



Python con Excel III Análisis Cluster II, Caso FilmMarvelStudio

Introducción al Análisis Cluster con Python en Excel, uso de las principales librerías. Marco conceptual básico. Separar columna de texto de valores.

Jose Ignacio González Gómez

Departamento de Economía, Contabilidad y Finanzas - Universidad de La Laguna

www.jggomez.eu

Ejercicio adaptado de:

V. 2.5

Contenido

Cuestiones técnicas relacionadas I. Introducción al Análisis Cluster con Excel en Python	2
El Poder del Análisis de Clúster, aproximación conceptual.....	2
Aplicaciones, segmentación de clientes, riesgo en banca, etc.....	2
¿Cómo funciona?	3
Requisitos de los datos para el Análisis de Clúster	3
El Proceso de Análisis de Clúster con Python y Excel	3
Fase I. Cargar Datos – Crear el dataframe	4
Fase II. Escalado de Datos: StandardScaler	4
Fase III. Agrupamiento. Técnica de Clustering: Aplicación de K-Means	4
Fase IV Añadir columna cluster al Dataframe	5
Fase V Análisis de los Clústeres Resultantes:.....	5
Caso Analisis Cluster II FilmMarvelStudio. Maven Analytics, perfil de las películas, análisis cluster	6
Presentacion	6
Objetivo.....	6
Información disponible	6
Análisis con Python para Excel.....	7
Introducción	7
Fase I. Cargar Datos. Crear el Dataframe	8
Fase II. Escalado de Datos: StandardScaler	9
Fase III. Creación de los Cluster, agrupar los datos con k-medias.....	11
Fase IV Añadir columna cluster al Dataframe	12
Fase V Análisis de los Clústeres Resultantes	13

Cuestiones técnicas relacionadas I. Introducción al Análisis Cluster con Excel en Python

Presentamos a continuación un resumen de los puntos clave y las ideas principales sobre el análisis de clúster utilizando Python y Excel, centrándonos en la accesibilidad, el proceso técnico y las metodologías de análisis de resultados.

El Poder del Análisis de Clúster, aproximación conceptual

El análisis de clúster se presenta como una "superpotencia analítica" útil para cualquier profesional, independientemente de su sector (sanidad, gobierno, marketing, cadena de suministro, servicio al cliente, etc.), resaltando que no es necesario ser un "mago de las matemáticas" para aprender y aplicar esta técnica.

El análisis de clúster es una técnica para agrupar elementos similares dentro de un conjunto de datos. Se considera una "superpotencia analítica" porque permite a los profesionales independientemente de su sector (sanidad, gobierno, marketing, cadena de suministro, servicio al cliente, etc.), obtener nuevas y potentes ideas a partir de sus datos, independientemente de su campo, resaltando que no es necesario ser un "mago de las matemáticas" para aprender y aplicar esta técnica.

Con un poco más de rigor el **análisis de clúster** es una técnica de **minería de datos y estadística multivariada** que permite **agrupar objetos o datos** en grupos (clústeres) de tal manera que los elementos dentro de un mismo grupo sean **más similares entre sí** que con los de otros grupos.

Es básicamente un método de segmentación **no supervisado**, porque no partes de etiquetas predefinidas, sino que dejas que los datos "hablen" y formen sus propios grupos. Está dirigido a profesionales que buscan ir más allá de las capacidades estándar de Excel.

Aplicaciones, segmentación de clientes, riesgo en banca, etc.

Segmentación de clientes en retail y e-commerce

Una empresa de retail podría usar análisis de clúster para analizar el comportamiento de compra (frecuencia, tipo de productos, gasto promedio) para agrupar a los clientes en clústeres:

- Compradores frecuentes, Clientes premium
- Compradores ocasionales
- Clientes sensibles al precio
- Clientes leales a marcas específicas
- Cazadores de ofertas

Esto permite **tomar decisiones más informadas** sobre promociones, productos y atención al cliente.

Análisis de riesgo en banca y seguros

Ejemplo: Un banco como BBVA utiliza clustering para agrupar a sus clientes según su perfil financiero:

- Bajo riesgo (ingresos estables, buen historial)
- Riesgo medio (deudas moderadas)
- Alto riesgo (morosidad, ingresos variables)

Así pueden ajustar condiciones de crédito, tasas de interés o estrategias de cobranza.

