofrece diversas soluciones tanto para el aprendizaje supervisado como para el no supervisado, además de una gran capacidad para preprocesar datos y evaluar modelos (Scikit-Learn, 2024).

Adicionalmente, es crucial destacar la amplia diversidad de opciones disponibles para visualizar y representar datos, algo indispensable en Ciencia de Datos. Ya sea para describir nuestras variables o mostrar información subyacente que no se percibe a simple vista, diversas librerías como Matplotlib, Seaborn, Plotly o ggplot2 posibilitan dar vida a nuestros datos mediante gráficos, mapas y representaciones.

Estas razones, junto con la posibilidad de recurrir a la comunidad para investigar soluciones y ejemplos, hacen de Python y sus librerías la elección ideal para este proyecto. En los próximos subepígrafes se detallará el proceso y los resultados obtenidos mediante el uso de estas herramientas.

3.4 Power BI

Power BI es una herramienta de análisis de datos que facilita la carga, transformación y visualización de conjuntos de datos, permitiendo generar información efectiva y de calidad. Aunque nos enfocaremos principalmente en el uso de código con Python, Power BI puede ser un soporte eficiente para desplegar gráficos y visualizaciones de manera más rápida y concluyente que mediante código. En particular, destacamos su eficacia al trabajar con mapas, geolocalización y puntos, como exploraremos más adelante.

A lo largo de este trabajo, su utilización será más bien anecdótica. No obstante, se empleará como apoyo en aquellas visualizaciones que requieran codificación especialmente compleja o de gran carga computacional.

4. Resultados

4.1 Fusión de datos y obtención del dataset objetivo

Como se ha mencionado en el epígrafe anterior, el conjunto de datos escogido cuenta con 9 archivos distintos y una amplia variedad de columnas que nos aportan información acerca de los pedidos generados online. Debemos seleccionar qué variables nos interesan de cara al estudio, además de preparar y analizar el dataset base.

Tras cargar los diferentes archivos, optamos por unir y seleccionar columnas utilizando las variables identificativas (ID) de pedidos, productos o reseñas. Así, por ejemplo, utilizamos la función 'merge' en Python para unir la tabla 'customers' con la tabla 'orders':

```
df_0 = pd.merge(customers,orders, on="customer_id", how="inner")
```

Siguiendo este procedimiento, detectamos que aumenta el número de filas. Esto se produce principalmente por la duplicidad de algunos valores, en concreto las ID de pedidos. Analizando detenidamente la incidencia, descubrimos que algunos pedidos están duplicados pero el método de pago varía. Esto se debe a que algunos clientes han utilizado varios métodos de pago dentro de un mismo pedido.

Para visualizar este problema, usamos el siguiente código, que nos muestra todas las filas duplicadas ordenadas por ID:

```
duplicados_order_payments = order_payments[order_payments['order_id'].duplicated(keep=False)].sort_values(by='order_id')

duplicados_order_payments.head()
```

	order_id	payment_sequential	payment_type	payment_installments	payment_value
80856	0016dfedd97fc2950e388d2971d718c7	2	voucher	1	17.92
89575	0016dfedd97fc2950e388d2971d718c7	1	credit_card	5	52.63
20036	002f19a65a2ddd70a090297872e6d64e	1	voucher	1	44.11
98894	002f19a65a2ddd70a090297872e6d64e	2	voucher	1	33.18
30155	0071ee2429bc1efdc43aa3e073a5290e	2	voucher	1	92.44

Ilustración 5. Pedidos duplicados ordenados por ID. **Fuente: Propia**

En la Ilustración 5, se aprecian pedidos duplicados con diferentes métodos de pago y distintas secuencias de pago. Esto confirma nuestra hipótesis acerca del uso de distintos métodos de pago dentro de un mismo pedido. Tomamos la decisión de agrupar los pedidos duplicados en

un método de pago que los englobe llamado “Varios”. De esta forma, garantizamos la coherencia del dataset y conseguimos eliminar las filas sobrantes.

De forma similar, encontramos también duplicidades para pedidos en los que se compran varios productos. En esos casos, se genera un registro por cada producto adquirido. Para esta particularidad, tomamos la decisión de mantener estas filas, al no existir un criterio claro que justifique la eliminación de unos pedidos y la prevalencia de otros, más allá de tratarse del mismo ID. Se trata de productos distintos, con diferentes categorías, precios y valoraciones. Por otro lado, contiene 23784 entradas, por lo que su eliminación podría sesgar el estudio al representar cerca de un 25% del total de los datos.

customer_id	0	customer_id	97916
customer_zip_code_prefix	0	customer_zip_code_prefix	14955
customer_city	0	customer_city	4108
customer_state	0	customer_state	27
order_id	0	order_id	97916
price	0	price	5948
freight_value	0	freight_value	6976
payment_type	0	payment_type	5
payment_value	0	payment_value	27474
product_id	0	product_id	32789
product_category_name	1598	product_category_name	73
order_status	0	order_status	7
order_purchase_timestamp	0	order_purchase_timestamp	97370
order_approved_at	15	order_approved_at	89533
review_id	0	review_id	97708
review_score	0	review_score	5
year	0	year	3
month	0	month	12
day_of_week	0	day_of_week	7
hour	0	hour	24
dtype: int64		dtype: int64	

Ilustración 6. Distribución de valores nulos (izquierda) y valores únicos (derecha) en las columnas del dataframe inicial. **Fuente: Propia**

Finalizada la fusión de las tablas, nos decantamos por explorar el conjunto de datos obtenido, transformando las variables en función de su tipología ya sean numéricas, categóricas decimales o fechas. Por otro lado, contamos con una variable fecha en formato ‘yyyy-mm-dd HH:mm:ss’. Para que los algoritmos sean capaces de procesar estos valores, debemos transformar esta variable en numérica. Para ello, separamos la fecha en cuatro columnas numéricas: ‘year’, ‘month’, ‘day’ y ‘hour’.

Tal y como refleja la Ilustración 6, procedemos finalmente a eliminar 1613 filas con valores nulos, además de las variables identificativas (ID) que no aportan valor al estudio. Tras este paso, el dataframe resultante se ha reducido significativamente, constando de 110757 entradas y 14 columnas únicas y con información relevante a los pedidos realizados. A continuación, se remite una breve descripción de cada una de las variables presentes en el dataframe final,

así como la relevancia que tienen de cara a la segmentación de los clientes del e-Commerce en grupos:

- Customer_zip_code_prefix: Variable categórica que representa el código postal del cliente. Hay 14955 valores únicos por lo que la información es extensa y muy dispersa.
- Ciudad del cliente: Esta es una variable categórica que abarca 4108 ciudades distintas. Será necesario emplear 'label encoding' para asignar una variable numérica a cada ciudad, ya que algunos algoritmos solo pueden interpretar variables numéricas. Esta técnica convierte las cadenas de texto en números enteros. Sin embargo, es importante destacar que podría ocasionar problemas en la estimación de valores para ciertos algoritmos de aprendizaje automático, ya que podrían malinterpretar que un número mayor indica un valor asociado más grande. En otras palabras, si no existe un orden de preferencia inherente, 'label encoder' podría no ser la solución más adecuada. Como alternativa, surge la técnica 'one-hot encoding', que convierte cada valor en una variable binaria. Esta variable toma el valor de 1 si pertenece a la categoría en cuestión y 0 si no pertenece (Raschka, S., & Mirjalili, V., 2017). En variables como la ciudad del cliente o el código postal antes visto, esto podría ser un problema, ya que crearía una gran cantidad de variables adicionales, generando dispersión y empeorando el rendimiento de los algoritmos (Géron, A., 2019).

  Existen varias alternativas que estudiaremos en epígrafes posteriores, por ejemplo, considerar la eliminación de la variable 'ciudad' y usar en su lugar la variable 'estado', la cual cuenta con tan solo 27 valores.
- Customer_state: Variable categórica con 27 estados diferentes. Podemos usar label-encoding o one-hot-encoding y ver la diferencia entre uno u otro. Al no existir superioridad entre los diferentes valores, sería más conveniente utilizar one-hot-encoding. Sin embargo, esto causaría la creación múltiples nuevas variables.
- Price: Variable numérica que representa el precio del pedido sin incluir el porte.
- Freight_value: Variable numérica que representa el precio del porte de cada pedido.
- Payment_type: Variable categórica que representa el tipo de método de pago. Será necesario realizar label-encoding para realizar los análisis.
- Payment_value: Se trata de una variable numérica que representa el valor total del pedido, siendo la suma del valor del artículo y del porte.
- Product_category_name: Variable categórica que representa a los 73 tipos de producto que se pueden comprar en la tienda online. Será necesario convertirla en numérica mediante label-encoding.

- Order_status: Variable categórica que representa los 7 estados en los que se puede encontrar un pedido. Habrá que realizar label-encoding ya que sí existe cierto orden o jerarquía entre ellos.
- Year (variable categórica del año).
- Month (variable categórica del mes).
- Day_of_week (variable categórica del día de la semana).
- Hour (variable categórica de la hora).
- Review_score: Esta variable categórica representa la nota que ha puntuado el cliente en cada pedido.

Tras esta breve descripción de las categorías, podemos procedemos a inspeccionar de forma superficial parte de la información que se puede extraer con el análisis exploratorio (EDA).

4.2 Análisis exploratorio de los datos (EDA)

4.2.1 Localización del cliente

Hay diversas cuestiones que podemos plantear con el análisis exploratorio del dataset del e-Commerce. Dado que contamos con las columnas de 'Código Postal', 'Ciudad' y 'Estado', podemos comenzar por tener una referencia geográfica de la localización de los clientes.

Empezando por el Código Postal, empezamos por analizar las frecuencias obtenidas en el conjunto de datos. Dado que contamos con 14955 códigos postales distintos y más de 100.000 entradas, podemos optar por mostrar los 10 valores más comunes, junto al recuento del resto de códigos postales menos frecuentes:

22790	149
22793	148
24220	141
24230	134
22775	122
35162	111
29101	106
11740	106
13087	104
22631	102
Otros	109534

Ilustración 7. Recuento de los códigos postales más comunes. **Fuente: Propia**

Como se puede observar en la Ilustración 7, el recuento de los 10 valores más comunes indica gran diversidad en los datos, ya que no existe un valor que destaque claramente sobre los otros, estando concentradas la mayoría de los clientes dentro del grupo "Otros". En el caso de mostrar

visualmente los datos, obtendríamos una gran representación de este grupo en comparación con el resto de códigos postales. Esta gran disparidad demuestra la importancia de analizar correctamente las variables de localización, por lo que investigaremos cada una de ellas de manera detallada y tomaremos decisiones en función de los resultados obtenidos.

Si nos centramos en la variable 'Estado', encontramos menor cantidad de valores únicos, con un total de 27. Realizando la misma operación, podemos obtener la distribución de los clientes por Estado.

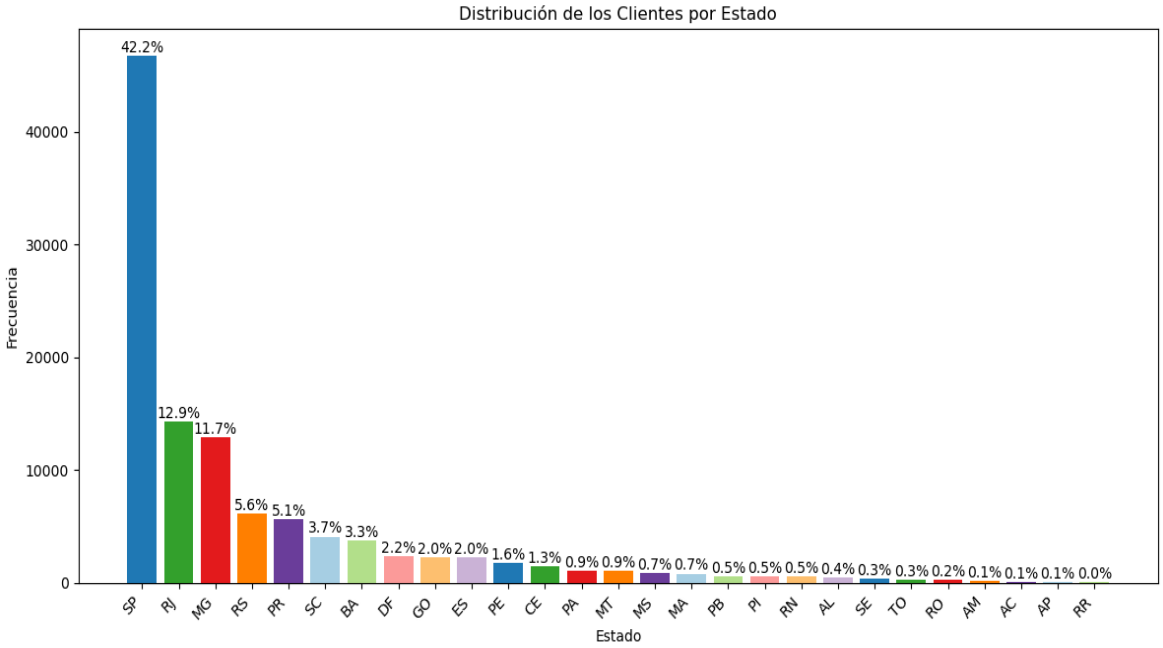


Ilustración 8. Distribución de los Clientes por Estado. **Fuente: Propia.**

En este caso, se observa una clara concentración de clientes procedentes de Sao Paulo (SP), Río de Janeiro (RJ) y Minas Gerais (MG). De hecho, se aprecia que más del 40% de los clientes provienen de Sao Paulo.

Esta información resulta relevante para la empresa, ya que permite llevar a cabo campañas de marketing focalizadas en aquellos estados con menor densidad de pedidos. También se puede interpretar que existe una mayor densidad de población en Sao Paulo y Río de Janeiro, lo que explica la mayor concentración de clientes. Sin embargo, hay un dato evidente: la prevalencia de Sao Paulo como el origen prototípico del usuario del comercio electrónico.

Dado que también contamos con la variable 'Ciudad', existen dos operaciones que podrían resultar de interés. En primer lugar, conocer el recuento de pedidos por ciudad nos permitirá identificar las principales ciudades donde se encuentran los clientes del e-Commerce. En segundo lugar, geolocalizar estos valores a través de un mapa de calor o 'heatmap', en el cual se pueda apreciar una mayor densidad cuanta mayor sea la frecuencia de pedidos en cada localización. Esta última tarea puede llevarse a cabo en Python mediante el uso de librerías como 'Geopandas'. Para ello,

utilizaríamos el archivo del conjunto de datos 'geolocation' que incluye la latitud y longitud de las ciudades para ubicar cada ciudad en su posición real dentro del mapa.

Sin embargo, debido al volumen de nuestros datos, se requiere un equipo potente para renderizar la información y ajustar el aspecto visual hasta lograr el formato deseado. En este contexto, surgen soluciones alternativas como Power BI, un servicio de Microsoft que facilita la carga de nuestro conjunto de datos y la representación fácil del recuento de pedidos por cada ciudad sin la necesidad de emplear código, y en un tiempo considerablemente menor. Para llevar a cabo este proceso, seguimos los siguientes pasos:

1. Fusionamos las columnas que contienen la información de latitud y longitud de cada ciudad, presentes en el archivo 'geolocation', con el dataframe bajo estudio. Agregamos otra columna que indique la frecuencia de pedidos en cada ciudad. Posteriormente, eliminamos duplicados y descartamos variables que no resulten relevantes, conservando únicamente aquellas relacionadas con las ciudades, las frecuencias de pedidos, la latitud y la longitud.

2. Exportamos el dataframe en formato xlsx para su posterior carga en Power BI.

3. En Power BI, utilizamos el objeto visual 'Mapa' disponible en la biblioteca de la plataforma.

4. Ajustamos cada columna del dataframe con el campo correspondiente en la configuración del objeto. Es decir, asignamos la frecuencia de pedidos a la variable 'Tamaño de Burbuja' y la latitud y longitud en sus respectivas secciones.

5. Además, consideramos la posibilidad de agregar una tabla que muestre el recuento de pedidos por ciudad, ordenando la información de mayor a menor.

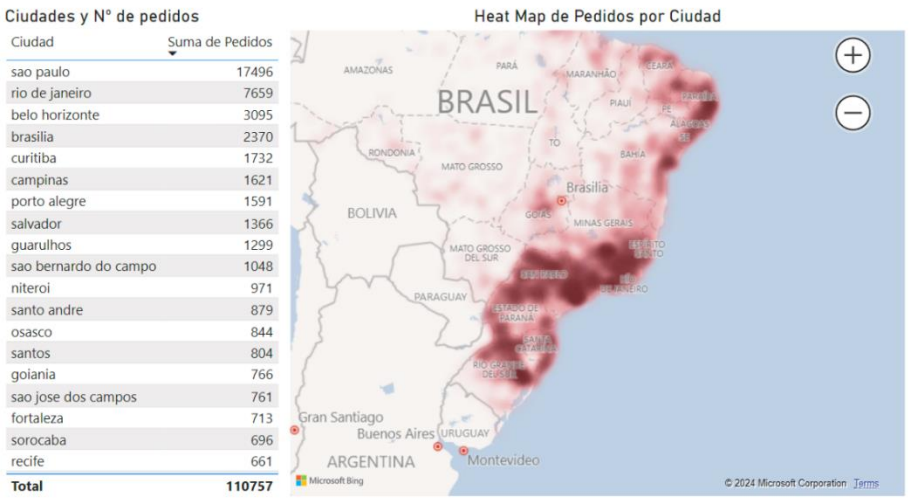


Ilustración 9. Tabla de Ciudades y Número de Pedidos junto a Mapa de Calor o 'Heat Map' de la distribución de estos pedidos en Brasil. **Fuente: Propia**

De esta manera, mostramos los resultados en la Ilustración 9, proporcionando una visión general de la distribución de los clientes en el mapa de Brasil. Además, hemos empleado un procedimiento más rápido y eficiente en comparación con la ejecución de código.

