Capstone Project | Master thesis

Proyecto End-to-End de Machine Learning

# DSMarket - your next generation store

**MÁSTER EN DATA SCIENCE & IA**

Nuclio Digital School

Barcelona


**Grupo Retail-3**

Bayarri, Eduardo

Cuenca, Victor

Jaén, Alan

Ruz, José Manuel


**Tutora**

Raquel Revilla

**Defensa**

Lunes, 16 de septiembre de 2024

**Entrega**

Jueves, 26 de septiembre de 2024

nuclio digital school

# Índice

# 1. Introducción

El presente proyecto se enmarca en la transformación digital de DSMarket, una cadena de centros comerciales que ha decidido modernizar sus procesos mediante la adopción de tecnologías basadas en datos, con el fin de optimizar sus operaciones y aumentar su competitividad. Durante los últimos años, el sector minorista ha experimentado una revolución tecnológica significativa, impulsada por el análisis de datos y la inteligencia artificial. DSMarket, como muchas empresas tradicionales, ha llegado tarde a esta transformación y se enfrenta a varios desafíos que requieren soluciones basadas en datos y enfoques innovadores.

Con el objetivo de mejorar la precisión en la toma de decisiones y aumentar la eficiencia operativa, DSMarket ha apostado por la implementación de técnicas avanzadas de ciencia de datos, centrándose en tres áreas críticas: predicción de ventas, segmentación de productos y tiendas, y optimización de inventarios. A través de este proyecto, se han utilizado algoritmos de machine learning, técnicas de clustering y herramientas de Business Intelligence (BI) para resolver estos problemas y ofrecer una ventaja competitiva a la empresa.

El proyecto se ha estructurado en varias fases clave, comenzando con un análisis exploratorio de datos (EDA) que permitió identificar patrones ocultos y estructurar los datos de manera adecuada para el modelado predictivo. Posteriormente, se desarrollaron modelos de predicción de ventas a 28 días vista, los cuales fueron integrados a los sistemas de inventario para automatizar y optimizar el reabastecimiento en las tiendas. También se aplicaron técnicas de agrupamiento para segmentar tanto productos como tiendas, lo que facilitó la creación de estrategias personalizadas para cada segmento.

Además de mejorar la precisión de las estimaciones actuales, este proyecto incluye la implementación de dashboards interactivos que permiten un monitoreo en tiempo real de los indicadores clave, ofreciendo a los directivos una visión clara y accionable para la toma de decisiones. Estas soluciones no solo abordan los problemas actuales de DSMarket, sino que también preparan a la empresa para el futuro, sentando las bases para una transformación más profunda hacia una organización data-driven.

Este documento describe en detalle los enfoques metodológicos adoptados, los resultados obtenidos y las conclusiones derivadas del análisis de datos y de la implementación de las soluciones propuestas. La implementación de estas herramientas y técnicas permitirá a

DSMarket adaptarse al entorno competitivo actual y continuar su proceso de digitalización con una infraestructura flexible y escalable.

# 2. Justificación y Contextualización

La transformación digital en el sector retail ha provocado un cambio radical en la forma en que las empresas gestionan sus operaciones, brindando nuevas oportunidades para optimizar procesos a través del análisis de datos. DSMarket, una cadena de centros comerciales, ha iniciado su transición hacia una empresa basada en datos, reconociendo la necesidad de mejorar la eficiencia y rentabilidad en áreas clave como la predicción de ventas y la gestión de inventarios. Este proyecto tiene como objetivo aplicar técnicas avanzadas de machine learning y análisis de datos para afrontar estos desafíos y lograr que las operaciones de la empresa sean más eficientes.

Lo que distingue a este proyecto es la colaboración de un equipo multidisciplinario de profesionales, cada uno con una sólida trayectoria en campos diversos, lo que permite abordar el reto desde diferentes perspectivas. El enfoque multidisciplinario ha sido crucial para entender y resolver problemas complejos dentro de DSMarket, ofreciendo una solución integral que optimiza tanto las decisiones estratégicas como operativas.

Bayarri, con formación en ciencias ambientales y un Máster en Ordenación Territorial y Gestión Ambiental, aporta un enfoque analítico y estratégico para trabajar con datos complejos. Su experiencia ha sido fundamental en la gestión de inventarios, ayudando a identificar formas de optimizar recursos y mejorar la eficiencia operativa de las tiendas.

Cuenca, especialista en Marketing y con estudios técnicos en Electrónica, contribuyó significativamente al análisis de tendencias de ventas y al comportamiento del cliente, lo que fue esencial para la segmentación de productos y tiendas. Su perfil técnico también ha sido clave en el desarrollo de dashboards interactivos, facilitando el monitoreo en tiempo real de los resultados.

Jaén, con un máster en Economía y Finanzas y una especialización en Gestión Cultural, ha aportado su experiencia en análisis financiero para evaluar el impacto económico de las soluciones implementadas. Su trabajo ha sido fundamental para estimar los beneficios financieros de la optimización de los procesos de reabastecimiento y la predicción de ventas, permitiendo presentar propuestas claras y precisas a la dirección de DSMarket.

Ruz, con un grado superior en Administración de Sistemas Informáticos y Redes (ASIR), ha liderado el desarrollo de infraestructuras tecnológicas robustas para gestionar grandes volúmenes de datos y automatizar procesos. Su experiencia ha sido clave para la integración y despliegue de las soluciones de machine learning, así como para asegurar que los sistemas de DSMarket puedan escalar de manera eficiente.

Esta combinación de perfiles no solo ha permitido abordar los desafíos desde múltiples ángulos —optimización operativa, gestión de inventarios, predicción de ventas y toma de decisiones estratégicas basadas en datos—, sino que también ha sentado las bases para que DSMarket adopte un enfoque más integral en su transformación digital. Gracias a este enfoque, la empresa ha podido aprovechar al máximo el potencial del análisis de datos y la inteligencia artificial, mejorando significativamente la toma de decisiones y aumentando su competitividad en el mercado actual.

Además, este proyecto ha proporcionado una oportunidad única para que los miembros del equipo apliquen y expandan sus conocimientos en un entorno real, utilizando herramientas avanzadas de ciencia de datos para resolver problemas específicos del sector retail. La colaboración interdisciplinaria y el uso de tecnología de vanguardia han sido esenciales para lograr soluciones innovadoras y escalables que preparan a DSMarket para el futuro.

# 3. Objetivos

## 3.1. Objetivo General

El objetivo general de este proyecto es desarrollar soluciones basadas en ciencia de datos que optimicen las operaciones y estrategias comerciales de DSMarket. Esto incluye la implementación de modelos predictivos de ventas, la segmentación de productos y tiendas, y la creación de herramientas de análisis que apoyen la toma de decisiones, contribuyendo así a la transformación de DSMarket en una empresa data-driven.

## 3.2. Objetivos Específicos

3.2.1. **Predecir las ventas a nivel de tienda y producto** mediante el desarrollo de modelos de machine learning que mejoren la precisión de las estimaciones actuales, proporcionando previsiones de ventas con un horizonte de 28 días.

3.2.2. **Identificar grupos de productos y tiendas con comportamientos similares** a través de la aplicación de técnicas de clustering, lo que permitirá optimizar campañas de marketing y ajustar las estrategias de inventario y precios.

3.2.3. **Desarrollar un sistema de Business Intelligence (BI)** que permita visualizar los resultados de los análisis de ventas y predicciones, facilitando un seguimiento en tiempo real de los indicadores clave para los equipos ejecutivos y de operaciones.

3.2.4. **Optimizar el proceso de reposición de inventario** mediante la integración de modelos predictivos que ajusten las cantidades de stock de manera más eficiente, minimizando el remanente y mejorando la rotación de productos en las tiendas.

# 4. Metodología

La metodología utilizada en este proyecto se basa en un enfoque estructurado, dividido en varias fases clave, siguiendo un enfoque de ciencia de datos end-to-end. A continuación, se describen los pasos seguidos en el desarrollo del proyecto:

## 4.1. Definición del problema

El primer paso fue identificar los principales desafíos que enfrenta DSMarket, una cadena de centros comerciales en proceso de transformación digital. Se plantearon diversos problemas que abarcan desde la predicción de ventas hasta la optimización del stock en tiendas. Este proceso fue guiado por el equipo directivo, especialmente la directora de finanzas y la directora de marketing, quienes asignaron las tareas iniciales.

## 4.2. Recopilación de datos

Se accedió a bases de datos proporcionadas por el equipo de tecnología de DSMarket. Estos datos incluyen información de ventas diarias, precios de productos y eventos relevantes que podrían afectar las ventas en las ciudades de Nueva York, Boston y Filadelfia. La recopilación se realizó utilizando herramientas de extracción de datos, las cuales almacenan las tablas relevantes en un entorno compartido para su análisis.

## 4.3. Exploración y limpieza de datos

Se llevó a cabo un análisis exploratorio inicial de los datos con el objetivo de comprender su estructura y características. Se identificaron problemas como valores faltantes y datos inconsistentes que se resolvieron mediante técnicas de limpieza de datos. Se estandarizaron formatos y se eliminaron outliers utilizando herramientas como Python y librerías como Pandas y NumPy.

## 4.4. Análisis y modelado

Una vez limpios los datos, se aplicaron técnicas de análisis estadístico y de machine learning. Para abordar los diferentes problemas planteados, se implementaron los siguientes enfoques:

4.4.1. **Análisis de ventas:** Se examinaron las tendencias de ventas globales y específicas por ciudad, así como la relación entre las variaciones de precios y los volúmenes de venta.

4.4.2. **Clustering de productos y tiendas:** Se utilizaron algoritmos de agrupamiento (clustering) para identificar grupos de productos con comportamientos similares y para analizar similitudes entre las tiendas, con el fin de optimizar campañas de marketing y estrategias de distribución.

4.4.3. **Predicción de ventas:** Se desarrolló un modelo de predicción de ventas basado en técnicas de machine learning supervisado, utilizando series temporales para prever las ventas a nivel de tienda y producto. Para este modelo, se emplearon técnicas de regresión y algoritmos como XGBoost y modelos autorregresivos integrados.

### 4.5. Evaluación del modelo

Para la evaluación de los modelos predictivos, se dividió el conjunto de datos en subconjuntos de entrenamiento, prueba y predicción, aplicando técnicas de validación cruzada para asegurar la robustez de los resultados. Las métricas utilizadas para la evaluación de los modelos incluyeron el error cuadrático medio (RMSE), que permitió medir la precisión de las predicciones.

### 4.6. Implementación y puesta en producción

Una vez seleccionados los modelos con mejores resultados, se desarrolló una solución automatizada, basada en la contenerización de los modelos con **Docker** y su integración mediante una **API REST** en los sistemas de reabastecimiento de inventarios de DSMarket. El equipo de tecnología evaluó los modelos con pruebas adicionales utilizando datos no vistos.

Se implementó un pipeline de **CI/CD** para automatizar la actualización continua de los modelos y un sistema de **monitorización** con **Prometheus** y **Grafana** para asegurar su correcto funcionamiento. Adicionalmente, se utilizó una estrategia de **Blue-Green Deployment** para garantizar un despliegue sin interrupciones en producción.

### 4.7. Interpretación de resultados

Los resultados obtenidos tras el análisis de datos permitieron generar recomendaciones concretas para mejorar las operaciones de DSMarket. Se identificaron productos con bajo

rendimiento en determinadas tiendas, lo que condujo a la implementación de estrategias de marketing más focalizadas y a la optimización del inventario en cada tienda. Además, el clustering de productos y tiendas permitió segmentar eficazmente los mercados, lo que sirvió como base para la personalización de ofertas, campañas promocionales y estrategias de reabastecimiento. Los modelos predictivos aplicados optimizaron la planificación de inventarios, reduciendo costos operativos y mejorando los márgenes de beneficio, especialmente en las categorías de Supermercado y Hogar y Jardín.

### 4.8. Comunicación de resultados

Finalmente, se elaboraron una serie de dashboards interactivos utilizando herramientas de Business Intelligence (BI), facilitando el seguimiento en tiempo real de las ventas y la evolución de los principales indicadores. Los resultados fueron presentados al equipo ejecutivo de DSMarket, destacando el impacto económico estimado y las mejoras en la eficiencia operativa.

# 5. Task 1: ANÁLISIS

## 5.1. Descripción de la Empresa

DSMarket es una cadena de centros comerciales con presencia en varias ciudades de los Estados Unidos, dedicada a la venta de una amplia variedad de productos que abarcan desde alimentos hasta bienes de consumo diario. Fundada bajo el nombre TradiStores, la compañía ha operado tradicionalmente como un negocio minorista convencional, con una fuerte dependencia de la experiencia y conocimientos del personal para la gestión de inventarios, precios y promociones.

En el marco de su reciente proceso de modernización, DSMarket ha emprendido un ambicioso plan de transformación digital para adaptarse a las nuevas exigencias del mercado. Esta transformación incluye la digitalización de procesos, la implementación de tecnologías avanzadas y, sobre todo, el aprovechamiento de los datos generados por la empresa para mejorar su eficiencia y competitividad. Uno de los principales impulsores de este cambio ha sido la contratación de Michelle Huggins como Directora de Transformación Digital, quien ha liderado iniciativas clave para aprovechar los grandes volúmenes de datos acumulados por la empresa.

DSMarket se encuentra en una etapa temprana de su madurez analítica, pero ha establecido un plan estratégico de cinco años para integrar soluciones avanzadas de ciencia de datos en todos los niveles de la organización. La empresa busca mejorar sus operaciones mediante la optimización de ventas, gestión de inventarios y personalización de campañas de marketing, posicionándose así como un actor competitivo en el sector retail.

El objetivo final de DSMarket es convertirse en una empresa completamente **data-driven**, donde las decisiones de negocio se basen en el análisis de datos y se respalden con tecnologías de inteligencia artificial que optimicen sus operaciones y mejoren la experiencia del cliente.

### 5.1.1. Tipología de negocio desarrollado

DSMarket opera en el sector minorista, específicamente dentro de la categoría de retail. Este sector se caracteriza por la venta directa de productos al consumidor final, tanto en tiendas físicas como a través de plataformas digitales. El negocio de DSMarket abarca una amplia gama de productos que incluyen alimentos, artículos de hogar, productos de belleza, electrónica, ropa y otros bienes de consumo cotidiano.

A lo largo de su trayectoria, DSMarket ha seguido un modelo de negocio tradicional de comercio físico, con centros comerciales ubicados en grandes ciudades de los Estados Unidos, como Nueva York, Boston y Filadelfia. Sin embargo, debido a la creciente digitalización del sector y los cambios en los hábitos de consumo, la compañía ha reconocido la necesidad de adaptarse al nuevo entorno competitivo. Esta transformación implica no solo una expansión hacia el comercio electrónico, sino también la integración de tecnologías avanzadas que permitan gestionar de manera más eficiente sus operaciones internas.

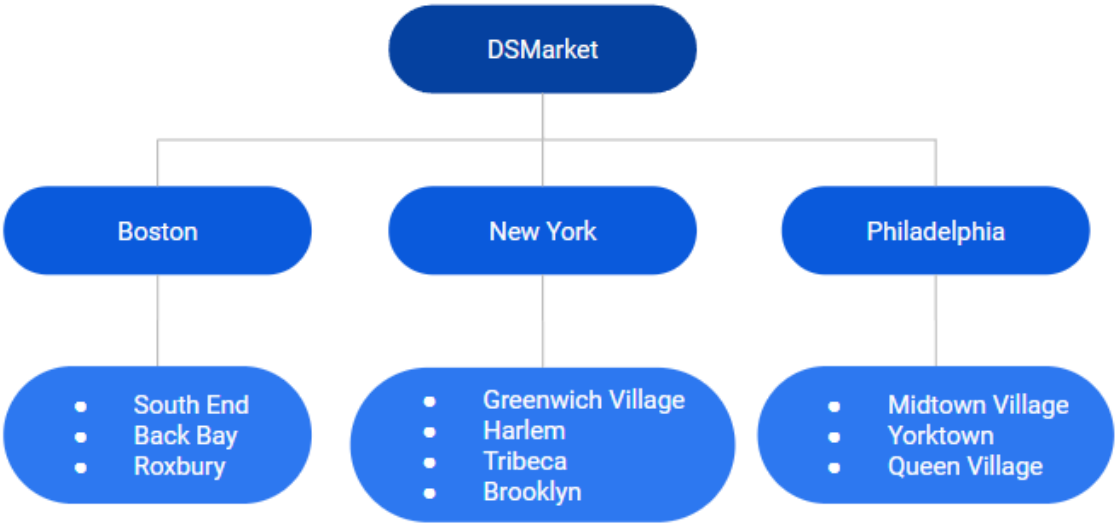
El modelo de negocio actual de DSMarket sigue los principios del retail omnicanal, donde los clientes pueden realizar sus compras tanto en tiendas físicas como a través de plataformas digitales. Este enfoque requiere que la empresa mantenga una operación altamente coordinada entre ambos canales, optimizando el manejo de inventarios, la logística de distribución y las estrategias de marketing.

Además, DSMarket se está moviendo hacia un modelo basado en datos, donde la analítica avanzada y las técnicas de machine learning juegan un papel fundamental para mejorar la toma de decisiones en áreas clave como la predicción de ventas, la segmentación de clientes y la personalización de campañas de marketing. El objetivo es que las decisiones operativas y estratégicas se basen en datos concretos y en predicciones precisas, lo que permitirá mejorar la eficiencia y satisfacer mejor las demandas del mercado.

En resumen, DSMarket se sitúa en el sector retail, con un enfoque de negocio omnicanal y una visión estratégica que prioriza la adopción de tecnologías basadas en datos para adaptarse a los cambios en el entorno competitivo y mejorar su desempeño en el mercado.

### 5.1.2. Organigrama

En esta sección, abordamos la estructura organizativa de las tiendas de DSMarket, distribuidas en tres ciudades clave: New York, Boston y Philadelphia. Esta estructura está basada en la ubicación de las tiendas y en los departamentos principales que encontramos dentro de cada una. El propósito de esta organización es garantizar que cada tienda esté alineada con las necesidades del mercado local, ofreciendo productos en categorías clave y operando de manera eficiente para satisfacer la demanda.




Esta distribución refleja la presencia de DSMarket en algunas de las ciudades más dinámicas del país, con cada tienda orientada a maximizar la cobertura de áreas residenciales y comerciales clave.

**Departamentos en las Tiendas**

Cada tienda de DSMarket sigue una estructura común en términos de departamentos y categorías de productos, lo que garantiza una oferta estandarizada y variada para los consumidores. Los departamentos principales que se encuentran en todas las tiendas de DSMarket son:

- **Accesorios**: Esta categoría abarca una amplia variedad de productos complementarios tanto para el hogar como para uso personal.
- **Hogar y Jardín**: En este departamento se ofrecen productos destinados al mantenimiento y la decoración del hogar, así como artículos para espacios exteriores.
- **Supermercado**: Este departamento incluye productos de primera necesidad, abarcando desde alimentos hasta productos frescos y bebidas, lo que asegura una oferta básica para las compras recurrentes de los clientes.

## 5.2. Exploratory Data Analysis (E.D.A.) y Limpieza de Datos

En este proyecto, el **Análisis Exploratorio de Datos (EDA)** fue esencial para detectar y corregir inconsistencias, así como para transformar los datos en un formato adecuado para el modelado predictivo. El EDA se centró en los datos de ventas, precios y eventos, y nos permitió ajustar el formato temporal, gestionar valores nulos, y consolidar la información necesaria para predecir las ventas de DSMarket a nivel de tienda, departamento y ciudad. Adicionalmente, el proceso de limpieza de datos fue clave para asegurar que la información estuviera en las condiciones óptimas para el análisis y modelado.

A continuación, se detallan los hallazgos clave y las decisiones tomadas en cada etapa del EDA y la limpieza de datos:

**1. Carga y Exploración Inicial de los Datos**

Trabajamos con tres conjuntos de datos principales:

- **Ventas de productos (item_sales.csv)**: Contiene información detallada de ventas diarias por tienda y producto.

```
[ ] df_sales.head()
```

| | id | item | category | department | store | store_code | region | d_1 | d_2 | d_3 | ... | d_1904 | d_1905 | d_1906 | d_1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | ACCESORIES_1_001_NYC_1 | ACCESORIES_1_001 | ACCESORIES | ACCESORIES_1 | Greenwich_Village | NYC_1 | New York | 0 | 0 | 0 | ... | 1 | 3 | 0 | |
| 1 | ACCESORIES_1_002_NYC_1 | ACCESORIES_1_002 | ACCESORIES | ACCESORIES_1 | Greenwich_Village | NYC_1 | New York | 0 | 0 | 0 | ... | 0 | 0 | 0 | |
| 2 | ACCESORIES_1_003_NYC_1 | ACCESORIES_1_003 | ACCESORIES | ACCESORIES_1 | Greenwich_Village | NYC_1 | New York | 0 | 0 | 0 | ... | 2 | 1 | 2 | |
| 3 | ACCESORIES_1_004_NYC_1 | ACCESORIES_1_004 | ACCESORIES | ACCESORIES_1 | Greenwich_Village | NYC_1 | New York | 0 | 0 | 0 | ... | 1 | 0 | 5 | |
| 4 | ACCESORIES_1_005_NYC_1 | ACCESORIES_1_005 | ACCESORIES | ACCESORIES_1 | Greenwich_Village | NYC_1 | New York | 0 | 0 | 0 | ... | 2 | 1 | 1 | |

5 rows × 1920 columns

- **Precios de productos (item_prices.csv)**: Proporciona los precios semanales de cada producto en cada tienda.

```
[ ] df_prices.head()
```

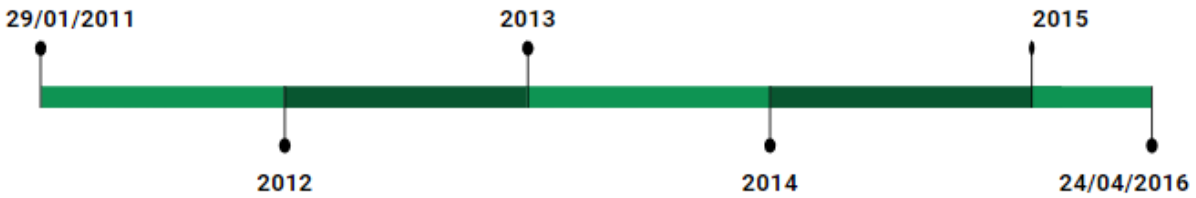
| | item | category | store_code | yearweek | sell_price |
|---|---|---|---|---|---|
| 0 | ACCESORIES_1_001 | ACCESORIES | NYC_1 | 201328.0 | 12.7414 |
| 1 | ACCESORIES_1_001 | ACCESORIES | NYC_1 | 201329.0 | 12.7414 |
| 2 | ACCESORIES_1_001 | ACCESORIES | NYC_1 | 201330.0 | 10.9858 |
| 3 | ACCESORIES_1_001 | ACCESORIES | NYC_1 | 201331.0 | 10.9858 |
| 4 | ACCESORIES_1_001 | ACCESORIES | NYC_1 | 201332.0 | 10.9858 |

- **Calendario de eventos (daily_calendar_with_events.csv)**: Incluye fechas importantes que influyen en las ventas, como eventos clave (Super Bowl, Thanksgiving, entre otros).

```
[ ] df_eventos.head()
```

| | date | weekday | weekday_int | d | event |
|---|---|---|---|---|---|
| **0** | 2011-01-29 | Saturday | 1 | d_1 | NaN |
| **1** | 2011-01-30 | Sunday | 2 | d_2 | NaN |
| **2** | 2011-01-31 | Monday | 3 | d_3 | NaN |
| **3** | 2011-02-01 | Tuesday | 4 | d_4 | NaN |
| **4** | 2011-02-02 | Wednesday | 5 | d_5 | NaN |

Durante la exploración inicial, observamos inconsistencias en los formatos y presencia de valores nulos en varias columnas, lo que indicaba la necesidad de aplicar procesos de limpieza y transformación de los datos.




### 2. Reformateo y Estandarización de los Datos de Ventas

El conjunto de datos de ventas originalmente estaba estructurado en un formato ancho, con una columna por cada día de ventas. Utilizamos la función **melt** de pandas para reformatear estos datos a un **formato largo**, donde cada fila representa las ventas de un producto en una tienda específica en un día concreto. Este cambio de formato fue fundamental para poder realizar un análisis temporal más preciso.

Además, realizamos una **optimización de memoria** convirtiendo varias columnas como `id`, **store** y **region** al tipo de dato **category**. Esto redujo significativamente el uso de memoria y mejoró el rendimiento durante el procesamiento, especialmente considerando el gran volumen de datos.

**3. Creación del Identificador Único de Producto y Tienda (ID)**

Un paso clave en la transformación de los datos fue la creación de una columna **id única** que combina el código del producto (**item**) y el código de la tienda (**store_code**). Este identificador nos permitió unir los diferentes conjuntos de datos (ventas, precios y eventos) de manera eficiente y precisa, manteniendo la integridad de la información a nivel tienda-producto.

**Ejemplo de IDs generados**:

- ACCESORIES_1_BOS_1: Accesorio vendido en la tienda "South End" de Boston.
- SUPERMARKET_3_NYC_4: Producto de supermercado en la tienda de Brooklyn, Nueva York.

**4. Estandarización y Corrección de los Datos Temporales**

El conjunto de datos cubre ventas desde 2011 hasta 2016, pero encontramos inconsistencias en las semanas del año. El formato original presentaba semanas incorrectas o atípicas:

- **2011**: Semanas de 05 a 52 (48 semanas en total).
- **2012 a 2015**: Incluyen 54 y 53 semanas, lo que no es estándar.
- **2016**: Termina en la semana 17.

Para estandarizar este formato:

- Eliminamos la semana 54, ya que representaba un valor atípico.
- Rellenamos valores nulos en las semanas utilizando el método **ffill (frontfill)** para completar los datos con la semana anterior.
- Ajustamos el conjunto de datos de ventas para que el año 2011 comenzará en la Semana 04.

La variable temporal **yearweek**, que combina el año y la semana, fue clave para realizar las uniones entre los conjuntos de ventas y precios, permitiéndonos realizar análisis semanales consistentes.