Investigación médica y farmacéutica

Ejemplo: Laboratorios como Pfizer agrupan pacientes según síntomas, respuestas a tratamientos o perfiles genéticos.

Esto permite desarrollar medicamentos personalizados y mejorar la eficacia de los ensayos clínicos.

Optimización logística y distribución

Ejemplo: Empresas como DHL o Mercadona usan clustering para agrupar zonas geográficas con patrones similares de demanda. Esto mejora rutas de entrega, reduce costes y optimiza inventarios.

Desarrollo de productos y UX

Ejemplo: Una empresa de software como Spotify analiza el uso de su app para agrupar usuarios según hábitos:

- Oyentes activos
- Usuarios de listas automáticas
- Exploradores de géneros

Así pueden mejorar la interfaz, lanzar nuevas funciones o ajustar el algoritmo de recomendación

¿Cómo funciona?

Sin etiquetas previas: No requiere datos clasificados.

Selección de variables: Se eligen las características relevantes para agrupar los datos.

Medición de similitud: Se usa una métrica (como la distancia euclidiana) para evaluar qué tan parecidos son los elementos.

Algoritmos de agrupamiento más comunes:

- *K-means*: El más usado. Divide los datos en k grupos según distancias (euclídeas normalmente). Agrupa datos en un número fijo de clústeres.
- *Jerárquico*: Crea una estructura de clústeres anidados, un dendrograma (árbol de similitudes). Muy útiles cuando no sabes cuántos grupos hay
- *DBSCAN*: Agrupa según densidad de datos.
- *Clúster espectral*: Usa álgebra lineal para reducir la complejidad

Evaluación de resultados: Se analiza la calidad de los clústeres formados (por ejemplo, con el índice de silueta).

Requisitos de los datos para el Análisis de Clúster

Para realizar un análisis de clúster efectivo, los datos deben cumplir con criterios específicos, especialmente al inicio.

- **Estructura de Datos:** Cada fila debe representar un "elemento de interés" (por ejemplo, clientes, pacientes, reclamaciones de seguros).
- Cada columna debe ser una "medición numérica".
- **Tipo de Datos:** Es crucial utilizar datos numéricos al principio; los datos categóricos o de cadena son conceptos más avanzados.
- **Ejemplo de Datos:** La fuente utiliza datos de comportamiento de clientes de supermercado, donde cada fila es un cliente individual y cada columna es una medición numérica de su comportamiento (compras de vino, ingresos, compras de carne, etc.).

En resumen, para comenzar con el análisis de clúster, es crucial tener datos donde cada fila sea un "elemento de interés" y cada columna sea una "medición numérica". Se deben evitar los datos categóricos o de cadena (texto) en las etapas iniciales, ya que son un concepto más avanzado.

El Proceso de Análisis de Clúster con Python y Excel

La integración de Python en Excel simplifica el proceso, permitiendo a los usuarios cargar, preprocesar y aplicar algoritmos de clustering.

Fase I. Cargar Datos – Crear el dataframe

Comenzamos por importar datos desde hojas de cálculo o Power Query de Excel a Python. Es recomendable usar tablas de Excel con encabezados y formatos de datos correctos.

Recordemos, un DataFrame es la forma en que Python representa tablas de datos completas, conceptualmente similar a una tabla de Excel. Es una estructura de datos bidimensional con etiquetas de columnas y filas que permite manipular y analizar datos de manera eficiente.

Fase II. Escalado de Datos: StandardScaler

Antes de aplicar K-Means, es fundamental escalar las columnas numéricas para ponerlas "en igualdad de condiciones". Esto es porque algunas columnas (como los ingresos) pueden tener valores mucho mayores que otras, lo que afectaría el rendimiento de K-Means.

Es decir, el escalado de datos es crucial porque las columnas numéricas en un conjunto de datos a menudo no están en la misma escala (por ejemplo, los ingresos son mucho mayores que el número de compras). Escalar las columnas las pone en igualdad de condiciones, lo que permite que el algoritmo K-Means funcione de manera más efectiva al evitar que las características con valores más grandes dominen el cálculo de las distancias.