En el contexto de nuestro estudio, estas visualizaciones destacan la relevancia de grandes ciudades como Sao Paulo y Río de Janeiro, así como la capital Brasilia, dentro del e-Commerce analizado. Nuevamente, la empresa podría realizar segmentaciones sin necesidad de recurrir a algoritmos, basándose exclusivamente en la ubicación de los clientes y ajustando su estrategia según el volumen de pedidos en cada área. Sin embargo, esta acción sería menos precisa que la segmentación a través de grupos de clientes con múltiples características comunes, la cual solo sería posible mediante el uso de algoritmos de clustering, siendo este el objetivo de nuestro estudio.

En la continuación del análisis de las variables presentes en nuestro conjunto de datos, queremos profundizar en el análisis de 'customer_zip_code_prefix', que cuenta con 14,955 códigos postales distintos. Dada la amplia variedad de posibilidades, esto podría resultar en un aumento de la dispersión de los datos, incluso con un conjunto extenso, como es el caso, con más de 100,000 entradas. Podemos intentar identificar los 100 códigos postales más frecuentes y agrupar el resto bajo un único valor llamado 'otros'. Si la frecuencia de los pedidos dentro de estos 100 códigos postales más comunes es lo suficientemente alta, podría permitir a los algoritmos clasificar a los clientes en función de esta variable. Con el siguiente código, realizamos un recuento de las frecuencias de cada código postal:

```
postal_counts = df['customer_zip_code_prefix'].value_counts()
```

A continuación, obtenemos los 100 valores más frecuentes y agrupamos el resto bajo un único valor llamado 'Otros':

```
top_10_postal_counts = postal_counts.head(100)

otros_postal_count = postal_counts[~postal_counts.index.isin(top_10_postal_counts.index)].sum()

top_10_postal_counts = top_10_postal_counts.append(pd.Series({f'Otros': otros_postal_count}))
```

Al mostrar por pantalla el recuento de los 100 códigos postales y el del grupo restante, obtenemos lo siguiente:

```
22790       149
22793       148
24220       141
24230       134
22775       122
           ...
28300        55
11250        55
22240        54
88015        54
Otros    103371
Length: 101, dtype: int64

Recuento del Valor 'otros': 103371
```

Ilustración 10. Recuento de los 100 códigos postales más frecuentes y del subgrupo 'otros'. **Fuente: Propia**

Como podemos observar, el código postal más frecuente cuenta con tan solo 149 pedidos, mientras que más de 103,000 pedidos se distribuyen entre las demás localizaciones. Esto indica una gran dispersión en la distribución de los datos de nuestra variable 'customer_zip_code_prefix', la cual se destaca aún más mediante el histograma presentado en la Ilustración 11.

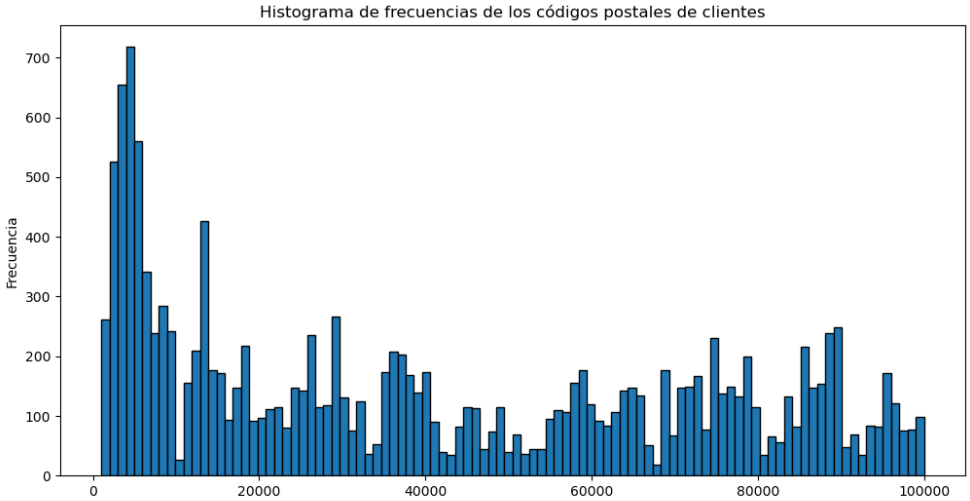


Ilustración 20 Histograma de frecuencias de los códigos postales de clientes. **Fuente: Propia.**

La distribución de códigos postales es bastante diversa a lo largo de todo el conjunto de datos, destacando ligeramente aquellos códigos comprendidos entre 1000 y 10,000. Estos podrían corresponder a códigos postales de ciudades donde se han registrado un mayor número de pedidos, como es el caso de Sao Paulo.

Con el siguiente código, obtenemos un dataframe que incluye únicamente los pedidos realizados en Sao Paulo. Además, mostramos por pantalla los códigos postales mínimo y máximo presentes en este subconjunto:

```
saopaulo_df = df.loc[df['customer_city'] == 'sao paulo']

saopaulo_df['customer_zip_code_prefix'].min()

saopaulo_df['customer_zip_code_prefix'].max()
```

Así, confirmamos que el código postal más bajo para Sao Paulo en nuestro conjunto de datos es 1003, mientras que el más alto es 8490. Esta información respalda la observación del histograma, el cual muestra una clara concentración de clientes en esta zona geográfica, mientras que el resto de localizaciones están más dispersas. Este escenario podría plantear un desafío al aplicar algoritmos de clustering, ya que sería necesario generar un mayor número de clústeres o segmentos para lograr una clasificación más precisa de los clientes.

Además, se presenta el problema de la trasnformación de los datos, un tema que abordaremos más adelante, y que discutirá cómo manejar una variable categórica con una diversidad tan amplia de valores.

4.2.2 Categorías de producto

En un e-commerce, las categorías de productos son agrupaciones que reúnen productos de una misma rama, sector o con características similares. En nuestro caso, contamos con 73 categorías únicas identificadas por la variable 'product_category_name'. Dada la amplitud de nuestro conjunto de datos, es crucial determinar si hay una concentración significativa de pedidos en ciertas categorías. Para lograr esto, empleamos el siguiente código, el cual calcula la frecuencia de cada categoría y la divide por 110.757, que corresponde al total de entradas en nuestro conjunto de datos. De esta manera, obtenemos la proporción que cada categoría representa dentro del conjunto global de datos:

```
product_category_name_counts = df['product_category_name'].value_counts()/110757*100

print(product_category_name_counts.head(10))
```

Al limitar la impresión por pantalla a 10, encontramos las diez categorías más frecuentes y su proporción:

```
cama_mesa_banho           10.054444
beleza_saude               8.705545
esporte_lazer              7.799056
moveis_decoracao           7.519164
informatica_acessorios     7.086685
utilidades_domesticas      6.268678
relogios_presentes         5.372121
telefonia                  4.076492
ferramentas_jardim         3.907654
automotivo                 3.803823
```

Ilustración 11. Las 10 categorías más frecuentes en el conjunto de datos y su frecuencia relativa. **Fuente: Propia**

Como se aprecia en la ilustración, la categoría más frecuente es 'cama_mesa_banho', que representa el 10% del conjunto de datos. Al sumar las frecuencias relativas de estas diez categorías, observamos que el 59.16% de los pedidos se distribuyen entre ellas, mientras que el 41% restante corresponde a categorías menos frecuentes.

4.2.3 Tipo de pago y estado del pedido

En el epígrafe 4.1 sobre fusión de datos y obtención del dataset objetivo, identificamos una problemática relacionada con los tipos de pagos presentes en los pedidos, debido a diversas secuencias de pago que duplicaban las entradas. La solución adoptada fue agrupar los pedidos con múltiples instancias bajo un único valor denominado 'varios'. Ahora, podemos realizar un recuento de cada tipo de pago y examinar su frecuencia relativa dentro del conjunto de datos utilizando un código similar al empleado para las categorías de productos.

Los resultados indican que más del 74% de los pedidos se pagaron con tarjeta de crédito, el 20% mediante boleto bancario, y el resto a través de cupones, tarjeta de débito y métodos de pago variados. Este último incluye el valor 'varios' creado anteriormente, representando únicamente el 2.9% del total de pedidos.

Además, llevamos a cabo un análisis similar para el estado de los pedidos, que reflejan las etapas desde el pago hasta la entrega del producto. Un e-commerce saludable debería mostrar un alto porcentaje de pedidos pagados y entregados, mientras que una incidencia elevada de cancelaciones podría indicar problemas. Al utilizar el código previamente mencionado para obtener la frecuencia de categorías o tipos de pago, descubrimos que el 97.9% de los pedidos en el conjunto de datos tienen el estado 'delivered', es decir, entregado. Solo un 0.5% de los pedidos fueron cancelados, y el resto se encuentra en proceso de envío o preparación del pedido.

Este resultado es crucial para el análisis de clustering posterior, ya que la inclusión de una variable altamente sesgada como esta podría conducir a resultados poco concluyentes y carentes de información, al agrupar a todos los clientes bajo un mismo grupo.

4.2.4 Puntuación de los pedidos

Otra variable de gran interés tanto para la empresa como en el contexto de este estudio es la puntuación de los pedidos. Esta puntuación es una valoración numérica, que va del 1 al 5, y refleja la satisfacción del cliente con la tienda y su compra. A pesar de que los valores son numéricos, consideramos esta variable como categórica, ya que cada valor representa un grupo de clientes con distintos niveles de satisfacción. Una mayor concentración de clientes en los valores 4 y 5 significaría que el negocio cumple en gran medida con las expectativas de sus usuarios. En este caso, mostramos un diagrama de barras en la Ilustración 12, representando la distribución de las puntuaciones en el conjunto de datos, así como la frecuencia relativa de cada uno.

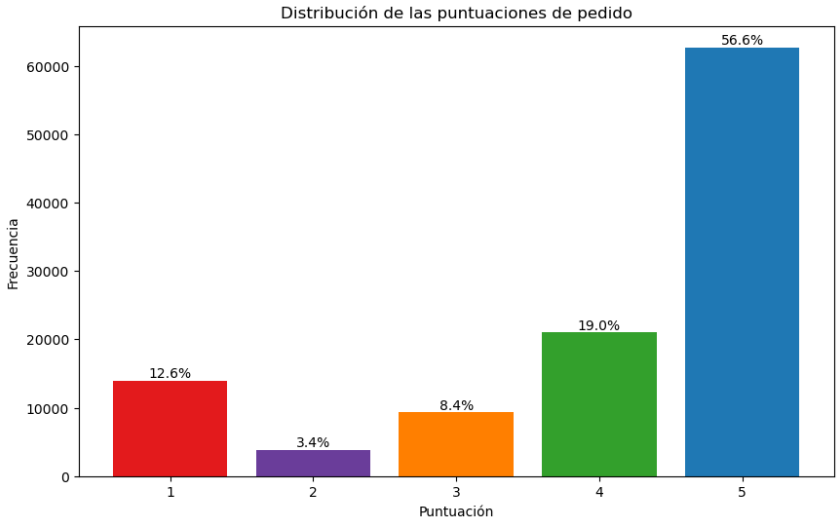


Ilustración 12. Distribución de las puntuaciones de pedido. **Fuente: Propia.**

El resultado es interesante desde el punto de vista de la segmentación de clientes, al encontrar que más del 56% de los usuarios han otorgado la máxima puntuación a su pedido. De la misma forma, la suma de valoraciones negativas, nos muestra que más del 15% han quedado insatisfechos con la compra o la tienda.

4.2.5 Distribución de la variable fecha y su uso en la segmentación

La última variable que exploraremos es la fecha de compra. Recordemos que previamente transformamos esta variable en diferentes agrupaciones con valores numéricos para que los algoritmos de clustering pudieran procesar los datos.

Siguiendo el código descrito en los subepígrafes anteriores, encontramos que en 2018 se realizaron la mayoría de los pedidos, representando más del 54% del conjunto de datos. Por otro lado, en 2016 tan solo se registró un 0.3%. Esto podría indicar que los comercios comenzaron sus actividades en ese año o que se volvieron populares desde entonces. En cualquier caso, no contamos con datos que confirmen esta teoría.

En relación con la segmentación de clientes, la fecha puede ser una variable interesante para identificar tendencias o tipologías de clientes en función del momento de la compra. Por ejemplo, podríamos encontrar grupos de clientes que concentran sus compras en los fines de semana o en días festivos. Esta información sería útil para reforzar el stock o aumentar el personal en días clave, como el Black Friday. Para verificar esta afirmación, utilizamos momentáneamente una versión de nuestro conjunto de datos donde la variable 'order_purchase_timestamp' no se había desglosado en diferentes columnas:

```
df_inicial['order_purchase_timestamp'] = pd.to_datetime(df_inicial['order_purchase_timestamp'])
df_inicial['fecha'] = df_inicial['order_purchase_timestamp'].dt.date
df_inicial['hora'] = df_inicial['order_purchase_timestamp'].dt.time
df_inicial['hora_str'] = df_inicial['order_purchase_timestamp'].dt.strftime('%H:%M:%S')
```

Con este código, creamos dos nuevas columnas: una para la fecha y otra para la hora. Esto se realiza porque al intentar obtener las frecuencias utilizando el formato inicial, solo se mostrará el recuento de los pedidos producidos en una hora específica. Con las columnas separadas, podemos aplicar nuestro código de recuento de pedidos en función de una variable. A continuación, presentamos los cinco primeros resultados:

```
2017-11-24    1366
2017-11-25     580
2017-11-27     480
2017-11-26     452
2018-08-06     430
```

El día con el mayor número de pedidos es el 24 de noviembre de 2017, coincidiendo con el Black Friday (último viernes del mes de noviembre). Los tres registros siguientes corresponden a días de la misma semana. De esta manera, se destaca la importancia de reflejar estas tendencias en el estudio de segmentación, el cual se abordará en próximos epígrafes.

4.3 Preprocesado de datos

4.3.1 Selección de variables

Antes de iniciar la ejecución de los diversos algoritmos de clustering, es necesario tomar decisiones sobre qué variables mantener y realizar las transformaciones necesarias para garantizar la máxima eficacia de nuestros métodos.

Inicialmente, hemos decidido eliminar la variable 'customer_zip_code_prefix'. Como se demostró en el epígrafe 4.2.1, existe una gran dispersión en la distribución de códigos postales de los pedidos. Dada la magnitud de nuestro conjunto de datos, conservar esta variable podría conducir a resultados poco concluyentes. Además, la razón principal para no incluir esta categoría es evitar una multidimensionalidad excesiva. Como mencionamos en epígrafes anteriores, a pesar de que la variable 'customer_zip_code_prefix' contiene valores numéricos, es categórica. Cada individuo pertenece a un código postal específico, y estos códigos son independientes y carecen de jerarquía. Incluir el código postal en nuestro modelo requeriría la aplicación de 'One-hot-encoding'. Este método convertiría cada código postal en una variable binaria, tomando un valor de 1 o 0 según si el individuo pertenece o no a ese código postal. Dado que esta variable tiene actualmente 14,955 valores únicos, aplicar 'One-hot-encoding' generaría 14,955 nuevas variables categóricas en nuestro conjunto de datos. Esto aumentaría significativamente las dimensiones de los datos, dificultando el procesamiento y la ejecución del modelo de segmentación.

De manera similar, hemos decidido excluir la variable 'order_status' debido a un problema contrario al observado con los códigos postales. Como se evidenció en el análisis exploratorio del conjunto de datos, esta variable está altamente concentrada en torno a uno de los estados de pedido, revelando que más del 95% de los pedidos en el conjunto de datos ya han sido entregados. Mantener esta variable podría dificultar que el algoritmo clasifique a los individuos en más de un grupo.

Finalmente, aunque no sea estrictamente necesario, hemos optado por eliminar la variable 'hour' que creamos inicialmente para separar los valores del campo con la fecha y hora del pedido. Consideramos que es preferible concentrarnos en años, meses y días de la semana para simplificar el estudio. No obstante, cabe destacar que siempre existe la posibilidad de reintroducir esta variable en estudios posteriores de mayor amplitud si se considera necesario.