**5. Limpieza de los Datos de Precios**

El conjunto de precios contenía información crucial para entender el comportamiento de las ventas, ya que los cambios en los precios pueden afectar la demanda de los productos. Durante la exploración de los datos de precios, encontramos varias inconsistencias:

- **Valores duplicados**: Se eliminaron registros repetidos para asegurar que cada combinación tienda-producto tuviera un único precio por semana.
- **Valores nulos:** En la columna de semanas (**yearweek**), los valores nulos fueron rellenados usando el valor anterior con el método **ffill**.
- **Formato de precios**: Redondeamos la columna **sell_price** a dos decimales, ya que algunos precios presentaban inconsistencias en el número de decimales, lo que podía afectar los análisis posteriores.

**6. Integración del Calendario de Eventos**

El conjunto de eventos proporcionaba contexto sobre factores externos que pudieran haber influido en las ventas. Este conjunto de datos incluía eventos como el Super Bowl, Thanksgiving, Ramadan, y Easter, entre otros. Durante el análisis, identificamos varios valores nulos en la columna **event**, que fueron imputados con la categoría "Sin Evento". Esto nos permitió evaluar el impacto de los días en los que no ocurría ningún evento relevante.

Adicionalmente, eliminamos los registros de eventos de 2016, ya que faltaban valores significativos en ese año, lo que podría haber afectado los resultados. Esta limpieza fue importante para asegurar que los datos utilizados fueran completos y consistentes.

**7. Unión de los Conjuntos de Datos**

Una vez limpiados y transformados los datos de ventas, precios y eventos, procedimos a realizar una serie de uniones para consolidar la información en un único conjunto. La unión de los datos se realizó en dos pasos:

- **Ventas + Eventos**: Utilizamos la columna **d** (días) como clave para combinar los datos de ventas con el calendario de eventos.

- **Unión con Precios**: Se realizó un **left join** con el conjunto de precios utilizando las columnas **id**, **año** y **semana** como claves.

Al hacer una unión de tipo left join, garantizamos que no se perdieran registros de ventas sin información de precios. Posteriormente, rellenamos los precios nulos utilizando los métodos de **frontfill** y **backfill**, lo que nos permitió completar las series de precios faltantes.

**8. Transformación y Agregación de Datos**

El siguiente paso fue agregar los datos a nivel semanal para facilitar el análisis temporal. Creamos nuevas variables como:

- **Año y Semana**: Extraídas de la columna de fecha, lo que permitió realizar análisis semanales y observar tendencias.
- **Trimestre**: Esta variable nos permitió estudiar los efectos estacionales en las ventas, como la subida de demanda en ciertos trimestres del año.

Esta agregación fue crucial para reducir la granularidad de los datos y enfocarnos en el análisis de las ventas por semana, lo que es más adecuado para realizar predicciones a corto y mediano plazo.

**9. Optimización de la Estructura de los Datos**

Finalmente, optimizamos el uso de memoria de los datos convirtiendo las columnas categóricas y numéricas a sus tipos de datos más eficientes:

- **Categorización**: Las columnas que contenían textos repetidos (como **id**, **store**, **region**) fueron convertidas al tipo **category** para reducir el tamaño del conjunto de datos.
- **Reducción de tamaño numérico**: Las columnas numéricas fueron convertidas a los tipos de datos **int** y **float** más ligeros, utilizando técnicas de reducción de tamaño para mejorar el rendimiento durante el modelado.

Este paso final nos permitió trabajar con un conjunto de datos más ligero y eficiente, listo para su uso en los modelos de predicción de ventas.

**Conclusión del EDA y Limpieza de Datos**

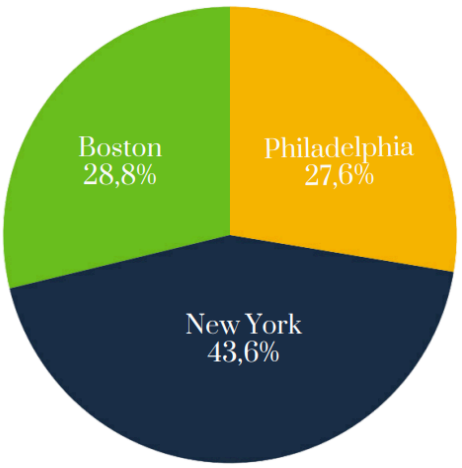
El EDA y la limpieza de datos fueron pasos fundamentales para preparar los datos de DSMarket y garantizar que fueran adecuados para el modelado predictivo. Durante este proceso, detectamos y solucionamos varias inconsistencias, como la falta de precios, valores nulos en los eventos y errores en el formato de semanas.

La creación de identificadores únicos para cada combinación tienda-producto nos permitió consolidar los datos y garantizar su integridad a lo largo del análisis. La optimización de memoria y la agregación de datos por semana fueron pasos cruciales para mejorar la eficiencia del modelo y asegurar que las predicciones se basaran en datos completos y precisos. Este análisis nos deja con un conjunto de datos robusto, limpio y listo para alimentar.

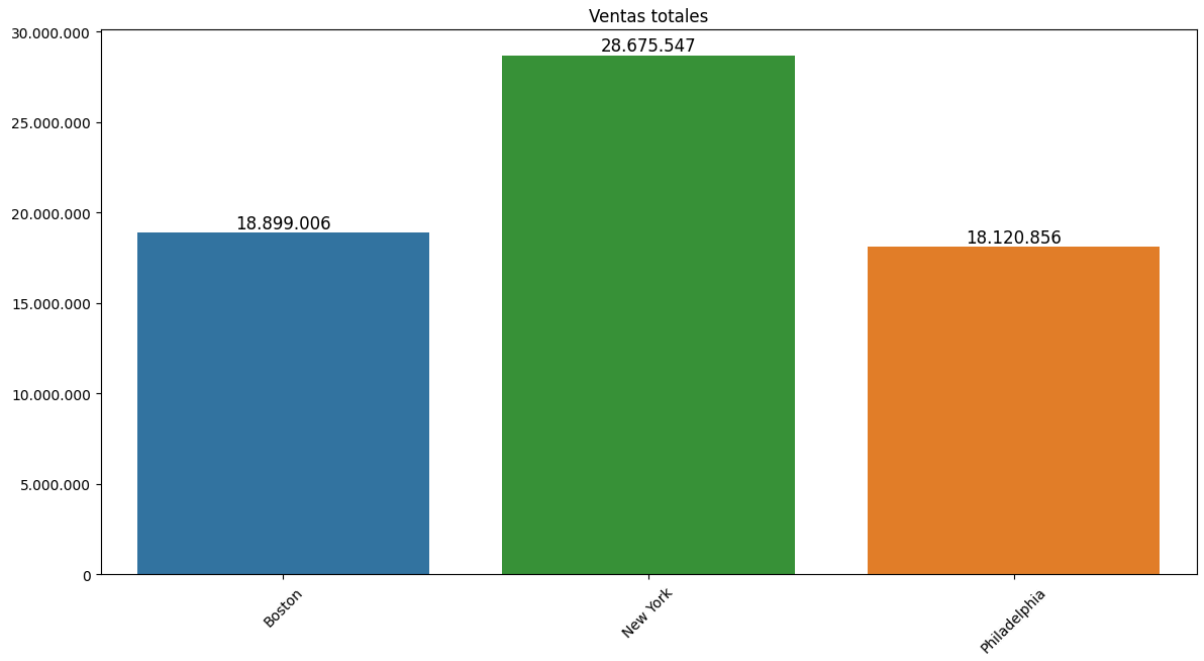
## 5.3. Análisis de datos

Los datos generales de DSMarket abarcan un período desde 2011 hasta principios de 2016. La información refleja la facturación total y el crecimiento anual por ciudad, junto con un desglose de ventas por tienda y producto.

A nivel global, la **facturación total** de DSMarket en este período asciende a **231 millones de dólares**, con un **crecimiento anual promedio del 12.8%**. Las ventas totales alcanzaron los **65 millones de dólares**, y las tres ciudades principales —**New York**, **Boston** y **Philadelphia**— se distribuyen el volumen de ventas de la siguiente manera:



Las categorías más relevantes incluyen **Accesorios**, **Hogar y Jardín**, y **Supermercado**, cada una con patrones de demanda específicos según la ubicación geográfica y la tienda. A continuación, se desglosa el análisis por ciudad, incluyendo los datos específicos de cada tienda en cada una de las áreas metropolitanas estudiadas.




### 5.3.1 New York

New York lidera en términos de ventas totales, con una participación del 43.6% del total de DSMarket, y un **crecimiento anual promedio del 11.9%**. La facturación total de la ciudad alcanza los **103.8 millones de dólares**, con ventas de **28.6 millones de dólares**. Este desempeño destaca la importancia de las tiendas situadas en zonas céntricas, que tienen un mayor volumen de ventas y crecimiento. En contraste, las tiendas periféricas muestran un crecimiento más moderado, pero estable.

- **Categorías más vendidas**: Hogar y Jardín.
- **Productos que más facturan**: Accesorios, impulsados por las tiendas céntricas.

**Tiendas en New York**

- **Tribeca**:
  - **Crecimiento**: 11.8%
  - **Ventas**: 5.6 millones de dólares
  - **Facturación**: 39.5 millones de dólares

    - **Observación**: Tribeca es la tienda con mayor facturación, debido principalmente a la demanda de productos de Hogar y Jardín. El poder adquisitivo de la zona y la sensibilidad a eventos de lujo impulsan las ventas en este segmento.
- **Greenwich Village**:
    - **Crecimiento**: 11.2%
    - **Ventas**: 7.2 millones de dólares
    - **Facturación**: 21.5 millones de dólares
    - **Observación**: El crecimiento en esta tienda se ha visto impulsado por la categoría de Accesorios y Hogar y Jardín. Es una tienda estable que responde bien a las promociones y eventos estacionales.
- **Brooklyn**:
    - **Crecimiento**: 14%
    - **Ventas**: 6 millones de dólares
    - **Facturación**: 15 millones de dólares
    - **Observación**: A pesar de su ubicación periférica, Brooklyn ha mostrado el crecimiento más rápido, con un enfoque en productos de consumo diario y Supermercado.
- **Harlem**:
    - **Crecimiento**: 11.8%
    - **Ventas**: 6 millones de dólares
    - **Facturación**: 27.7 millones de dólares
    - **Observación**: Harlem tiene un desempeño mixto. A pesar de no ser una tienda céntrica, muestra una alta facturación en la categoría de Supermercado.

**Conclusiones sobre New York**

New York es la ciudad más importante para DSMarket, con las tiendas de **Tribeca** y **Greenwich Village** liderando en ventas y facturación. Estas áreas de alto poder adquisitivo impulsan la demanda de productos de Hogar y Jardín, mientras que las tiendas en Brooklyn y Harlem mantienen un crecimiento estable, centrado principalmente en la categoría de Supermercado.

### 5.3.2 Boston

Boston contribuye con el 28.8% de las ventas totales de DSMarket, con un **crecimiento anual promedio del 10.3%**. La facturación total en esta ciudad asciende a **66 millones de dólares**, mientras que las ventas alcanzan los **19 millones de dólares**. Las tiendas en Boston muestran una distribución equilibrada de ventas, con un comportamiento estable en las tres tiendas principales.

- **Categorías más vendidas**: Supermercado.
- **Productos que más facturan**: Productos de consumo diario y frescos.

**Tiendas en Boston**

- **South End**:
  - **Crecimiento**: 10.5%
  - **Ventas**: 5.6 millones de dólares
  - **Facturación**: 19.3 millones de dólares
  - **Observación**: South End es la tienda más rentable en Boston. Las ventas de productos de Supermercado representan el mayor porcentaje de ingresos, con un enfoque en productos de consumo diario y frescos.
- **Back Bay**:
  - **Crecimiento**: 8.3%
  - **Ventas**: 7.2 millones de dólares
  - **Facturación**: 25.2 millones de dólares
  - **Observación**: Back Bay ha mostrado un crecimiento más lento en comparación con otras tiendas de Boston. La categoría de Accesorios no ha tenido el desempeño esperado, lo que sugiere la necesidad de ajustes estratégicos.
- **Roxbury**:
  - **Crecimiento**: 12.7%
  - **Ventas**: 6 millones de dólares
  - **Facturación**: 22 millones de dólares
  - **Observación**: Roxbury ha superado las expectativas, con un crecimiento sostenido en la categoría de Supermercado. A pesar de estar en la periferia, su desempeño ha sido notable.

**Conclusiones sobre Boston**

En Boston, la tienda de **South End** lidera en términos de rentabilidad, mientras que **Roxbury** muestra un crecimiento sólido en productos de consumo diario. **Back Bay**, aunque tiene un buen volumen de ventas, enfrenta desafíos en la categoría de Accesorios, lo que requiere una revisión de su estrategia de marketing.

### 5.3.3 Philadelphia

Philadelphia representa el 27.6% de las ventas totales de DSMarket, con un crecimiento anual promedio del 17.3%, **el más alto entre las tres ciudades analizadas**. La facturación total es de **60 millones de dólares**, mientras que las ventas alcanzan los **18 millones de dólares**. Las tiendas en Philadelphia han mostrado un crecimiento acelerado, especialmente en las áreas de productos de lujo y accesorios.

- **Categorías más vendidas**: Accesorios.
- **Productos que más facturan**: Moda y productos de lujo.

**Tiendas en Philadelphia**

- **Midtown Village**:
  - **Crecimiento**: 25%
  - **Ventas**: 5.2 millones de dólares
  - **Facturación**: 18.2 millones de dólares
  - **Observación**: Midtown Village es la tienda de mayor crecimiento en Philadelphia. La demanda de productos de lujo y accesorios ha impulsado este crecimiento, siendo una tienda clave para la expansión en este segmento.
- **Queen Village**:
  - **Crecimiento**: 31.5%
  - **Ventas**: 6.5 millones de dólares
  - **Facturación**: 21.6 millones de dólares
  - **Observación**: Queen Village ha experimentado un crecimiento notable en la categoría de Accesorios, consolidándose como una de las tiendas más exitosas de DSMarket.
- **Yorktown**:
  - **Crecimiento**: 4.3%
  - **Ventas**: 6.5 millones de dólares

- **Facturación**: 20.7 millones de dólares
- **Observación**: A pesar de su crecimiento más modesto, Yorktown tiene potencial para mejorar con ajustes en la oferta de productos de lujo y estrategias de marketing específicas.

**Conclusiones sobre Philadelphia**

Philadelphia destaca por su crecimiento acelerado, especialmente en las tiendas de **Midtown Village** y **Queen Village**, que lideran en la categoría de accesorios y productos de lujo. **Yorktown** tiene un desempeño más moderado, pero con el potencial de crecimiento si se implementan las estrategias adecuadas.

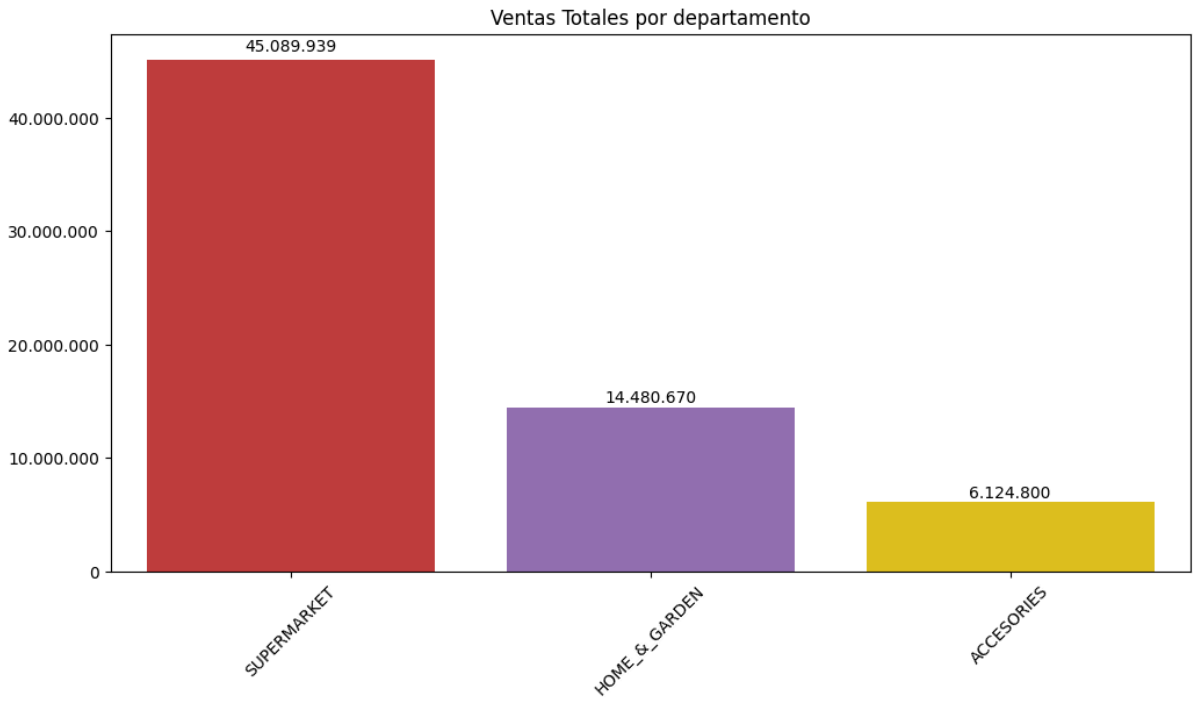
**Datos Más Relevantes**

El análisis detallado por ciudad y tienda revela los siguientes datos clave:

- **New York** es el mercado más importante, con **Tribeca** y **Greenwich Village** liderando en facturación.
- **Boston**, aunque más pequeño, muestra estabilidad, con **South End** como la tienda más rentable, y **Roxbury** superando las expectativas en productos de Supermercado.
- **Philadelphia** ha tenido el crecimiento más acelerado

### 5.3.4 Análisis Global de los Departamentos

Los departamentos de **Accesorios**, **Hogar y Jardín**, y **Supermercado** se distribuyen de manera equilibrada en todas las tiendas de DSMarket, pero con variaciones significativas en su desempeño según la ubicación geográfica y el perfil de la tienda.




## Accesorios

- **Ventas totales**: 62 millones de dólares.
- **Crecimiento anual**: 15.2%.
- **Participación en las ventas globales**: 26.8%.

Este departamento muestra un fuerte desempeño en tiendas ubicadas en áreas céntricas y de alto poder adquisitivo, particularmente en **New York** y **Philadelphia**. La categoría de accesorios tienen una demanda constante en las zonas urbanas más exclusivas. Es relevante señalar que los artículos de este departamento son especialmente sensibles a eventos promocionales y de temporada, lo que genera picos de ventas significativos durante el cuarto trimestre del año.

### Ciudades y Tiendas Relevantes para Accesorios

- **New York**:
    - **Tribeca** y **Greenwich Village**: Estas tiendas lideran en la venta de accesorios, con un crecimiento del 17.5% en comparación con el año anterior. **Tribeca** genera el 22% de las ventas globales de accesorios, con un total de **13.6 millones de dólares**. Los productos son los más demandados, especialmente durante el cuarto trimestre del año, coincidiendo con las festividades.

    - **Harlem** y **Brooklyn**: Aunque las tiendas periféricas de New York no tienen el mismo volumen de ventas que las del centro, estas tiendas han experimentado un crecimiento sólido en accesorios, con un aumento del 12%, generando en conjunto **7.8 millones de dólares**. Aquí se observa una demanda más orientada hacia accesorios funcionales y económicos.
- **Philadelphia**:
    - **Midtown Village** y **Queen Village**: En Philadelphia, el departamento de Accesorios ha crecido un 19.3%, siendo la categoría más importante para las tiendas de Midtown Village y Queen Village, que juntas generan **15.8 millones de dólares**.
- **Boston**:
    - **South End**: Aunque Boston no lidera en la categoría de Accesorios, la tienda de South End ha mostrado un buen desempeño, con un crecimiento del 11.7% en esta categoría. Sin embargo, **Back Bay** y **Roxbury** han tenido un rendimiento inferior al promedio, lo que sugiere que esta categoría tiene menos demanda en estas zonas.

## Hogar y Jardín

- **Ventas totales**: 85 millones de dólares.
- **Crecimiento anual**: 13.5%.
- **Participación en las ventas globales**: 36.7%.

El departamento de Hogar y Jardín es el más grande en términos de ventas. Este departamento tiene un comportamiento más uniforme en las tres ciudades, con picos de ventas durante el verano y los fines de semana largos, cuando los clientes tienden a comprar artículos relacionados con mejoras en el hogar y el jardín. **New York** destaca en esta categoría, con un rendimiento especialmente fuerte en las tiendas de Tribeca y Greenwich Village.

**Ciudades y Tiendas Relevantes para Hogar y Jardín**

- **New York**:
    - **Tribeca**: Es la tienda más fuerte en esta categoría, con una facturación de **24 millones de dólares** en productos de Hogar y Jardín. El crecimiento en esta tienda ha sido constante, con un 14.9% de aumento anual.

    - **Greenwich Village**: Ha tenido un comportamiento similar, con un crecimiento del 12.8% en Hogar y Jardín, y ventas totales de **19 millones de dólares** en esta categoría.
- **Boston**:
    - **South End** y **Roxbury**: Estas tiendas han mostrado un buen rendimiento en Hogar y Jardín, especialmente en productos de jardinería y muebles para exteriores. South End, con **15 millones de dólares** en ventas de esta categoría, ha experimentado un crecimiento del 11.3%, mientras que Roxbury ha registrado un aumento del 10.2%.
- **Philadelphia**:
    - **Midtown Village**: Con **12 millones de dólares** en ventas, Midtown Village es la tienda más exitosa en Philadelphia en esta categoría. El crecimiento del 10.9% en Hogar y Jardín refleja la estabilidad de esta tienda en productos de mejora del hogar.

## Supermercado

- **Ventas totales**: 84 millones de dólares.
- **Crecimiento anual**: 14.8%.
- **Participación en las ventas globales**: 36.5%.

El departamento de Supermercado tiene una participación casi igual a la de Hogar y Jardín, pero con una mayor consistencia en las ventas a lo largo del año. Este departamento incluye productos de primera necesidad. Las tiendas ubicadas en áreas periféricas, como Brooklyn y Harlem en New York o Roxbury en Boston, tienen un alto rendimiento en esta categoría debido a su enfoque en productos de consumo diario.

### Ciudades y Tiendas Relevantes para Supermercado

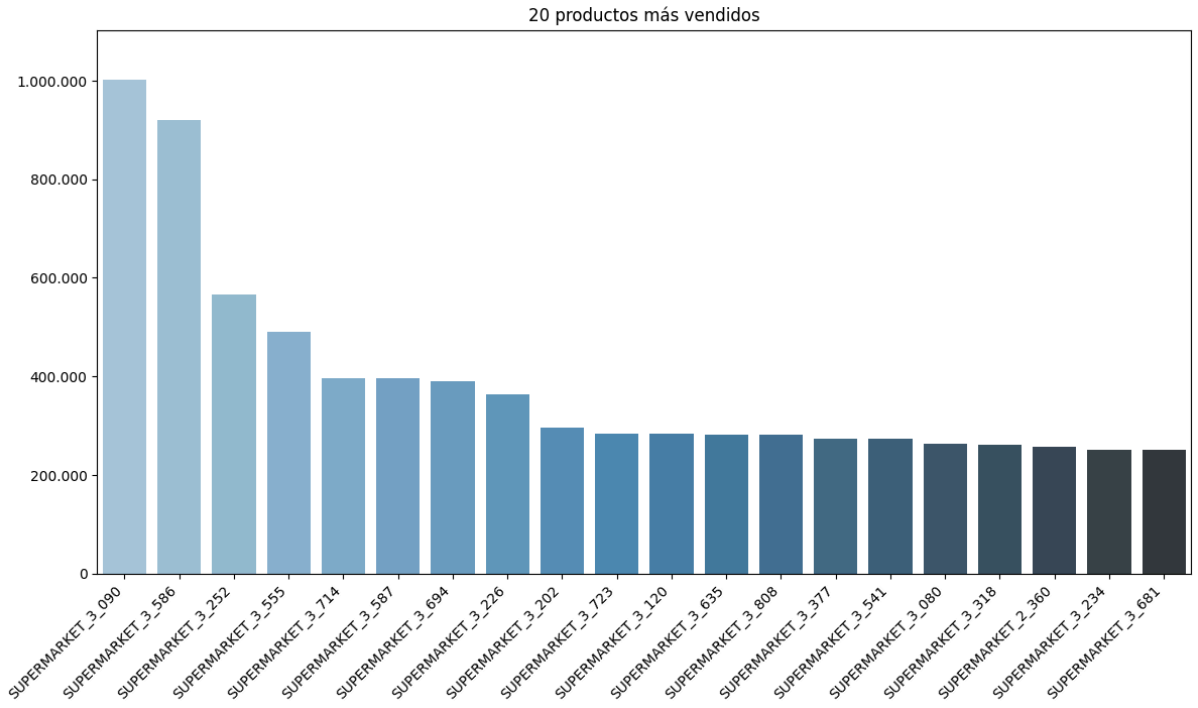
- **New York**:
    - **Harlem** y **Brooklyn**: Estas tiendas son las más fuertes en la categoría de Supermercado, con **13 millones de dólares** en ventas combinadas. El crecimiento en estas áreas ha sido del 15.2%, impulsado por la demanda de productos frescos y de primera necesidad.
- **Boston**:
    - **South End**: La tienda de South End es la más rentable en esta categoría, con **16 millones de dólares** en ventas y un crecimiento del 12.4%.

- **Roxbury**: También ha tenido un buen desempeño en esta categoría, con **11.8 millones de dólares** en ventas y un crecimiento del 14.1%.

- **Philadelphia**:
  - **Yorktown**: Es la tienda más relevante en la categoría de Supermercado en Philadelphia, con **10 millones de dólares** en ventas y un crecimiento del 10.8%.

### 5.3.5 Análisis Global de los Productos

**1. Productos más vendidos**




Este gráfico muestra los **20 productos más vendidos**, clasificados por unidades vendidas.