Esto se logra usando un algoritmo de Python de la clase "**Standard Scaler**", que escala todas las columnas y crea un objeto scaler, para que todas las columnas estén en igualdad de condiciones, creando un nuevo dataframe con los datos escalados. El escalado puede resultar en números negativos y positivos, lo cual no es un problema para K-means

Así **StandardScaler** proporciona la funcionalidad para que todas las columnas numéricas estén escaladas, en igualdad de condiciones, es decir transforma los datos para que sus valores tengan una escala y magnitud similares, sin cambiar la distribución subyacente de los datos.

Fase III. Agrupamiento. Técnica de Clustering: Aplicación de K-Means

Una vez escalado los datos procedemos a agruparlos para lo cual aplicaremos el algoritmo más popular K-Means, diseñado para agrupar datos en conjuntos. Un requisito fundamental es que se debe especificar de antemano el número de clústeres (K) que se desea encontrar en los datos. Así es necesario:

Determinación del Número de Clústeres (K): K-Means requiere que se especifique de antemano el número de clústeres (k, por ejemplo, n_clusters=3). Aunque existen técnicas de optimización, normalmente se asume o comienza con un k de 3 que es una elección razonable.

Especificar el número de intentos (n_init). Se crea un objeto KMeans, especificando el número de clústeres y el número de intentos (n_init), ya que K-means utiliza aleatorización y realizar múltiples intentos puede ayudar a encontrar la mejor agrupación. Se recomienda n_init=50 para ejecutar múltiples clustering y seleccionar el mejor debido a la aleatoriedad del algoritmo. Luego se aplica fit() a los datos escalados.

Resultados: Etiquetas de Clúster (metodo fit) : Después de "ajustar" el modelo K-Means a los datos escalados, se utiliza el método fit con los datos escalados para realizar la agrupación. El resultado son las "etiquetas" o asignaciones de clúster para cada fila (por ejemplo, 0, 1, 2), es decir las "etiquetas" que son las asignaciones de clúster para cada fila del conjunto de datos. Estas etiquetas, a menudo representadas como números (por ejemplo, 0, 1, 2), indican a qué clúster pertenece cada punto de datos (0, 1, 2 en el ejemplo de 3 clústeres).

Integración y Salida de Resultados. Combinación de Etiquetas con Datos Originales: Después de obtener las asignaciones de clúster, estos se integran con los datos originales para su análisis. Las asignaciones de clúster se integran fusionándolas como una nueva columna, generalmente llamada "**cluster**", en el **DataFrame original sin escalar**. Esto

permite analizar las características de los clústeres en el contexto de los datos originales y comprensibles

Es decir, las asignaciones de clúster se añaden como una nueva columna al DataFrame original (no escalado) para facilitar el análisis, así el dataframe modificado, incluye la columna de asignación de clúster, se puede exportar directamente a una hoja de cálculo de Excel cambiando el tipo de valor de "python object" a "excel value". Esto permite ver todos los números originales junto con la asignación de clúster para cada fila.

Fase IV Añadir columna cluster al Dataframe

Conocido el cluster al que pertenece cada registro procedemos a añadir una nueva columna con un nombre identificativo, por ejemplo 'Cluster' al DataFrame y así tenemos la tabla ya preparada para el análisis o estudio.

Fase V Análisis de los Clústeres Resultantes:

Una vez que los datos con las asignaciones de clúster están en Excel, los dos métodos más frecuentes para analizar los clústeres son:

1. El uso de tablas dinámicas para ver promedios u otras métricas (edad, ingresos, compras totales de alimentos) para cada clúster. Esto permite identificar diferencias entre los clústeres (por ejemplo, el clúster 2 es más joven que el 1 y el 0; el clúster 0 tiene ingresos más altos).
2. Visualizaciones de datos (gráficos) para una exploración visual de cómo las características individuales se distribuyen dentro y entre los clústeres (Análisis Exploratorio de Datos o EDA). Las visualizaciones permiten entender no solo los promedios, sino también la distribución de los datos dentro de cada clúster (por ejemplo, el rango de valores bajos y altos).

Combinación con Machine Learning - "Lo Mejor": La "mejor forma" de analizar los clústeres, según la fuente, es combinar el análisis de clúster con otras técnicas de machine learning.

Caso Analisis Cluster II FilmMarvelStudio. Maven Analytics, perfil de las películas, análisis cluster

Fuente: [Aprendizaje automático con Python en Excel: ¿Lo mejor de ambos mundos? \(¡Tutorial completo!\) - YouTube](#)

Presentacion

Objetivo

Marvel Studios se dedica a la producción de películas de superhéroes basadas en los personajes icónicos que aparecen en los cómic de Marvel Comics.