La eliminación de las variables anteriores se ha realizado mediante el uso del método ‘drop’ de Python, siguiendo el código de ejemplo siguiente:

```
df = df.drop("customer_zip_code_prefix", axis=1)
```

4.3.2 Transformación de variables

La siguiente etapa del preprocesamiento de datos implica examinar el tipo y formato de cada variable, así como llevar a cabo las transformaciones necesarias para avanzar con el análisis y la ejecución de algoritmos. Realizamos una copia del dataframe actual y aplicamos el método ‘info’ para mostrar el número de columnas, entradas, variables y tipología de las variables:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 110757 entries, 0 to 110756
Data columns (total 11 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   customer_city         110757 non-null  object
 1   customer_state        110757 non-null  object
 2   price                 110757 non-null  float64
 3   freight_value         110757 non-null  float64
 4   payment_type          110757 non-null  object
 5   payment_value         110757 non-null  float64
 6   product_category_name 110757 non-null  object
 7   review_score          110757 non-null  int64
 8   year                  110757 non-null  int64
 9   month                 110757 non-null  int64
 10  day_of_week           110757 non-null  int64
dtypes: float64(3), int64(4), object(4)
memory usage: 9.3+ MB
```

Ilustración 13. Impresión por pantalla de la información principal del dataframe. **Fuente: Propia**

Observamos que el conjunto de datos cuenta con cuatro variables categóricas, cuatro numéricas y tres decimales. Nuestro objetivo a continuación es convertir las variables categóricas en numéricas para que los algoritmos de clustering puedan procesarlas correctamente. Comenzamos por la variable 'customer_city', que posee 4093 valores únicos sin jerarquía. Es decir, pertenecer a una ciudad no implica un rango mayor que pertenecer a otra. Por lo tanto, si asignamos valores numéricos a esta variable, el algoritmo podría asumir que ser Río de Janeiro es más grande que Brasilia solo por haber asignado a la primera el valor 3 y a la segunda el valor 1. Para abordar esto, debemos utilizar el método 'One-Hot-Encoding', convirtiendo cada ciudad en una categoría binaria (Raschka, S., & Mirjalili, V., 2017).

Como mencionamos anteriormente, aplicar 'One-Hot-Encoding' a categorías con múltiples valores únicos puede generar multidimensionalidad, creando numerosas variables y complicando nuestro modelo. Para evitar esto, podemos optar por transformar previamente la variable ciudad, utilizando las diez ciudades con mayor representación en el conjunto de datos y agrupando el resto en una categoría llamada 'Otras ciudades'. Con el siguiente código, obtenemos las 10 ciudades más frecuentes del conjunto de datos:

```
top_cities = X['customer_city'].value_counts().nlargest(10).index
```

A continuación, podemos usar la expresión ‘lambda’ para que toda ciudad que no sea una de las más frecuentes se sustituya por el término ‘Otras ciudades’:

```
X['customer_city'] = X['customer_city'].apply(lambda x: x if x in top_cities else 'otras ciudades')
```

```
otras ciudades          71480
sao paulo               17496
rio de janeiro           7659
belo horizonte           3095
brasilia                 2370
curitiba                 1732
campinas                 1621
porto alegre             1591
salvador                 1366
guarulhos                1299
sao bernardo do campo    1048
```

Ilustración 14. Listado de las 10 ciudades más frecuentes y subgrupo ‘otras ciudades’ junto al recuento de pedidos. **Fuente: Propia.**

Ahora que hemos conseguido reducer la dimensión de la variable ‘customer_city’, podemos proceder a aplicar el método ‘One-Hot-Encoding’. Para ello, utilizamos el siguiente código:

```
X = pd.get_dummies(X, columns=['customer_city'], prefix='city')
```

La sección 'prefix=city' del código se utiliza para que el nombre de cada nueva variable comience con esta palabra, generando nuevas categorías como 'city_brasilia' o 'city_curitiba'.

La siguiente variable a transformar es 'customer_state', que cuenta con 27 valores únicos. La solución es la misma que en el caso de las ciudades, aplicando el método 'One-Hot-Encoding' y aumentando, de cierta manera, la dimensionalidad de nuestro conjunto de datos al agregar 27 nuevas variables binarias. Por lo tanto, aplicamos el código anterior y transformamos estos datos en variables con valores 0 y 1.

A continuación, sería el turno de la variable 'product_category_name'. Como mostramos en el epígrafe de exploración de datos, esta variable cuenta con 73 valores únicos. Por otro lado, cerca del 60% de los datos se agrupan en torno a 10 de sus valores. Por lo tanto, decidimos agrupar el 40% restante alrededor de un valor ficticio llamado 'otras categorías'. De esta forma, evitamos aumentar la dimensionalidad en exceso al aplicar 'One-Hot-Encoding', agregando 11 variables nuevas en lugar de 73.

Finalmente, la última categoría sobre la que aplicar 'One-Hot-Encoding' es la de 'payment_type', por lo que agregamos cinco nuevas variables a nuestro conjunto. Sería posible discutir si es necesario realizar el mismo procedimiento con las variables 'year', 'month' y 'day_of_week' o ‘review_score’. En estos casos, consideramos más factible dejarlas en formato numérico al existir cierta ordinalidad y evitar aumentar aún más la dimensionalidad del conjunto de datos, que en este punto ya cuenta con 61 columnas.

4.3.3 Estandarización

Conforme a la comunidad científica, la estandarización se presenta como un paso previo crucial e indispensable antes de aplicar algoritmos de aprendizaje automático. Trabajar con datos numéricos que poseen diferentes escalas puede generar resultados anómalos, ya que muchos algoritmos funcionan de manera más eficaz cuando los datos están adaptados a la misma escala (Géron, A., 2019).

Estandarizar implica transformar cada uno de los datos en el conjunto para aproximarse a una media de 0 y una desviación estándar de 1 (Albon, C., 2018). Aunque existen métodos en librerías como 'Scikit-Learn' (sklearn) que permiten estandarizar conjuntos de datos, también es posible aplicar la fórmula directamente, tal como la presentan los autores mencionados:

$Z= (X-\mu) / \sigma$

En esta fórmula, el valor estandarizado (Z) es igual a la diferencia entre X (variable original) y μ (media de la variable original X), dividido por σ (desviación estándar de la variable original X). Así pues, procedemos a utilizar el siguiente código para realizar la estandarización de nuestro conjunto de datos:

```
x_estandarizado = (X - X.mean() ) / X.std()
X.describe()
x_estandarizado.describe()
```

Utilizando el método ‘describe’, podemos imprimir por pantalla las estadísticas principales cada variable presente en nuestro conjunto de datos y comparar nuestro conjunto original con el estandarizado. Así lo mostramos en la siguiente ilustración:

	price	freight_value	payment_value
count	110757.000000	110757.000000	110757.000000
mean	120.502017	20.012748	179.979834
std	181.919374	15.814897	271.122309
min	0.850000	0.000000	9.590000
25%	39.900000	13.080000	65.690000
50%	74.900000	16.290000	114.180000
75%	134.900000	21.170000	195.690000
max	6735.000000	409.680000	13664.080000

	price	freight_value	payment_value
count	1.107570e+05	1.107570e+05	1.107570e+05
mean	4.105811e-17	-8.416913e-17	-2.155551e-17
std	1.000000e+00	1.000000e+00	1.000000e+00
min	-6.577200e-01	-1.265437e+00	-6.284611e-01
25%	-4.430645e-01	-4.383682e-01	-4.215435e-01
50%	-2.506716e-01	-2.353950e-01	-2.426943e-01
75%	7.914486e-02	7.317479e-02	5.794494e-02
max	3.635950e+01	2.463925e+01	4.973438e+01

Ilustración 15. Estadísticos descriptivos del conjunto de datos original (izquierda) y estandarizado (derecha).
Fuente: Propia

Podemos observar en la Ilustración 15 que la estandarización de los datos ha tenido éxito, ya que en variables como ‘price’, la desviación estándar ha pasado de 181.9 a 1, mientras que la media cambia 120.5 a un valor aproximado a cero (4.105811e-17).

Otros métodos alternativos consisten en la aplicación de técnicas provenientes de librerías de código abierto. En el caso de Scikit-Learn (2024), se definen y ponen a disposición los métodos Min-Max y Robust Scaler. El primero de ellos escala nuestro conjunto de datos a un rango determinado, utilizando el siguiente código:

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 1))
x_estandarizado = scaler.fit_transform(X)
x_estandarizado = pd.DataFrame(x_estandarizado,columns=X.columns)
```

Por otro lado, Robust Scaler es un método que escala nuestro conjunto de datos adaptándolo a su rango intercuartílico (rango entre el primer y el tercer cuartil).

```
from sklearn.preprocessing import RobustScaler
robust_scaler = RobustScaler()
robust_scaler.fit_transform(X)
x_estandarizado = robust_scaler.fit_transform(X)
```

Conocer y aplicar diferentes métodos de estandarización puede ser la base para obtener mejores resultados, como demostraremos en los próximos epígrafes, donde haremos uso de las opciones aquí definidas.

El siguiente paso, será visualizar nuestro conjunto de datos estandarizado y observar si la correlación entre las distintas variables de la muestra.

4.3.4 Correlación de las variables

Aunque muchos de los modelos de aprendizaje automático sean robustos a cierto grado de colinealidad entre las variables, resulta interesante estudiar brevemente su relación y tomar acciones en el caso de que sea muy evidente. Como veremos más adelante, algoritmos como K-Means se basan en la distancia Euclidiana, por lo que una optimización de la correlación resultaría en una mayor agrupación de las muestras, dando lugar a clústeres de mejor calidad (Nagar, A. 2020).

El método más estandarizado para medir la correlación de las variables es el Coeficiente de Correlación de Pearson, un estadístico numérico que varía entre -1 y 1. Si el resultado es un valor cercano a uno, hablamos de correlación positiva, lo que significa que un incremento de la variable ‘A’ conlleva también un aumento de la variable ‘B’. En cambio, un valor del coeficiente cercano a -1 nos indica correlación negativa, por lo que un incremento de la variable ‘A’ se traduciría en un decrecimiento de la variable ‘B’, estableciendo así una relación inversamente proporcional (Géron, A., 2019).

En el conjunto de datos que nos ocupa, representaremos tanto la matriz de correlación de las variables como un mapa de calor que demuestre la relación entre las mismas. Para ello, utilizamos el método 'corr()' incluido en la librería Pandas de Python para obtener la matriz y los estadísticos de correlación:

```
x_estandarizado.corr(method='pearson')
```

	price	freight_value	payment_value	review_score	year
price	1.000000	0.412875	0.761766	-0.004421	-0.002269
freight_value	0.412875	1.000000	0.385303	-0.037308	0.033146
payment_value	0.761766	0.385303	1.000000	-0.085252	0.000858
review_score	-0.004421	-0.037308	-0.085252	1.000000	-0.008161
year	-0.002269	0.033146	0.000858	-0.008161	1.000000

Ilustración 16. Matriz de correlación de las primeras cinco variables del conjunto de datos.
Fuente: Propia

En la Ilustración 16, podemos concluir que existe una fuerte correlación positiva entre las variables 'payment_value' y 'price', con un estadístico de 0.76. Esto se debe a que 'payment_value' es el importe total del pedido, siendo la suma de las variables 'price' y 'freight_value', por lo que se encuentran estrechamente relacionadas.

De la misma forma, existe correlación moderada entre la variable 'price' y la variable 'freight_value' con un estadístico de 0.41. Aunque no llega a ser una correlación fuerte, podríamos argumentar que el precio del pedido determina el porte del transportista a pagar. Así, hay productos de grandes dimensiones como electrodomésticos que requieren un transporte especial y esto encarece el pedido. Nos interesaría representar su relación lineal y tomar decisiones a partir del gráfico. Para ello, decidimos realizar una transformación logarítmica sobre ambas variables, lo cual ayudará a resaltar la relación entre las mismas dentro del gráfico (Holtz, Y., 2023).

```
df['log_price'] = np.log1p(df['price'])
df['log_freight_value'] = np.log1p(df['freight_value'])
plt.figure(figsize=(8, 6))
sns.regplot(x='log_price', y='log_freight_value', data=df, line_kws={'color': 'red'})
plt.title('Relación entre Log(Price) y Log(Freight Value) con Línea de Regresión Roja')
plt.xlabel('Log(Price)')
plt.ylabel('Log(Freight Value)')
plt.show()
```

El código realiza primero la transformación logarítmica de las variables 'price' y 'freight_value', para luego representarlas en un gráfico de dispersión, dibujando en color rojo la línea de regresión lineal.

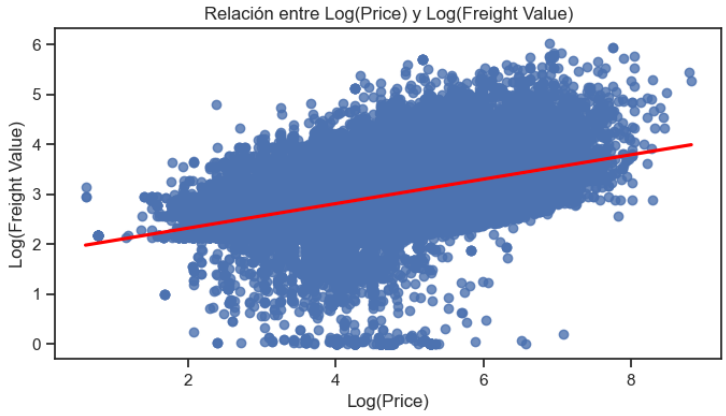


Ilustración 17. Gráfico de dispersión que muestra la relación lineal entre las variables 'price' y 'freight_value'. **Fuente: Propia.**

La Ilustración 17 nos muestra que la mayoría de los puntos siguen la línea de regresión, con la excepción de algunos valores en la mitad inferior y extremo superior derecha de la visualización. Esto confirma que existe cierta linealidad entre las variables y nos otorga un argumento adicional para proceder al descarte de la variable 'freight_value' de nuestro conjunto de datos.

```
x_estandarizado = x_estandarizado.drop("payment_value", axis=1)
x_estandarizado = x_estandarizado.drop("freight_value", axis=1)
```

Finalmente, procedemos a la eliminación de 'total_payment' y 'freight_value de nuestro conjunto de datos y continuamos con el análisis de valores atípicos.

4.3.5 Valores atípicos

Una de las formas más claras y visuales de detectar anomalías en nuestro conjunto de datos es realizar una visualización de un diagrama de caja o 'box plot'. Una de las librerías más comunes para representar este tipo de diagrama, es Seaborn, que describe cómo interpretar este gráfico:

> El cuadro muestra los cuartiles del conjunto de datos mientras que las líneas que se extienden (los 'bigotes') representan el resto de la distribución, excepto por los puntos que se determinan como 'valores atípicos' utilizando un método que es una función del rango intercuartílico (Seaborn, s.f.).

De esta forma, procedemos a aplicar el siguiente código en nuestro conjunto de datos estandarizado, desactivando el nombre de las variables con la línea "ax.set_xticks([])" para mayor claridad:

```
plt.figure(figsize=(10, 5))
ax = sns.boxplot(data=x_estandarizado)
ax.set_xticks([])
plt.title(u'Representación de cajas de las variables independientes X')
plt.ylabel('Valor de la variable normalizada')
_ = plt.xlabel('Variables estandarizadas')
plt.show()
```

El resultando se puede ver en la Ilustración 16 que presentamos a continuación:

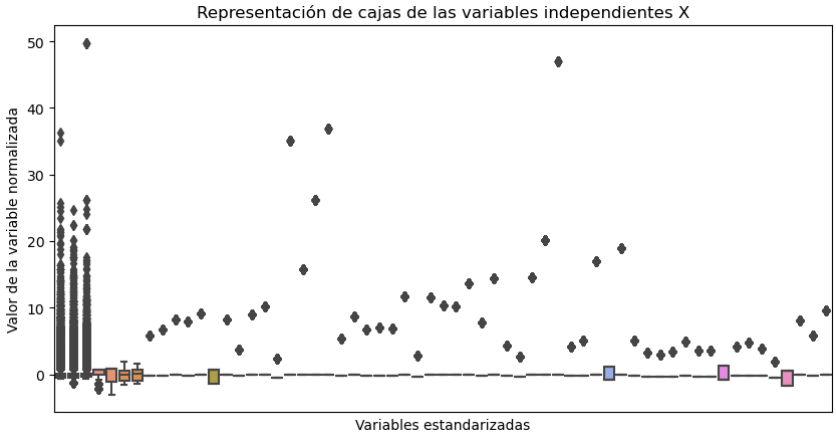


Ilustración 18. Representación de los diagramas de caja de las variables del conjunto de datos estandarizado. **Fuente: Propia**

En la Ilustración 18, se observa la presencia de valores atípicos (outliers) en la mayoría de las variables, representados por rombos de color negro. Además, se perciben variables que no muestran ninguna caja, lo cual puede atribuirse principalmente a las variables binarias creadas en pasos anteriores.

La literatura científica respalda la reducción o eliminación de valores atípicos para mejorar la precisión de los algoritmos de clustering. En el caso del algoritmo K-Means, por ejemplo, calcular la distancia entre varios grupos con características similares se vuelve más complejo cuando existen valores difíciles de clasificar entre ellos. La eliminación de estos valores atípicos maximizaría la distancia dentro de un mismo clúster y minimizaría la distancia con respecto a otros clústeres (Barai (Deb), A., y Dey, L., 2017).

Sin embargo, debemos tener en cuenta que no todos los valores extremos constituyen un valor atípico. Podemos encontrar valores atípicos que proceden de un error en la muestra, de fallos en la estandarización de los datos, suposiciones erróneas de la distribución o bien se trata de valores legítimos dentro de la población de estudio (Osborne, J. W., y Overbay, A., 2019). Esto significa que debemos estudiar variable a variable y tratar de entender si esos valores pueden ser realmente una anomalía. Por otro lado, tendremos que decidir si eliminarlos o continuar con ellos.

En el conjunto de datos que nos ocupa, observamos mayor densidad de valores atípicos dentro de las variables como 'price', es decir; el precio pagado por los productos del pedido. Si observamos los valores mínimos y máximos, encontramos que, aun siendo valores extremos, podrían formar parte de la normalidad operacional del e-Commerce. Por ejemplo, un precio máximo del pedido de 6735 (unidades monetarias) puede ser un caso único dentro de la actividad del comercio, pero después de todo, posible. Por otro lado, si observamos los valores máximos y mínimos de las variables binarias, siguen siendo 0 y 1, por lo que entendemos que los valores atípicos apreciados en la gráfica son obra de la estandarización del conjunto de datos y, por lo tanto, no son errores como tal.