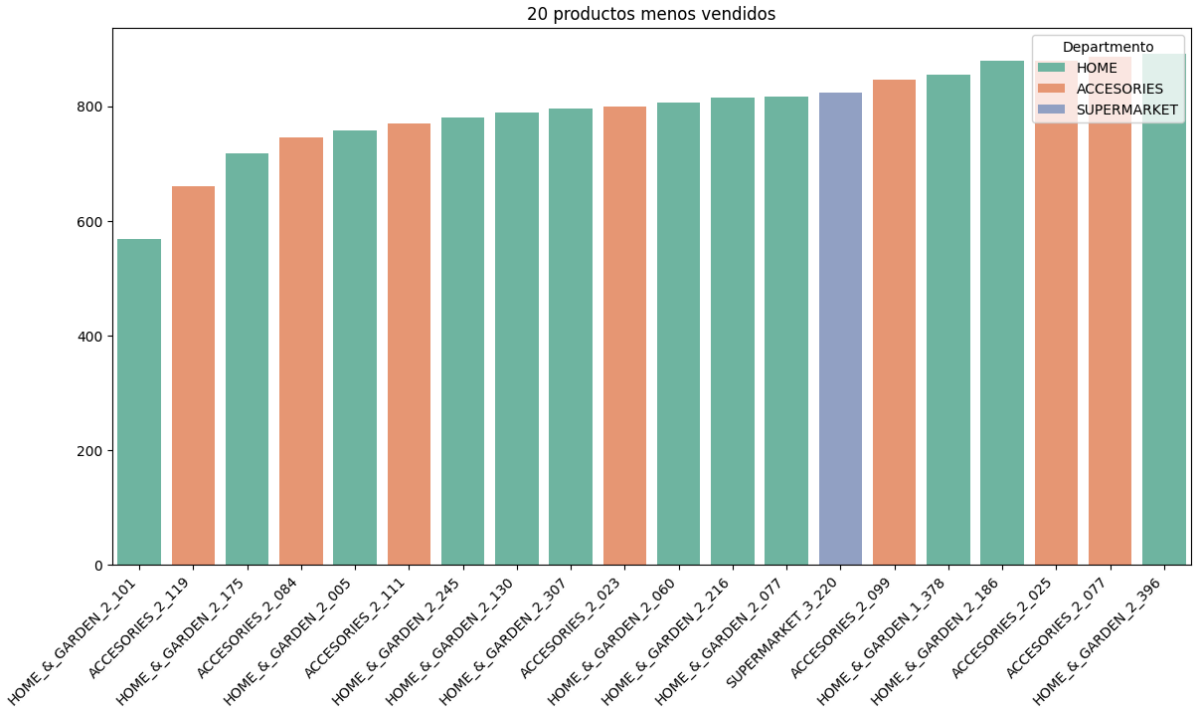
**Puntos a destacar:**

1. **Producto más vendido**: **SUPERMARKET_3_090** lidera las ventas con más de **1 millón de unidades**, seguido de **SUPERMARKET_3_586** con cerca de **900,000 unidades**. Esto destaca la fuerte demanda de estos productos esenciales.
2. **Diferencia notable en ventas**: Aunque todos los productos están en el top 20, hay una diferencia significativa entre los primeros y últimos. Los productos más vendidos

han alcanzado cifras mucho más altas, mientras que los últimos apenas superan las **300,000 unidades**.

3. **Predominio de productos esenciales**: Los productos más vendidos pertenecen a la categoría de Supermercado, lo que subraya la alta demanda de productos de consumo diario, clave en las tiendas periféricas y suburbanas.

## 2. Productos menos vendidos

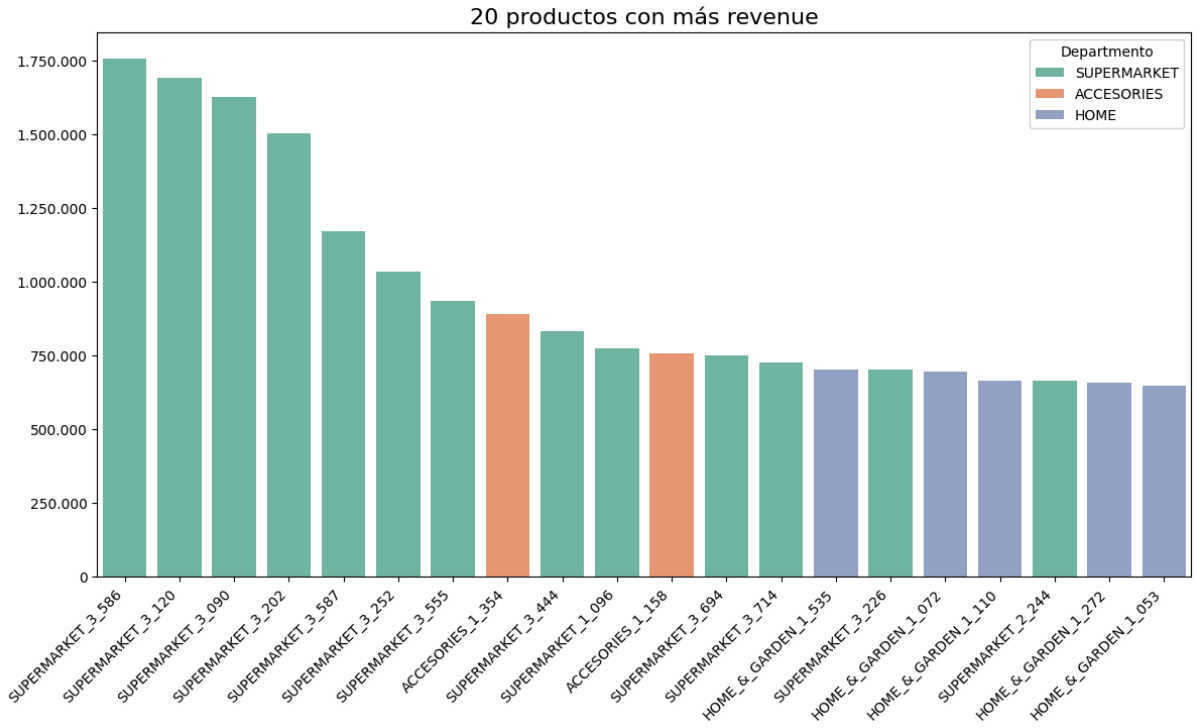



Este gráfico representa los **20 productos menos vendidos**, a continuación, presentamos los puntos clave:

1. **Productos del departamento de Hogar y Jardín** dominan la lista: La mayoría de los productos menos vendidos pertenecen a la categoría de **Hogar y Jardín**, lo que indica una menor demanda en comparación con otras categorías, especialmente en productos más específicos o estacionales.
2. **Accesorios con menor rotación**: Los productos del departamento de **Accesorios** también aparecen varias veces en la lista de los menos vendidos, destacando aquellos que probablemente pertenecen a nichos que no son tan populares en todas las tiendas.
3. **Supermercado en la lista**: Aunque el producto **SUPERMARKET_3_220** aparece en esta lista, es el único de la categoría de Supermercado, lo que sugiere que incluso

los productos menos vendidos de Supermercado tienen una rotación relativamente alta comparada con los otros departamentos.

### Productos con más revenue

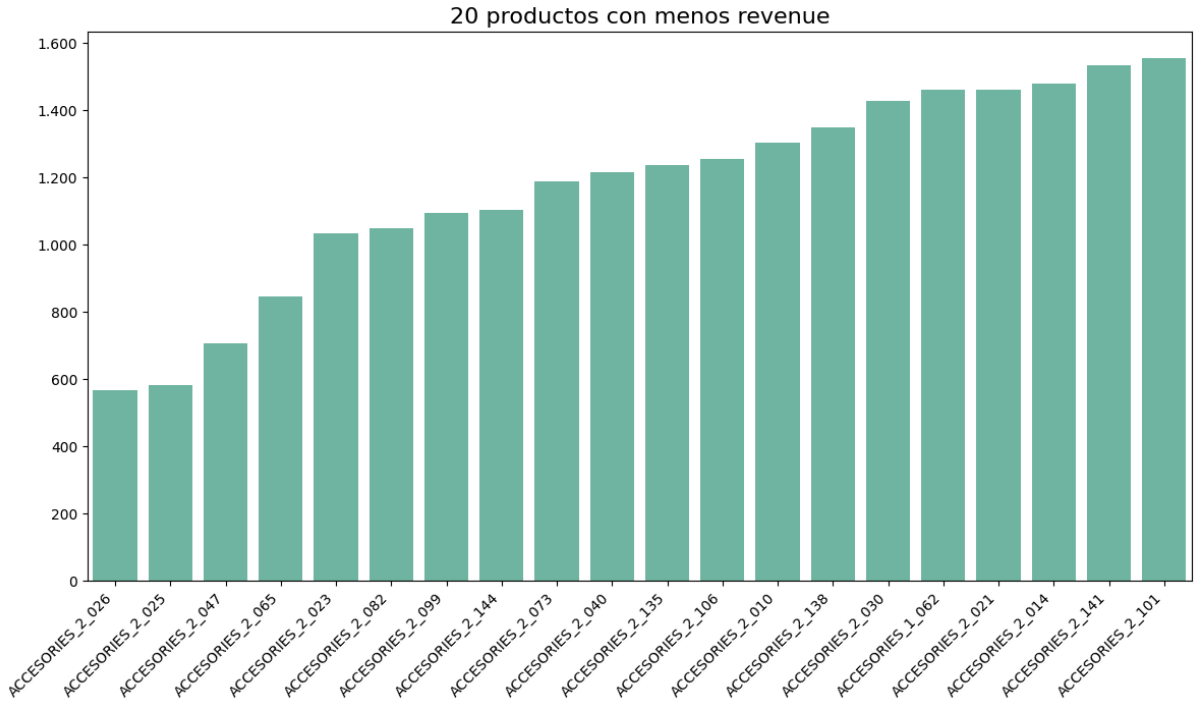



Este gráfico muestra los **20 productos con más revenue**, a continuación, los puntos clave:

1. **Dominio de Supermercado**: Los productos del departamento de **Supermercado** dominan la lista, ocupando los primeros puestos con ingresos superiores a **1.5 millones de dólares**. En particular, los productos **SUPERMARKET_3_586**, **SUPERMARKET_3_120**, y **SUPERMARKET_3_090** encabezan las ventas, lo que subraya la importancia de esta categoría en el revenue total.
2. **Accesorios con alto revenue**: Aunque los productos de **Accesorios** no dominan en unidades vendidas, logran aparecer en la lista de productos más rentables. **ACCESORIES_1_354** y **ACCESORIES_1_096** generan ingresos cercanos al millón de dólares, destacando por sus mayores márgenes de beneficio.
3. **Presencia de Hogar y Jardín**: Los productos de **Hogar y Jardín** también están presentes en la lista, aunque con ingresos algo menores. Sin embargo, siguen siendo importantes, con varios productos generando entre **700,000** y **900,000 dólares**.

Los productos de **Supermercado** dominan en términos de revenue, seguidos por los de **Accesorios**, que destacan por generar altos ingresos a pesar de venderse en menores cantidades. **Hogar y Jardín**, aunque menos representado, sigue siendo relevante para la generación de ingresos en DSMarket.

### Productos con menos revenue




Este gráfico muestra los **20 productos con menos revenue**, todos pertenecientes al departamento de **Accesorios**. A continuación, los puntos clave:

1. **Bajo revenue generalizado en Accesorios**: Todos los productos en esta lista pertenecen a la categoría de **Accesorios**, lo que indica que, aunque esta categoría puede tener márgenes altos, ciertos productos específicos no logran generar ingresos significativos.
2. **Productos con ingresos mínimos**: Los productos con menores ingresos, como **ACCESORIES_2_026** y **ACCESORIES_2_025**, generan menos de **600 dólares**, lo que subraya una baja rotación o demanda limitada en comparación con otros artículos.
3. **Ligeras variaciones en el revenue**: Aunque todos los productos tienen bajos ingresos, las variaciones entre ellos no son muy grandes. El producto con más

revenue en esta lista (**ACCESORIES_2_101**) genera cerca de **1,600 dólares**, lo que sigue siendo significativamente bajo comparado con productos de otras categorías.

# 6. Task 2: CLUSTERING

El procedimiento de agrupamiento, mejor conocido como Clustering, es una técnica de aprendizaje no supervisado que tiene como objetivo agrupar un conjunto de datos en grupos (clusters) de tal manera que los elementos dentro de un mismo grupo sean más similares entre sí que con los elementos de otros grupos. Este proceso permite identificar patrones y estructuras ocultas en los datos sin la necesidad de contar con etiquetas o categorías predefinidas, lo que lo convierte en una herramienta esencial para explorar y segmentar grandes volúmenes de información.

## 6.1. Metodología

En este proyecto, el clustering se utiliza para agrupar productos (**ítems**) que comparten comportamientos similares en términos de ventas, precios, estacionalidad y sensibilidad a eventos. El mismo proceso se aplicará al número de tiendas examinadas en este trabajo.

A continuación, se describen los pasos clave involucrados y técnicas empleadas en ambos procesos de clustering aplicados:

| | | |
|---|---|---|
| **1** | **Feature Engineering** | Proceso de creación de nuevas variables (features) a partir de los datos existentes que pueden aportar información valiosa al modelo de clustering. Esto incluye la generación de ratios de ventas, sensibilidad al precio, impacto de eventos, entre otros. |
| **2** | **Evaluación de Features** | Análisis de las características creadas para determinar su relevancia en el modelo. Se evalúa su capacidad para aportar información y su relación con otras variables. |
| | | **Feature Variance**<br>Cálculo de la varianza de cada característica para entender cuánta variación existe entre los ítems. Características con varianza extremadamente baja pueden no ser útiles y podrían ser descartadas. |
| | | **Feature Correlation**<br>Evaluación de la correlación entre las características para identificar relaciones lineales. Variables con alta correlación positiva o negativa podrían ser redundantes y podrían fusionarse o eliminarse. |
| **3** | **Estandarización de datos** | Proceso de escalar las características para que todas tengan la misma importancia, ajustando sus valores a una escala común. Generalmente se aplica la **normalización** o **estandarización** (media 0 y desviación estándar 1). |

| | | |
|---|---|---|
| **4** | **Análisis de los Componentes Principales (PCA)** | Técnica de reducción de dimensionalidad que transforma las características originales en **componentes principales** que explican la mayor variabilidad en los datos, ayudando a simplificar el modelo sin perder demasiada información. |
| **5** | **Selección del Algoritmo de Clustering** | Proceso de elegir el algoritmo más adecuado para agrupar los datos. Para este ejercicio se selecciona **K-Means** debido a su simplicidad y eficacia en datos numéricos escalados. |
| **6** | **Aplicación del Clustering K-Means** | Se aplica el algoritmo K-Means, que agrupa los ítems en un número predefinido de clusters minimizando la distancia interna entre los puntos del mismo grupo. |
| **7** | **Evaluación del número óptimo de clusters** | Proceso de determinar el número adecuado de clusters para el modelo. Esto se hace evaluando la dispersión dentro de los grupos y utilizando herramientas como la curva del codo y el Silhouette Score. |
| | | **Curva del codo**<br>Método gráfico para identificar el número óptimo de clusters. Se basa en encontrar el punto en el que la **inercia** (suma de las distancias al centroide) disminuye de forma significativa, formando un "codo". |
| | | **Silhouette Score**<br>Métrica para evaluar la **calidad** de los clusters. Un **Silhouette Score** alto (cercano a 1) indica que los puntos están bien agrupados en su cluster y alejados de otros clusters. |
| **8** | **Interpretación de los resultados** | Análisis de los clusters formados para entender las características que definen cada grupo y cómo se relacionan con las variables originales. Esto ayuda a identificar patrones clave en los datos. |
| **9** | **Visualización y análisis de los clusters** | Creación de gráficos y representaciones visuales (mapas de calor, gráficos de dispersión, etc.) que permiten explorar y comunicar las diferencias entre los clusters de manera clara. |
| **10** | **Asignación de labels** | Asignación de etiquetas o nombres descriptivos a cada cluster para facilitar su interpretación. Los labels se basan en las características predominantes en cada grupo. |
| **11** | **Generación de fichas para cada cluster** | Creación de perfiles o "fichas" detalladas para cada cluster, que describen sus principales características, comportamientos y patrones. Esto permite una visión clara y estructurada de cada grupo y sus diferencias. |

### 6.1.1. Ratios vs. valores absolutos

En este estudio, optamos por utilizar ratios en lugar de valores absolutos para analizar el comportamiento de ventas de los productos. La principal razón de esta elección radica en la

necesidad de obtener una comparación más equitativa y significativa entre ítems con diferentes volúmenes de ventas.

Los valores absolutos, como el número total de ventas en un trimestre, pueden sesgar el análisis hacia los productos con mayores volúmenes, dejando de lado aquellos con ventas menores pero con patrones de comportamiento igualmente relevantes. Por el contrario, los ratios de ventas trimestrales permiten observar qué proporción de las ventas totales de un ítem ocurrió en un trimestre específico, lo que normaliza las diferencias en el tamaño de las ventas y facilita la comparabilidad entre productos.

Al utilizar ratios, podemos identificar patrones estacionales o comportamientos recurrentes que serían difíciles de detectar con los valores absolutos. Por ejemplo, dos productos con volúmenes de ventas muy diferentes pueden tener ratios similares, lo que indicaría que ambos tienen una distribución de ventas similar a lo largo del año, incluso si el número total de unidades vendidas es diferente.

Esta elección también es crucial en el contexto de clustering, ya que algoritmos como KMeans tienden a agrupar productos según sus características. Los ratios permiten que el algoritmo identifique productos con patrones de ventas similares, independientemente de su volumen total, lo que nos proporciona clusters más útiles para el análisis del comportamiento de ventas.

Finalmente, procedemos entonces a generar nuevas variables de valores absolutos, que a su vez nos permitirán obtener el ratio de dichos valores.

## 6.2. Feature engineering

Analizando las variables que ya contiene nuestro dataset, nos percatamos que no todas son directamente útiles para el clustering, ya que algunas son categóricas y otras pueden no ser suficientemente descriptivas. Necesitamos transformar estas variables en algo más informativo. Esto significa que necesitamos generar nuevas características mediante el proceso de *Feature Engineering*, dinámica que consiste en generar variables que capturen el comportamiento relevante de los productos.

Las variables que generamos para este ejercicio parten de la información que tenemos a disposición: fechas, cifras de ventas temporales y regionales, precio de venta e información externa como impacto de eventos, etc.

### 6.2.1. Variables relacionadas a tiempo

En el Data Frame original creamos (y retomamos) variables relacionadas al tiempo y los eventos, fundamentales para identificar patrones temporales y el impacto de eventos en el comportamiento de las ventas.

| Variable | Descripción |
|---|---|
| `date` | Convertimos la columna yearweek (año y semana del año) en un formato de **fecha completa.** |

```
[ ]  #Convertimos la columna yearweek (año y semana del año) en un formato de fecha completa.
     df['date'] = pd.to_datetime(df['yearweek'].astype(str) + '0', format='%Y%W%w')
```

Tomamos la columna **yearweek**, la convertimos a texto y le agregamos un "0" al final para que represente el último día de la semana. Luego, usamos **pd.to_datetime** con el formato de año y semana (%Y%W%w) para generar una fecha válida.

| | |
|---|---|
| `month` | Extraemos el **número de mes** de la columna date recién creada, para agregar información sobre la estacionalidad de las ventas. |

```
[ ]  #Extraemos el número de mes de la columna date recién creada.
     df['month'] = df['date'].dt.month
```

Usamos el atributo **.dt.month** para obtener el mes de cada fecha en la columna date.

| | |
|---|---|
| `quarter` | Extraemos el **trimestre del año al que pertenece cada fecha**, ya que los patrones de ventas pueden variar significativamente por trimestre. |

```
[ ]  #Extraemos el trimestre del año al que pertenece cada fecha.
     df['quarter'] = df['date'].dt.to_period('Q')
```

Con el método **.dt.to_period('Q')**, convertimos las fechas en periodos trimestrales (por ejemplo, '2024Q1' representa el primer trimestre de 2024).

| | |
|---|---|
| `is_event` | Creamos una **variable binaria para indicar si hubo un evento o no** en una semana determinada. |

```
[ ]  #Creamos una variable binaria para indicar si hubo un evento o no en una semana determinada.
     df['is_event'] = np.where(df['event'] == 'Sin_Evento', 0, 1)
```

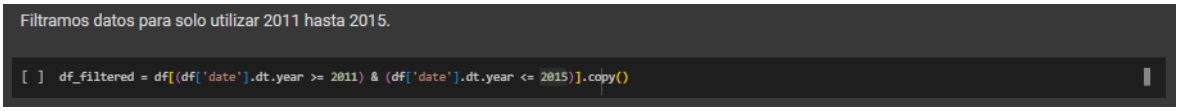
Usamos **np.where** para crear una nueva columna **is_event** donde si el valor en la columna event es 'Sin_Evento', asignamos un 0 (sin evento), y en caso contrario, un 1 (con evento).

#### 6.2.1.1. Selección de períodos a analizar

Antes de comenzar la dinámica de clustering, partimos del hecho de que nuestro dataset cuenta con información desde 2011-01-30 hasta 2016-04-24. Es decir, faltan datos del primer mes del 2011 y de gran parte de 2016, como podemos observar en el siguiente diagrama:


Consideramos una estrategia óptima para nuestro ejercicio de clustering limitar el análisis al período 2011-2015 dado que la ausencia de datos para parte de 2011 y gran parte de 2016 podría afectar negativamente la calidad del modelo del clustering generando una inconsistencia temporal, sesgando los resultados.

Al entender que las ventas de los productos son cíclicas y dependen de las temporadas y eventos que ocurren a lo largo de un año completo, podemos perder patrones consistentes que pueden proveer información estacional importante sobre las ventas trimestrales o el impacto de eventos que solo obtenemos si el rango temporal es completo.




Dicho esto, procedemos a filtrar nuestros datos para trabajar con datos obtenidos desde 2011 hasta 2015. Estos datos los almacenamos en la variable **df_filtered**.

### 6.2.2. Variables relacionadas a ventas

**Ventas totales**

| | |
|---|---|
| `total_sales` | **Ventas totales** por cada ítem. |

```
[ ]  df_total_sales = df_filtered.groupby('item')['ventas'].sum().reset_index()
     df_total_sales.columns = ['item', 'total_sales']
     df_total_sales
```

Agrupamos los datos filtrados (df_filtered) por la columna item y calcular la suma total de ventas por cada ítem.

El método **groupby('item')** agrupa los datos por el ítem, y **['ventas'].sum()** calcula la suma total de ventas de cada ítem. **reset_index()** se usa para convertir el índice del agrupamiento en una columna normal, creando un nuevo DataFrame.

Seguidamente, renombramos las columnas.

**Ventas totales por región para cada ítem**

| | |
|---|---|
| `ventas_Boston` | **Ventas totales en la región de Boston** para dado ítem. |
| `ventas_New_York` | **Ventas totales en la región de New York** para dado ítem. |
| `ventas_Philadelphia` | **Ventas totales en la región de Philadelphia** para dado ítem. |

```
[ ]  df_ventas_region = df_filtered.groupby(['item', 'region'])['ventas'].sum().reset_index()
     df_ventas_region = df_ventas_region.pivot(index='item', columns='region', values='ventas')
     df_ventas_region.columns = ['ventas_Boston', 'ventas_New York', 'ventas_Philadelphia']
     df_ventas_region.reset_index(inplace=True)
```

Agrupamos las ventas por ítem y por región, y luego sumamos las ventas para cada combinación de ítem y región.

Utilizamos **pivot** para transformar los datos de manera que los ítems sean las filas **(index='item')**, las regiones sean las columnas **(columns='region')**, y los valores en esas celdas sean las ventas **(values='ventas')**.

Seguidamente, renombramos las columnas y restablecemos el índice.

**Ratio de ventas regionales para cada ítem**

| | |
|---|---|
| `ratio_ventas_Boston` | **Ratio de las ventas en Boston** respecto a las ventas totales. |
| `ratio_ventas_Nueva_York` | **Ratio de las ventas en Nueva York** respecto a las ventas totales. |
| `ratio_ventas_Filadelfia` | **Ratio de las ventas en Philadelphia** respecto a las ventas totales. |

```
[ ]  # Combina las características de ventas regionales calculando la proporción de ventas en cada región respecto a las ventas totales.
     ratio_ventas_Boston = df_ventas_region['ventas_Boston'] / df_total_sales['total_sales']
     df_total_sales['ratio_ventas_Boston'] = ratio_ventas_Boston

[ ]  ratio_ventas_Nueva_York = df_ventas_region['ventas_New York'] / df_total_sales['total_sales']
     df_total_sales['ratio_ventas_Nueva_York'] = ratio_ventas_Nueva_York

[ ]  ratio_ventas_Filadelfia = df_ventas_region['ventas_Philadelphia'] / df_total_sales['total_sales']
     df_total_sales['ratio_ventas_Filadelfia'] = ratio_ventas_Filadelfia

[ ]  df_total_sales
```

Agrupamos las ventas por ítem y región, luego calcula los ratios de ventas de cada ítem en tres ciudades (Boston, Nueva York y Filadelfia) en relación a sus ventas totales.

Finalmente, agregamos estos ratios al DataFrame **df_total_sales** para analizar cómo se distribuyen las ventas por región. Esto nos permite obtener una medida más concisa de la distribución regional de las ventas, calculando la proporción de ventas en cada región en relación con las ventas totales.

**Ventas totales por trimestre para cada ítem.**

| | |
|---|---|
| `ventas_T1` | **Ventas totales durante el primer trimestre** para dado ítem. |
| `ventas_T2` | **Ventas totales durante el segundo trimestre** para dado ítem. |
| `ventas_T3` | **Ventas totales durante el tercer trimestre** para dado ítem. |
| `ventas_T4` | **Ventas totales durante el cuarto trimestre** para dado ítem. |

```
[ ]  #Agrupamos las ventas por cada ítem y trimestre para sumar las ventas de cada ítem en cada trimestre.
     df_sales_quarter = df_filtered.groupby(['item', 'trimestre'])['ventas'].sum().reset_index()

     #Usamos pivot para reorganizar los datos de tal manera que cada ítem tenga sus ventas distribuidas en columnas por trimestre.
     df_sales_quarter = df_sales_quarter.pivot(index='item', columns='trimestre', values='ventas')
     df_sales_quarter.columns = ['ventas_T1', 'ventas_T2', 'ventas_T3', 'ventas_T4']

     # Creamos una nueva columna que sume las ventas de todos los trimestres para obtener las ventas totales de cada ítem.
     df_sales_quarter['total_sales'] = df_sales_quarter[['ventas_T1', 'ventas_T2', 'ventas_T3', 'ventas_T4']].sum(axis=1)
```

Agrupamos los datos por las columnas **item** y **trimestre** utilizando **groupby**, luego se calcula la suma de ventas por cada combinación. El método **reset_index()** se usa para convertir el resultado en un DataFrame regular.