Para evaluar el éxito de estas producciones, contamos con un fichero (dataframe) que recoge información detallada sobre diversas películas producidas por Marvel Studios a lo largo de los años sobre la aceptación de estas películas, tanto del público como de la crítica especializada, así como sobre sus resultados económicos.

En este archivo, cada fila corresponde a una película, mientras que las columnas recogen diferentes métricas numéricas asociadas.

El propósito de este estudio no es realizar una predicción sobre el éxito futuro de las películas, sino segmentarlas según su rendimiento. A través de esta segmentación, se busca identificar cuáles han sido las películas con el mejor desempeño global, considerando tanto la aceptación del público y de la crítica como los resultados financieros.

Para alcanzar el objetivo, vamos a llevar a cabo un **análisis de clústeres** utilizando **Python en Excel**, destacando la importancia de que los datos sean **numéricos** para un clustering efectivo.

Se explica el proceso de **escalar los datos** para que las columnas tengan el mismo peso y luego se aplica el **algoritmo K-means** para agrupar clientes según su comportamiento de compra. Finalmente, se ilustra cómo **analizar los clústeres resultantes** utilizando **tablas dinámicas** y **visualizaciones de datos** dentro de Excel para obtener perspectivas significativas.

El proceso implica cargar los datos en Python dentro de Excel (crear el dataframe) a partir de la tabla de Excel y seleccionar las características relevantes, la escalada de datos usando técnicas de Python como la normalización, y la determinación del número óptimo de clusters mediante un gráfico de inercia. Finalmente, se interpreta los resultados del clustering para identificar películas de Marvel de alto rendimiento y cómo visualizar los clusters dentro de Excel.

Información disponible

Como hemos comentado disponemos de una tabla que contiene la producción de la cinematográfica y presentamos en la **¡Error! No se encuentra el origen de la referencia.** el formato y contenido de las columnas o campos disponibles. Destacar que los valores monetarios están expresados en M\$ que significa millones de dólares.

	A	B	C	D	E	F
1	film	categoría	recaudación mundial (M\$)	% del presupuesto recuperado	puntuación de los críticos (%)	puntuación del público (%)
2	Ant-Man: El hombre hc Ant-Man	Ant-Man	518,0 M\$	398%	83%	85%
3	Ant-Man y la Avispa	Ant-Man	623,0 M\$	479%	87%	80%
4	Vengadores: La era de l Vengadores		1.395,0 M\$	382%	76%	82%
5	Vengadores: End game	Vengadores	2.797,0 M\$	699%	94%	90%
6	Vengadores: Infinity W	Vengadores	2.048,0 M\$	683%	85%	91%
7	Black Panther: Pantera	Black Panthe	1.336,0 M\$	668%	96%	79%

Ilustración 1

El numero de columnas o campos disponibles con información relacionada con cada registro es de 19 en la **¡Error! No se encuentra el origen de la referencia.** presentamos el detalle de cada una de ellas.

1	film	<i>Título de la película</i>
2	categoría	<i>Categoría</i>
3	recaudación mundial (M\$)	<i>Ingresos mundiales</i>
4	% del presupuesto recuperado	<i>% del presupuesto recuperado</i>
5	puntuación de los críticos (%)	<i>puntuación de los críticos (%)</i>
6	puntuación del público (%)	<i>puntuación del público (%)</i>
7	desviación entre el público y los críticos (%)	<i>desviación entre el público y los críticos (%)</i>
8	presupuesto (M\$)	<i>presupuesto (M\$)</i>
9	recaudación nacional (M\$)	<i>recaudación nacional (M\$)</i>
10	recaudación internacional (M\$)	<i>recaudación internacional (M\$)</i>
11	primer fin de semana (M\$)	<i>primer fin de semana (M\$)</i>
12	segundo fin de semana (M\$)	<i>segundo fin de semana (M\$)</i>
13	Caída entre el primer y el segundo fin de semana	<i>Caída entre el primer y el segundo fin de semana</i>
14	% de la recaudación del fin de semana de estreno	<i>% de la recaudación del fin de semana de estreno</i>
15	% de la recaudación nacional	<i>% de la recaudación nacional</i>
16	% de la recaudación internacional	<i>% de la recaudación internacional</i>
17	% del presupuesto del fin de semana de estreno	<i>% del presupuesto del fin de semana de estreno</i>
18	Año	<i>Año de estreno</i>
19	Fuente	<i>Enlace a los datos de la película</i>

Nota: M\$ significa millones de dólares

Tabla 1

Para el diseño del modelo de aprendizaje no supervisado mediante clustering (agrupamiento o segmentación de las películas en grupos), sino segmentar las películas en grupos se ha decidido seleccionar las 4 siguientes características (rellenado en gris): “recaudación mundial (M\$)”, “% del presupuesto recuperado”, “puntuación de los críticos (%)” y “puntuación del público (%)”.