La solución que entendemos más óptima dentro de este conjunto de datos es la de ejecutar los algoritmos usando un modelo que contenga valores atípicos y realizar un segundo modelo en el que se mantengan. Por otro lado, dado que el problema parece deberse a la estandarización, además de la propuesta en el apartado 4.3.3, ofreceremos también métodos alternativos como la transformación Min-Max y Robust Scaler. De esta forma, podremos comparar resultados y medir la eficiencia de uno y otro.

De cara a eliminar a los valores atípicos, nos decantamos por el método del rango intercuartílico, inventado por John Tuckey. Este rango abarcaría todo aquello comprendido entre el primer y tercer cuartil, considerando como atípicos los valores fuera de Q3+1,5*IQR o de Q1-1,5*IQR (Oracle, s.f). Así pues, sustituimos todos los valores fuera el rango mencionado, por los límites inferiores (Q1-1.5IQR) y superiores (Q3+1.5IQR), siguiendo el siguiente código:

```
for k in list(x_estandarizado.columns):
    IQR = np.percentile(x_estandarizado[k],75) - np.percentile(x_estandarizado[k],25)
    limite_superior = np.percentile(x_estandarizado[k],75) + 1.5*IQR
    limite_inferior = np.percentile(x_estandarizado[k],25) - 1.5*IQR
    x_estandarizado[k] = np.where(x_estandarizado[k] limite_superior,limite_superior,x_estandarizado[k])
    x_estandarizado[k] = np.where(x_estandarizado[k] < limite_inferior,limite_inferior,x_estandarizado[k])
```

Procedemos a representar de nuevo nuestro conjunto de datos estandarizado tras la aplicación de la técnica IQR:

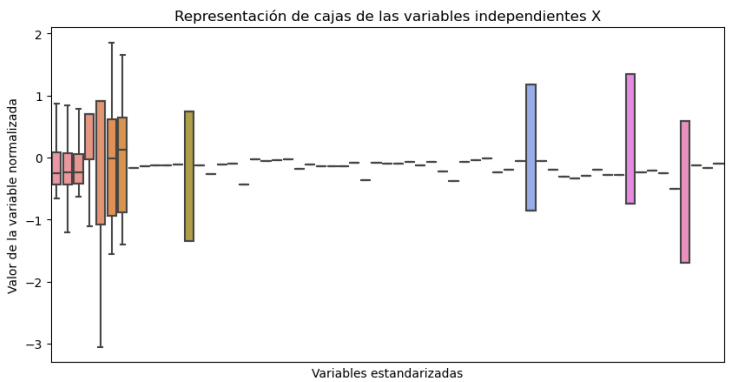


Ilustración 19. Representación de los diagramas de caja de las variables del conjunto de datos estandarizado tras la eliminación de valores atípicos. **Fuente: Propia**

Observamos en la gráfica 19 que la eliminación de valores atípicos ha resultado exitosa, no apreciando signos de valores atípicos, por lo que podemos continuar con el análisis de la normalidad.

4.3.6 Normalidad de las variables

Dado que el objetivo principal de este estudio es la aplicación y comparación de diferentes algoritmos de clustering, es esencial tener en cuenta los diversos aspectos a los que estos algoritmos son sensibles. Uno de estos aspectos críticos es la distribución de los datos. Algunos algoritmos, como K-Means, tienden a mostrar un rendimiento óptimo en presencia de distribuciones Gaussianas o normales, donde la varianza se mantiene constante en todo el conjunto de datos (Scikit-Learn, 2024). Cabe destacar que en la sección 4.3.3 se llevó a cabo la estandarización del conjunto de datos para normalizar la media y la varianza, aproximándolas a 0 y 1, respectivamente. No obstante, es importante señalar que esta normalización no garantiza automáticamente que el resultado siga una distribución normal. Por lo tanto, es esencial realizar pruebas estadísticas específicas y visualizaciones para evaluar la distribución de las variables después de la estandarización.

En una primera instancia, se descartó el uso del test de Shapiro-Wilk, ya que se considera más apropiado para muestras pequeñas, mientras que nuestro conjunto de datos consta de cientos de miles de registros. En su lugar, se optó por utilizar el test de Anderson-Darling, el cual se basa en la prueba de Kolmogorov-Smirnov, proporcionando el valor crítico de cada variable para llevar a cabo la evaluación de normalidad (Brownlee, J., 2019). El test de Anderson-Darling puede ser implementado en nuestro conjunto de datos utilizando la librería SciPy, siguiendo el siguiente código:

```
for column in x_estandarizado.columns:
  data = x_estandarizado[columna]
  result = anderson(data)
  print(f'Columna: {column}')
  print(f'  Estadístico de prueba: {result.statistic}')
  print(f'  Valores críticos: {result.critical_values}')
  alpha = 0.05
  for i, cv in enumerate(result.critical_values):
    if result.statistic < cv:
      print(f'  No podemos rechazar la hipótesis nula en el nivel de significancia {result.significance_level[i]}%. Los datos parecen
provenir de una distribución normal.'
      break:
      else:
      print("  Rechazamos la hipótesis nula. Los datos no parecen provenir de una distribución normal.")
```

Al utilizar este método, el código devuelve el nombre de cada variable, el resultado del test de Anderson, el estadístico de prueba y el veredicto del test. A continuación, se presenta una breve representación de los resultados obtenidos al evaluar la normalidad de las variables en nuestro conjunto de datos:

Columna: Price
Estadístico de prueba: 14317.10683763682
Valores críticos: [0.576 0.656 0.787 0.918 1.092]
Rechazamos la hipótesis nula. Los datos no parecen provenir de una distribución normal.

Columna: review_score
Estadístico de prueba: 13922.52956173064
Valores críticos: [0.576 0.656 0.787 0.918 1.092]
Rechazamos la hipótesis nula. Los datos no parecen provenir de una distribución normal.

Los resultados del test muestran consistencia a lo largo de todas las variables evaluadas. El valor del estadístico de prueba es significativamente superior a los valores críticos, lo que conduce al rechazo de la hipótesis nula que asume la normalidad de la distribución. Como resultado, se concluye que ninguna de las variables en nuestro conjunto de datos sigue una distribución normal. Para respaldar estos hallazgos, procedemos a una representación visual mediante un gráfico Cuantil-Cuantil, también conocido como 'Q-Q plot', que ilustra la distribución de la variable en comparación con una línea de referencia de normalidad (Brownlee, J., 2019). En este contexto, utilizaríamos el siguiente código para generar el gráfico de cada variable.

```
for column in x_estandarizado.columns:
data = x_estandarizado[column]
plt.figure(figsize=(8, 5))
probplot(data, plot=plt)
plt.title(f'Q-Q Plot de {column}')
plt.show()
```

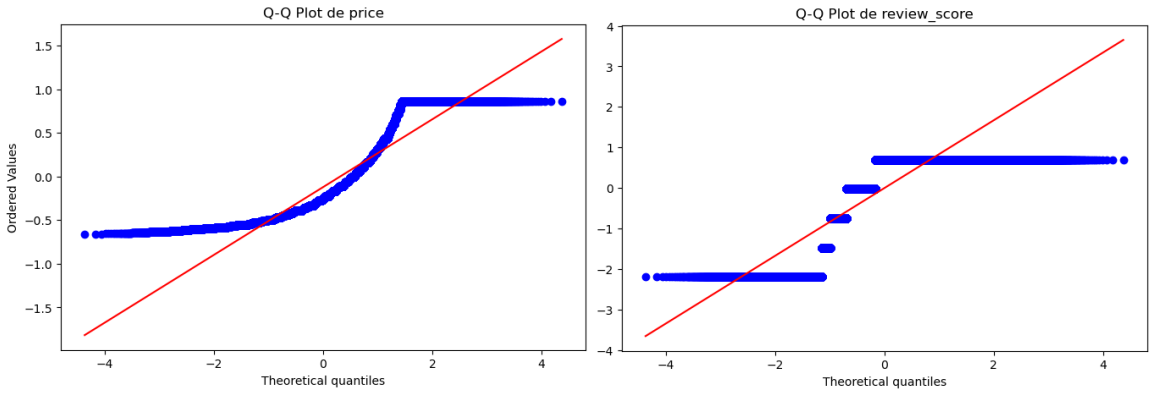


Ilustración 20. Representación del 'Q-Q plot' de las variables 'price' y 'review_score'. **Fuente: Propia**

En la Ilustración 20 podemos observar un ejemplo de dos de nuestras variables. En el caso de 'price', la distribución no parece seguir la línea de referencia, observando desviaciones significativas desde el cuantil -4 al -2 u desde el 2 al 4. En lo referente a la variable 'review_score', en ningún caso parece seguir la curva de la normalidad. Estos rasgos parecen repetirse con la representación de todas las variables del conjunto de datos. La conclusión, por lo tanto, es evidente. A pesar de haber estandarizado, ninguna de nuestras variables parece seguir una distribución normal.

Los resultados de estos tests nos indican de antemano que la aplicación de algoritmos de clutering sobre este conjunto de datos podría arrojar resultados poco concluyentes o, al menos, de menor calidad que si se aplicaran sobre variables que siguiesen una distribución normal. Por lo tanto, esto no debe considerarse motivo para descartar el estudio, sino más bien una advertencia sobre los posibles resultados y un indicativo de la dificultad que conllevan los análisis de segmentación en poblaciones reales, donde la cantidad de datos puede ser considerable.

4.4 Aplicación de los algoritmos de clustering

Una vez obtenido, analizado y transformado el conjunto de datos proveniente de un e-Commerce, procedemos a aplicar diversos algoritmos de aprendizaje no supervisado para segmentar nuestros datos en clústeres, los cuales pueden presentar niveles de diferenciación variables. Es importante tener en cuenta que los resultados pueden variar entre distintas poblaciones y dependerán de las consideraciones tomadas durante la transformación de los datos. Además, algunos algoritmos pueden arrojar resultados diferentes según su configuración, por lo que cada paso debe ser analizado y justificado adecuadamente.

En este contexto, emplearemos algoritmos que clasifican los datos en función de su proximidad a puntos denominados centroides, mientras que otros se basan en la búsqueda de densidades o en el establecimiento de jerarquías. Como resultado, se observarán diferencias en la eficacia con la que generan los clústeres, lo que conducirá a disparidades entre unos resultados y otros.

4.4.1 K-Means

Este algoritmo pertenece a los denominados modelos de clustering por prototipos. Esto implica que identifica cada clúster a través de un punto llamado centroide o medoide. En el caso de los centroides, se trata del promedio de puntos con características similares, de ahí el nombre "K-Means" o "K-Medias". Por otro lado, el medoide representa un punto frecuentado por un gran

número de datos y constituye una versión alternativa del algoritmo conocida como "K-Medoides", la cual abordaremos en el próximo epígrafe (Raschka, S., & Mirjalili, V., 2017).

En el contexto del K-Means, es necesario especificar el valor de *k*, que representa el número de clústeres que se desea generar. Por lo tanto, es crucial tener en cuenta que un valor inadecuado de *k* puede resultar en un rendimiento ineficiente del algoritmo. Aunque *k* sea desconocida, podemos emplear una representación gráfica denominada 'Prueba del Codo', que "traza la suma de las diferencias al cuadrado entre cada punto de datos y su centroide asignado para diferentes valores de *k*" (Barrios, A., 2023).

Además, es importante destacar que este algoritmo asume que los clústeres tienen forma esférica y que las características presentan una varianza similar (Nagar, A., 2020). Conociendo estos preceptos básicos, procederemos a aplicar el algoritmo utilizando la librería Scikit-Learn y realizaremos las configuraciones necesarias para maximizar sus resultados.

```
cluster_range = range(1,20)
cluster_wss=[]
for cluster in cluster_range:
    model = KMeans(cluster)
    model.fit(df)
    cluster_wss.append(model.inertia_)

plt.figure(figsize=[20,6])
plt.title('Curva WSS para encontrar el valor óptimo de clústers o grupos')
plt.xlabel('# grupos')
plt.ylabel('WSS')
plt.plot(list(cluster_range),cluster_wss,marker='o')
plt.show()
```

El código genera un modelo de clustering mediante el algoritmo K-Means y visualiza la Curva de la Suma de Cuadrados para diferentes valores de *k*, facilitando la identificación del número óptimo de clústers.

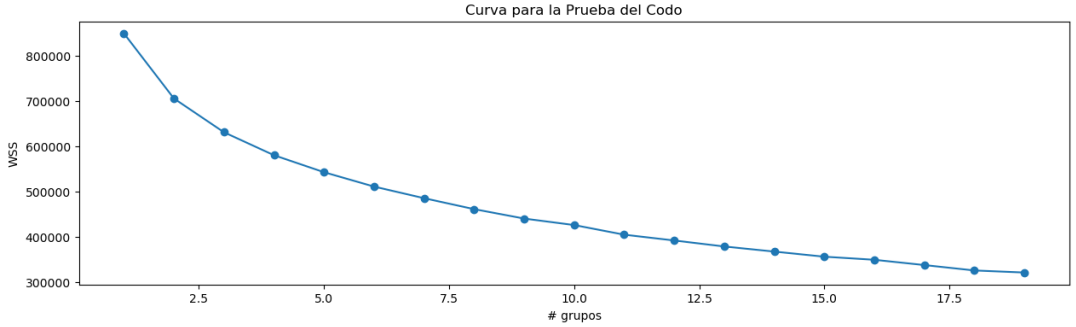


Ilustración 21. Representación de la Curva de la Suma de Cuadrados para la realización de la Prueba del Codo.
Fuente: Propia.

En la Ilustración 21, se puede apreciar que la singularidad de esta prueba radica en determinar el número de grupos o *k* que maximiza la suma de los cuadrados. Este punto generalmente coincide con 'el codo de la curva', es decir, las coordenadas exactas donde la disminución de la suma de cuadrados comienza a suavizarse. En el ejemplo propuesto, podríamos enfrentarnos a la elección entre *k*=2 o *k*=3. A continuación, procederemos a ejecutar nuestro algoritmo:

```
model = KMeans(n_clusters=3,random_state=0, init='random')
model.fit(df)
labels = model.labels_
df_total = df.copy()
df_total['cluster']=model.predict(df)
```

El código anterior construye un modelo de clustering mediante el algoritmo K-Means con *k*=3. Luego, genera etiquetas y las aplica a una copia de nuestro conjunto de datos, permitiéndonos visualizar la distribución en los tres clústeres correspondientes. En otras palabras, podemos identificar el número de pedidos del e-Commerce asignados a los clústeres 0, 1 y 2.

```
df_total.cluster.value_counts().plot(kind='bar', figsize=(10,4))
plt.title('Recuento de pedidos por grupo')
plt.xlabel('Grupo')
_ = plt.ylabel('Recuento')
```

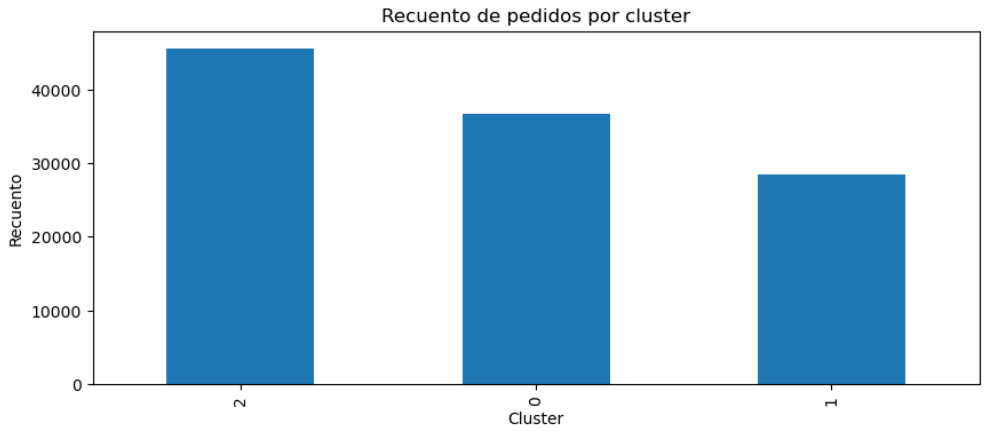


Ilustración 22. Recuento de pedidos por clúster. **Fuente: Propia**

Observamos que nuestro modelo ha logrado distribuir de manera equitativa nuestro conjunto de datos entre los distintos clústeres.

Aunque parece que el algoritmo puede generar y clasificar nuestra muestra, esto no garantiza su eficiencia. Para determinar cuál de los valores de *k* realmente maximiza el número de clústeres y produce una división adecuada, podemos analizar la Curva de la Silueta. Esta curva nos proporciona un estadístico llamado Coeficiente o Índice de la Silueta, el cual indica "cómo de cerca está cada punto de un clúster respecto a los puntos de sus clústeres vecinos" (Gonzalo, Á., 2019).

Para su implementación, utilizamos el método 'silhouette_score' de la librería Scikit-Learn:

```
from sklearn.metrics import silhouette_score
silhouette_avg = silhouette_score(df, labels)
silhouette_avg
```

Este código genera un índice con un rango de -1 a 1. Cuanto más cercano a 1, indica que los puntos se han asignado correctamente a un clúster. Si está más cercano a 0, sugiere que distintos clústeres se superponen. En cambio, si el estadístico es negativo y cercano a -1, evidencia que el algoritmo no ha clasificado correctamente nuestro conjunto de datos (Scikit-Learn, 2024). En este estudio, hemos seguido la Prueba del Codo para seleccionar el valor de *k* y posteriormente hemos calculado el Coeficiente de la Silueta para evaluar su optimalidad. En casos donde este valor es muy bajo, hemos ajustado el valor de *k* y repetido el proceso hasta alcanzar el número de clústeres que maximizan nuestra curva y obtienen una puntuación más alta en la prueba de la silueta.