Usamos pivot para reorganizar los datos de tal manera que cada ítem tenga sus ventas distribuidas en columnas por trimestre.

Seguidamente renombramos las columnas para que reflejen claramente que el contenido corresponde a las ventas de cada trimestre.

**Ratio de las ventas para cada ítem por trimestre respecto a las ventas totales**

| | |
|---|---|
| `ventas_T1_ratio` | **Ratio de las ventas en Trimestre 1** respecto a las ventas totales. |
| `ventas_T2_ratio` | **Ratio de las ventas en Trimestre 2** respecto a las ventas totales. |

| | |
|---|---|
| `ventas_T3_ratio` | **Ratio de las ventas en Trimestre 3** respecto a las ventas totales. |
| `ventas_T4_ratio` | **Ratio de las ventas en Trimestre 4** respecto a las ventas totales. |

```
#Para cada trimestre, se calcula el porcentaje de las ventas totales que ocurrió en ese trimestre.
df_sales_quarter['ventas_T1_ratio'] = df_sales_quarter['ventas_T1'] / df_sales_quarter['total_sales']
df_sales_quarter['ventas_T2_ratio'] = df_sales_quarter['ventas_T2'] / df_sales_quarter['total_sales']
df_sales_quarter['ventas_T3_ratio'] = df_sales_quarter['ventas_T3'] / df_sales_quarter['total_sales']
df_sales_quarter['ventas_T4_ratio'] = df_sales_quarter['ventas_T4'] / df_sales_quarter['total_sales']

#Rellenamos los valores faltantes (NaN) con ceros, en caso de que un ítem no haya tenido ventas en uno o más trimestres.
df_sales_quarter.fillna(0, inplace=True)

#Restablecemos el índice.
df_sales_quarter.reset_index(inplace=True)
df_sales_quarter.drop(columns=['total_sales', 'ventas_T1', 'ventas_T2', 'ventas_T3', 'ventas_T4'], inplace=True)
df_sales_quarter
```

Se divide las ventas de cada trimestre (**ventas_T1**, **ventas_T2**, etc.) por las ventas totales (**total_sales**). Esto genera nuevas columnas para cada trimestre que contienen los ratios (**ventas_T1_ratio**, **ventas_T2_ratio**, etc.).

Usamos **fillna(0)** para reemplazar cualquier valor nulo en el DataFrame por un 0, asegurando que no queden valores vacíos en las columnas.

Se usa el método **drop()** para eliminar las columnas innecesarias, dejando solo los ratios.

### 6.2.3. Variables relacionadas a precio

En esta sección del análisis, desarrollamos varias variables clave relacionadas al precio para obtener una mejor comprensión de su impacto en el comportamiento de ventas de cada ítem. Estas variables son fundamentales para identificar la volatilidad de ventas, el precio promedio y la sensibilidad al precio de los ítems, elementos que pueden influir significativamente en la agrupación dentro de nuestro ejercicio de clustering.

**Volatilidad de Ventas por ítem**

| | |
|---|---|
| `sales_volatility` | Captura cuán inestables son las ventas de un producto. Los ítems con alta volatilidad podrían tener ventas más erráticas. |

```
[ ]  df_sales_volatility = df_filtered.groupby('item')['ventas'].std().reset_index()
     df_sales_volatility.columns = ['item', 'sales_volatility']
```

Calculamos la **desviación estándar** de las ventas por ítem, lo que refleja la **volatilidad** o variabilidad de las ventas a lo largo del tiempo.

Un valor más alto indica que las ventas de ese ítem fluctúan más mes a mes.

**Precio Promedio por Ítem**

| | |
|---|---|
| `avg_price` | Nos permite identificar qué productos tienden a ser más baratos o caros, y cómo podría relacionarse su precio con las ventas. |

```
[ ]  df_avg_price = df_filtered.groupby('item')['sell_price'].mean().reset_index()
     df_avg_price.columns = ['item', 'avg_price']
```

Calculamos el **precio promedio** de cada ítem en el periodo 2011-2015 agrupando el dataset por ítem aplicando **.mean()** al valor de **sell_price**.

### Sensibilidad al Precio por Ítem

| | |
|---|---|
| `price_sensitivity` | Refleja cómo los cambios en el precio afectan las ventas de un ítem. Productos con alta sensibilidad podrían requerir estrategias de precios más ajustadas para maximizar sus ventas. |

```
df_price_sensitivity = df_filtered.groupby('item').apply(lambda x: x['sell_price'].corr(x['ventas'])).reset_index()
df_price_sensitivity.columns = ['item', 'price_sensitivity']
df_price_sensitivity
```

Calculamos la **sensibilidad al precio** para cada **item** utilizando la correlación entre el precio de venta (**sell_price**) y las ventas (**ventas**).
Agrupamos los datos por cada **item** y para cada grupo (**ítem**) se calcula la correlación de Pearson entre el precio de venta (**sell_price**) y el número de ventas (ventas). Esto refleja cómo los cambios en el precio están relacionados con los cambios en las ventas.

El resultado lo convertimos en un Data Frame con las columnas **'item'** y '**price_sensitivity**'.

**Manera de interpretarlo**

- **Valores cercanos a -1**: Relación negativa fuerte, las ventas bajan cuando el precio sube.
- **Valores cercanos a 1**: Relación positiva fuerte, las ventas aumentan cuando el precio sube.
- **Valores cercanos a 0**: No hay relación clara entre precio y ventas.

### Característica de Precio Relativo

| | |
|---|---|
| `precio_relativo` | Variable que normaliza el precio de cada artículo comparándolo con el precio promedio de los artículos de su misma categoría. Esto ayuda a entender cómo se posiciona el precio de un artículo en relación con otros similares. |

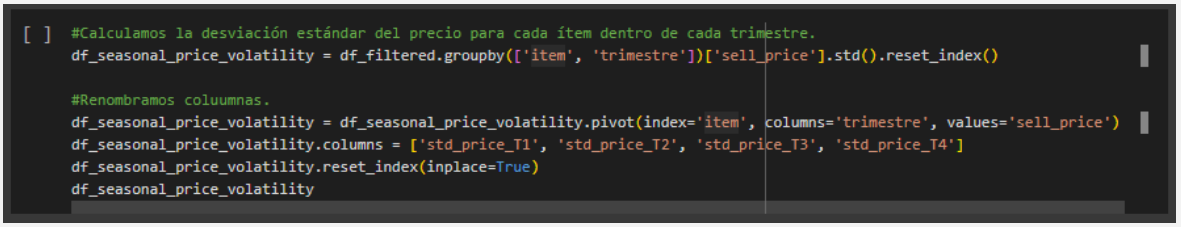
```
[ ]  #Calculamos el precio promedio de cada categoría y asigna ese valor a cada ítem dentro de la categoría correspondiente.
     precio_promedio_categoria = df_filtered.groupby('category')['sell_price'].transform('mean')

     # Creamos una nueva característica de precio relativo.
     df_precio_relativo = df_avg_price['avg_price'] / precio_promedio_categoria

     df_precio_relativo
```

Calculamos el precio relativo de cada ítem respecto al promedio de su categoría. El objetivo es medir si un ítem es más caro o más barato que otros productos similares dentro de su misma categoría, lo que nos proporciona una perspectiva útil para entender su posicionamiento de precio dentro de su segmento.

Un precio relativo más alto puede indicar un producto premium dentro de su categoría, mientras que un valor bajo podría señalar un producto económico en comparación con sus competidores.

**Volatilidad Estacional de Precios por Trimestre**

| | |
|---|---|
| **std_price_T1** | Volatilidad Estacional de Precios para el Trimestre 1. |
| **std_price_T2** | Volatilidad Estacional de Precios para el Trimestre 2. |
| **std_price_T3** | Volatilidad Estacional de Precios para el Trimestre 3. |
| **std_price_T4** | Volatilidad Estacional de Precios para el Trimestre 4. |



```
[ ]  #Calculamos la desviación estándar del precio para cada ítem dentro de cada trimestre.
     df_seasonal_price_volatility = df_filtered.groupby(['item', 'trimestre'])['sell_price'].std().reset_index()

     #Renombramos coluumnas.
     df_seasonal_price_volatility = df_seasonal_price_volatility.pivot(index='item', columns='trimestre', values='sell_price')
     df_seasonal_price_volatility.columns = ['std_price_T1', 'std_price_T2', 'std_price_T3', 'std_price_T4']
     df_seasonal_price_volatility.reset_index(inplace=True)
     df_seasonal_price_volatility
```

Agrupamos el DataFrame por la columna **category** y se calcula la media de los precios de venta (**sell_price**). Luego, el método **transform('mean')** asegura que este valor promedio se asigne a cada ítem dentro de su categoría.

Seguidamente, tomamos el precio promedio del ítem (**avg_price**) y se divide entre el precio promedio de la categoría a la que pertenece. Este ratio indica si un ítem es más caro o más barato en relación con otros ítems de su misma categoría.

De esta manera, medimos cómo fluctúa el precio de un producto durante diferentes estaciones o períodos de tiempo (trimestres en este caso).

### 6.2.4. Variables relacionadas a crecimiento de ventas

**Crecimiento anual de ventas para cada ítem entre los años 2011 y 2015**

| | |
|---|---|
| **growth_2012** | Crecimiento anual de ventas 2011-2012. |
| **growth_2013** | Crecimiento anual de ventas 2012-2013. |
| **growth_2014** | Crecimiento anual de ventas 2013-2014. |
| **growth_2015** | Crecimiento anual de ventas 2014-2015. |

```
[ ]  #Definimos una función que tiene como objetivo calcular el crecimiento anual de ventas de 2011 a 2015 para cada ítem.
     def calculate_annual_sales_growth(df):

       # Filtramos datos para obtener datos soolo del año 2011 hasta 2015.
       df_filtered = df[(df['año'] >= 2011) & (df['año'] <= 2015)]

       #Agrupamos por ítem y año, añadiendo las ventas.
       sales_by_item_year = df_filtered.groupby(['item', 'año'])['ventas'].sum().reset_index()

       #Pivotamos la tabla para tener los años como columnas.
       sales_pivot = sales_by_item_year.pivot(index='item', columns='año', values='ventas')

       #Calculamos la tasa de crecimiento.
       sales_pivot['growth_2012'] = ((sales_pivot[2012] - sales_pivot[2011]) / sales_pivot[2011]) * 100
       sales_pivot['growth_2013'] = ((sales_pivot[2013] - sales_pivot[2012]) / sales_pivot[2012]) * 100
       sales_pivot['growth_2014'] = ((sales_pivot[2014] - sales_pivot[2013]) / sales_pivot[2013]) * 100
       sales_pivot['growth_2015'] = ((sales_pivot[2015] - sales_pivot[2014]) / sales_pivot[2014]) * 100

       return sales_pivot[['growth_2012', 'growth_2013', 'growth_2014', 'growth_2015']]
```

Definimos una función que tiene como objetivo calcular el crecimiento anual de ventas de 2011 a 2015 para cada ítem.

```
     #Apliacamos la función al dataset df.
     annual_sales_growth = calculate_annual_sales_growth(df)

     annual_sales_growth
```

Aplicamos la función al dataset df.

```
[ ]  annual_sales_growth.replace([np.inf, -np.inf], np.nan, inplace=True)
     annual_sales_growth.fillna(0, inplace=True)
     annual_sales_growth
```

Reemplazamos los valores infinitos (**np.inf** y **-np.inf**) por **NaN** en el DataFrame que hemos llamado **annual_sales_growth**, para evitar errores al trabajar con estos valores.
Rellenamos los valores **NaN** con ceros (**fillna(0)**), asegurandonos que no haya valores faltantes en el DataFrame.

```
[ ]  annual_sales_growth['average_growth'] = annual_sales_growth.mean(axis=1)
     annual_sales_growth
```

Finalmente, calculamos el promedio del crecimiento anual para cada ítem a lo largo de los años y creamos una nueva columna **average_growth** en el mismo Dataframe que almacena el resultado.

### 6.2.5. Variables relacionadas a eventos

**Ratio de Impacto de Eventos por Trimestre.**

| | |
|---|---|
| **event_impact_ratio_T1** | Ratio de Impacto de Eventos para el Trimestre 1. |
| **event_impact_ratio_T2** | Ratio de Impacto de Eventos para el Trimestre 2. |
| **event_impact_ratio_T3** | Ratio de Impacto de Eventos para el Trimestre 3. |
| **event_impact_ratio_T4** | Ratio de Impacto de Eventos para el Trimestre 4. |

```
[ ]  #Calculamos las ventas totales durante eventos por ítem y trimestre.
     event_sales = df_filtered[df_filtered['is_event'] == 1].groupby(['item', 'trimestre'])['ventas'].sum().reset_index()
     event_sales.rename(columns={'ventas': 'event_sales'}, inplace=True)

     #Calculamos las ventas totales por ítem y trimestre.
     total_sales_per_quarter = df_filtered.groupby(['item', 'trimestre'])['ventas'].sum().reset_index()
     total_sales_per_quarter.rename(columns={'ventas': 'total_sales_per_quarter'}, inplace=True)

     #Unión de las ventas totales con las ventas durante eventos.
     df_event_impact = pd.merge(event_sales, total_sales_per_quarter, on=['item', 'trimestre'], how='left')

     #Calculamos el ratio de impacto de eventos.
     df_event_impact['event_impact_ratio_per_quarter'] = df_event_impact['event_sales'] / df_event_impact['total_sales_per_quarter']

     #Reorganizamos el DataFrame con trimestres como columnas.
     df_event_impact = df_event_impact.pivot(index='item', columns='trimestre', values='event_impact_ratio_per_quarter')
     df_event_impact.rename(columns={1: 'event_impact_ratio_T1', 2: 'event_impact_ratio_T2', 3: 'event_impact_ratio_T3', 4: 'event_impact_ratio_T4'}, inplace=True)
     df_event_impact
```

Con este código calcula el impacto de los eventos en las ventas trimestrales de cada ítem, generando un ratio que muestra qué proporción de las ventas de un ítem en un trimestre ocurrió durante eventos.
Luego reorganizamos los datos para que los trimestres sean columnas y los ítems sean filas.

Finalmente, combinamos todas las características previamente calculadas (ventas totales, precio promedio, sensibilidad al precio, ventas por trimestre, impacto de eventos y crecimiento anual) en un solo DataFrame llamado **df_features**, utilizando el ítem como clave de unión. Seguidamente, establecemos la columna **item** como índice para que sea fácil acceder a las características de cada producto.

```
[104] df_features = df_total_sales \
          .merge(df_sales_quarter, on='item', how='left') \
          .merge(df_sales_volatility, on='item', how='left') \
          .merge(df_avg_price, on='item', how='left') \
          .merge(df_price_sensitivity, on='item', how='left') \
          .merge(df_seasonal_price_volatility, on='item', how='left') \
          .merge(df_annual_sales_growth, on='item', how='left') \
          .merge(df_event_impact, on='item', how='left')

      df_features.set_index('item', inplace=True)

      df_features
```

Este es el dataset “madre” del cual seleccionaremos las variables que nos interesa explorar en las distintas iteraciones que haremos a lo largo de nuestro ejercicio.

## 6.3. Clustering de productos: Iteración 1

En nuestra primera iteración, evaluaremos variables globales para obtener un panorama global sobre cuáles variables podemos descartar con el fin de simplificar nuestro modelo. El objetivo aquí es construir un modelo con una visión amplia y variada del comportamiento de los productos; sin embargo, no es recomendable analizar todas las variables al mismo tiempo ya que, a medida que aumentamos el número de variables (o dimensiones), los

datos se vuelven más dispersos y distantes entre sí. En alta dimensionalidad, los puntos de datos tienden a alejarse unos de otros, lo que dificulta que el algoritmo de K-Means identifique grupos bien definidos.

### 6.3.1. Evaluación de Features

Al diseñar un modelo de clustering o cualquier otro modelo de aprendizaje automático, la calidad de las características (o features) es crucial. Dos herramientas clave para evaluar nuestras variables son la varianza de las características (feature variance) y la correlación entre características (feature correlation). Estas métricas nos ayudan a entender la importancia de las características y su impacto en el modelo.

#### 6.3.1.1. Feature Variance

La varianza mide la dispersión o distribución de los valores de una característica en relación con su media. En otras palabras, nos indica cuánta variación existe en una característica entre los diferentes ítems o registros en el dataset.

```
feature_variances = df_features.var()

print("Feature Variances:\n", feature_variances)
```

```
Feature Variances:
 total_sales                1.727335e+09
ratio_ventas_Boston         9.135408e-03
ratio_ventas_Nueva_York     1.281527e-02
ratio_ventas_Filadelfia     1.007430e-02
ventas_T1_ratio             2.500128e-03
ventas_T2_ratio             2.362395e-03
ventas_T3_ratio             2.604914e-03
ventas_T4_ratio             3.951024e-03
sales_volatility            2.235551e+02
avg_price                   2.054069e+01
precio_relativo             4.936928e-01
price_sensitivity           2.445322e-02
std_price_T1                9.393868e-02
std_price_T2                9.447836e-02
std_price_T3                7.636687e-02
std_price_T4                1.026768e-01
growth_2012                 3.420251e+08
growth_2013                 1.582832e+07
growth_2014                 2.562686e+07
growth_2015                 2.513637e+07
average_growth              2.545284e+07
event_impact_ratio_T1       1.190489e-03
event_impact_ratio_T2       1.302401e-03
event_impact_ratio_T3       4.515192e-04
event_impact_ratio_T4       3.126910e-04
dtype: float64
```

En el contexto de un modelo de clustering, la varianza de las características es crucial porque una alta varianza indica que una característica tiene un rango amplio de valores y podría contener información valiosa para diferenciar los ítems. Por ejemplo, si algunas ventas totales son muy altas y otras son bajas, esto probablemente ayudará al modelo a agrupar productos de manera efectiva.

Al analizar la varianza de las características en nuestro caso, observamos que variables como total_sales tienen una varianza muy alta (1.72e+09), lo que sugiere una gran disparidad en las ventas entre los ítems. Esta característica puede ser muy útil para distinguir productos con altos y bajos volúmenes de ventas.

Por otro lado, una baja varianza sugiere que los valores de la característica están muy concentrados alrededor de la media, lo que podría implicar que esta característica no está aportando suficiente información. En algunos casos, las características con baja varianza podrían incluso ser eliminadas, ya que no contribuyen significativamente al proceso de agrupamiento o clasificación.

En nuestro caso, podemos observar que características como los ratios de ventas por ciudad o trimestre tienen varianzas bajas, lo que indica que la mayoría de los productos tienen una distribución de ventas similar en esos segmentos. Esto podría sugerir que estas características son menos influyentes al intentar distinguir diferentes clusters.

En base a estos resultados, las acciones que tomaríamos serían proceder a eliminar características con varianza extremadamente baja (por ejemplo, características que no varían mucho, como algunas relacionadas con ratios de ventas ya que no están proporcionando mucha información diferenciadora.)

Por otra parte, debemos también considerar estandarizar características ya que las características con varianzas extremadamente altas, como total_sales, pueden dominar el modelo de clustering. De esta manera nos aseguramos que características dominantes no tengan un peso desproporcionado en el modelo.

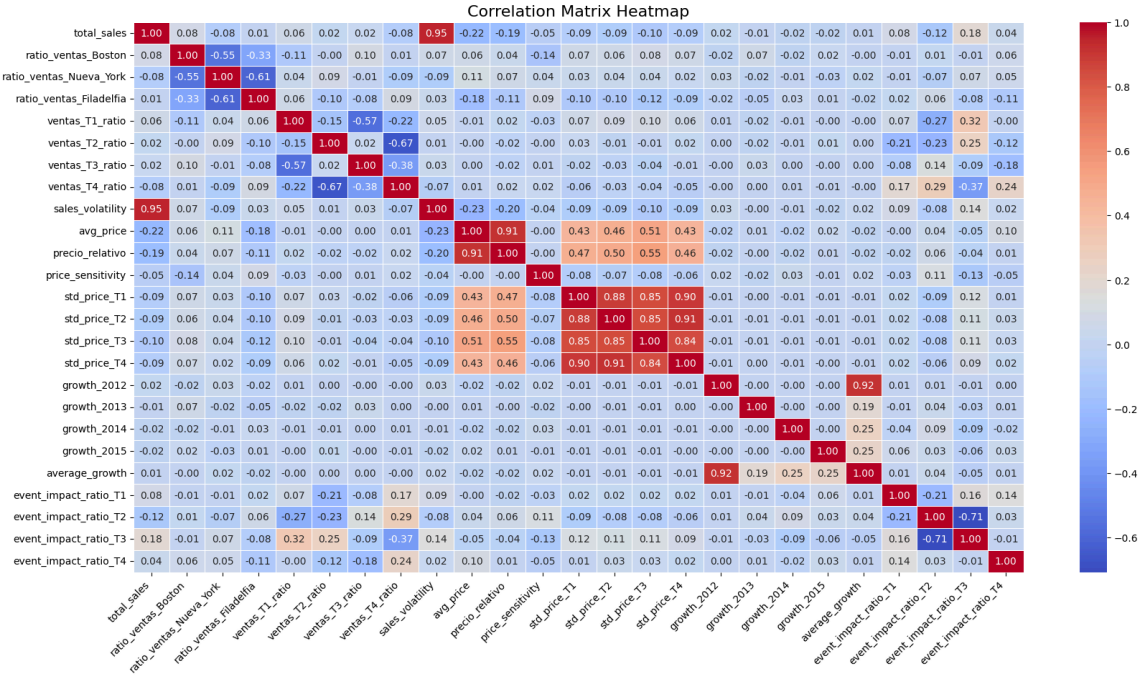
No obstante, antes de considerar descartar cualquier variable, procedemos a evaluar la correlación entre características para poder obtener una decisión más acertada.

#### 6.3.1.2. Feature Correlation

La correlación de Pearson es importante porque nos permite identificar si dos características están fuertemente relacionadas. Si dos características tienen una alta correlación positiva o negativa, esto puede sugerir que ambas están proporcionando información redundante al modelo. Al aplicarse la correlación entre variables, se mide la fuerza y dirección de la relación lineal entre las dos variables analizadas.

El valor de la correlación varía entre -1 (correlación negativa perfecta) y 1 (correlación positiva perfecta). Una correlación positiva perfecta implica que, cuando una característica aumenta, la otra también lo hace proporcionalmente; mientras que por el contrario, una correlación negativa perfecta (cuando una característica aumenta, la otra disminuye proporcionalmente). Un 0 significa que no hay relación lineal entre las dos características.

A continuación se presentan algunos insights clave basados en el análisis del mapa de calor de las variables:




**Altas correlaciones:** Si dos características están altamente correlacionadas (por ejemplo, más de 0.8 o menos de -0.8), puede ser innecesario incluir ambas en el modelo, ya que están aportando información similar. En este caso, podríamos eliminar una de ellas o crear una nueva característica combinada que represente ambas.

1. **Correlación alta entre total_sales y sales_volatility.** Los productos con mayores ventas totales tienden a tener una mayor volatilidad en las ventas.
2. **Correlación alta entre las desviaciones estándar de precios.** Los valores de correlación muy altos (superiores a 0.85) entre las desviaciones estándar trimestrales de precios indican que, si un producto tiene alta variabilidad en el precio durante un trimestre, es probable que esa variabilidad también esté presente en los otros trimestres.
3. **Correlación entre ratio_ventas_Boston y total_sales.** Existe una correlación positiva alta entre las ventas totales y el ratio de ventas en Boston.
4. **Impacto de eventos por trimestres.** Hay correlación moderada a fuerte entre los ratios de impacto de eventos en diferentes trimestres, por ejemplo, **event_impact_ratio_T2** y **event_impact_ratio_T1** tienen una correlación de 0.29.

**Bajas correlaciones:** Las características con bajas correlaciones entre sí aportan información única al modelo, lo que es deseable, ya que queremos que cada característica agregue algo nuevo al análisis.

1. **Relación entre el ratio de ventas trimestrales y ventas_T2_ratio y ventas_T4_ratio.** Hay una correlación negativa significativa entre el ratio de ventas del T2 y el ratio de ventas del T4. Esto sugiere que los productos que tienen un buen desempeño en el segundo trimestre tienden a tener un desempeño más bajo en el cuarto trimestre, lo que puede deberse a factores estacionales.
2. **Baja correlación general entre el avg_price y otras características.** El precio promedio de los productos parece tener baja correlación con otras características como las ventas totales o la volatilidad, lo que podría indicar que el precio no está influyendo fuertemente en el comportamiento general de las ventas.

#### 6.3.1.3. Conclusiones de la Evaluación de Features

En base a los resultados de Feature Variance y el mapa de correlación de variables, podemos hacer una selección de las características más relevantes para el primer ejercicio de clustering y descartar aquellas que no aportan valor significativo o que están altamente correlacionadas con otras variables:

- **Las variables con baja varianza** las hemos de descartar porque no aportan suficiente variabilidad al modelo, lo que significa que todos los productos tienen valores similares en esas características, haciendo difícil que sean útiles para separar clusters. Como pudimos observar, las ventas distribuidas por trimestre muestran baja variabilidad en general, lo que puede implicar que los productos tienen un comportamiento estacional similar y estas variables no aportan suficiente diferenciación.
- **Las variables con alta correlación entre sí** pueden ser redundantes y no es necesario incluir ambas en el análisis. Pudimos identificar algunas variables que tienen una correlación significativa (mayor a 0.85) y pueden ser eliminadas para reducir la redundancia. Las listamos a continuación.