Queremos realizar un análisis de cluster para agrupar las películas por similitudes. Brevemente, señalar que esta es una técnica estadística multivariante que clasifica un conjunto de individuos u objetos en grupos (clústeres) basándose en la semejanza de sus características, de modo que los elementos dentro de un mismo grupo sean muy similares y diferentes de los de otros grupos. Su objetivo es descubrir patrones ocultos en los datos y agruparlos de forma que sean más fáciles de comprender, siendo útil en diversas áreas como la biología, la economía o el marketing.

Así vamos a agrupar las películas en grupo similares, es decir procedemos a segmentar los clientes en grupos similares según sus características.

Análisis con Python para Excel

Introducción

Vamos a sintetizar el proceso que vamos a seguir con los scripts a desarrollar en Python y que son los siguientes:

Fase I: Definimos el Dataframe df	Definimos cuál es nuestra tabla de datos (dataframe) y le asignamos el nombre df
Fase II a: Dividimos las columnas numéricas de las de texto creando dos dataframe df_texto y df_numericas	Como nuestra tabla de datos (df) contiene columnas de texto y numéricas, necesitamos separarlas para poder llevar a cabo el proceso de normalización o estandarización, creando así un dataframe solo con las numéricas y que llamamos df_numericas.

Fase II b: Escalar los datos numéricos antes de agruparlos con k-medias df_X	Sobre tabla anterior (df_numericas) procedemos a normalizar los valores y creamos una nueva tabla que la llamamos df_X que contiene los valores normalizados y preparados para aplicar el algoritmo Kmedias
Fase III: Agrupamos obteniendo los Cluster que definimos como k_means.labels_	Aquí aplicamos el algoritmo K_medias para agrupar y obtenemos los grupos para cada registro en un array k_means.labels_
Fase IV: Fusionar las etiquetas de los clústeres al dataframe original (df) df['Cluster'] = k_means.labels_	Ahora vamos a añadir una columna df['Cluster'] a nuestra tabla inicial (df) Cluster con los valores anteriores de las etiquetas k_means.labels_
Fase V: Generar el DataFrame con el Cluster df	Generamos ahora el df completo con todas las columnas de texto y numéricas así como la del cluster los que nos permitirá establecer la base para las tablas y gráficos dinámicos.

Pasamos a explicar el código paso a paso.

Fase I. Cargar Datos. Crear el Dataframe

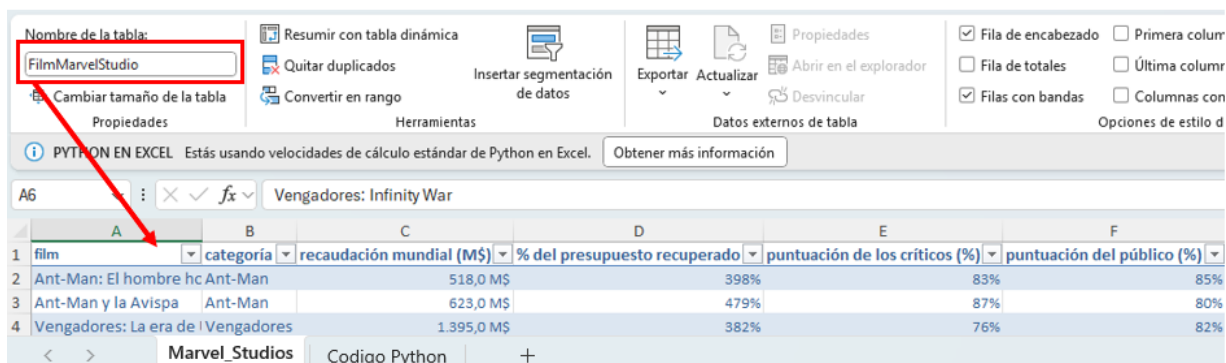


Ilustración 2

Como podemos observar de la Ilustración 2 disponemos de 2 hojas, la primera con el nombre de la pestaña **Marvel_Studios** que contiene la tabla de datos con el nombre “FilmMarvelStudios”.