Además, es importante recordar las anomalías presentes en nuestro conjunto de datos, que incluyen la presencia de valores atípicos, desviaciones de la normalidad y variaciones en el proceso de estandarización. Por esta razón, hemos llevado a cabo la prueba K-Means en seis versiones diferentes de nuestro conjunto de datos:

- Conjunto de datos estandarizado con fórmula y sin presencia de valores atípicos: El valor que maximiza la curva es k=3, obtienen una puntuación de 0.1712 en la prueba de la silueta. Dado que este es el mejor valor que hemos podido lograr, nos indica que el algoritmo ha conseguido generar 3 clústeres distintos pero que estos se solapan en cierta medida, por lo que el resultado no es muy óptimo.
- Conjunto de datos estandarizado con fórmula permitiendo la presencia de valores atípicos: En este caso, la Curva de la Silueta no nos permitía intuir el número de clústeres óptimo, por lo que obtenemos la puntuación del Índice de la Silueta para distintos valores de k. El resultado es que k=2, nos otorga una puntuación de 0.2542 en el estadístico, por lo que ha conseguido definir unos clústeres con mayores diferencias que eliminando los valores atípicos.
- Conjunto de datos estandarizado mediante Min-Max y sin presencia de valores atípicos: En este caso, encontramos una gran diferencia en cuanto al número de clústeres óptimos se refiere. Debemos elegir k=11 para obtener una puntuación de 0.3517 en el estadístico de la curva. Esto podría significar que hay mayores diferencias en los valores de los registros de este modelo, comparado con los anteriores. Por lo tanto, al incrementar la dispersión de los datos, el número de clústeres se incrementa también.
- Conjunto de datos estandarizado mediante Min-Max permitiendo la presencia de valores atípicos: Esta versión de nuestro conjunto de datos nos arroja resultados mucho más

pobres que las anteriores. Sólo conseguimos maximizar la curva con k=5 y una puntuación máxima de 0.1342 por lo que se trata de clústeres con escasas diferencias entre ellos que tienen a solapar sus valores.

- Conjunto de datos estandarizado mediante Robust Scaler y sin presencia de valores atípicos: Esta versión del conjunto de datos consigue maximizar la curva con k=2 y una puntuación de 0.2543. Nuevamente, nos indica que nuestro algoritmo K-Means consigue su propósito de clasificar nuestra muestra en dos clústeres. Sin embargo, estos cuentan con puntos comunes, solapando unos valores con otros. Por lo tanto, en una posible representación gráfica nos resultaría complicado diferenciar ambos grupos.
- Conjunto de datos estandarizado mediante Robust Scaler permitiendo la presencia de valores atípicos: Finalmente, esta última versión nos arroja unos resultados mucho más claros. En este caso, k=2 maximiza nuestra curva con una puntuación de 0.623. Esto significa que, para esta muestra de datos del e-Commerce, estandarizar mediante Robust Scaler y permitir que existan valores atípicos consigue aportar mayor información al algoritmo para poder clasificar los registros del conjunto de datos. El valor de puntuación de la curva, cercano a 1 es resultado de mayor calidad a la hora de clasificar, junto con una mayor separación de los valores del clúster A respecto al clúster B.

Una vez hemos encontrado nuestro número de clústeres óptimo, resultaría interesante representar gráficamente cómo nuestro conjunto de datos se divide entre los grupos. Sin embargo, nos encontramos con una barrera principal, la alta dimensionalidad de nuestro conjunto, que impide que podamos representar todas las características en un único gráfico. Por lo tanto, debemos llevar a cabo una reducción de dimensionalidad.

Uno de los métodos más utilizados para reducir la dimensionalidad de un conjunto de datos, es el Análisis de Componentes Principales o PCA. Este método "toma datos de alta dimensionalidad y los proyecta en un nuevo subespacio con igual o menos dimensiones que el original" (Raschka, S., & Mirjalili, V., 2017, p.142). Para ello, hace uso de las diferencias de varianza en los datos, identificando aquellas direcciones donde su valor es mayor. Por lo tanto, esto demuestra la importancia de la estandarización como proceso previo a la reducción de dimensionalidad. Si nuestro conjunto no fuese escalado para que su media tomase valor cero y la desviación estándar 1, habría descompensación entre las distintas variables y el algoritmo las favorecería frente al resto (Amat Rodrigo, J., 2017).

Finalmente, existen diferentes opiniones dentro de la literatura a la hora de definir el número de dimensiones o componentes principales al que resulta más óptimo reducir. Sin embargo, el método más utilizado es el de la ratio de varianza explicada, representado también a través del

gráfico de la curva 'Dimensiones-Varianza Explicada', donde podemos aplicar la Prueba del Codo (Géron, A., 2019). El punto óptimo, sería aquel número de componentes que explicasen al menos el 95% de la varianza. Ejecutamos el siguiente código procedente nuevamente de la librería Scikit-Learn:

```
pca = PCA()
principal_components = pca.fit_transform(df)
explained_variance_ratio_cumulative = np.cumsum(pca.explained_variance_ratio_)
target_variance_ratio = 0.95
num_components_for_target_variance = np.argmax(explained_variance_ratio_cumulative >= target_variance_ratio) + 1
print(f'Número de componentes para el {target_variance_ratio*100}% de varianza explicada:
{num_components_for_target_variance}')

plt.figure(figsize=(10, 6))
plt.plot(range(1, len(explained_variance_ratio_cumulative) + 1), explained_variance_ratio_cumulative, marker='o')
plt.axvline(x=num_components_for_target_variance, color='red', linestyle='--', label=f'{target_variance_ratio*100}% de varianza
explicada')
plt.xlabel('Número de Componentes Principales')
plt.ylabel('Varianza Explicada Acumulativa')
plt.title('Curva de Varianza Explicada')
plt.legend()
plt.show()
```

El código anterior calcula en primer lugar el número de componentes que explican el 95% de la varianza del conjunto de datos. A continuación, representa la curva 'Dimensiones-Varianza Explicada' estableciendo una línea sobre el número de componentes óptimo y devolviendo por pantalla a cuántos componentes debemos reducir.

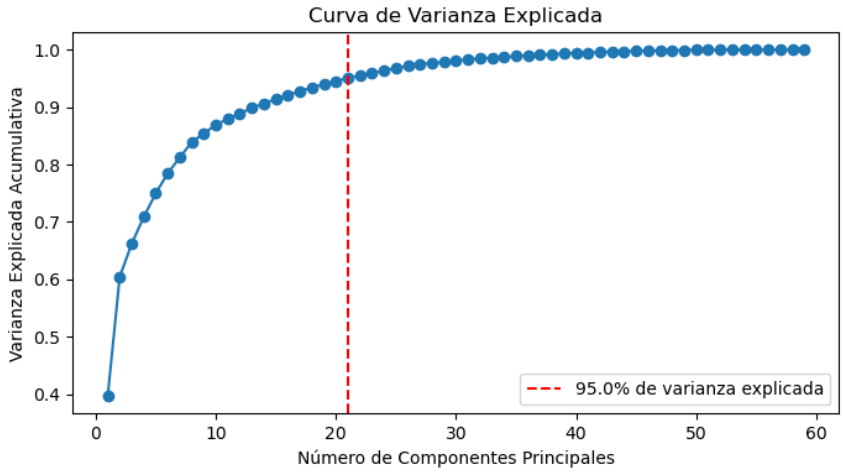


Ilustración 23. Curva de la varianza explicada por el número de componentes principales. Fuente: Propia.

La curva demuestra que, si reducimos a 21 componentes principales nuestro conjunto de datos, seguiremos explicando una gran parte de su varianza. Sin embargo, esto no funcionará muy bien de cara a visualizaciones, donde nos interesa encontrar la mayor cantidad de información posible de forma rápida y eficaz. Por lo tanto, decidimos reducir a 10 componentes, suponiendo un 87%

de la varianza explicada, todavía un indicador muy alto. Ahora sí nos permitirá representar los clústeres que hemos generado con el algoritmo K-Means y cómo se agrupan entre ellos los componentes principales.

```
sns.pairplot(pca, hue='cluster', palette='Set1')
plt.show()
```

El código anterior genera un gráfico de dispersión de pares utilizando Seaborn (sns.pairplot). Este gráfico visualiza la relación entre los componentes principales obtenidos mediante el Análisis de Componentes Principales (PCA). Los puntos en el gráfico están coloreados según los clústeres obtenidos mediante el algoritmo K-Means, y se utiliza una paleta de colores ('Set1') para diferenciar visualmente entre los clústeres.

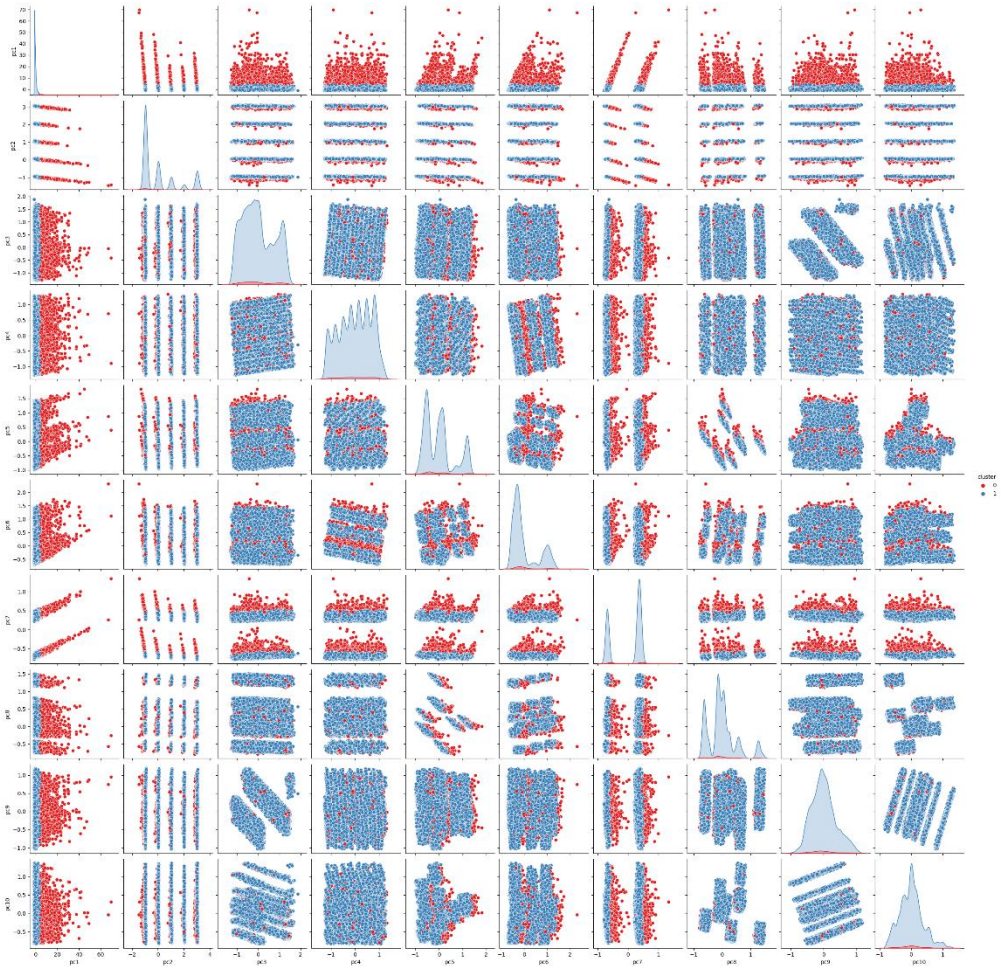


Ilustración 24. Representación visual del gráfico de dispersión de cada pareja de componentes principales. **Fuente: Propia**

La Ilustración 24 demuestra que una representación 2D no es el método más eficaz, en este caso, para diferenciar nuestros clústeres. Sin embargo, se puede observar que los gráficos tienen mayor calidad informacional en las primeras parejas de componentes principales, aquellas con mayor porcentaje de varianza explicada.

Para finalizar con la exposición del algoritmo K-Means y su representación mediante Análisis de Componentes Principales, podemos optar por realizar una visualización en 3D de los tres primeros componentes, utilizando el siguiente código, que mapea además cada clúster para identificarlo con color:

```
fig = plt.figure(figsize=(14, 14))
ax = fig.add_subplot(111, projection='3d')
scatter = ax.scatter(pca['pc1'], pca['pc2'], pca['pc3'], c=pca['cluster'], cmap='coolwarm', s=100)
ax.set_xlabel('PC1')
ax.set_ylabel('PC2')
ax.set_zlabel('PC3')
legend_labels = [plt.Line2D([0], [0], marker='o', color='w', label='Cluster 0',
markerfacecolor='red', markersize=10), plt.Line2D([0], [0], marker='o', color='w', label='Cluster 1', markerfacecolor='blue',
markersize=10)]
ax.legend(handles=legend_labels, title='Clusters')
plt.show()
```

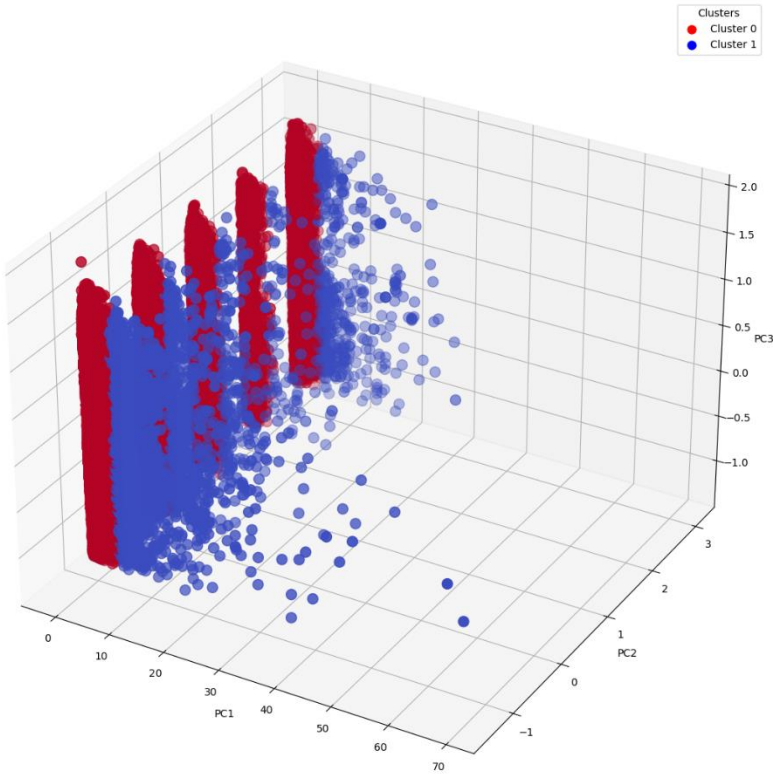


Ilustración 25. Representación en 3D de la dispersión de los tres primeros componentes principales y su agrupación en los clústeres generados mediante K-Means. **Fuente:Propia**

Este gráfico resulta mucho más evidente y nos permite observar la distribución de ambos clústeres. Visualmente, no existe mucha distancia entre ambas agrupaciones, pero sí hay una clara división entre los valores de uno y otro, con escasos solapamientos. Finalmente, dados los datos proporcionados, el e-Commerce podría obtener un listado de los registros pertenecientes a uno y otro grupo, así como obtener una media de los valores de cada uno, de forma que podrá identificar un perfil más o menos definido de cada agrupación, de cara a dirigir acciones futuras.

4.4.2 K-Medoides (K-Medoids)

Tal como señalamos en la introducción del epígrafe anterior, K-Medoides, también conocido como 'K-Medoids', se presenta como una alternativa al algoritmo K-Means. En lugar de utilizar centroides, este algoritmo hace uso de medioides, que son 'objetos que minimizan la suma de las distancias con otros objetos dentro del clúster' (Soleymani, A., 2022). En contraste con K-Means, K-Medoides destaca por su mayor resistencia a los valores atípicos. Esto se debe a que los medoides representan el punto central de cada clúster (Jin, X., Han, J., 2011).

La aplicación de K-Medoides sigue un proceso similar al de K-Means, donde la tarea clave es determinar el número óptimo de clústeres, *k*. Este proceso puede realizarse mediante la Prueba del Codo y la puntuación del Índice de la Silueta. Sin embargo, nos encontramos con un obstáculo al intentar ejecutar el algoritmo en nuestro conjunto de datos. Aparentemente, nuestro conjunto de datos es demasiado extenso, lo que genera un uso excesivo de la memoria y obstaculiza la ejecución correcta del algoritmo. Esta limitación se evidencia en la Ilustración 26.