**Variables a descartar**

| | |
|---|---|
| **Baja Varianza** | **Ratios de impacto de eventos**<br>● **event_impact_ratio_T3**<br>● **event_impact_ratio_T4**<br><br>**Ratios de ventas por trimestre**<br>● **ventas_T1_ratio**<br>● **ventas_T2_ratio** |

| | |
|---|---|
| | • **ventas_T3_ratio**<br><br>**Precio relativo**<br>Aunque no es la varianza más baja, es relativamente pequeña en comparación con otras variables, lo que indica que no hay mucha variabilidad en la posición de los productos dentro de su categoría.<br>• **precio_relativo** |
| **Alta correlación** | **Variables de desviaciones estándar de precios**<br>Tienen una alta correlación con **std_price_T**1. Sería suficiente con mantener una de estas.<br>• **std_price_T2**<br>• **std_price_T3**<br>• **std_price_T4**<br><br>**Variables relativas a crecimiento**<br>• **growth_2014** muestra una alta correlación con **growth_2012** (0.92). |

### Variables a mantener

| | |
|---|---|
| **Mantenemos Alta varianza** | **Variables de ventas**<br>• **total_sales:** Clave para entender la distribución de ventas entre los productos.<br>• **sales_volatility:** Muestra la dispersión en las ventas. Esto es importante para identificar productos estables frente a productos con fluctuaciones.<br><br>**Precio**<br>• **avg_price:** Aporta variabilidad moderada y es una característica clave para entender el comportamiento de los productos en función del precio.<br>• **price_sensitivity:** Mide cómo reaccionan las ventas ante cambios de precio, esencial para productos con alta elasticidad de precio. |
| **Mantenemos Información única** | **Ratios de ventas regionales**<br>Aunque no tienen una varianza extremadamente alta, aportan información clave sobre el comportamiento de ventas en diferentes regiones, lo que puede ser útil para segmentar los productos geográficamente.<br>• **ratio_ventas_Boston**<br>• **ratio_ventas_Nueva_York**<br>• **ratio_ventas_Filadelfia** |

Finalizamos nuestro proceso de evaluación de características creando una variable llamada **selected_features,** la cual almacenará nuestro Data Frame **df_features** sólo con las columnas (variables) que nos interesan mantener para cada iteración que haremos.

Las variables que no nos interese mantener para ciertas iteraciones procedemos a excluirlas (comentarlas). Y con esto, ya estamos listos para nuestro próximo paso: la Estandarización de Datos.

### 6.3.2. Estandarización de Datos

Nuestro siguiente paso consiste en **normalizar** las características. Dado que las características tienen diferentes escalas (por ejemplo, el precio se mide en diferentes unidades que las ventas), es esencial estandarizar los datos. Para asegurar que el algoritmo de clustering pueda agrupar los productos de manera efectiva, se realizó una Estandarización de Datos mediante técnicas de normalización. Este proceso garantiza que todas las características tengan una media de 0 y una desviación estándar de 1, lo que evita que variables con escalas más grandes dominen el cálculo de distancias entre puntos.

```
[ ]  from sklearn.preprocessing import StandardScaler

     #Seleccionamos variables para nuestra primera iteracion.
     selected_features = df_features[[
         'total_sales',
         'ratio_ventas_Boston',
         'ratio_ventas_Nueva_York',
         'ratio_ventas_Filadelfia',
         'avg_price',
         'price_sensitivity',
         #'ventas_T1_ratio',
         #'ventas_T2_ratio',
         #'ventas_T3_ratio',
         #'ventas_T4_ratio',
         #'event_impact_ratio_T1',
         #'event_impact_ratio_T2',
         #'event_impact_ratio_T3',
         #'event_impact_ratio_T4',
         'growth_2012',
         'growth_2013',
         'growth_2014',
         'growth_2015',
         'average_growth'
         ]]

     selected_features = selected_features.fillna(0)

     #Normalizamos datos.
     scaler = StandardScaler()
     scaled_features = scaler.fit_transform(selected_features)
```

*Diagrama: Estandarización de datos para las variables escogidas, Iteración 1.*

Para lograr esto, utilizamos el objeto **StandardScaler** de la biblioteca sklearn, que transforma los datos de tal manera que cada característica tenga una **media de 0** y una **desviación estándar de 1**. Este paso asegura que ninguna característica, por tener un rango de valores mayor (como **total_sales**), domine el modelo a la hora de determinar las distancias entre puntos.

### 6.3.3. Análisis de Componentes Principales (PCA)

El **Análisis de Componentes Principales (PCA)** es una técnica de reducción de dimensionalidad que permite transformar un conjunto de variables correlacionadas en un conjunto más pequeño de variables no correlacionadas, llamadas componentes principales.

El PCA es útil durante el proceso de Clustering porque se encarga de reducir el número de dimensiones de los datos, lo que puede facilitar la agrupación y la visualización, especialmente cuando trabajamos con muchas variables. No obstante, el PCA es sensible a las escalas de las variables; por esto la necesidad de estandarizar nuestros datos como ya hemos explicado en el apartado anterior.

Para nuestro ejercicio hemos seleccionado los **6 primeros componentes principales**, cada uno con una varianza explicada de **[0.182, 0.157, 0.133, 0.107, 0.091, 0.090]**, donde el primer componente explica el mayor porcentaje de la varianza y cada componente sucesivo contribuye en menor medida. Estos, en total, suman un **76.07% de la varianza total** del conjunto de datos original lo cual significa que el 76.07% de la estructura original del dataset (su variabilidad) se mantiene al trabajar con 6 componentes principales, lo que reduce considerablemente la dimensionalidad sin perder demasiada información.

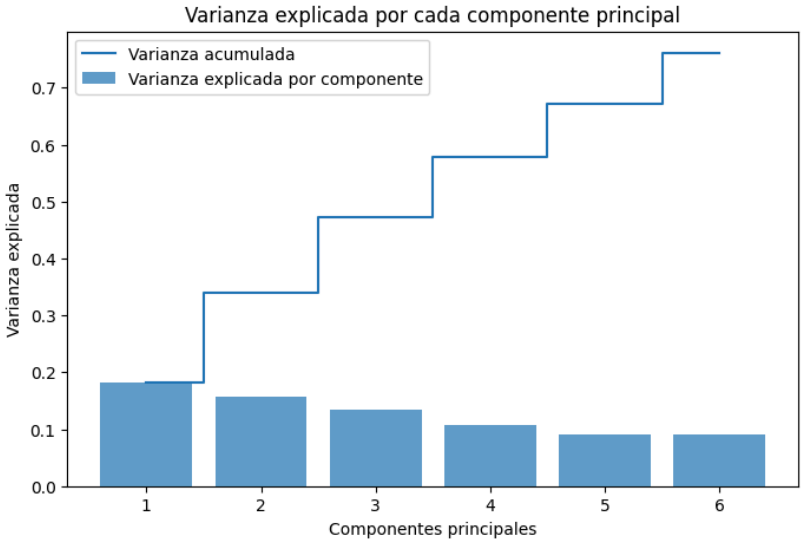



*Diagrama: Análisis de Componentes Principales (PCA), Iteración 1.*

Esta reducción de dimensionalidad facilita el posterior análisis de clustering al operar en un espacio más manejable sin perder información relevante, y permite interpretar mejor los resultados del modelo en términos de grupos formados.

### 6.3.4. K-Means: Selección y aplicación del algoritmo de Clustering

Existen diversos algoritmos de clustering, cada uno con sus propias fortalezas y debilidades. En este proyecto se utilizó el algoritmo K-Means, uno de los más populares debido a su simplicidad y eficiencia para agrupar grandes conjuntos de datos.

K-Means funciona agrupando los datos en un número K de clusters predefinido. El algoritmo busca minimizar la distancia cuadrática entre los puntos dentro de un cluster y el centroide (el punto central) del mismo cluster. Los pasos principales del algoritmo son:

1. Seleccionar K centroides de manera aleatoria.
2. Asignar cada punto de datos al cluster cuyo centroide esté más cercano.
3. Recalcular los centroides en función de los puntos asignados a cada cluster.
4. Repetir los pasos 2 y 3 hasta que los centroides ya no cambien significativamente, es decir, hasta la convergencia.

El proceso de clustering fue implementado con el siguiente flujo:

- **Inicialización:** Se eligió un valor inicial de ***K***, que representa el número de clusters deseado. En este proyecto, se probaron varias configuraciones de ***K*** para encontrar el número óptimo de grupos que mejor segmentara los productos.
- **Asignación de Clusters:** A cada producto se le asignó un cluster basado en la similitud de las características, medida por la distancia euclidiana a los centroides de los clusters.
- **Reajuste de Centroides:** Tras asignar los productos a los clusters iniciales, se recalcularon los centroides de cada grupo como el promedio de todos los puntos en ese cluster. Este proceso se repitió hasta que los productos dejaron de cambiar de cluster, lo que indica que el modelo ha convergido.

## 6.3.5. Evaluación del Número Óptimo de Clusters

### 6.3.5.1. Curva del Codo

Una de las decisiones más importantes en el clustering es elegir el número adecuado de clusters K. Para ello, se utilizó el **método del codo (Elbow Method)**, que consiste en calcular la suma de las distancias cuadráticas entre los puntos y sus centroides para diferentes valores de K. Este valor es conocido como la **inercia** del modelo. Se selecciona el valor de K donde la reducción de inercia deja de ser significativa, formando una curva en forma de codo.

En nuestro caso, el "codo" de la curva se observa alrededor de **k = 4**. Esto indica que **4 clusters** es un buen compromiso entre el número de grupos y la inercia. Después de este punto, la inercia sigue disminuyendo, pero a un ritmo mucho más lento, lo que sugiere que agregar más clusters no mejorará significativamente la compactación de los datos.

Usar más de 4 clusters podría llevar a **sobreajustar** el modelo, mientras que usar menos podría no capturar toda la estructura presente en los datos. Así, en este análisis de clustering, **k = 4** parece ser una opción razonable y justificada para agrupar los datos de manera efectiva.

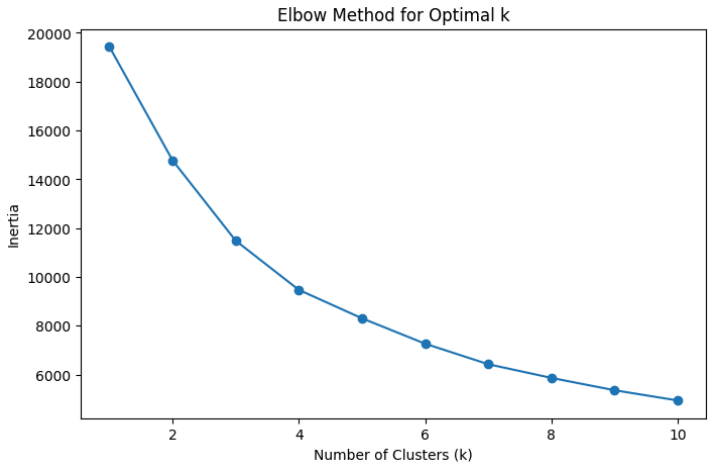



*Diagrama: Curva del Codo, Iteración 1.*

Con base en este análisis, utilizaré **4 clusters** para los siguientes pasos del ejercicio de clustering con K-Means, ya que este número parece balancear bien la simplicidad del modelo y la precisión de los grupos obtenidos.

Este resultado será seguido por la interpretación de los clusters formados y el análisis de las características principales de cada grupo.

#### 6.3.5.2. Silhouette Score

Aparte de la Curva del Codo, otra técnica evaluada fue la **silhouette score** la cual mide qué tan bien separadas están las muestras dentro de sus clusters asignados en comparación con otros clusters. Un puntaje cercano a 1 indica que los clusters están bien definidos.

En el siguiente diagrama, el eje Y muestra la calidad de los clusters formados. Podemos observar que **k = 2** tiene el **Silhouette Score más alto** (cerca de **0.9**), lo que indica que los datos probablemente se agrupan mejor en 2 clusters. Los puntos dentro de cada cluster están bien separados de los demás, y los clusters formados son de alta calidad.

A partir de **k = 3**, el Silhouette Score cae drásticamente a alrededor de **0.2-0.3**, y se mantiene relativamente constante. Este valor indica que la calidad de los clusters es **baja** para estos números de clusters, lo que sugiere que los clusters se superponen o no están bien definidos.

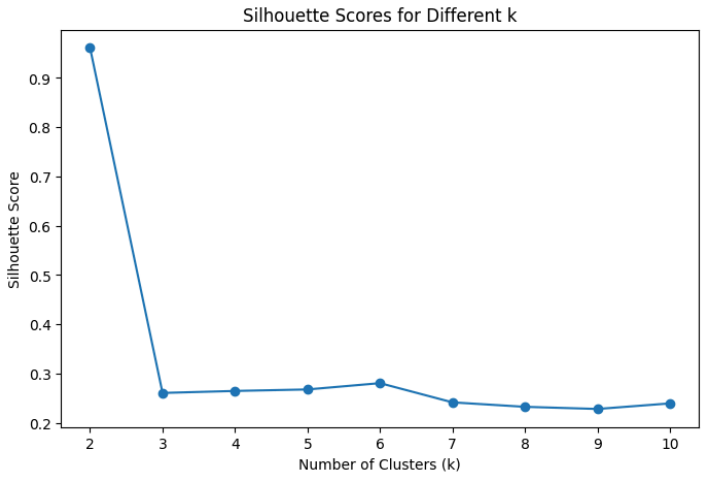



*Diagrama: Silhouette Score, Iteración 1.*

Por otra parte, aunque el **k = 2** tiene el puntaje de silueta más alto, consideramos que esta podría ser una solución demasiado simple porque necesitaríamos más granularidad en el análisis por lo cual podríamos explorar **k = 3 o 4**, aunque con una ligera reducción en la calidad de los clusters. Procedemos a realizar el ejercicio con 4 clusters y analizar sus resultados.

### 6.3.6. Interpretación de los Resultados

Una vez finalizado el proceso de clustering, se analizó la composición de cada cluster para entender qué productos fueron agrupados y por qué. Al analizar las distintas variables usadas por el algoritmo, podemos identificar patrones ocultos en los datos y segmentar los productos de manera que se puedan diseñar estrategias de marketing más efectivas o realizar análisis de ventas más detallados.

#### 6.3.6.1. Visualización y Análisis de Clusters

Finalmente, se realizó una visualización de los clusters para facilitar la interpretación de los resultados. Gráficos como los diagramas de dispersión nos ayudaron a mostrar las relaciones entre las características clave dentro de los grupos.

En base al gráfico de *Visualización de Clusters (IT1) Iteración 1*, podemos observar algunos aspectos clave sobre la distribución de los datos en los primeros dos componentes principales del análisis de PCA y los clusters formados por el algoritmo K-Means:

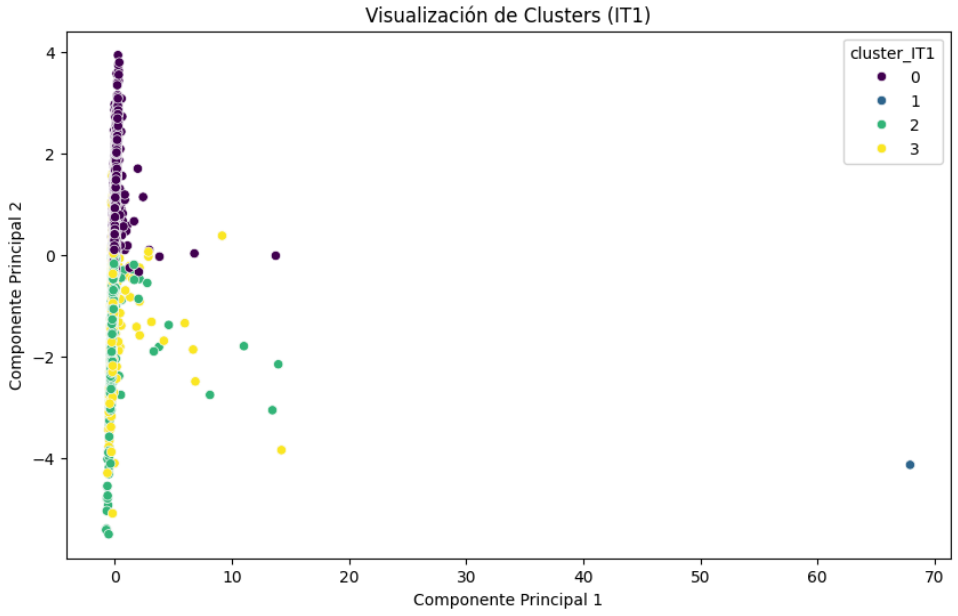



*Diagrama: Visualización de Clusters, Iteración 1.*

El alto grado de solapamiento entre los clusters puede indicar que los datos no están bien separados en las primeras dos componentes principales. Esto podría implicar que más componentes principales son necesarios para una mejor separación y que el modelo se beneficiaría de una exploración de más variables para afinar nuestro modelo.

Sumado a esto, tenemos un outlier bastante marcado: el punto aislado en el cluster azul (cluster 1). Esto nos indica que quizás haya que investigar más si este es un outlier legítimo o si son necesarios ajustes adicionales para lograr una separación más clara entre los grupos.

#### 6.3.6.2. Fichas de los Clusters: Iteración 1

Un mapa de calor para analizar los clusters nos permite visualizar las diferencias y similitudes entre los grupos en cuanto a sus estadísticas descriptivas (como la media, la desviación estándar, y los percentiles) para cada característica.

Junto a los datos reflejados, el uso de colores nos ayuda a identificar patrones de forma rápida. Por ejemplo, en este mapa, podemos ver claramente que el cluster 1 está conformado solo por un producto, mientras que el cluster 0 tiene la gran mayoría de los productos.

| cluster_IT1 | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| count | 1392.000000 | 1.000000 | 894.000000 | 762.000000 |

*Diagrama: Visualización de mapa de calor. Fichas de clusters, Iteración 1.*

Si bien es cierto que podríamos analizar los productos en base a las características de estos clusters generados por el algoritmo, quizás sería propicio intentar otra iteración e investigar si podemos conseguir una distribución de clusters menos desproporcionada -aunque siempre atentos si el mismo outlier sigue destacando en otras iteraciones con la ayuda de esta técnica.

## 6.4. Iteración 2. Enfoque en ventas totales y patrones generales

Realizar distintas iteraciones en el clustering es importante porque permite explorar el comportamiento de los datos desde diferentes perspectivas o enfoques específicos. Cada iteración puede enfocarse en un conjunto particular de características (ventas, precio, crecimiento, impacto de eventos, etc.), lo que ayuda a obtener insights más profundos ya

que cada perspectiva revela patrones únicos que podrían no ser visibles si se analizan todas las variables al mismo tiempo.

### 6.4.1. Evaluación y selección de features

En esta segunda iteración nuestro propósito será comprender las principales tendencias de ventas, las contribuciones regionales y la distribución trimestral mientras se minimiza el ruido de variables de menor impacto.

| Variables escogidas | Razón |
|---|---|
| **Ventas Totales y Regionales**<br>'total_sales',<br>'ratio_ventas_Boston',<br>'ratio_ventas_Nueva_York',<br>'ratio_ventas_Filadelfia', | **total_sales**<br>Mide el desempeño general de un producto.<br><br>**Ratios de ventas por ciudad**<br>(ratio_ventas_Boston, Nueva_York, Filadelfia)<br>Ayudan a identificar la distribución geográfica de las ventas y cómo varía entre diferentes mercados. |
| **Estacionalidad de Ventas**<br>'ventas_T1_ratio',<br>'ventas_T2_ratio',<br>'ventas_T3_ratio',<br>'ventas_T4_ratio', | **Ratios de ventas por trimestre**<br>(ventas_T1_ratio, T2, T3, T4)<br>Permiten observar patrones estacionales, destacando si un producto tiene mayor demanda en ciertos trimestres del año. |
| **Dinámicas de Precio**<br>'std_price_T1',<br>'std_price_T2',<br>'std_price_T3',<br>'std_price_T4',<br>'avg_price',<br>'price_sensitivity', | **Desviaciones estándar del precio por trimestre**<br>(std_price_T1, T2, T3, T4)<br>Revelan fluctuaciones de precio y cómo varían a lo largo del año.<br><br>**avg_price:** Mide el precio promedio del producto.<br><br>**price_sensitivity:** Identifica la elasticidad del precio, mostrando cómo responden las ventas a las variaciones en el precio |
| **Impacto de eventos estacional**<br>'event_impact_ratio_T1',<br>'event_impact_ratio_T2',<br>'event_impact_ratio_T3',<br>'event_impact_ratio_T4', | **Ratios de impacto de eventos por trimestre**<br>(event_impact_ratio_T1, T2, T3, T4)<br>Analizan cómo eventos externos (promociones, feriados, etc.) afectan las ventas en distintos periodos del año. |

Sin embargo, al evaluar nuestras features para esta segunda iteración, nos vemos en la necesidad de descartar las siguientes variables porque tienen muy baja varianza y alta correlación:

| | |
|---|---|
| **Variables con baja varianza** | 'event_impact_ratio_T1',<br>'event_impact_ratio_T2',<br>'event_impact_ratio_T3',<br>'event_impact_ratio_T4',<br>'price_sensitivity', |
| **Variables altamente correlacionadas** | 'std_price_T1',<br>'std_price_T2',<br>'std_price_T3',<br>'std_price_T4', |

A pesar que las variables de ***ventas_T1_ratio, ventas_T2_ratio, ventas_T3_ratio, y ventas_T4_ratio***, presentan muy baja variana, pero nos interesa mantenerlas en el modelo para que aporten un valor significativo en la estacionalidad.

Estas variables combinan volumen de ventas, estacionalidad, precios y eventos, lo que permite tener una visión 360° del comportamiento de los productos y cómo varían según diferentes factores clave.

### 6.4.2. Aplicación de PCA y KMeans

**Análisis de Componentes Principales (PCA)**

Con 6 componentes, explicamos el 92.40% de la varianza original:

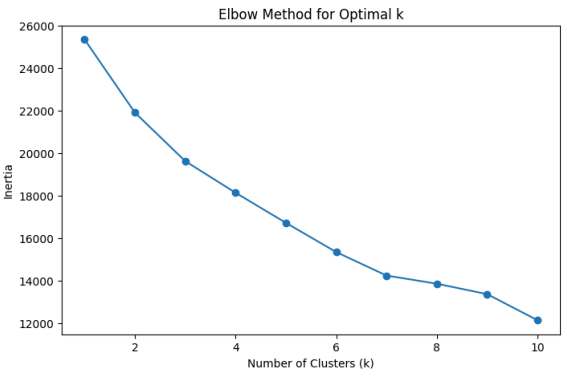
Varianza explicada por cada componente:

- 0.21559436
- 0.18988632
- 0.16832645
- 0.14195601
- 0.12389146
- 0.08430792

**Elbow Curve**

En este gráfico, observamos que la pendiente empieza a aplanarse a partir de k = 7 o k = 8.

Esto sugiere que el número óptimo de clusters estaría entre 7 y 8, ya que después de ese punto, la reducción en la inercia no mejora significativamente la calidad del agrupamiento.




Procedemos con 8 clusters generados. Procedemos a analizar los clusters para hacer una comparación con nuestra primera iteración.

### 6.4.3. Interpretación y visualización de los clusters

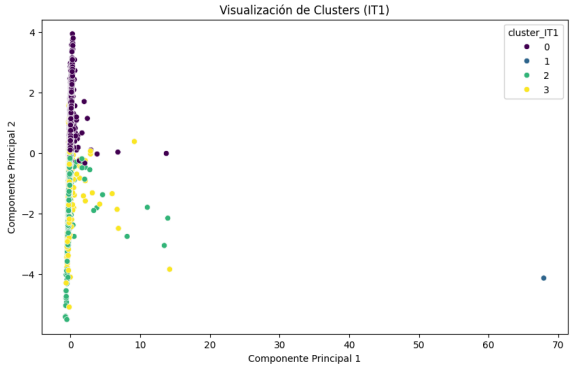



*Diagrama: Visualización de Clusters, Iteración 1.*

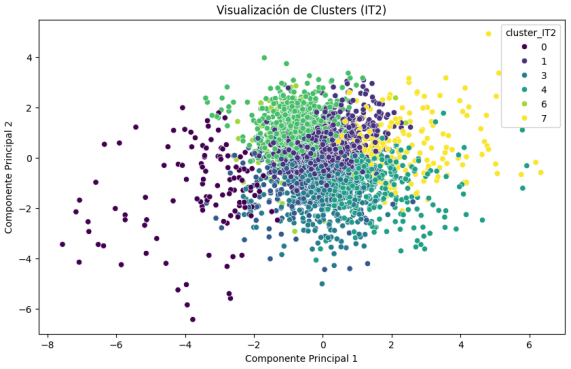



*Diagrama: Visualización de Clusters, Iteración 2.*

Como podemos observar, la Iteración 1 (IT1) muestra un caso más simple con poca variabilidad y alta superposición entre clusters, y un outlier claro en el cluster 1 que podría estar afectando el análisis.

En cambio, nuestra Iteración 2 (IT2) ofrece una mejor separación de clusters y captura más variabilidad entre los datos, lo que permite una segmentación más detallada con menos solapamiento. Aún así, algunos clusters siguen solapándose en IT2; no obstante, la distribución es más equilibrada y hay menos puntos extremos.

En conclusión, IT2 parece proporcionar una segmentación más útil y detallada que IT1, debido a la mayor separación entre los clusters y una mejor captura de la variabilidad en los datos.

**Nota**
Para la Iteración 3 y 4, por favor mirar los resultados en el Notebook:
***Cluster_Productos.ipynb***

## 6.5. Iteración 5. Enfoque en ventas desde una perspectiva de crecimiento.

En nuestra quinta iteración nuestro propósito será hacer un estudio de las ventas generales y a nivel regional, a la vez que utilizamos variables de crecimiento anual y promedio.

### 6.5.1. Evaluación y selección de features

No tomaremos en cuenta variables de impacto de eventos en esta iteración ya que nos interesa en esta iteración priorizar como se han comportado los productos en cuanto a crecimiento directamente.

| Variables escogidas | Razón |
|---|---|
| **total_sales** | Es una variable fundamental para medir el volumen de ventas de los productos. Tiene una alta varianza y es clave para capturar el desempeño general. |
| **ratio_ventas_Boston** | Aunque tiene baja correlación con la mayoría de las variables, ofrece una visión específica de una región. Mantendría esta variable como representativa. |
| **ratio_ventas_Nueva_York** | A pesar de la alta correlación negativa con el ratio de ventas de Filadelfia (-0.60), es útil para capturar patrones específicos de esta región. |
| **ratio_ventas_Filadelfia** | También aporta información útil y tiene una correlación moderada con otras variables. |
| **ventas_T4_ratio** | De los ratios estacionales, este tiene mayor varianza y puede capturar dinámicas estacionales. El resto (T1, T2, T3) podrían descartarse debido a su baja varianza y/o correlación. |
| **avg_price** | El precio promedio es una variable clave para entender el posicionamiento de los productos y su relación con el comportamiento de ventas. |
| **average_growth** | Esta variable refleja el crecimiento general de las ventas, y aunque tiene baja correlación con las demás, puede ayudar a identificar productos con diferentes tendencias de crecimiento. |
| **Variables descartadas** | **Razón** |
| **Ratios estacionales T1, T2, T3** | Baja varianza y algunas correlaciones significativas (como el ratio de T2 con T4). |

### 6.5.2. Aplicación de PCA y KMeans

**Análisis de Componentes Principales (PCA)**

Con 5 componentes, explicamos el 89.51% de la varianza original.