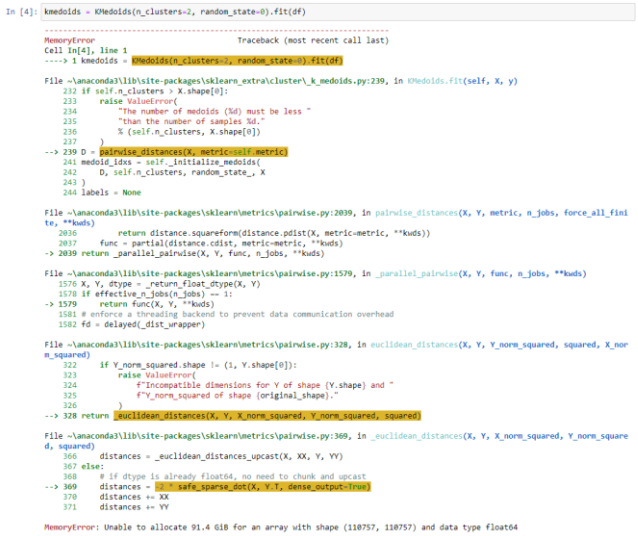


Ilustración 26. Error de memoria al ejecutar el algoritmo K-Medoides. **Fuente: Propia**

Nuestra solución es crear una muestra aleatoria de nuestro conjunto de datos lo suficientemente pequeña para poder ejecutar el algoritmo. Para ello utilizamos el siguiente código:

```
subset_df = df.sample(frac=0.2, random_state=42)
```

Una vez obtenida nuestra muestra representativa, que consta del 20% de los datos del conjunto original, procedemos a visualizar la Curva de la Suma de Cuadrados. Además, calculamos su puntuación para cada uno de los seis subconjuntos donde aplicamos anteriormente el algoritmo K-Means. Recordamos una vez más, que contamos con diferentes versiones del conjunto de datos en función de la estandarización utilizada y la presencia o no de valores atípicos.

```
cluster_range = range(1, 20)
cluster_wss = []
for cluster in cluster_range:
    model = KMedoids(n_clusters=cluster)
    model.fit(subset_df)
    cluster_wss.append(model.inertia_)
```

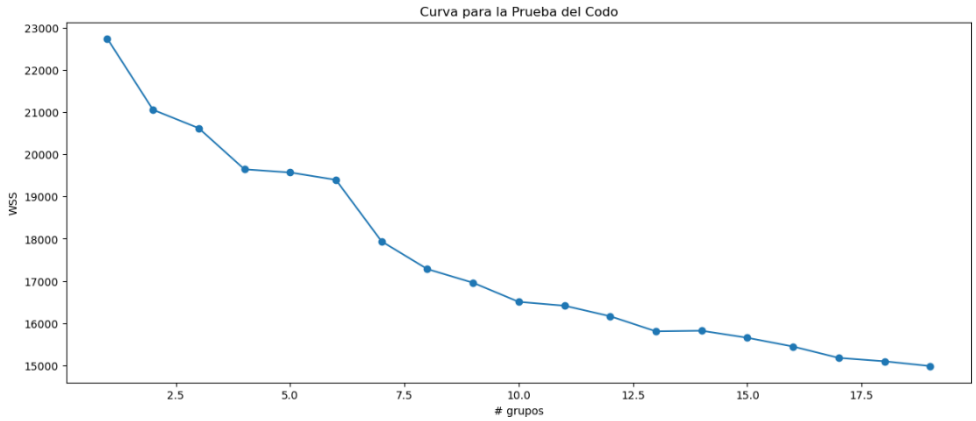


Ilustración 27. Curva para la Prueba del Codo y ejecución de K-Medoides en subconjunto de datos estandarizado mediante Robust Scaler y con presencia de valores atípicos. **Fuente: Propia**

En este caso, el código tiene en cuenta que la curva debe optimizarse según el algoritmo K-Medoides. Observamos que ninguna de las curvas generadas en los distintos subconjuntos nos permite definir el valor de *k* con facilidad. Por lo tanto, designaremos aquel que proporcione la mejor puntuación del Índice de la Silueta:

- Conjunto de datos estandarizado con fórmula y sin presencia de valores atípicos: El valor que maximiza la curva es *k*=3, obteniendo una puntuación de 0.1202 en la prueba de la silueta. Este resultado es muy pobre, indicando un solapamiento de los clústeres y una incapacidad para agrupar los datos correctamente.
- Conjunto de datos estandarizado con fórmula permitiendo la presencia de valores atípicos: El resultado es que *k*=3 es el número de clústeres que mejor puntuación obtienen en el estadístico de la silueta, con 0.029 puntos, un valor insuficiente para lograr una agrupación correcta.
- Conjunto de datos estandarizado mediante Min-Max y sin presencia de valores atípicos: En este caso, k=10 es el valor que optimiza la curva, obteniendo una pobre puntuación de 0.0137. Se trata de valores que siguen muy lejos del objetivo de crear clústeres bien definidos.
- Conjunto de datos estandarizado mediante Min-Max permitiendo la presencia de valores atípicos: Esta versión del conjunto maximiza la curva con *k*=5 y una puntuación máxima de 0.0132. Al igual que en su homólogo sin valores atípicos, K-Medoides sigue sin distribuir eficazmente los datos en clústeres.

- Conjunto de datos estandarizado mediante Robust Scaler y sin presencia de valores atípicos: Esta versión del conjunto maximiza la curva con k=2 y una puntuación de 0.1218. Aunque es el mejor de los resultados obtenidos hasta el momento, aún no se logra definir adecuadamente los clústeres.
- Conjunto de datos estandarizado mediante Robust Scaler permitiendo la presencia de valores atípicos: Este subconjunto vuelve a ser el que mejores datos obtiene. En el caso de K-Means llegó a obtener un 0.623 en la puntuación de la silueta. Sin embargo, en este caso, sólo logra 0.2332 para k=2. Trataremos de representar los clústeres que K-Medoides define para este conjunto, sin embargo, la escasa puntuación nos hace indicar que no será posible distinguir adecuadamente cada grupo.

Dada la alta dimensionalidad del conjunto de datos, es necesario recurrir nuevamente al Análisis de Componentes Principales para su representación. Este método revela que 21 de las características de nuestro conjunto explican el 95% de la varianza. Hemos decidido seleccionar únicamente 10 de estos componentes principales, los cuales aún concentran un 87% de la varianza total. Esta elección nos permite representar de manera más eficiente los diferentes clústeres en un espacio de menor dimensión.

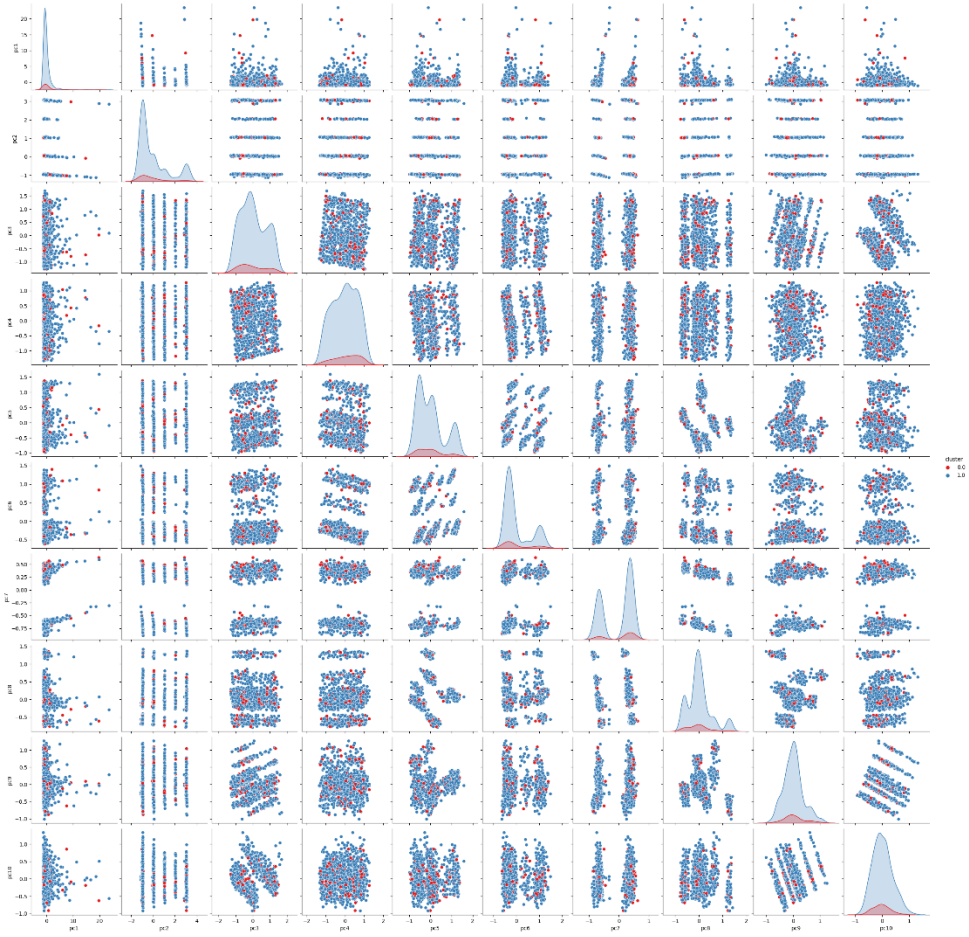


Ilustración 28. Representación de los clústeres obtenidos mediante K-Medoides a través de sus componentes principales. **Fuente Propia**

La Ilustración 28 nos proporciona rápidamente una visión de la difícil apreciación de los clústeres generados por el algoritmo K-Medoides. Los grupos 0 y 1 están representados en color rojo y azul, respectivamente. Este gráfico de pares muestra claros solapamientos en todas sus versiones, con puntos rojos presentes en los espacios ocupados por los azules. De esta observación, podemos concluir que K-Medoides no ha sido efectivo y que es necesario proceder con el análisis de algoritmos alternativos que puedan resultar más óptimos para nuestro conjunto de datos.

4.4.3 Clustering Jerárquico

El Clustering Jerárquico es un enfoque completamente distinto al que veníamos practicando con algoritmos como K-Means y k-Medoides. El algoritmo comienza considerando cada observación como un clúster individual y, a medida que avanza, agrupa iterativamente los clústeres más similares hasta formar una jerarquía de grupos y subgrupos con valores similares entre ellos y diferentes a los de otros clústeres. Existen dos variantes principales de Clustering Jerárquico: el Clustering Jerárquico Divisivo, donde se inicia con un solo clúster que se divide en grupos más pequeños a lo largo del proceso, y el Clustering Jerárquico Aglomerativo, que comienza con clústeres individuales y va agrupándolos para formar clústeres más grandes, finalizando con un clúster que abarca todo el conjunto de datos (Raschka, S., & Mirjalili, V., 2017).

Estas divisiones se establecen en base a un orden o jerarquía que puede ser representada en un gráfico de árbol de decisión llamado dendrograma, mostrando la relación de similitud entre los diferentes clústeres y subclústeres. A diferencia de K-Means o K-Medoides, esto hace innecesario proporcionar el número de clústeres que se quieren generar. Tan solo habrá que visualizar el dendrograma para observar una estructura clara de la distribución de los clústeres (Hastie, T., et, Al., 2001). No obstante, es importante comprender que hay una diferencia entre las divisiones que lleva a cabo el algoritmo y la conveniencia de mantener la estructura completa de clústeres. Al igual que en otros métodos, es crucial encontrar el número de clústeres que maximice la distribución de nuestro conjunto de datos. Algunas fuentes sugieren la posibilidad de cortar horizontalmente el dendrograma en el punto donde se minimizan las distancias entre los clústeres. Sin embargo, esta decisión dependerá del problema y del conjunto de datos, por lo que puede considerarse una opción subjetiva.

En este estudio hemos optado por mantener un criterio de coherencia al evaluar la elección del número de clústeres, por lo que utilizaremos el Índice de la Silueta, un coeficiente que nos ha sido útil previamente para determinar el número óptimo de clústeres en otros algoritmos. Además, introduciremos otro índice conocido como Coeficiente de Correlación Cofenética, el cual mide la similitud entre las distancias de los clústeres y las presentes en las agrupaciones del conjunto de

datos original (Saraçli, et al., 2013). En este contexto, se trata de un valor comparativo, entendiendo que a medida que el índice aumenta, la calidad de los clústeres generados es mejor. Esto nos permitirá comprender la eficiencia del algoritmo en un conjunto de datos, comparándolo con su aplicación en otro distinto (Gere, A., 2023).

A la hora de ejecutar el algoritmo, nos enfrentamos al mismo desafío que surgió al considerar la implementación de K-Medoides. El tamaño de nuestro conjunto de datos es demasiado grande, lo que resulta en errores de memoria insuficiente, impidiendo la interacción con el conjunto de datos completo. Por lo tanto, debemos comenzar creando, una vez más, una muestra de nuestro conjunto de datos:

```
subset_df = df.sample(frac=0.2, random_state=42)
```

A continuación, podemos proceder a representar el dendrograma, donde se establecerá la estructura de clústeres de nuestro conjunto de datos. Utilizamos el método de Ward en el Clustering Jerárquico, que consiste en "unir solo las agrupaciones que incrementan mínimamente la suma de errores cuadráticos dentro del clúster"(Raschka, S., y Mirjalili, V., 2017., p.364), o lo que es equivalente, minimizar el incremento de la varianza:

```
plt.figure(figsize=(10, 7))
dendogram = sch.dendrogram(sch.linkage(subset_df, method="ward"))
plt.title('Dendrograma')
plt.xlabel('Pedidos')
plt.ylabel('Distancia')
plt.show()
```

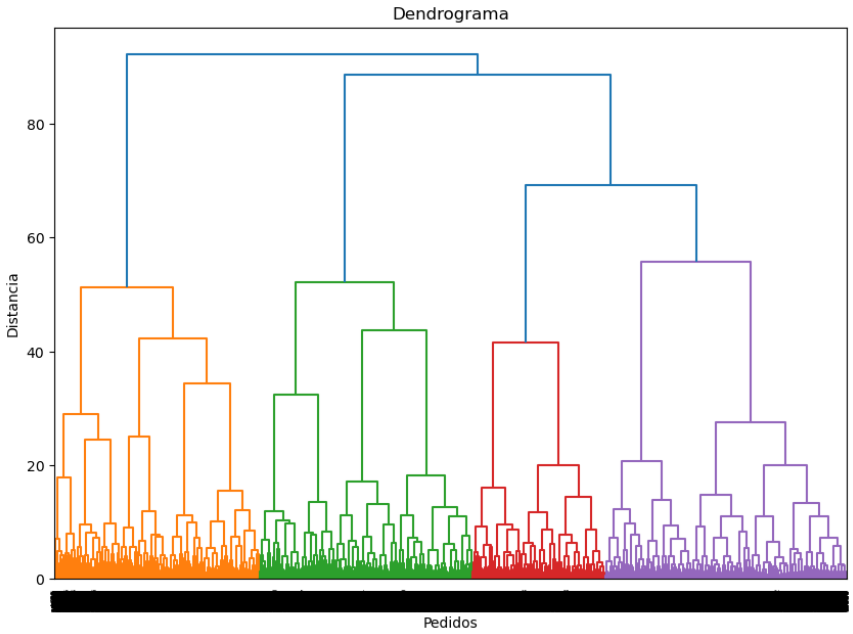


Ilustración 29. Dendrograma que establece los clústeres generados mediante Clustering Jerárquico para los diferentes pedidos del conjunto de datos. **Fuente: Propia**

La Ilustración 29 nos muestra el dendrograma generado dentro de un conjunto de datos normalizado mediante Min-Max y sin presencia de valores atípicos. Si seguimos las indicaciones referidas en la literatura, podríamos realizar un corte horizontal tras las líneas de color azul, que forman los clústeres con mayor distancia de la figura. Esto nos dejaría con un mínimo de 6 clústeres en los que dividir eficientemente nuestro conjunto de datos. Sin embargo, para constatar las observaciones que podemos llevar a cabo visualmente, realizaremos, como hemos mencionado anteriormente, el Índice de la Silueta y el Coeficiente de Correlación Cofenética. De esta forma, podremos alcanzar una conclusión con mayor información disponible, así como contrastar los resultados obtenidos en los diferentes subconjuntos.

Para el cálculo del Coeficiente de Correlación Cofenética, utilizaremos la librería SciPy. En primer lugar, calcularemos la matriz de distancias del conjunto de datos original. A continuación, generaremos otra matriz con las distancias creadas mediante el método de Ward y, finalmente, calcularemos el coeficiente según lo descrito en el siguiente código:

```
from scipy.cluster.hierarchy import dendrogram, linkage, cophenet
from scipy.spatial.distance import pdist
distance_matrix = pdist(subset_df)
linkage_matrix = linkage(distance_matrix, method='ward')
coph_corr, coph_dist = cophenet(linkage_matrix, distance_matrix)
print(f"Coeficiente de correlación cofenética: {coph_corr}")
Coeficiente de correlación cofenética: 0.5759596973728278
```

Procedemos a llevar a cabo el análisis del Índice de la Silueta y Coeficiente de Correlación Cofenética en los diferentes subconjuntos de datos estudiados en secciones anteriores:

- Conjunto de datos estandarizado con fórmula y sin presencia de valores atípicos: El dendrograma indica que k=3 puede ser un número de clústeres apropiado. Tras ejecutar el Índice de la Silueta, nos confirma que este es el valor que maximiza la curva, con una puntuación de 0.1711. Por otro lado, el Coeficiente de Correlación Cofenética nos arroja un valor de 0.5759. Aunque debemos contextualizar este último índice, la prueba de la silueta ya nos alerta de escasa distancia entre los clústeres.
- Conjunto de datos estandarizado con fórmula permitiendo la presencia de valores atípicos: El resultado es que k=2 es el número de clústeres con mayor puntuación en el índice de la silueta, con 0.2513 puntos. En lo referente al Coeficiente de Correlación Cofenética, consigue un valor de 0.5056. Observamos que, a pesar de que este número de clústeres optimiza mejor la Curva de la Silueta, el estadístico de las distancias cofenéticas es inferior al obtenido en el anterior subconjunto.
- Conjunto de datos estandarizado mediante Min-Max y sin presencia de valores atípicos: En este caso, k=12 es el número de clústeres que optimizan la curva, obteniendo

puntuación de 0.3678. El Coeficiente de Correlación Cofenética arroja un valor de 0.6980.

- Conjunto de datos estandarizado mediante Min-Max permitiendo la presencia de valores atípicos: Esta versión adopta *k*=4 con una puntuación de 0.1468. Sin embargo, el Coeficiente de Correlación Cofenética se mantiene a la altura de anteriores subconjuntos, con un valor de 0.6011.
- Conjunto de datos estandarizado mediante Robust Scaler y sin presencia de valores atípicos: Esta versión del conjunto maximiza la curva con *k*=2 y una puntuación de 0.2540. El Coeficiente de Correlación Cofenética obtiene, por su parte, un valor de 0.6614.
- Conjunto de datos estandarizado mediante Robust Scaler permitiendo la presencia de valores atípicos: Este subconjunto obtiene una puntuación de 0.7074 en el Índice de la Silueta para *k*=2. Sin embargo, el Coeficiente de Correlación Cofenética sólo ha conseguido un valor de 0.5288.

Tras revisar los datos anteriores, podemos llegar a diversas conclusiones. En primer lugar, el Coeficiente de Correlación Cofenética ha demostrado ser sensible a la presencia de valores atípicos, respaldando esta observación autores como Gere, A. (2023) y estudios previos (Sokal, R. R., & Rohlf, F. J., 1962). Además, se destaca nuevamente que el modelo estandarizado mediante Robust Scaler y con presencia de valores atípicos es el que obtiene la mejor puntuación en el Índice de la Silueta. Sin embargo, como mencionamos anteriormente, es crucial considerar la evaluación de ambos estadísticos, Silueta y Coeficiente de Correlación Cofenética. En este último caso, el subconjunto de datos mencionado obtiene una puntuación más baja que los demás. De hecho, su dendrograma ha resultado más complicado de interpretar, como se muestra en la siguiente ilustración:

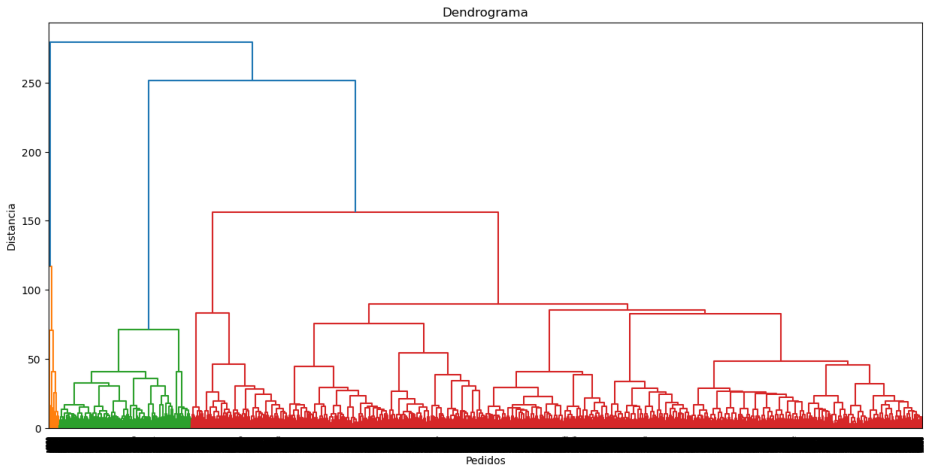


Ilustración 30. Dendrograma que representa los clústeres generados mediante Clustering Jerárquico en el conjunto de datos estandarizado mediante Robust Scaler y con presencia de valores atípicos**. Fuente: Propia**

Tras llegar a estas conclusiones, hemos decidido que el conjunto más óptimo para su representación es aquel estandarizado mediante Min-Max sin la presencia de valores atípicos. Este conjunto ha obtenido 0.3678 en el Índice de la Curva y 0.6980 en el Coeficiente de Correlación Cofenética.

El dendrograma de este subconjunto se presenta en la Ilustración 29, la cual analizamos en pasos anteriores. Inicialmente, estimamos que podríamos generar un mínimo de 6 clústeres, pero, como hemos comprobado, finalmente se ha aumentado hasta 12 para encontrar el punto que maximiza la Curva de la Silueta.

Aunque el dendrograma es una visualización clásica para el clustering jerárquico, resulta complicado obtener información precisa en conjuntos de datos con muchas dimensiones, ya que no podemos diferenciar los valores individuales desde los cuales parte el algoritmo a la hora de clasificar. Por lo tanto, recurrimos nuevamente al Análisis de Componentes Principales para reducir la dimensionalidad de nuestro conjunto de datos. Aplicando el método de la Curva de la Varianza explicada, que analizamos en el epígrafe 4.4.1 con el algoritmo K-Means, concluimos que el 95% de la varianza puede explicarse con solo un componente. Por lo tanto, llevaremos a cabo este proceso de reducción de dimensionalidad para 2 componentes principales y representaremos gráficamente los resultados, como hemos hecho en algoritmos anteriores.

```
pca = PCA(n_components=2)
principal_components = pca.fit_transform(subset_df)
pc_df = pd.DataFrame(data=principal_components, columns=['pc1', 'pc2'])
pca_result = pd.concat([pc_df, df_total['cluster'].reset_index(drop=True)], axis=1)
pca_result.head()
```

Con este código reducimos el subconjunto de datos que hemos obtenido de aplicar Clustering Jerárquico a tan sólo 2 componentes y lo imprimimos por pantalla. A continuación, procedemos a visualizar en 2D un diagrama de dispersión de cómo se distribuyen los valores de ambos componentes principales entre los 12 clústeres. Dado que, en esta ocasión, contamos con un gran número de agrupaciones, añadimos una línea de código que establece una leyenda que relaciona cada clúster con su correspondiente color, de cara a hacer más fácil su entendimiento.

```
plt.figure(figsize=(10, 5))
for cluster in pca_result['cluster'].unique():
    subset_cluster = pca_result[pca_result['cluster'] == cluster]
    plt.scatter(subset_cluster['pc2'], subset_cluster['pc1'], label=f'Cluster {cluster}')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.title('Visualización en 2D de las dos primeras componentes principales')
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```

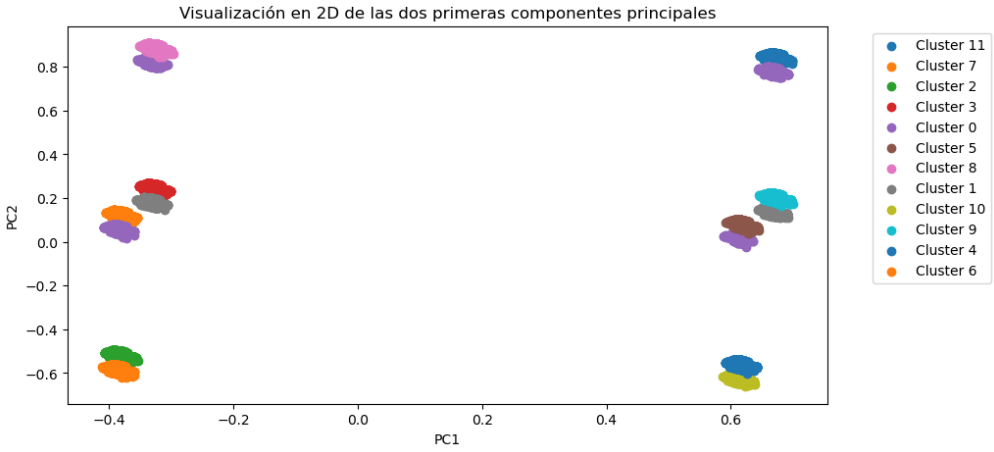


Ilustración 31. Diagrama de dispersión de los componentes principales y su reparto en clústeres generados mediante Clustering Jerárquico. **Fuente: Propia**

Observamos que algunos de los clústeres generados por el algoritmo exhiben una gran distancia en el gráfico, separándose aquellos que se encuentran en x=-0.4 y en x=0.6. Dentro de estos dos grupos, observamos subdivisiones, algunas en posiciones centrales y otras en los extremos. Cada uno de los clústeres se empareja específicamente con otro a una corta distancia, aunque sin llegar a solapar puntos. Esta configuración sigue el esquema que analizamos en el dendrograma de la Ilustración 29, donde existe una jerarquía que divide los datos en grupos y subgrupos. En dicha figura mencionada, si realizamos un corte horizontal entre y=40 e y=60, podríamos identificar los 12 clústeres representados aquí.

4.4.4 Clustering DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) es un algoritmo de clustering que se basa en la densidad de los datos en un conjunto, identificando clústeres y manejando el ruido, es decir, valores atípicos. Este modelo es particularmente adecuado para conjuntos de datos extensos, ya que no requiere conocimiento previo de los parámetros y es capaz de descubrir clústeres de diversas formas, como esféricas, lineales o alargadas. Además, DBSCAN muestra una resistencia significativa ante la presencia de valores extremos, los cuales no se asignan claramente a un clúster (Ester, M., Kriegel, H.-P., Sander, J., & Xu, X., 1996).

Este algoritmo identifica puntos centrales (core points) para cada clúster basándose en el número de vecinos que se encuentran dentro de un radio específico, conocido como épsilon (ε). Todo valor que se encuentre dentro de esta área, será clasificado dentro de ese clúster. Por otro lado, existen algunos puntos centrales que no pertenecen a ningún clúster ya que no se encuentran dentro del área de influencia de ninguno. En tales casos, el algoritmo los clasificará como puntos

de ruido (Raschka, S., & Mirjalili, V., 2017). Por lo tanto, este tipo de clustering genera divisiones en función de la densidad de los valores y cómo estos se agrupan y distribuyen de manera natural.

Para aplicar este algoritmo en nuestro conjunto de datos, volvemos a acudir a Scikit-Learn, una librería muy extendida en el mundo del Aprendizaje Automático.

Al igual que en anteriores métodos como K-Medoides y Clustering Jerárquico, encontramos la necesidad de comenzar reduciendo el tamaño de nuestra muestra, ya que el procesamiento con el conjunto de datos completo da lugar al error presentado en la Ilustración 26, donde se muestra la insuficiente memoria para ejecutar completamente el proceso. Para ello, volvemos a generar un subconjunto aleatorio con el 20% de los datos del original:

```
subset_df = df.sample(frac=0.2, random_state=42)
```

A continuación, importamos el método DBSCAN de la librería Scikit-Learn y ejecutamos un algoritmo para ε=3 y min_samples=2. Épsilon, como hemos comentado, mide la distancia que tiene que haber entre dos valores para poder considerarse vecinos de clúster. Por el otro lado, min_samples indica el número de valores necesarios para poder considerase un punto central o core point. La elección de estos valores la primera vez que probamos el algoritmo es estimada y debe repetir si el número de clústeres no es el adecuado, tanto por exceso como por defecto. En el caso que nos ocupa, trataremos de reproducir el algoritmo en cada uno de los subconjuntos de datos que hemos analizado, variando su método de estandarización y la presencia de valores atípicos. Además, nos ayudaremos, una vez más, del Coeficiente de la Silueta para poder tener una idea de cómo de bien se separan los clusters unos de otros.

- Conjunto de datos estandarizado con fórmula y sin presencia de valores atípicos: En este caso, logramos encontrar una generación estable de clústeres para ε=2.5 y min_samples=30. Como resultado, se generaron 10 clústeres diferentes y se identificaron los valores atípicos dentro de un undécimo clúster. El Coeficiente de la Silueta Resultante es de 0.27.
- Conjunto de datos estandarizado con fórmula permitiendo la presencia de valores atípicos: En este caso, hemos logrado generar 5 clústeres y un sexto con valores atípicos para ε=15 y min_samples=40. El Coeficiente de la Silueta obtenido es de 0.477, algo mejor que su contraparte sin valores atípicos.
- Conjunto de datos estandarizado mediante Min-Max y sin presencia de valores atípicos: Para este subconjunto, se han obtenido 15 clústeres para ε=0.75 y min_samples=20. El Coeficiente de la Silueta obtiene 0.563 puntos.
- Conjunto de datos estandarizado mediante Min-Max permitiendo la presencia de valores atípicos: En esta versión, encontramos dificultades significativas para poder encontrar valores de ε y min_samples que generen un número razonable de clústers. Al aumentar

sus valores, el algoritmo tiende a agrupar todos los datos de la muestra en un único clúster. Sin embargo, al reducir, el algoritmo dividía el subconjunto en un gran número de clústeres. Finalmente, con ε=1.5 y min_samples=100 se lograron generar 45 clústeres con un Coeficiente de la Silueta de 0.347. Esto nos puede indicar que nuestro conjunto de datos no cuenta con una gran densidad y, a pesar del número de entradas, su dispersión es tal que algoritmos de densidad se muestran altamente sensibles y pueden resultar ineficaces a la hora de clasificar nuestros valores.

- Conjunto de datos estandarizado mediante Robust Scaler y sin presencia de valores atípicos: En el caso de este subconjunto, encontramos resultados similares a los comentados en la anterior muestra. El algoritmo tiene dificultades a la hora de clasificar nuestros datos, tendiendo a agrupar la mayoría de ellos en un único clúster. Finalmente, logra generar 8 clústeres y un noveno con valores que DBSCAN considera atípicos. Para ello, se ha configurado ε=1 y min_samples=10, alcanzando una puntuación de 0.122 en el Coeficiente de la Silueta.
- Conjunto de datos estandarizado mediante Robust Scaler permitiendo la presencia de valores atípicos: Finalmente, el algoritmo logra generar 4 clústeres para este subconjunto, uno de ellos con valores que considera atípicos. Ha sido necesario configurar ε=2.5 y min_samples=10, logrando la máxima puntuación del Coeficiente de la Silueta en este tipo de clustering, con 0.629 puntos. Sin embargo, debemos destacar nuevamente la desproporción existente entre los diferentes clústeres, concentrándose la mayoría de puntos en el grupo 0.

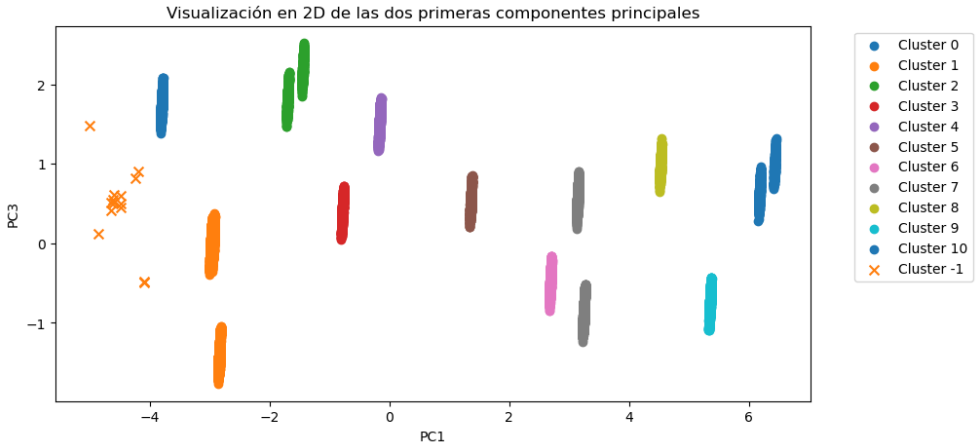
Una vez ejecutado el algoritmo en nuestros subconjuntos, debemos destacar la dificultad por encontrar un valor adecuado para ε y min_samples. Este método ha demostrado una alta sensibilidad frente a nuestro conjunto de datos, pudiendo discutir que éste cuenta con una excesiva dispersión, algo que impediría medir adecuadamente la densidad de los datos y, por lo tanto, clasificarlos en grupos.

Para la representación de los resultados, haremos uso del Análisis de Componentes Principales para mostrar la distribución de los datos en los clústeres obtenidos. Comenzaremos el diagrama de dispersión del conjunto de datos estandarizado mediante fórmula de estandarización y sin presencia de valores atípicos, el cual no ha obtenido la mejor puntuación, pero cuya figura simboliza muy bien la distribución de nuestros datos:

```
plt.figure(figsize=(10, 5))
for cluster in pca_result['Cluster'].unique():
  subset_cluster = pca_result[pca_result['Cluster'] == cluster]
  if cluster == -1:
    plt.scatter(subset_cluster['pc1'], subset_cluster['pc3'], label='Cluster -1', marker='x', s=50)
  else:
```

```
    plt.scatter(subset_cluster['pc1'], subset_cluster['pc3'], label=f'Cluster {cluster}')
plt.xlabel('PC1')
plt.ylabel('PC3')
plt.title('Visualización en 2D de las dos primeras componentes principales')
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```

Este código es similar al utilizado anteriormente para representar la distribución de otros clústeres. Sin embargo, añadimos una línea condicional que modifica el marcador para los valores del clúster -1. La documentación de Scikit-Learn establece que el algoritmo clasificará en el clúster número -1 a todos aquellos valores que considere atípicos. Dado que este subconjunto se ha dividido en 11 clústeres, debemos tomar precauciones que aseguren el completo entendimiento del gráfico.



Illustración 32. Representación de la distribución de los componentes principales entre los clústeres generados mediante DBSCAN. **Fuente: Propia**

La Ilustración nos muestra que, a pesar de obtener sólo 0.27 puntos en la curva de la silueta, los clústeres parecen estar separados entre sí de forma adecuada. Por otro lado, podemos observar, representados por el símbolo “x”, los valores que el algoritmo ha considerado como atípicos y que no ha sido capaz de integrar dentro del resto de clústeres.

Finalmente, representamos de forma similar el conjunto estandarizado mediante Robust Scaler en el que permitimos la presencia de valores atípicos. Este modelo es el que ha obtenido la puntuación más alta en la curva de la silueta, con 0.629. Este gráfico demuestra, por un lado, la clara tendencia del algoritmo a concentrar los datos en el clúster 0, hasta tal punto que clasifica como atípicos valores que otros modelos sí son capaces de clasificar. De hecho, el clúster -1 es el segundo con mayor número de puntos, por lo que evidencia la teoría que exponíamos anteriormente: Un conjunto de datos ampliamente disperso puede resultar inadecuado para la aplicación de algoritmos de densidad.