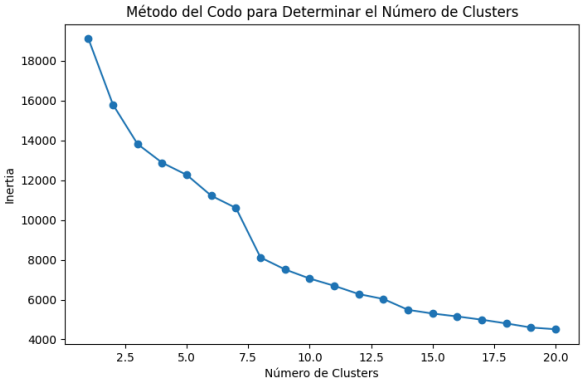
Varianza explicada por cada componente:

- 0.24923474
- 0.19826612
- 0.17215155
- 0.14292248
- 0.13256538

**Elbow Curve**

La inercia disminuye rápidamente hasta k = 7, donde ocurre un punto de inflexión. Después de este punto, la reducción de la inercia es menos pronunciada.

Esto sugiere que 7 clusters podría ser el número óptimo para segmentar los datos, ya que más clusters no aportan mejoras significativas en la compactación del modelo.




### 6.5.3. Interpretación y visualización de los clusters

En el siguiente gráfico se demuestra la distribución de 7 clusters (IT5) utilizando las dos primeras componentes principales (PCA). Cada color representa un cluster diferente.

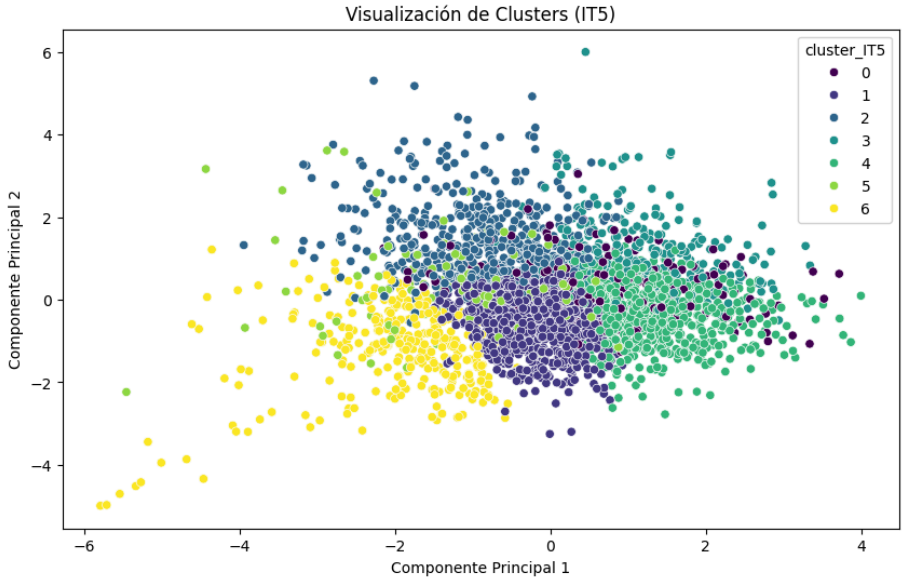



*Diagrama: Visualización de Clusters, Iteración 5.*

Aparte de poder observar que los clusters en general están bien distribuidos, con el cluster amarillo (5) claramente diferenciado en la parte izquierda y el cluster verde (6) en la derecha, tambien podemos notar que existe solo un moderado solapamiento moderado: hay cierta superposición en el centro, especialmente entre los clusters 1, 2, 3, y 4 (azules y verdes), lo que indica que los puntos en esas áreas tienen características similares.

Finalmente, la mayoría de los clusters tienen un tamaño similar, lo que sugiere que la segmentación es relativamente equitativa.

### 6.5.4. Interpretación del mapa de calor de clusters.

| | cluster_IT5 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|---|
| total_sales | count | 145.000000 | 947.000000 | 520.000000 | 340.000000 | 689.000000 | 99.000000 | 309.000000 |
| | mean | 13743.779310 | 14846.616684 | 15102.240385 | 6177.650000 | 16352.503628 | 181627.070707 | 17251.527508 |
| | std | 20259.996122 | 15562.953232 | 18299.801299 | 6139.111239 | 19278.468732 | 132443.186400 | 18850.788220 |
| | min | 932.000000 | 525.000000 | 346.000000 | 583.000000 | 463.000000 | 74521.000000 | 767.000000 |
| | 25% | 3146.000000 | 4511.000000 | 3012.500000 | 2207.000000 | 5102.000000 | 116208.000000 | 4898.000000 |
| | 50% | 6018.000000 | 9322.000000 | 7720.500000 | 3862.000000 | 9779.000000 | 139432.000000 | 11305.000000 |
| | 75% | 12917.000000 | 19588.500000 | 20582.750000 | 8474.500000 | 18936.000000 | 195864.000000 | 21960.000000 |
| | max | 115874.000000 | 96273.000000 | 105544.000000 | 51294.000000 | 153990.000000 | 942717.000000 | 129786.000000 |

**Ventas totales (total_sales)**
En términos de ventas totales, los clusters destacan por diferencias en el nivel de ventas promedio y su dispersión. Algunos clusters tienen productos que son claros "ganadores" en términos de ventas, mientras que otros son más consistentes, aunque con ventas bajas o moderadas.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| ventas_T4_ratio | count | 145.000000 | 947.000000 | 520.000000 | 340.000000 | 689.000000 | 99.000000 | 309.000000 |
| | mean | 0.114100 | 0.260071 | 0.288297 | 0.263293 | 0.263462 | 0.237236 | 0.281254 |
| | std | 0.052297 | 0.043896 | 0.073258 | 0.044608 | 0.044776 | 0.040170 | 0.060398 |
| | min | 0.000000 | 0.174506 | 0.060995 | 0.138096 | 0.152347 | 0.107395 | 0.128354 |
| | 25% | 0.071736 | 0.231965 | 0.244910 | 0.237225 | 0.235119 | 0.219289 | 0.244786 |
| | 50% | 0.120277 | 0.253286 | 0.270943 | 0.259720 | 0.256212 | 0.244062 | 0.272882 |
| | 75% | 0.157502 | 0.280732 | 0.308785 | 0.285944 | 0.285603 | 0.257397 | 0.302894 |
| | max | 0.203563 | 0.536470 | 0.658950 | 0.512158 | 0.585903 | 0.318073 | 0.576615 |

**Ratio de ventas T4 (ventas_T4_ratio)**

- **Clusters con fuerte comportamiento estacional (Clusters 1, 2, 3, 4, 6)**: Estos grupos destacan por tener un porcentaje significativo de sus ventas en el Trimestre 4, lo que indica que los productos en estos clusters probablemente dependen de temporadas o eventos que impulsan las ventas al final del año.
- **Clusters con ventas menos estacionales (Cluster 0)**: Los productos en este cluster no muestran una gran concentración de ventas en el T4, lo que sugiere que las ventas están más distribuidas a lo largo del año, sin una dependencia fuerte de la estacionalidad.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| ratio_ventas_Boston | count | 145.000000 | 947.000000 | 520.000000 | 340.000000 | 689.000000 | 99.000000 | 309.000000 |
| | mean | 0.279713 | 0.239844 | 0.411883 | 0.296896 | 0.213602 | 0.317471 | 0.229707 |
| | std | 0.076612 | 0.054669 | 0.074727 | 0.075884 | 0.055377 | 0.082381 | 0.081972 |
| | min | 0.112231 | 0.026423 | 0.288031 | 0.119174 | 0.024328 | 0.116227 | 0.010123 |
| | 25% | 0.221822 | 0.203363 | 0.354071 | 0.243895 | 0.174839 | 0.267362 | 0.165516 |
| | 50% | 0.277369 | 0.245258 | 0.395737 | 0.301742 | 0.215484 | 0.311241 | 0.235429 |
| | 75% | 0.336800 | 0.283112 | 0.452286 | 0.340023 | 0.252342 | 0.356186 | 0.288569 |
| | max | 0.515445 | 0.350199 | 0.743156 | 0.642363 | 0.353806 | 0.610750 | 0.472195 |

**Ratio de Ventas en Boston (ratio_ventas_Boston)**

- Cluster 2 tiene el mayor promedio de ventas en Boston con un 41.1%, seguido por el Cluster 0 (27.9%).
- Cluster 4 tiene el menor porcentaje (21.4%).
- Conclusión: Los productos en el Cluster 2 tienden a depender más de las ventas en Boston.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| ratio_ventas_Nueva_York | count | 145.000000 | 947.000000 | 520.000000 | 340.000000 | 689.000000 | 99.000000 | 309.000000 |
| | mean | 0.485166 | 0.452827 | 0.350854 | 0.502139 | 0.583958 | 0.404911 | 0.303447 |
| | std | 0.111006 | 0.046607 | 0.075899 | 0.071047 | 0.056013 | 0.088352 | 0.085884 |
| | min | 0.220201 | 0.326653 | 0.091363 | 0.296143 | 0.486007 | 0.134909 | 0.000266 |
| | 25% | 0.416149 | 0.419409 | 0.306443 | 0.456534 | 0.542614 | 0.355050 | 0.260830 |
| | 50% | 0.493302 | 0.456009 | 0.355474 | 0.507356 | 0.574743 | 0.401544 | 0.316989 |
| | 75% | 0.549587 | 0.487790 | 0.403540 | 0.549985 | 0.614133 | 0.461063 | 0.360725 |
| | max | 0.745913 | 0.569009 | 0.518880 | 0.675615 | 0.828623 | 0.633313 | 0.456230 |

**Ratio de Ventas en Nueva York (ratio_ventas_Nueva_York).** El Cluster 4 está muy concentrado en Nueva York, mientras que el Cluster 2 depende menos de este mercado.

- **Cluster 4** tiene el mayor promedio de ventas en Nueva York con 58.4%.
- Mientras que el **Cluster 2** tiene el menor promedio (35%).

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| ratio_ventas_Filadelfia | count | 145.000000 | 947.000000 | 520.000000 | 340.000000 | 689.000000 | 99.000000 | 309.000000 |
| | mean | 0.235121 | 0.307328 | 0.237263 | 0.200964 | 0.202440 | 0.277618 | 0.466846 |
| | std | 0.074703 | 0.046181 | 0.066490 | 0.064602 | 0.047458 | 0.091425 | 0.098982 |
| | min | 0.073453 | 0.185818 | 0.045516 | 0.061493 | 0.044662 | 0.123894 | 0.342751 |
| | 25% | 0.176323 | 0.273354 | 0.194312 | 0.157440 | 0.172587 | 0.225990 | 0.407967 |
| | 50% | 0.223125 | 0.304628 | 0.237083 | 0.191240 | 0.205288 | 0.272583 | 0.442613 |
| | 75% | 0.288569 | 0.341294 | 0.280563 | 0.232286 | 0.231779 | 0.320800 | 0.493745 |
| | max | 0.425075 | 0.454365 | 0.426749 | 0.435173 | 0.344100 | 0.681457 | 0.989611 |

**Ratio de Ventas en Filadelfia (ratio_ventas_Filadelfia).** El Cluster 6 tiene una fuerte dependencia de las ventas en Filadelfia, en contraste con otros clusters que muestran una menor dependencia.

- **Cluster 6** muestra el mayor promedio de ventas en Filadelfia con 46.6%.
- Mientras que el **Cluster 1** tiene un promedio más moderado (30.7%).

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| avg_price | count | 145.000000 | 947.000000 | 520.000000 | 340.000000 | 689.000000 | 99.000000 | 309.000000 |
| | mean | 5.866043 | 4.051156 | 5.231987 | 13.727842 | 4.987378 | 1.709902 | 4.116271 |
| | std | 3.159027 | 2.216856 | 3.403695 | 6.461891 | 2.885216 | 1.039597 | 3.260937 |
| | min | 1.175507 | 0.310000 | 0.310000 | 3.949511 | 0.398114 | 0.237915 | 0.237956 |
| | 25% | 3.093912 | 2.459778 | 2.752386 | 8.870325 | 2.998967 | 1.178640 | 2.337184 |
| | 50% | 6.229522 | 3.557147 | 4.645261 | 12.332143 | 4.380868 | 1.227173 | 3.014592 |
| | 75% | 7.418860 | 4.999294 | 6.837613 | 16.293506 | 6.482335 | 1.929978 | 4.721754 |
| | max | 14.860743 | 16.131085 | 28.554110 | 39.500206 | 15.262879 | 5.979346 | 20.658500 |

**Precio Promedio (avg_price).**
El **Cluster 3** contiene productos de mayor precio, mientras que los productos en los **Clusters 1, 4, y 5** parecen ser de bajo valor.

- **Cluster 3** tiene el precio promedio más alto con 13.72 unidades monetarias, mientras que el **Cluster 5** tiene el precio más bajo con 1.71.
- **Cluster 0** también tiene precios bajos con una media de 5.87, lo que podría sugerir productos de menor valor en comparación con Cluster 3.
- **Cluster 2** muestra un precio promedio moderado de 5.23, mientras que el Cluster 1 y **Cluster 4** presentan precios promedio bajos de 4.05 y 4.98, respectivamente.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| average_growth | count | 145.000000 | 947.000000 | 520.000000 | 340.000000 | 689.000000 | 99.000000 | 309.000000 |
| | mean | 125.180791 | 338.291695 | 399.694751 | 240.307082 | 591.993258 | 91.648816 | 276.223940 |
| | std | 779.557742 | 2917.663757 | 3028.086124 | 1166.482474 | 9552.980875 | 722.628574 | 2071.540458 |
| | min | -69.689495 | -50.017752 | -57.060627 | -26.910782 | -32.972000 | -50.681600 | -45.749957 |
| | 25% | -3.751316 | -0.822836 | 1.000588 | 0.936332 | -1.876732 | -2.325894 | 5.495120 |
| | 50% | 5.998675 | 11.773256 | 14.430148 | 12.717472 | 8.046751 | 7.405975 | 19.227009 |
| | 75% | 18.101816 | 38.748114 | 46.426530 | 35.442403 | 25.908613 | 21.828979 | 55.211639 |
| | max | 8220.437761 | 50610.265915 | 53000.000000 | 11312.500000 | 242476.046947 | 7189.762182 | 30625.000000 |

**Crecimiento Promedio (average_growth).**
Cluster 2 y Cluster 1 contienen productos con alto crecimiento, mientras que el Cluster 5 tiene una tendencia negativa, con productos que enfrentan disminución en ventas.

- **Cluster 2** muestra el mayor crecimiento promedio con 399.69%, seguido de **Cluster 1** con 338.29%. Estos dos clusters reflejan productos que han experimentado un crecimiento notable en ventas.
- **Cluster 0** y **Cluster 6** tienen crecimientos más moderados con 125.18% y 276.22%, respectivamente.
- **Cluster 5** muestra un decrecimiento significativo, con una media de -50.68%, lo que indica productos con ventas decrecientes.

### 6.5.5. Comparación entre clusters

| Cluster (CIX) | CI0 | CI1 | CI2 | CI3 | CI4 | CI5 | CI6 |
|---|---|---|---|---|---|---|---|
| **Cantidad** | 145 | **947** | 520 | 340 | 689 | 99 | 309 |
| **%** | 4.75% | **31.06%** | 17.05% | 11.15% | 22.60% | 3.25% | 10.13% |
| **Ventas Totales (avg)** | 13,743 | 14,846 | 15,102 | **6,177 (el más bajo)** | 16,352 | **181,627 (el más alto)** | 17,251 |

**Ratio de Ventas**

| Cluster (CIX) | CI0 | CI1 | CI2 | CI3 | CI4 | CI5 | CI6 |
|---|---|---|---|---|---|---|---|
| **Ratio Boston** | Media | Media | **Domina** | Media | Media | Media | Media |
| **Ratio NY** | Media | Media | Media | **Domina** | **El más alto** | **Alta** | **Alta** |
| **Ratio Filadelfia** | Media | Media | Media | Media | Media | **Alta** | **Alta** |
| **Ventas T4** | **Bajo** | Media | Media | Media | Media | Media | Media |
| **Precio Promedio** | 5.87 | 4.05 | 5.23 | **13.73 (el más alto)** | 4.99 | **1.71 (el más bajo)** | 4.12 |
| **Crecimiento Promedio** | 125 | 338 | 399 | 240 | **591 (el más alto)** | **7189 alta variabilidad** | **276 alta estabilidad** |

### 6.5.6. Observaciones Generales por Clusters

| | |
|---|---|
| Cluster 0 | **Productos de Crecimiento Moderado y Precio Medio**<br>• Tamaño: 145 productos, representando el 4.75% del total.<br>• Ventas Totales: Media baja de 13,743 unidades vendidas por producto.<br>• Presencia Regional: Mayor ratio de ventas en Nueva York (48%) y Filadelfia (23%).<br>• Ratio de Ventas T4: Relativamente bajo, indicando que no dependen mucho de ventas en el cuarto trimestre.<br>• Crecimiento Promedio: Leve crecimiento positivo (125), pero lejos de los top performers. |
| Cluster 1 | **Productos de Bajo Precio y Alto Crecimiento**<br>• Tamaño: 947 productos, 31.06% del total.<br>• Ventas Totales: Media de 14,846 unidades por producto, ventas bastante consistentes.<br>• Presencia Regional: Distribución balanceada en Nueva York (45%) y Boston (24%), con buen rendimiento en Filadelfia (30%).<br>• Ratio de Ventas T4: Buena performance en el cuarto trimestre (26% de las ventas).<br>• Crecimiento Promedio: Crecimiento positivo sólido (338), indicando productos bien establecidos en el mercado. |
| Cluster 2 | **Productos de Crecimiento Elevado y Precio Moderado**<br>• Tamaño: 520 productos, representando el 17.05%.<br>• Ventas Totales: Media de 15,102 unidades vendidas. |

| | |
|---|---|
| | ● Presencia Regional: Mayor ratio de ventas en Boston (41%) y Filadelfia (24%).<br>● Ratio de Ventas T4: Buen rendimiento en el cuarto trimestre (28%).<br>● Crecimiento Promedio: Buen crecimiento (399), indicativo de productos que están ganando tracción. |
| Cluster 3 | **Productos Premium de Precio Alto**<br>● Tamaño: 340 productos, el 11.15% del total.<br>● Ventas Totales: Media baja de 6,177 unidades por producto.<br>● Presencia Regional: Fuerte presencia en Nueva York (50%), menor en Boston y Filadelfia.<br>● Ratio de Ventas T4: Bajo rendimiento en el cuarto trimestre (26%).<br>● Crecimiento Promedio: Crecimiento bajo (240), lo que refleja productos con menor expansión en el mercado. |
| Cluster 4 | **Productos de Bajo Crecimiento y Precio Medio**<br>● Tamaño: 689 productos, 22.60% del total.<br>● Ventas Totales: Media de 16,352 unidades vendidas, con alta estabilidad.<br>● Presencia Regional: Muy fuerte en Nueva York (58%) y Filadelfia (20%).<br>● Ratio de Ventas T4: Buen rendimiento en el cuarto trimestre (26%).<br>● Crecimiento Promedio: Crecimiento notable (591), indicativo de productos líderes en su segmento. |
| Cluster 5 | **Productos en Declive y Bajo Precio**<br>● Tamaño: 99 productos, representando el 3.25%.<br>● Ventas Totales: Media muy alta de 181,627 unidades por producto.<br>● Presencia Regional: Mayor ratio de ventas en Nueva York (40%) y Filadelfia (27%).<br>● Precio Promedio: Precio bajo comparado con el volumen de ventas (1.70).<br>● Crecimiento Promedio: Crecimiento muy bajo (-50), posiblemente por saturación del mercado. |
| Cluster 6 | **Productos de Precio Moderado y Crecimiento Positivo**<br>● Tamaño: 309 productos, el 10.13% del total.<br>● Ventas Totales: Media de 17,251 unidades vendidas.<br>● Presencia Regional: Buen rendimiento en Nueva York (30%) y Filadelfia (23%).<br>● Ratio de Ventas T4: Buen rendimiento en el cuarto trimestre (28%).<br>● Crecimiento Promedio: Crecimiento acelerado (276), lo que indica una expansión prometedora en el mercado. |

## 6.6. Conclusiones: Clustering de Productos.

En este análisis de *clustering* se realizaron varias iteraciones con el objetivo de identificar patrones generales en ventas y comportamientos de productos a través de diferentes variables, tales como ventas totales, precios promedios, ratios de ventas por ciudad (Boston, Nueva York, Filadelfia), y crecimiento promedio. A lo largo de estas iteraciones, cada una se enfocó en diferentes combinaciones de variables para optimizar la segmentación de los productos.

La quinta iteración fue la que ofreció los mejores resultados, logrando una segmentación más clara y diferenciada. En esta iteración final, se incluyeron variables clave como los ratios de crecimiento de ventas y las ventas en distintas ciudades y trimestres, y el precio

medio, lo que permitió agrupar los productos de manera más representativa y ofrecer insights más profundos sobre los comportamientos de ventas a nivel regional y temporal.

En resumen, los productos fueron agrupados en siete clusters, cada uno con características distintivas en cuanto a ventas, crecimiento y comportamiento regional. Este análisis ha proporcionado una comprensión más clara de cómo se comportan los productos en diferentes mercados, siendo la última iteración la más precisa para identificar patrones específicos de ventas, permitiendo a la empresa tomar decisiones más informadas sobre estrategias de marketing y distribución en diferentes regiones.

## 6.7. Clustering de tiendas

La metodología empleada para el Clustering de tiendas fue muy similar al de Clustering de Productos. Sin embargo, la perspectiva del Feature engineering fue realizada desde el punto de vista de las tiendas en lugar de productos.
También requerimos del previamente detallado filtrado de datos para incluir solo registros entre los años 2011 y 2015.

A continuación, explicamos brevemente los pasos desarrollados.

### 6.7.1. Feature engineering

**Variables relacionadas a tiempo**

| | |
|---|---|
| **date** | La fecha para obtenida de yearweek. |
| **month** | El mes derivado de la fecha. |
| **quarter** | El trimestre del año. |
| **is_event** | Variable binaria para marcar si hubo un evento o no, en función de la columna event. |

```
[ ]  df['date'] = pd.to_datetime(df['yearweek'].astype(str) + '0', format='%Y%W%w')
     df['month'] = df['date'].dt.month

[ ]  df['quarter'] = df['date'].dt.to_period('Q')

[ ]  df['is_event'] = np.where(df['event'] == 'Sin_Evento', 0, 1)
```

**Variables relacionadas a ventas**

| | |
|---|---|
| **Cálculos de Proporción de Ventas** | Se calcula la proporción de ventas de cada tienda con respecto a las ventas totales. Esto permite identificar el peso o participación de cada tienda en las ventas generales del conjunto de datos. |

```
df_store_sales = df_filtered.groupby('store_code')['ventas'].sum().reset_index()
df_store_sales.rename(columns={'ventas': 'total_ventas_store'}, inplace=True)
df_store_sales['total_ventas'] = df_filtered['ventas'].sum()
df_store_sales['store_sales_ratio'] = df_store_sales['total_ventas_store'] / df_store_sales['total_ventas']
df_store_sales
```

| | |
|---|---|
| **Ventas Promedio por Categoría** | Se agrupan las ventas por tienda y por categoría, y luego se calcula el promedio de ventas por categoría en cada tienda. Este análisis puede ayudar a comparar el rendimiento de las categorías de productos entre tiendas. Las columnas que se generan incluyen nombres como avg_sales_cat_X para cada categoría. |

```
df_avg_sales_category = df_filtered.groupby(['store_code', 'category'])['ventas'].mean().reset_index()
df_avg_sales_category.columns = ['store_code', 'category', 'avg_sales_per_category']
df_avg_sales_category = df_avg_sales_category.pivot(index='store_code', columns='category', values='avg_sales_per_category')
df_avg_sales_category.columns = ['avg_sales_cat_' + col for col in df_avg_sales_category.columns]
df_avg_sales_category.reset_index(inplace=True)
df_avg_sales_category
```

| | |
|---|---|
| **Ratio de Ventas por Semana o Trimestre** | Se calcula el ratio de ventas por semana o trimestre para medir la estabilidad o estacionalidad de las ventas de cada tienda. Esto ayuda a capturar variaciones temporales en las ventas. |

```
df_sales_quarter = df_filtered.groupby(['store_code', 'trimestre'])['ventas'].sum().reset_index()
df_sales_quarter = df_sales_quarter.pivot(index='store_code', columns='trimestre', values='ventas')
df_sales_quarter.columns = ['ventas_T1', 'ventas_T2', 'ventas_T3', 'ventas_T4']
df_sales_quarter.reset_index(inplace=True)
df_sales_quarter
```

| | |
|---|---|
| **Ratio de Eventos a Ventas** | Este ratio puede indicar cómo los eventos (promociones, descuentos, etc.) impactan en las ventas totales de la tienda. |
| **Ratio de Ventas en Días de Evento** | Mide el impacto de los días de eventos en las ventas totales. Puede mostrar si las ventas de una tienda dependen en gran medida de eventos promocionales |

```
df_store_event_sales = df_filtered.groupby(['store_code', 'is_event'])['ventas'].sum().reset_index()

df_store_event_sales = df_store_event_sales.pivot(index='store_code', columns='is_event', values='ventas')
df_store_event_sales.columns = ['event_sales', 'no_event_sales']

df_store_event_sales['total_store_sales'] = df_filtered.groupby('store_code')['ventas'].sum()
df_store_event_sales['count_events'] = df_filtered.groupby('store_code')['is_event'].sum()

df_store_event_sales['ratio_eventos_ventas'] = df_store_event_sales['count_events'] / df_store_event_sales['total_store_sales']

df_store_event_sales['ratio_ventas_en_eventos'] = df_store_event_sales['event_sales'] / df_store_event_sales['total_store_sales']

df_store_event_sales.drop(columns=['event_sales','no_event_sales','total_store_sales','count_events'], inplace=True)

df_store_event_sales.reset_index(inplace=True)

df_store_event_sales
```