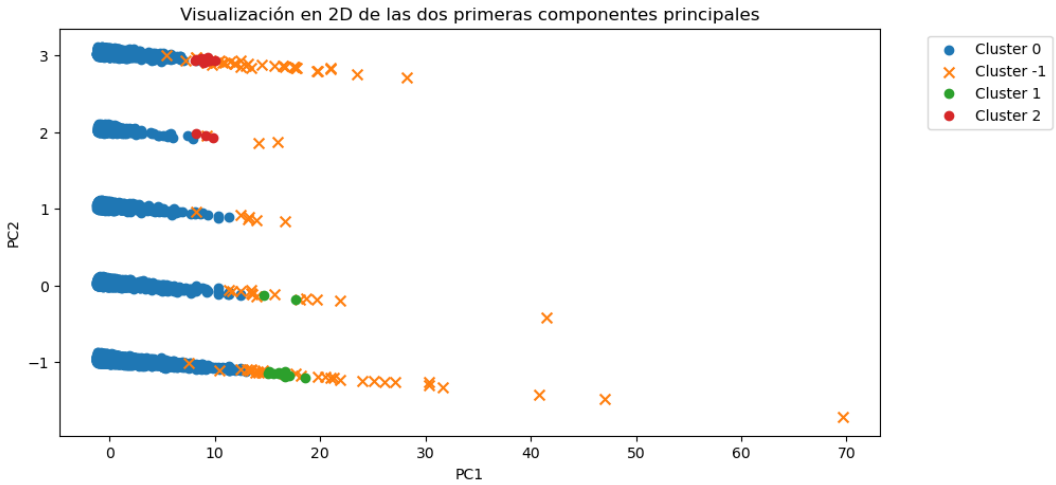


Ilustración 33. Representación de la distribución de los componentes principales entre los clústeres generados mediante DBSCAN. **Fuente: Propia**

Si no existe una alta densidad de elementos dentro de un área o áreas definidas, el modelo se centrará en concentraciones más pequeñas, aumentando significativamente el número de clústeres, o abarcará la mayor cantidad de datos posibles y siendo incapaz de distinguir otras diferencias. Por otro lado, debemos tener en cuenta que este algoritmo se ha ejecutado sobre muestras aleatorias extraídas de conjuntos que además han sido estandarizados de maneras distintas y con diferentes formas de tratar los valores atípicos. Por lo tanto, la densidad de cada subconjunto podría ser significativamente diferente y de ahí las dificultades encontradas con la aplicación de DBSCAN.

5. Conclusiones

Tras el desarrollo de los métodos y algoritmos recogidos a lo largo de este estudio, hemos logrado alcanzar un entendimiento avanzado sobre la relevancia del Aprendizaje Automático en el ámbito del Comercio Electrónico. Teniendo en cuenta el conjunto de datos utilizado, así como las técnicas empleadas para segmentar los datos de clientes, procedemos a presentar las siguientes conclusiones:

- La dimensión del conjunto de datos es un factor clave a la hora de ejecutar algoritmos de clustering. La elección de las características objetivo es un proceso crucial que se debe realizar antes de llevar a cabo cualquier intento de segmentación. Un conjunto de datos con demasiadas variables o con escaso valor informativo, tiene como consecuencia la ineficiencia de los algoritmos, así como la obtención de resultados poco concluyentes.
- En relación con la dimensión del conjunto de datos, es imprescindible valorar la tipología de las variables que se han escogido para generar clústers. Los algoritmos de clustering sólo son capaces de procesar valores numéricos, por lo que la inclusión de variables categóricas requiere la transformación de las mismas mediante técnicas como Label Encoder o One-Hot-Encoding. La primera sustituye cada valor único en formato cadena por números. Esta elección es más apropiada para variables categóricas cuyos valores cuentan con un orden o jerarquía, en tanto que los algoritmos de clustering podrían entender que una entrada vale más que otra, al estar representado por un número superior. Por el contrario, One-Hot-Encoding es la técnica ideal para variables cuyos valores no siguen jerarquía alguna, es decir, no existe diferencia de tamaño ni pondera un valor más que otro.

  Considerando la elección de One-Hot-Encoding, la encontramos un factor determinante a tener en cuenta. Su ejecución conlleva la creación de nuevas variables binarias, por lo que en conjuntos de datos como el analizado en este estudio, conllevan el incremento de la dimensionalidad en tantas nuevas categorías como valores únicos tenga la variable original.
- Como solución al incremento de la dimensionalidad, destacamos el agrupamiento de valores mediante ingeniería de características. Es decir, conociendo nuestro conjunto de datos, podemos crear nuevas variables o puntos en los que englobar un grupo de valores. Utilizamos esta técnica, por ejemplo, para representar las 10 ciudades con mayor número de pedidos y agrupar el resto de ellos en el valor ‘otros’. De esta forma, lo que antes habría supuesto aumentar en más de 4000 variables nuestro conjunto de datos, ahora sólo se incrementa en 11 tras la aplicación de One-Hot-Encoding.

- La mayoría de algoritmos requieren un conjunto de datos estandarizado para conseguir mejores resultados a la hora de generar clústeres. En este estudio, comprobamos que no sólo es importante estandarizar, sino también la elección del método para llevarla a cabo. Ante la incertidumbre, decidimos ejecutar nuestros algoritmos de clustering en conjuntos de datos estandarizados mediante diversas técnicas, obteniendo diferencias significativas en el rendimiento de los mismos. Así, para este conjunto de datos, la estandarización mediante Robust Scaler ha resultado la más eficaz.
- Otro factor imprescindible que debemos destacar es el estudio de los valores atípicos. La mayor parte de la literatura aconseja su eliminación como paso previo a la ejecución de algoritmos de clustering. Sin embargo, resulta crucial su representación y el análisis de los datos para comprender el origen de dichos valores. Desestimar los valores atípicos sin estudio previo, puede llevar a una pérdida significativa de información que algunos algoritmos son capaces de procesar y aprovechar. Al igual que con la estandarización, concluimos que, ante la duda, resulta muy útil ejecutar los algoritmos en conjuntos con presencia de valores atípicos para así comparar con los resultados de sus contrapartes donde se han reducido o eliminado.
- Hemos llevado a cabo la ejecución de los algoritmos K-Means, K-Medoides, Jerárquico Aglomerativo y DBSCAN en nuestro conjunto de datos. Los resultados han sido variados en función del método utilizado. Sin embargo, podemos destacar la alta puntuación del subconjunto de datos estandarizado mediante Robust Scaler y con presencia de valores atípicos. Este ha conseguido el mejor rendimiento para la mayoría de algoritmos, destacando de nuevo la necesidad de estudiar y considerar la presencia de anomalías.
- Entre los algoritmos utilizados, queremos hacer las siguientes observaciones:
    - Los métodos Jerarquizado Aglomerativo y K-Means han sido los más eficientes en clasificar los clientes, dado el conjunto de datos utilizado.
    - La dispersión de los valores del conjunto ha sido un hándicap a la hora de maximizar los resultados del clústering. Destacamos la incapacidad del algoritmo DBSCAN para desarrollar todo su potencial en conjuntos sin una densidad clara de valores.
    - El tamaño del conjunto de datos ha ocasionado problemas a la hora de ejecutar clústering Jerárquico Aglomerativo, K-Medoides y DBSCAN, al saturar la memoria utilizada de la computadora. La solución propuesta en estos casos, ha sido la creación de un subconjunto con el 20% de los valores del original. Esto, sin embargo, ha podido tener consecuencias en cuanto a la calidad de los resultados obtenidos.

Finalmente, queremos concluir este trabajo destacando la importancia del Aprendizaje Automático y la Ciencia de Datos en un sector como el Comercio Electrónico. El auge del e-Commerce ha facilitado a compañías de todo el mundo la obtención de datos de sus usuarios de forma masiva y eficaz. La utilización de esta información es una ventaja definitiva frente a negocios tradicionales, al poder utilizar metodologías y técnicas modernas de segmentación con la que conocer a los usuarios potenciales, fidelizar clientes y ganar recurrencia de compra, así como diversificar negocios y llevar a cabo estrategias a gran escala basándonos en el conocimiento real y sólo visible mediante un tratamiento adecuado de los datos.

6. Referencias Bibliográficas

Albon, C. (2018). *Machine Learning with Python Cookbook: Practical Solutions from preprocessing to deep learning.* O'Reilly Media, Inc., pp. 83-84.

Amat Rodrigo, J. (2017, junio). *Análisis de Componentes Principales (Principal Component Analysis, PCA) y t-SNE.* Recuperado el 21 de enero de 2024, de https://www.cienciadedatos.net/documentos/35_principal_component_analysis bajo licencia CC BY-NC-SA 4.0.

Amazon Ads (2023). *Segmentación de mercado: Definición, tipos, beneficios y ejemplos.* Recuperado el 28 de diciembre de 2023, de https://advertising.amazon.com/library/guides/market-segmentation

Baena Graciá, Verónica. (2012). *Fundamentos de marketing.* UOC.

Barai (Deb), A., y Dey, L. (2017). *Outlier Detection and Removal Algorithm in K-Means and Hierarchical Clustering. World Journal of Computer Application and Technology, 5*(2), 24-29. DOI: 10.13189/wjcat.2017.050202. p. 24. Recuperado el 14 de enero de 2024, desde https://www.hrpub.org/download/20170830/WJCAT2-13708454.pdf

Barrios, A. (2023, 8 de agosto). *Tutorial del Algoritmo K-Means en Python.* LatinXinAI. Medium. Recuperado el 20 de enero de 2024, de https://medium.com/latinxinai/tutorial-del-algoritmo-k-means-en-python-d8055751e2f3#:~:text=M%C3%A9todo%20del%20codo%20(Elbow%20Method,curva%20similar%20a%20un%20codo.

Brownlee, J. (2019, August 8). *A Gentle Introduction to Normality Tests in Python.* Machine Learning Mastery. https://machinelearningmastery.com/a-gentle-introduction-to-normality-tests-in-python/

Comisión Nacional de los Mercados y la Competencia, (2023). *Volumen de negocio de comercio electrónico CNMC.* Comisión Nacional de los Mercados y la Competencia. data.cnmc.es. Recuperado el 28 de diciembre de 2023, de https://es.statista.com/estadisticas/496407/ingresos-por-ventas-en-el-comercio-electronico-en-espana/

Editorial Elearning. (n.d.). *Dirección de marketing.* Vértice.

Editorial Elearning. (n.d.). *UF0032 - Venta online.* Vértice.

Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. (1996). *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise.* Institute for Computer Science, University of Munich. Oettingenstr. 67, D-80538 München, Germany. Recuperado de https://file.biolab.si/papers/1996-DBSCAN-KDD.pdf

Fraguela, Noelia (2023). *Top 20: las mejores herramientas de opiniones online para tu eCommerce (2023).* Marketing 4Ecommerce. Recuperado el 28 de diciembre de 2023, de https://marketing4ecommerce.net/mejores-herramientas-opiniones-online/

Gere, A. (2023). *Recommendations for validating hierarchical clustering in consumer sensory projects. Current Research in Food Science, 6*(100522). Recuperado el 23 de enero de 2024, desde https://doi.org/10.1016/j.crfs.2023.100522

Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow.* O'Reilly Media, Inc., p. 71.

Gonzalo, Á. (2019). *Segmentación utilizando K-Means en Python.* Machine Learning Para Todos. Recuperado de https://machinelearningparatodos.com/segmentacion-utilizando-k-means-en-python/

Hastie, T., Tibshirani, R., y Friedman, J. (2001). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction (2da ed.).* Springer Science & Business Media.

Holtz, Y. (2023). *Scatterplot and log scale in Matplotlib.* Python Graph Gallery. Recuperado el 19 de enero de 2024, de https://python-graph-gallery.com/scatterplot-and-log-scale-in-matplotlib/

Jin, X., Han, J. (2011). *K-Medoids Clustering.* Sammut, C., Webb, G.I. (eds) Encyclopedia of Machine Learning. Springer, Boston, MA. Recuperado el 22 de enero de 2024, desde https://doi.org/10.1007/978-0-387-30164-8_426

Kütz, Martin (2016). *Introduction to Ecommerce: Combining Business and Information Technology.* Martin Kütz y Bookboon.com.

Mahesh, B. (2020). *Machine Learning Algorithms - A Review. International Journal of Science and Research (IJSR), 9*(1), p. 381. Recuperado el 30 de diciembre de 2023, en: https://www.ijsr.net

Miguel Santesmases Mestre. (2012). *Marketing.* Ediciones Pirámide.

Mitchell, T. M. (1997). *Machine Learning.* McGraw-Hill Science/Engineering/Math.

Müller, A. C., & Guido, S. (2017). *Introduction to Machine Learning with Python.* O'Reilly Media, Inc.

Nagar, A. (2020, January 26). *K-means Clustering — Everything you need to know.* Analytics Vidhya. Medium. Recuperado el 18 de enero de 2024, desde https://medium.com/analytics-vidhya/k-means-clustering-everything-you-need-to-know-175dd01766d5

Oracle. (s.f.). *IQR (Rango intercuartílico).* Oracle® Fusion Cloud EPM Trabajo con Planning. Recuperado el 14 de enero de 2024, desde https://docs.oracle.com/cloud/help/es/pbcs_common/PFUSU/insights_metrics_IQR.htm#PFUSU-GUID-CF37CAEA-730B-4346-801E-64612719FF6B

Osborne, J. W., y Overbay, A. (2019). *The power of outliers (and why researchers should ALWAYS check for them). Practical Assessment, Research, and Evaluation, 9*(1), Article 6. https://doi.org/10.7275/qf69-7k43

Raschka, S., & Mirjalili, V. (2017). *Python Machine Learning.* Packt Publishing., p. 116.

Ródenas Gascó, V. J. (2015). *Consulta SQL para exportar productos con atributos en Prestashop.* Recuperado el 31 de diciembre de 2023, de https://victor-rodenas.com/2015/12/20/consulta-sql-para-exportar-productos-con-atributos-en-prestashop/

Rois, Susana (2023). *Pequeña (gran) historia del eCommerce en España.* Marketing 4Ecommerce. Recuperado el 28 de diciembre de 2023, de https://marketing4ecommerce.net/historia-del-ecommerce-en-espana/

Saraçli, S., Doğan, N., & Doğan, İ. (2013). *Comparison of hierarchical cluster analysis methods by cophenetic correlation. Journal of Inequalities and Applications, 2013*(1), 203. Recuperado el 23 de enero de 2024, desde https://doi.org/10.1186/1029-242X-2013-203

Scikit-Learn (2024). *Getting Started.* Recuperado el 02 de enero de 2024, desde: https://scikit-learn.org/stable/getting_started.html

Scikit-Learn. (2024). *Demonstration of k-means assumptions.* Recuperado el 18 de enero de 2024, de https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_assumptions.html

Scikit-Learn. (2024). *MinMaxScaler.* Recuperado el 17 de enero de 2024, de https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

Scikit-Learn. (2024). *RobustScaler.* Recuperado el 17 de enero de 2024, de https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html

Scikit-Learn. (2024). *silhouette_score.* En: *scikit-learn: Machine Learning in Python.* Recuperado de https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html

Scikit-Learn. (2024). *sklearn.cluster.DBSCAN. Scikit-learn: Machine Learning in Python.* Recuperado el 24 de enero de 2024, desde https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html

Seaborn. (2012-2023, Michael Waskom). (s.f.). *Boxplot.* Documentación de Seaborn. Recuperado el 14 de enero de 2024, desde: https://seaborn.pydata.org/generated/seaborn.boxplot.html

Sionek, A. (2017). *Brazilian E-Commerce Public Dataset by Olist.* Kaggle. Recuperado el 02 de enero de 2024, desde: https://www.kaggle.com/datasets/olistbr/brazilian-ecommerce

Smith, W. R. (1956). *Product Differentiation and Market Segmentation as Alternative Marketing Strategies. Journal of Marketing, 21*(1), 3-8. Recuperado el 29 de diciembre de 2023, en: http://www.jstor.org/stable/1247695

Sokal, R. R., & Rohlf, F. J. (1962). *The comparison of dendrograms by objective methods. Taxon, 11*(2), 33-40. Recuperado el 23 de enero de 2024, desde https://doi.org/10.2307/1217208

Soleymani, A. (2022, March 11). *What is K-Medoids Clustering and When should you use it instead of K-Means.* Medium. https://medium.com/@ali.soleymani.co/beyond-scikit-learn-is-it-time-to-retire-k-means-and-use-this-method-instead-b8eb9ca9079a

Terra, J (2023). *Why Python is essential for Data Analysis and Data Science.* Simplilearn. Recuperado el 03 de enero de 2024, desde: https://www.simplilearn.com/why-python-is-essential-for-data-analysis-article

Turban, E., King, D., Lee, K. L., Liang, T. P., & Turban, C. T. (2015). *Electronic Commerce.* Springer International Publishing.

Yankelovich, Daniel., Meer, David. (2006). *Rediscovering Market Segmentation. Harvard Business Review.* Recuperado el 28 de diciembre de 2023, de https://hbr.org/2006/02/rediscovering-market-segmentation

Zhao, Bohan. (2022). *Research on Using Market Segmentation to do Recommendation in E-commerce.* En *Proceedings of the 2022 7th International Conference on Financial Innovation and Economic Development (ICFIED 2022)* (pp. 3017-3022). Atlantis Press. Recuperado el 28 de diciembre de 2023, en: https://www.atlantis-press.com/article/125971546.pdf