**Variables relacionadas a crecimiento anual**

| | |
|---|---|
| **Ratio de Crecimiento de Ventas Anual** | La calculamos utilizando la siguiente formula: Fórmula para sacar Ratio de Crecimiento de Ventas Anual = (Ventas Año Actual - Ventas Año Anterior) / Ventas Año Anterior |

```
# Fórmula para sacar Ratio de Crecimiento de Ventas Anual = (Ventas Año Actual - Ventas Año Anterior) / Ventas Año Anterior

df_annual_sales = df_filtered.groupby(['store_code', df_filtered['date'].dt.year])['ventas'].sum().reset_index()

df_annual_sales.rename(columns={'ventas': 'annual_sales', 'date': 'year'}, inplace=True)

df_annual_sales_pivot = df_annual_sales.pivot(index='store_code', columns='year', values='annual_sales')

df_annual_sales_pivot['growth_rate'] = (df_annual_sales_pivot[df_annual_sales_pivot.columns[-1]] - df_annual_sales_pivot[df_annual_sales_pivot.columns[-2]]) / df_annua

df_annual_sales_pivot.reset_index(inplace=True)

df_annual_sales_pivot.columns = ['store_code', 'ventas_2011', 'ventas_2012', 'ventas_2013', 'ventas_2014', 'ventas_2015', 'growth_rate']
df_annual_sales_pivot
```

**Variables relacionadas a precio**

| | |
|---|---|
| **Precio de Venta Promedio Ajustado por Categoría** | Normaliza el precio de venta promedio considerando el número de categorías o departamentos, para comparar mejor tiendas con diferentes mix de productos. |

```
df_avg_price_category = df_filtered.groupby(['store_code', 'category'])['sell_price'].mean().reset_index()
df_avg_price_category.columns = ['store_code', 'category', 'avg_price_per_category']

df_category_count = df_filtered.groupby('store_code')['category'].nunique().reset_index()
df_category_count.rename(columns={'category': 'category_count'}, inplace=True)

df_avg_price_category = pd.merge(df_avg_price_category, df_category_count, on='store_code')

df_avg_price_category['adjusted_avg_price'] = df_avg_price_category['avg_price_per_category'] / df_avg_price_category['category_count']

df_avg_price_category_pivot = df_avg_price_category.pivot(index='store_code', columns='category', values='adjusted_avg_price')
df_avg_price_category_pivot.columns = ['adj_avg_price_cat_' + col for col in df_avg_price_category_pivot.columns]
df_avg_price_category_pivot.reset_index(inplace=True)

df_avg_price_category_pivot
```

### 6.7.2. Evaluación de Features: Feature Variance y Feature Correlation

Las variables con baja varianza aportan muy poca información diferenciadora al modelo de clustering, ya que no permiten distinguir entre tiendas.

```
Feature Variances:
 store_sales_ratio              8.892531e-04
avg_sales_cat_ACCESORIES        1.296326e+00
avg_sales_cat_HOME_&_GARDEN     2.788694e+00
avg_sales_cat_SUPERMARKET       1.038792e+01
ventas_T1                       1.694675e+11
ventas_T2                       2.160827e+11
ventas_T3                       2.569362e+11
ventas_T4                       1.837887e+11
ratio_ventas_en_eventos         2.035208e-06
ventas_2011                     9.216219e+10
ventas_2012                     1.727057e+11
ventas_2013                     1.537257e+11
ventas_2014                     1.718276e+11
ventas_2015                     1.399478e+11
growth_rate                     3.506920e-03
dtype: float64
```

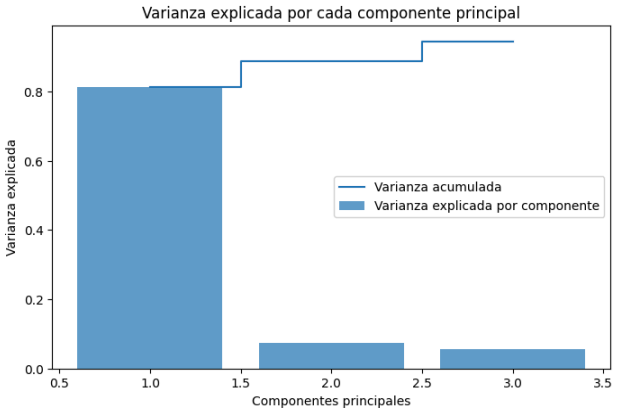
- **store_sales_ratio** tiene una varianza muy baja.
- **ratio_ventas_en_eventos** también tiene una varianza extremadamente baja

Ambas variables podrían descartarse, ya que no contribuirán significativamente a la diferenciación de grupos.

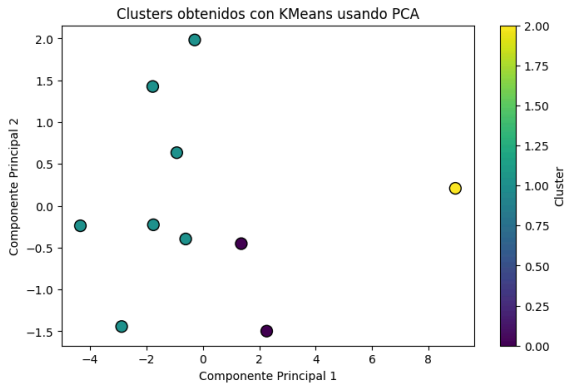
### 6.7.3. PCA, KMeans y Elbow Curve

Después de estandarizar los datos, procedemos a aplicar el PCA y conseguimos una varianza del 94.41%. Hemos disminuido **3 componentes principales:**

- [0.8131869
- 0.07468813
- 0.05620918].




```
[ ] from sklearn.decomposition import PCA

    n_components=3

    pca = PCA(n_components)
    A = pca.fit_transform(scaled_features)

    explained_variance = pca.explained_variance_ratio_
    print("Varianza explicada por cada componente:", explained_variance)

→ Varianza explicada por cada componente: [0.8131869  0.07468813 0.05620918]

[ ] cumulative_variance = explained_variance.cumsum()
    exp_var_ratio_sum = sum(explained_variance)

    print(f"Con {n_components} componentes, explicamos el {exp_var_ratio_sum*100:.2f}% de la varianza original.")

→ Con 3 componentes, explicamos el 94.41% de la varianza original.
```

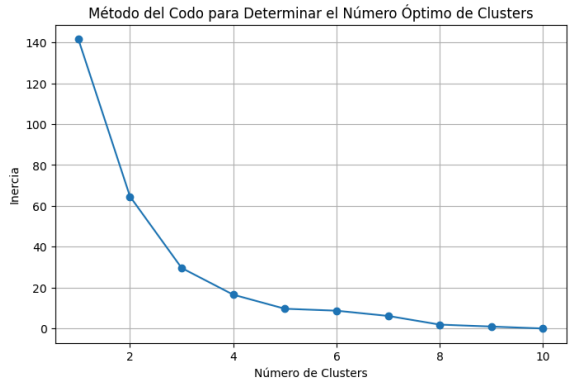



**KMeans**

En esta gráfica del PCA, cada punto en la gráfica representa una tienda en las nuevas dimensiones de PCA.

Por lo que podemos observar, al menos 3 clusters están identificados con los colores.

*Diagrama: Clusters obtenidos con KMeans usando PCA (Tiendas)*




**Elbow Curve**

En esta gráfica, el codo parece estar alrededor de **3 o 4 clusters**, lo que sugiere que es el número óptimo de clusters para el análisis.

Procederemos con 3 clusters.

*Diagrama: Método del codo para Determinar el Número Optimo de Clusters (Tiendas)*

## 6.7.4. Asignación de labels y Análisis de las Fichas

| Segmentos Tiendas | Tiendas Supermercado Estable | Tiendas Diversificadas con Baja Estacionalidad | Tienda de Alto Volumen y Multicategoría |
|---|---|---|---|
| **Código tienda** | *BOS_2, NYC_1* | *BOS_1, BOS_3, NYC_2, NYC_4 PHI_1, PHI_2, PHI_3* | *NYC_3* |
| **Vol** | **2 tiendas (20.00%)** | **7 tiendas (70.00%)** | **1 tienda (10.00%)** |
| **Ratio Ventas** | 11.44%, con poca variabilidad. | Promedio de 8.56%. | **17.18% (el ratio más alto).** |
| **Ventas Promedio Categorías** | Accesorios: 4.56<br>Home & Garden: 4.83<br>**Supermercado: 12.42.** | Accesorios: 3.12<br>Home & Garden: 3.41<br>**Supermercado: 9.12** | **Accesorios: 5.78**<br>**Home & Garden: 8.78**<br>**Supermercado: 17.96** |
| **Análisis Estacional** | **Ventas distribuidas de manera homogénea T1-4.** | **Cierta variabilidad estacional.** Ventas ligeramente más altas en el T3. | **Ventas altas y constantes** T1-T4. Sugiere **operación de alto volumen sin estacionalidad marcada.** |
| **Ventas Anuales** | **Crecimiento consistente en ventas** | **Distribución en ventas relativamente uniforme**, pero con menor rendimiento. | **Ventas muy altas y estables** en un rango alto hasta 2015. |
| **Crecimiento Promedio** | 1.86%, lo que sugiere un **crecimiento modesto pero estable.** | Promedio de crecimiento anual de 8.43%, **el crecimiento más alto entre los clusters.** | A pesar de altas ventas, muestra un **ligero crecimiento negativo.** |

## 6.8. Conclusiones: Clustering de Tiendas.

**Cluster 0: "Tiendas Supermercado Estable"**

- Este cluster agrupa tiendas que muestran un desempeño estable a lo largo del año, con un enfoque importante en ventas de supermercado y buena respuesta a eventos de ventas.
- Este cluster continúa mostrando estabilidad en términos de crecimiento y ventas, sin fluctuaciones significativas en los ingresos anuales. Las tiendas en este grupo siguen un patrón de ventas constante y predecible.
- Este cluster representa tiendas con ventas estables y una participación alta en la categoría de supermercados. Se recomienda mantener una estrategia de promociones regulares y eventos para mantener su desempeño estable.
- Podría beneficiarse de implementar estrategias de fidelización de clientes y mejoras operativas para seguir incrementando lentamente el volumen de ventas, manteniendo el enfoque en la estabilidad.

**Cluster 1: "Tiendas Diversificadas con Baja Estacionalidad"**

- Estas tiendas tienen una distribución de ventas relativamente uniforme a lo largo de todas las categorías, pero con un rendimiento ligeramente menor en comparación con los otros clusters. Parecen ser tiendas más generalistas con poca estacionalidad.
- Las tiendas en este cluster experimentan un crecimiento notable, aunque algunas lo hacen a un ritmo más acelerado que otras. Esto puede deberse a la diversidad en su enfoque de productos y mercados.
- Este cluster representa la mayoría de las tiendas, con ventas distribuidas uniformemente a lo largo del año y en diferentes categorías. Podrían beneficiarse de campañas que se enfoquen en aumentar ventas en categorías específicas (como accesorios) o aprovechar épocas del año donde las ventas son más bajas.
- Podría beneficiarse de procesos que fomenten mejores prácticas entre las tiendas de mayor y menor rendimiento para reducir la variabilidad.

**Cluster 2: "Tienda de Alto Volumen y Multicategoría"**

- Este cluster representa una tienda con un desempeño excepcionalmente alto en todas las categorías de ventas y una fuerte influencia de eventos, lo que la distingue como una operación de gran volumen y muy diversificada.

- A pesar de sus volúmenes de ventas significativamente altos, esta tienda muestra señales de estancamiento o ligera disminución en el crecimiento. Esto podría indicar saturación de mercado o una falta de innovación en su enfoque de ventas.
- La tienda en este cluster tiene un rendimiento excepcional en todas las categorías y muestra una alta sensibilidad a eventos de ventas. Debería ser un modelo a seguir para las otras tiendas; replicar sus estrategias de eventos y promociones podría mejorar el rendimiento de los otros clusters.0
- Podría beneficiarse de implementar estrategias de renovación de productos y marketing para revitalizar el crecimiento. Esto puede incluir campañas específicas para introducir nuevos productos o servicios, o explorar nuevas categorías de ventas.

# 7. Task 3: PREVISIÓN DE VENTAS

A continuación, detallamos el enfoque desarrollado para la previsión de ventas en DSMarket, que es crucial para el caso de uso de reabastecimiento y la planificación estratégica. Se ha diseñado un modelo predictivo que proyecta las ventas a nivel tienda-producto durante un período de 28 días (equivalente a 4 semanas). Este enfoque parte de las predicciones a nivel individual de tienda y producto, y posteriormente agrega estas predicciones para obtener ventas agregadas por tienda, departamento y ciudad. A continuación, describimos los pasos clave seguidos en el análisis y la construcción del modelo de previsión.

## 6.9. Metodología

| | | |
|---|---|---|
| 1 | **Preparación de los Datos** | **Conversión de fechas:** Asegurar que la columna de fechas esté en formato adecuado (datetime). |
| | | **Filtrado de datos:** Selección de las columnas necesarias, eliminación de duplicados y tratamiento de valores nulos. |
| | | **Tipificación de variables:** Convertir variables categóricas en formato adecuado. |
| 2 | **Feature Engineering** | **Variables de tiempo:** Extracción de características como el día de la semana, mes, trimestre, festivos, o eventos especiales. |
| | | **Codificación de variables categóricas:** Hardcodear o codificar variables categóricas como región, tienda y categoría de producto mediante técnicas como Label Encoding o One-Hot Encoding. |
| | | **Lagged y Variability features:** Crear características de "retraso" (lags) y variaciones, para capturar patrones temporales pasados (e.g., ventas de la semana anterior). |
| 3 | **División de Datos** | **Entrenamiento/Validación:** Dividir los datos en conjunto de entrenamiento y validación. Utilizar parte de los datos históricos para entrenar y reservar un conjunto para evaluar la precisión del modelo. |
| | | **Test:** Preparar un dataset con las fechas futuras a predecir, manteniendo el mismo formato y estructura que el dataset de entrenamiento |
| 4 | **Selección del Modelo de Forecasting** | **Modelos comunes:**<br>● **Prophet:** Desarrollado por Facebook, especialmente útil para datos con estacionalidad.<br>● **XGBoost/LightGBM:** Modelos de machine learning para predicción multivariable. |

| | | |
|---|---|---|
| | | **Selección de hiperparámetros:** Ajuste de parámetros como la estacionalidad, número de lags o el tamaño de ventana temporal utilizando GridSearch para probar diferentes combinaciones de parámetros para obtener los más óptimos. |
| 5 | **Entrenamiento del Modelo** | **Entrenamiento del algoritmo:** Ajustar el modelo usando el conjunto de entrenamiento y aplicando los hiperparámetros más óptimos para obtener la mayor fiabilidad del modelo. |
| 6 | **Evaluación del Modelo** | **Métricas de evaluación:** Calcular métricas de error como el RMSE (Raíz del Error Cuadrático Medio), muy sensible a los errores de predicción respecto al valor real. |
| 7 | **Predicción** | **Predicción sobre el conjunto de validación:** Comparar las predicciones con los valores reales para ajustar el modelo. |
| | | **Predicción de fechas futuras:** Aplicar el modelo sobre el dataset preparado con las fechas futuras para generar las predicciones. |

## 6.10. Desarrollo

### 6.10.1. Exploración Inicial de Datos y Preprocesamiento

Comenzamos realizando un análisis exploratorio de los datos (EDA), donde identificamos que DSMarket cuenta con **3049 productos únicos** distribuidos en distintas tiendas y regiones (New York, Boston y Philadelphia).

```
df['item'].nunique() # Tenemos 3049 items unicos con los que hacer timeseries
```

```
3049
```

Los datos que utilizamos para las predicciones incluyen las ventas históricas, los precios, eventos especiales (como el Super Bowl o Thanksgiving), y la categorización de productos en departamentos como Accesorios, Hogar y Jardín, y Supermercado.

Una de las primeras tareas fue asegurarnos de que la columna **yearweek**, que contenía información de la semana del año, fuera convertida a un formato de fecha (**datetime**) la cual nos servirá más adelante para realizar las predicciones del mes de mayo de 2016 (4 semanas).

```
# Convertiremos la columna de "yearweek" a formato datetime para realizar tareas de forecasting
# para ello necesitamos que la columna este en formato string primero y despues la cambiaremos a Datetime
df['yearweek'] = df['yearweek'].astype(str)

# Convertir 'yearweek' a datetime usando el primer día de la semana
df['date'] = pd.to_datetime(df['yearweek'] + '1', format='%Y%W%w')  # El 1 agrega el primer día de la semana (lunes)
```

Esta transformación fue necesaria para aplicar correctamente los modelos de series temporales y analizar los patrones de ventas a lo largo del tiempo.

Además, generamos variables adicionales como la facturación (`revenue`), multiplicando las ventas por el precio de cada producto, lo que nos proporcionó una visión más precisa del comportamiento económico de cada tienda.

```
df2['revenue']=df2['ventas'] * df2['sell_price'] #creamos la variable revenue (facturación)
```

Por último, y después de realizar los cambios oportunos en el dataset, comprobamos cómo había variado el rango de fechas con las cuales tendríamos que trabajar para el modelo.

```
MIN_DATE = df["date"].min()
MAX_DATE = df["date"].max()
print(f"Min date is {MIN_DATE}\nMax date is {MAX_DATE}")
```

```
Min date is 2011-01-24 00:00:00
Max date is 2016-04-18 00:00:00
```

Las fechas con las que contábamos por el momento eran todas las fechas informadas de nuestro dataset, y con el fin de crear predicciones para el mes siguiente, que no estaba informado, creamos las filas necesarias para cada producto con las nuevas fechas a predecir.

```
last_date = df3["date"].max()

# Generamos nuevas fechas
new_dates = pd.date_range(start=last_date + pd.Timedelta(days=7), periods=4, freq="W-MON")
```

```
new_dates
```

```
DatetimeIndex(['2016-04-25', '2016-05-02', '2016-05-09', '2016-05-16'], dtype='datetime64[ns]', freq='W-MON')
```

Nos guardamos en una variable independiente la fecha máxima de la cual teníamos información en nuestro dataset, para luego generar otra variable donde guardaremos las 4 semanas de predicción que se nos pidió.

```
last_records = df3.groupby('id').last().reset_index()

new_rows = last_records.loc[last_records.index.repeat(len(new_dates))]
new_rows['date'] = list(new_dates) * len(last_records)
new_rows['ventas'] = 0
new_rows['revenue'] = 0
new_rows['semana'] = new_rows['date'].dt.isocalendar().week
new_rows['trimestre'] = new_rows['date'].dt.quarter
new_rows['año'] = new_rows['date'].dt.year

df_future = pd.concat([df3, new_rows], ignore_index=True)
```

Primero, filtramos el dataset por **ID**, ya que este campo está compuesto por el nombre del **ítem** y el código de tienda, lo que lo hace único para cada **ítem** en cada tienda. De esta manera, podemos respetar la posible variabilidad de precio que pueda existir entre el mismo ítem en diferentes tiendas.

Luego, creamos las variables necesarias para que el nuevo dataset, con las fechas a predecir, tenga el mismo formato que el dataset principal. Forzamos las columnas de ventas y facturación (**revenue**) a cero, con el fin de luego concatenarlo correctamente con nuestro dataset principal.

### 1.1 Preprocesamiento:

Inicialmente, la información en nuestro dataset está organizada de manera que cualquier persona pueda entender a qué se refieren los valores de cada variable, asignando a cada una su correspondiente tipo de dato (String, Float, Integer, etc.).

Sin embargo, nuestro modelo, al ser un algoritmo matemático, sólo puede procesar valores numéricos. Estos valores reciben un peso basado en la correlación que el algoritmo encuentra con nuestra variable objetivo (**target**). Por lo tanto, el último paso consiste en adaptar las variables no numéricas a un formato numérico que el modelo pueda consumir.

En particular, las variables que consideramos importantes para la predicción son **Región, Categoría, Tienda, y Evento**. Decidimos "*hardcodear*" estas variables, es decir, asignar un código numérico a cada valor distinto dentro de ellas, para asegurarnos de no perder la referencia de su significado original.

```
# Definir el mapeo manual
region_mapping = {
    'New York': 1,
    'Philadelphia': 2,
    'Boston': 3
}

# Asignar el orden numérico manualmente
df2['region'] = df2['region'].map(region_mapping).astype("int64")
```

Empezamos asignando un número a cada región, mapeando primero la etiqueta que posee originalmente la región con el número que nosotros hemos querido asignarle.

```
# Definir el mapeo manual
store_mapping = {
    'Greenwich_Village': 0,
    'Harlem': 1,
    'Tribeca': 2,
    'Brooklyn': 3,
    'Midtown_Village': 4,
    'Yorktown': 5,
    'Queen_Village': 6,
    'South_End': 7,
    'Roxbury': 8,
    'Back_Bay': 9,
}

# Asignar el orden numérico manualmente a la columna 'store'
df2['store'] = df2['store'].map(store_mapping).astype("int64")
```

A continuación, siguiendo un criterio parecido al anteriormente, mapeamos las tiendas de **New York** del 0 al 3, las de **Philadelphia** del 4 al 6 y por último las últimas tiendas de **Boston** del 7 al 9.

```
# Definir el mapeo manual
dptm_mapping = {
    'ACCESORIES': 0,
    'HOME_&_GARDEN': 1,
    'SUPERMARKET': 2

}

# Asignar el orden numérico manualmente a la columna 'store'
df2['category'] = df2['category'].map(dptm_mapping).astype("int64")
```

En el caso del mapeo de categorías y eventos no tuvimos un criterio en específico, simplemente el único objetivo era dejar informado cada cambio para luego, al inspeccionar los resultados del time series poder relacionar cada valor numérico con su etiqueta original.

```
# Definir el mapeo manual
eve_mapping = {
    'Sin_Evento': 0,
    'SuperBowl': 1,
    'Ramadan starts': 2,
    'Thanksgiving': 3,
    'NewYear': 4,
    'Easter': 5
}

# Asignar el orden numérico manualmente a la columna 'store'
df2['event'] = df2['event'].map(eve_mapping).astype("int64")
```

### 6.10.2. Creación de Nuevas Características (Features)

Para mejorar la precisión del modelo, generamos un conjunto de características adicionales basadas en las ventas históricas y otros factores. Utilizamos técnicas de ***lagging*** (características retrasadas) y **variaciones porcentuales** para capturar patrones de ventas anteriores y su evolución a lo largo del tiempo.

Para automatizar esta tarea en las variables seleccionadas, creamos una **Clase**[1]. Según su definición, una clase en Python es una estructura de programación que permite definir un conjunto de métodos y atributos que describen un objeto o entidad.

Dicho de forma más sencilla, una clase es como una mini-librería creada por nosotros, donde almacenamos funciones que podemos llamar cuando sea necesario.

```
class FeatureGenerator(object):
    '''
    Esto es una clase, nos servirá para montar una especie de "librería" de funciones
    propia para crear nuestras variables customizadas para mejorar nuestro dataset de TimeSeries.
    '''


    def __init__(self, full_df, gb_list):
        '''
        Se inicializa pasando un DataFrame y un listado de columnas las cuales nos servirán
        para generar nuevas variables. La última línea de esta función es para renombrar las
        columnas nuevas y añadirlas al DataFrame.
        '''
        self.full_df = full_df
        self.gb_list = gb_list
        self.objective_column_name = "_".join(gb_list)
```

Desgranando la clase por partes encontramos primero la declaración de la clase, el nombre que recibe esta, para luego encontrarnos con la función que declara las variables que le queramos pasar a la clase que en este caso son 2, **full_df** que recibirá el df que nosotros

[1] https://docs.python.org/es/3/tutorial/classes.html

queramos y **gb_list** que recibirá una lista de esas variables con las quales queremos generar nuevas características. La última línea que observamos en la imagen nos servirá para renombrar las nuevas columnas de features generadas automáticamente.

```
def generate_gb_df(self):
    '''
    Agrupamos el DataFrame original y calculamos sumas y medias de 'ventas' por cada grupo.
    El resultado es un DataFrame con características agregadas que pueden ayudar a identificar tendencias.
    '''
    def my_agg(full_df_, args):
        names = {
            '{}_sum'.format(args): full_df_['ventas'].sum(),
            '{}_mean'.format(args): full_df_['ventas'].mean(),
        }
        return pd.Series(names)

    # Aplicar la función agregada
    gb_df_ = self.full_df.groupby(self.gb_list).apply(my_agg, args=self.objective_column_name).reset_index()
    self.gb_df_ = gb_df_
```

A continuación, vemos la función dentro de la clase que se encargará de hacer los cálculos con cada variable que le introducimos, en este caso en particular, le ordenamos que nos haga la suma y la media en función del target ventas, que ya hemos introducido dentro de la clase para facilitar tiempo. Todo lo que vayamos generando se va a guardar de momento en un dataset temporal llamado **gb_df_.**

```
def generate_shift_features(self, suffix):
    '''
    Crea características retrasadas (lags) y variaciones de las mismas.
    '''
    name_ = self.objective_column_name + "_" + suffix

    for i in range(1, 4):
        # Crear la característica shift
        self.gb_df_['{}_shift_{}'.format(name_, i)] = self.gb_df_.groupby(self.gb_list[1:])[name_].transform(lambda series: series.shift(i))

        # Crear la variación porcentual, solo para los primeros 3 lags
        self.gb_df_['{}_var_pct_{}'.format(name_, i)] = self.gb_df_.groupby(self.gb_list[1:])[name_].transform(lambda series: (series - series.shift(i)) / series.shift(i))

    # Reemplazar NaN, inf, -inf
    self.gb_df_.fillna(-1, inplace=True)
    self.gb_df_.replace([np.inf, -np.inf], -1, inplace=True)

def return_gb_df(self):
    '''
    Retorna el DataFrame con las características shift y variacion generadas.
    '''
    self.generate_shift_features(suffix="sum")
    self.generate_shift_features(suffix="mean")

    return self.gb_df_
```

Para un mejor desarrollo de nuevas variables, tendremos la opción de llamar a la función para generar características que se utilizan en time series para que el modelo aprenda de todas las variaciones que han existido anteriormente para llegar al resultado que obtenemos en la actualidad. Estas características son:

- **Lagging**: Introducimos variables que registran las ventas de las últimas semanas (*lags*), lo que nos permitió identificar tendencias a corto plazo en las ventas.
- **Variaciones porcentuales**: Calculamos las variaciones porcentuales de las ventas entre diferentes períodos de tiempo, con el objetivo de detectar aumentos o caídas significativas que pudieran influir en la predicción futura.

Por último, tendremos la función para recuperar ese dataset que hemos ido creando dentro de la clase que va a contener todas las nuevas features que hemos generado para añadir a nuestro dataset original.

Estas características se crearon tanto a nivel de tienda-producto, como a nivel de categoría y ciudad, lo que permitió capturar comportamientos de ventas agregados y específicos.

### 6.10.3. División de los Datos (Train/Test Split)

Dividimos el conjunto de datos en tres subconjuntos: **entrenamiento**, **validación** y **test**. El conjunto de entrenamiento incluía datos hasta el año 2014, el conjunto de validación se basó en datos entre 2015 y el conjunto de test se centró en 2016 más las predicciones de mayo, de las cuales no teníamos datos reales.

```
print(f"Our train index is {train_index[0]} - ... - {train_index[-1]}\n")
print(f"Our validation index is {valida_index[0]} - ... - {valida_index[-1]}\n")
print(f"Our test index is {test_index[0]} - ... - {test_index[-1]}\n")

##print(f"Our prediction index is {pred_index[0]} - ... - {pred_index[-1]}")
```

```
Our train index is 2011-01-24 00:00:00 - ... - 2014-12-29 00:00:00

Our validation index is 2015-01-05 00:00:00 - ... - 2015-12-28 00:00:00

Our test index is 2016-01-04 00:00:00 - ... - 2016-05-16 00:00:00
```

Este enfoque de división temporal asegura que el modelo esté entrenado en datos históricos y validado en un conjunto separado, lo que nos ayuda a verificar la capacidad del modelo para generalizar y hacer predicciones precisas en datos no vistos anteriormente.

### 6.10.4. Modelado: XGBoost y Búsqueda de Hiperparámetros

Para la predicción de las ventas, utilizamos **XGBRegressor**, de la familia de **XGBoost**, un modelo basado en árboles de decisión, conocido por su alto rendimiento en tareas tanto de regresión como clasificación. **XGBRegressor** es ideal para trabajar con conjuntos de datos grandes y complejos como el nuestro y multivariantes, ya que permite manejar interacciones no lineales entre las variables.

Antes de entrenar el modelo final, aplicamos una búsqueda de hiperparámetros con **GridSearchCV**, una técnica que nos permitió probar diferentes combinaciones de

parámetros (como la profundidad de los árboles, la tasa de aprendizaje y el número de estimadores) y seleccionar la combinación que produjera los mejores resultados.

```
xgbgs = xgb.XGBRegressor(random_state = 239)

parameters = {
    'objective':['reg:linear'],
    'learning_rate': [0.04,0.1,0.2],
    'max_depth': [5,10,15],
    'n_estimators': [300,500,1000],

    }

xgb_grid = GridSearchCV(xgbgs,
                        parameters,
                        scoring='neg_mean_squared_error',
                        cv = 5,
                        n_jobs = 5,
                        verbose=True)
```

Los parámetros que testeamos para ajustar nuestro modelo són pocos ya que **XGBRegressor** es un modelo sencillo pero efectivo y no queríamos forzar el cambio de parámetros que el propio modelo ya ajusta internamente.

Estos parámetros son:

- **Learning Rate:** La tasa de aprendizaje, sirve para controlar cuánto se ajustan los pesos del modelo con respecto al error observado en cada iteración.
- **Max depth:** Profundidad máxima, indica el número máximo de divisiones o niveles que puede tener cada árbol.
- **N Estimators:** el número de árboles que se entrenan en el conjunto durante el proceso.

El resultado del **Grid Search** refleja que los mejores parámetros para nuestro modelo son:

```
Fitting 5 folds for each of 27 candidates, totalling 135 fits
Mejores parámetros: {'learning_rate': 0.04, 'max_depth': 15, 'n_estimators': 1000, 'objective': 'reg:linear'}
```

Una vez seleccionados los mejores parámetros, entrenamos el modelo final utilizando estos ajustes, maximizando así la precisión de las predicciones y añadimos un ***Early Stopping Round*** de 20 para que cuando el modelo haga overfitting durante 20 iteraciones, el modelo deje de entrenarse.

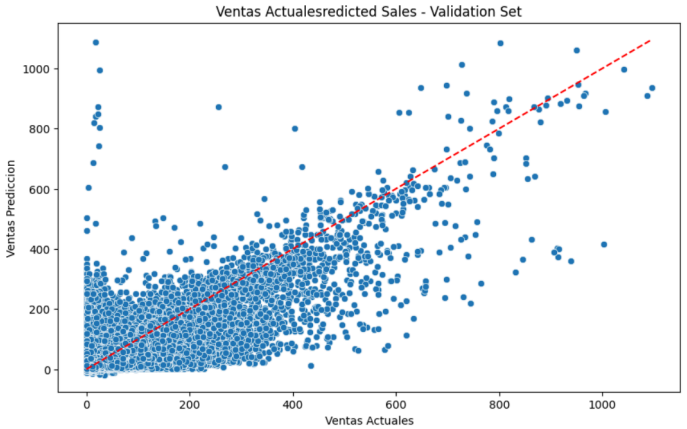
```
model = xgb.XGBRegressor(eval_metric = "rmse",
                         seed = 239,
                         learning_rate= 0.04,
                         max_depth= 15,
                         n_estimators= 1000,
                         objective= "reg:linear",
                         early_stopping_rounds=20
                         )

model.fit(
    X_train,
    Y_train,
    eval_set = [(X_train, Y_train), (X_valida, Y_valida)],
    verbose = True,
)
```

### 6.10.5. Evaluación del Modelo

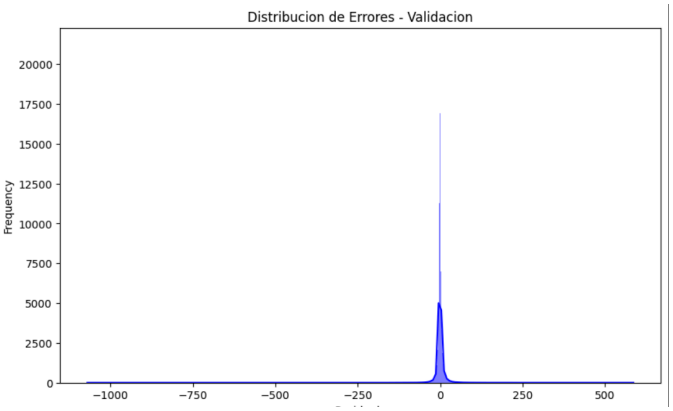
El modelo fue evaluado utilizando **RMSE** (*Root Mean Squared Error*) tanto en el conjunto de entrenamiento como en el conjunto de validación. Los resultados mostraron un **RMSE de 13.03** en validación, en otras palabras, por cada predicción de cada ítem, nuestro modelo será capaz de acertar con un margen de error máximo de ±13 ventas a la semana, lo que indica un nivel de precisión moderado en las predicciones realizadas, poco útil para los productos con pocas ventas, y más certero para aquellos con volúmenes de ventas altos.

Además, realizamos una serie de gráficos comparativos entre las **ventas reales** y las **ventas predichas** para visualizar la precisión del modelo. Estos gráficos incluyeron:

**Un gráfico de dispersión** que mostró una fuerte correlación entre las ventas reales y las predicciones.



**Un histograma de los residuos** (errores de predicción), que presentó una distribución cercana a cero, lo que indica que el modelo no estaba sesgado hacia ningún extremo en particular.



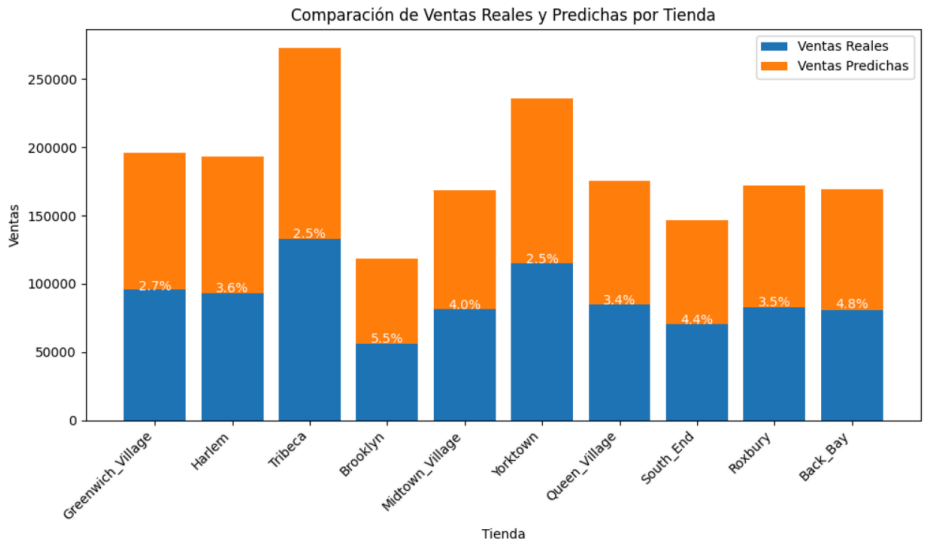

Para obtener un indicador más generalizado de nuestras predicciones, utilizamos la media de errores entre predicción y valor real de ventas obteniendo un valor de 0.3 ventas de error de media, indicando que por lo general el modelo hace muy bien su trabajo.

### 6.10.6. Predicción de las Próximas 4 Semanas

Una vez validado el modelo, realizamos la predicción de ventas para las siguientes **4 semanas** (28 días). Estas predicciones fueron agregadas a nivel de tienda, departamento y ciudad, y comparadas con los valores reales para verificar su precisión.

Para evaluar la calidad de las predicciones, creamos gráficos de barras apiladas que comparaban las **ventas reales** con las **ventas predichas** a nivel de **categoría**, **región** y **tienda**. Estos gráficos no solo confirmaron que el modelo era preciso, sino que también revelaron las diferencias porcentuales entre las predicciones y los valores reales.

Como podemos observar en el siguiente gráfico para ejemplificar los resultados, podemos ver que el modelo va a tener mayor capacidad predictiva en cuanto las ventas sean elevadas. En este caso, **Yorktown** y **Tribeca** son las tiendas con mayores ventas pudiendo predecir las ventas con una certeza de 2.5% respecto a los valores de venta reales y conforme el volumen de ventas es menor, el porcentaje de error aumenta hasta un 5.5% en el caso de la tienda de **Brooklyn**.




## 6.11. Beneficios del Enfoque Implementado

El enfoque desarrollado para la previsión de ventas ofrece varios beneficios clave para DSMarket:

- **Predicciones precisas a nivel de tienda-producto**: Esto permite una planificación más eficiente del reabastecimiento de inventarios y una mejor gestión de la demanda.
- **Escalabilidad**: El modelo puede ser fácilmente ajustado para realizar predicciones a diferentes niveles de agregación (por departamento, tienda o ciudad), lo que proporciona flexibilidad a la hora de tomar decisiones estratégicas.
- **Optimización del inventario**: Al prever con precisión las ventas futuras, DSMarket puede ajustar sus niveles de inventario para evitar tanto el exceso como la falta de stock, optimizando así los costos de almacenamiento y mejorando la experiencia del cliente.

## 6.12. Aspectos para mejorar nuestro modelo

- El modelo consta de un error máximo de predicción (**RMSE)** moderadamente alto, lo que nos indica que las predicciones de aquellos productos de baja rotación (pocas ventas por semana) serán poco certeras.

- El modelo pierde capacidad de predicción cuanto más lejos en el tiempo queramos predecir, esto puede ser debido a que todas nuestras variables originales y las generadas por nosotros, están contempladas en un espacio de tiempo de semana tras semana, con lo que al intentar predecir las ventas de más allá de una semana, el modelo no realiza bien los cálculos.

El enfoque predictivo desarrollado para DSMarket proporciona una herramienta potente para la planificación operativa, asegurando que las decisiones de reabastecimiento estén respaldadas por datos precisos y actualizados. Sin embargo, para mantener su efectividad, será necesario entrenar el modelo semanalmente.

El modelo resultará especialmente útil para predecir las ventas de productos con alta rotación, aunque podría ser menos efectivo para aquellos con un bajo ratio de ventas.

Finalmente, sería recomendable desarrollar un modelo adicional enfocado en las predicciones mensuales de ventas, el cual podría ser utilizado por la gerencia para prever los gastos de stock y la facturación mes a mes. Mientras que el modelo actual está orientado principalmente a la optimización del reabastecimiento, este nuevo modelo cubriría mejor las necesidades de previsión de gastos.

# 7. Task 4: CASO DE USO DE REABASTECIMIENTO DE VENTA

En esta última tarea del proyecto, abordamos la implementación de una solución automatizada para optimizar el reabastecimiento de productos en las tiendas de DSMarket. La solución se basa en los modelos de predicción de ventas que desarrollamos previamente y está diseñada para integrarse de forma eficiente con los sistemas internos de la empresa. Para ello, hemos empleado un enfoque de **MLOps** (Machine Learning Operations), junto con herramientas como **Docker** y pipelines de **CI/CD** (Integración Continua y Despliegue Continuo), con el objetivo de garantizar una infraestructura flexible, escalable y mantenible para la automatización del proceso de reabastecimiento. A continuación, se resumen los pasos clave de la metodología empleada:

## 7.1. Metodología

| | | |
|---|---|---|
| 1 | **Contenerización del Modelo** | Utilizamos **Docker** para empaquetar el modelo de predicción de ventas junto con todas sus dependencias en un contenedor. Un contenedor es una unidad ligera de software que incluye el código, las bibliotecas y configuraciones necesarias para ejecutar el modelo de manera consistente en cualquier entorno (servidores locales o en la nube). Esto garantiza que no haya discrepancias entre el entorno de desarrollo y producción. **Imagen Docker** es el archivo que contiene todo lo necesario, y **Contenedor Docker** es la instancia ejecutable de esa imagen. |
| 2 | **Despliegue de API REST** | Desplegamos una **API REST** para que las tiendas puedan interactuar con el modelo en tiempo real. Una API REST permite que los sistemas internos envíen datos de ventas actuales (como ventas diarias por tienda) y reciban como respuesta predicciones para los próximos 28 días. Esta API funciona como un puente entre el sistema de DSMarket y el modelo, facilitando la toma de decisiones automáticas sobre el reabastecimiento de inventario. Las API REST se basan en métodos HTTP como **GET** y **POST** para la comunicación entre sistemas. |
| 3 | **CI/CD (Integración/Despliegue Continuo)** | Implementamos un pipeline de **CI/CD** (Continuous Integration/Continuous Deployment), que automatiza la integración de nuevos cambios en el modelo y su despliegue continuo en producción. **Integración Continua (CI)** garantiza que las nuevas versiones del código se integren y validen automáticamente mediante pruebas, mientras que **Despliegue Continuo (CD)** asegura que, una vez validadas, las versiones se despliegan automáticamente en producción sin necesidad de intervención manual. Esto garantiza que DSMarket siempre utilice las predicciones más actualizadas. |

| | | |
|---|---|---|
| 4 | **Monitorización del Modelo** | Para garantizar el rendimiento del modelo, se implementó un sistema de **monitorización continua** utilizando herramientas como **Prometheus** y **Grafana**. Estas herramientas miden métricas clave como el tiempo de respuesta y la precisión de las predicciones. Además, si se detectan cambios significativos en los datos de entrada (un fenómeno conocido como **Data Drift**), el sistema puede activar un reentrenamiento automático del modelo, asegurando que las predicciones sigan siendo precisas en un entorno cambiante. |
| 5 | **Despliegue Blue-Green** | Aplicamos una estrategia de **Blue-Green Deployment**, en la que se mantienen dos entornos paralelos: uno (Blue) en funcionamiento y otro (Green) con la nueva versión del modelo. Solo cuando se valida la nueva versión en el entorno Green, todo el tráfico se redirige hacia este, garantizando una transición suave y evitando interrupciones en el servicio. Este enfoque es ideal para minimizar riesgos durante el despliegue de actualizaciones. |

## 7.2. Desarrollo

### 7.2.1. Contenerización del Modelo con Docker

**Docker** es una herramienta que permite encapsular una aplicación (en este caso, el modelo de predicción de ventas) junto con todas sus dependencias en un **contenedor**. Un contenedor es una unidad de software que incluye todo lo necesario para que el modelo funcione de manera consistente en cualquier entorno. Esto significa que independientemente del sistema operativo o las configuraciones del servidor, el modelo podrá ejecutarse sin problemas.

Para este proyecto, utilizamos Docker para **contenerizar** el modelo de predicción de ventas. Esto implica crear una imagen de Docker que incluya el código del modelo, las bibliotecas de Python necesarias (como `pandas` para la manipulación de datos y **XGBoost** para el modelo de predicción), y los archivos de datos requeridos. Esta imagen se puede desplegar fácilmente en cualquier servidor o en la nube.

- **Contenerización**: Proceso de empaquetar una aplicación y sus dependencias en una imagen ligera y portable.
- **Imagen Docker**: El archivo que contiene el modelo, dependencias y configuraciones necesarias.
- **Contenedor Docker**: La instancia de ejecución de la imagen en un entorno de producción o desarrollo.

Este enfoque de contenerización nos asegura que el modelo de predicción siempre se ejecutará de la misma forma, eliminando posibles inconsistencias entre entornos (por ejemplo, entre el entorno de desarrollo y producción).

### 7.2.2. Despliegue de una API REST para el Modelo

Una vez que el modelo ha sido contenerizado, el siguiente paso fue **desplegar una API REST** que permita a los sistemas internos de DSMarket interactuar con el modelo en tiempo real. Una **API REST (Representational State Transfer)** es una interfaz que permite la comunicación entre diferentes sistemas a través de internet. Es muy utilizada en el desarrollo de aplicaciones web debido a su simplicidad y escalabilidad.

A través de esta API, los sistemas de DSMarket pueden enviar datos de ventas actuales (por ejemplo, ventas diarias de una tienda) y recibir como respuesta las predicciones de ventas futuras. Estas predicciones se generan para cada tienda y producto, con un horizonte temporal de 28 días, permitiendo así planificar de forma precisa el reabastecimiento.

- **API**: Es un conjunto de reglas que permite a diferentes aplicaciones interactuar entre sí. En este caso, la API actúa como un puente entre el sistema de DSMarket y nuestro modelo de predicción.
- **REST**: Un estilo arquitectónico para diseñar APIs que permite la transferencia de datos a través de internet utilizando métodos HTTP como GET y POST.

El modelo está expuesto como un servicio web accesible a través de la API, lo que significa que cualquier tienda de DSMarket puede enviar datos y recibir predicciones en tiempo real. Esto es especialmente útil en un entorno distribuido como el de una cadena de tiendas, donde la necesidad de tener una plataforma centralizada que gestione las predicciones es crucial.

### 7.2.3. Implementación de CI/CD para Despliegue Continuo

Una parte fundamental para garantizar la **automatización y actualización continua** de los modelos de machine learning es el uso de **CI/CD**. **CI (Continuous Integration)** y **CD (Continuous Deployment)** son prácticas que automatizan la integración y el despliegue de nuevas versiones del modelo, asegurando que siempre se utilicen las versiones más recientes y optimizadas.

- **Integración Continua (CI)**: Se refiere al proceso en el que el código de los modelos se actualiza automáticamente cuando hay nuevos cambios (por ejemplo, nuevos datos para entrenar el modelo). Cada vez que se hacen cambios en el código o en los datos, se ejecutan pruebas automáticas para verificar que el nuevo modelo funciona correctamente.
- **Despliegue Continuo (CD)**: Este proceso asegura que, una vez que el nuevo modelo ha pasado todas las pruebas, se despliega automáticamente en producción, sustituyendo al modelo anterior. Esto permite que cualquier mejora o ajuste en el modelo esté disponible inmediatamente, sin necesidad de intervención manual.

En nuestro caso, implementamos un pipeline de CI/CD que se encarga de:

- **Automatizar la construcción de nuevas versiones del modelo**.
- **Ejecutar pruebas automáticas** para garantizar la precisión de las predicciones.
- **Desplegar las nuevas versiones** de manera segura y escalable en los servidores de DSMarket, sin interrumpir el funcionamiento de la API.

Este enfoque permite a DSMarket beneficiarse de las últimas actualizaciones en el modelo sin riesgos, asegurando que el sistema esté siempre utilizando las predicciones más precisas posibles.

### 7.2.4. Monitorización del Rendimiento del Modelo

Una vez que el modelo está en producción, es esencial asegurarse de que sigue funcionando correctamente y que las predicciones continúan siendo precisas a lo largo del tiempo. Para esto, implementamos un sistema de **monitorización del rendimiento** del modelo en producción.

- **Monitorización del modelo**: Proceso de seguimiento y evaluación continua del desempeño del modelo para detectar problemas, como una caída en la precisión de las predicciones o cambios en los datos de entrada que puedan afectar su rendimiento.

Utilizamos herramientas como **Prometheus** y **Grafana** para medir y visualizar las métricas clave del modelo, como el **tiempo de respuesta** de las predicciones y la **precisión** en función de los datos reales que van entrando. Esto nos permite detectar cualquier problema a tiempo.

Además, implementamos un sistema de **reentrenamiento automático**. En el caso de que los datos de ventas cambien de manera significativa (por ejemplo, por cambios estacionales o en el comportamiento de los consumidores), el sistema puede detectar estos cambios y activar automáticamente un nuevo ciclo de entrenamiento del modelo, utilizando los datos más recientes para mejorar las predicciones.

- **Data Drift**: Fenómeno en el que los datos de entrada cambian con el tiempo, lo que puede hacer que el modelo sea menos preciso si no se actualiza regularmente.

### 7.2.5. Estrategia de Despliegue Blue-Green

Para evitar interrupciones durante el despliegue de nuevas versiones del modelo, utilizamos una estrategia de **Blue-Green Deployment**. Este enfoque consiste en mantener dos entornos de producción paralelos: uno que está en funcionamiento (Blue) y otro con la nueva versión del modelo (Green). Una vez que la nueva versión ha sido probada y validada en el entorno Green, todo el tráfico de la aplicación se redirige a este entorno, garantizando una transición suave y sin fallos.

- **Blue-Green Deployment**: Método que permite despliegues seguros al tener dos entornos paralelos, minimizando riesgos y asegurando una disponibilidad continua del servicio.

Esto es particularmente importante en un entorno crítico como el de DSMarket, donde cualquier interrupción en la capacidad de predecir ventas o gestionar inventarios podría afectar las operaciones de las tiendas.

## 7.3. Beneficios de la Solución

La implementación de esta solución de MLOps y Docker en el proceso de reabastecimiento de inventarios de DSMarket ha aportado una serie de beneficios clave:

1. **Automatización del proceso de reabastecimiento**: La solución permite que las decisiones de reabastecimiento se realicen de forma automatizada y basada en datos actualizados, reduciendo la intervención manual y mejorando la eficiencia operativa.

2. **Predicciones en tiempo real**: Gracias a la API REST, las tiendas de DSMarket pueden recibir predicciones en tiempo real, lo que permite ajustar rápidamente los niveles de inventario en función de la demanda prevista.
3. **Escalabilidad**: El uso de Docker asegura que el sistema pueda ser escalado a nivel de toda la empresa sin problemas de compatibilidad. Ya sea que se despliegue en una tienda o en cien, el sistema se mantiene consistente.
4. **Mantenimiento continuo y mejoras sin interrupciones**: Con la implementación de CI/CD, DSMarket puede beneficiarse de las últimas mejoras en el modelo sin sufrir interrupciones en sus operaciones diarias. Además, el sistema de monitorización asegura que cualquier problema de rendimiento sea detectado y corregido a tiempo.
5. **Reducción de costos**: Al optimizar el reabastecimiento y minimizar el exceso de inventario, DSMarket puede reducir significativamente los costos asociados con el almacenamiento de productos no vendidos, mejorando a su vez la disponibilidad de productos clave.

En resumen, la solución implementada no solo garantiza una gestión eficiente del inventario basada en datos, sino que también prepara a DSMarket para el futuro, dotándola de una infraestructura robusta y escalable que puede seguir evolucionando a medida que se integran nuevas tecnologías.

# 9. Argumentación de los Objetivos Planteados

Los objetivos definidos al inicio del proyecto han sido alcanzados satisfactoriamente, logrando la implementación de soluciones de ciencia de datos que optimizan diversas áreas críticas para DSMarket. En primer lugar, se desarrolló un modelo predictivo de ventas que ha mejorado la precisión de las estimaciones a 28 días vista, lo que ha permitido una mayor eficiencia en la gestión de inventarios. Los enfoques de clustering aplicados a productos y tiendas permitieron segmentar eficazmente el mercado, facilitando campañas de marketing más personalizadas y estrategias ajustadas a las necesidades de cada tienda. Además, se implementaron dashboards de Business Intelligence (BI) para el monitoreo continuo de los indicadores clave, brindando al equipo ejecutivo una visión clara y accionable para la toma de decisiones en tiempo real.

Otro de los objetivos críticos, el caso de uso de reabastecimiento de tiendas, se ha implementado exitosamente con la integración de los modelos predictivos en los sistemas de inventario. Esto ha permitido minimizar el stock remanente y optimizar la rotación de productos, especialmente en la categoría de supermercado, lo cual es fundamental para mejorar los márgenes de beneficio y reducir costos operativos. En conclusión, se ha avanzado considerablemente en la transformación digital de DSMarket, sentando las bases para una empresa más data-driven.

# 10. Conclusiones

El proyecto llevado a cabo en DSMarket marca un hito importante en la digitalización de la empresa, transformando procesos críticos mediante el uso de la ciencia de datos. A través del desarrollo de modelos predictivos, el análisis exploratorio de datos y la segmentación por clustering, se ha optimizado tanto la predicción de ventas como la gestión de inventarios. Esto no solo ha permitido mejorar la eficiencia operativa, sino también ofrecer un mejor servicio al cliente mediante la personalización de campañas y estrategias de reabastecimiento.

Los resultados obtenidos reflejan que New York es el mercado más relevante para DSMarket, con una alta demanda en categorías como Hogar y Jardín, mientras que Boston y Philadelphia han mostrado estabilidad y crecimiento en productos de consumo diario y accesorios, respectivamente. La correcta implementación de los modelos predictivos y su integración en los sistemas de inventarios ha permitido mejorar los márgenes de beneficio, asegurando que DSMarket está en el camino correcto para consolidarse como una empresa impulsada por datos.

En resumen, los objetivos planteados han sido cumplidos exitosamente, y las soluciones implementadas dejan a DSMarket en una posición favorable para continuar su proceso de digitalización. Este proyecto ha sentado las bases para que la empresa continúe aprovechando el poder de los datos en la toma de decisiones estratégicas y operativas, lo que será clave para su competitividad a largo plazo.