La otra hoja o pestaña que deberemos crear con el nombre **Codigo Python** es donde vamos a escribir los scripts necesarios de Python, para nuestro análisis.

Ilustración 3

Comenzamos definiendo el Dataframe que llamaremos como df al ser el nombre más corriente y para ello vamos a la celda D2 y seleccionamos la tabla origen de los datos con los encabezados.

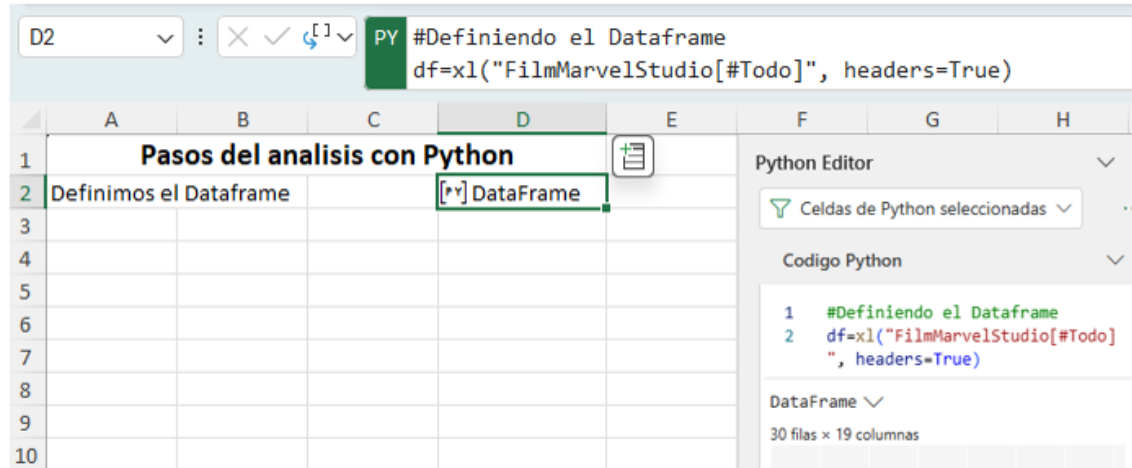


Ilustración 4

El script Python:

```
#Definiendo el Dataframe
df=xl("FilmMarvelStudio[#Todo]", headers=True)
```

Fase II. Escalado de Datos: StandardScaler

Separando columnas de texto de las numéricas

Antes de escalar debemos tener en cuenta que todas las columnas de nuestro Dataframe deben ser numéricas, en caso de que tengamos dos tipos de columnas debemos convertir o transformar las de texto en numérica antes de escalar y para ello tenemos varias soluciones, una de ellas que es la que hemos elegido es separar el dataframe en dos una para las columnas con números y otro para las de texto.

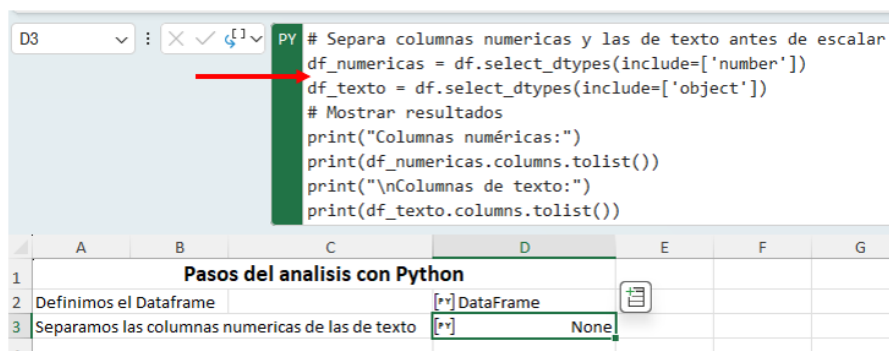


Ilustración 5

Así podemos proceder con el siguiente script

1. # Separa columnas numericas y las de texto antes de escalar
2. df_numericas = df.select_dtypes(include=['number'])
3. df_texto = df.select_dtypes(include=['object'])
4. # Mostrar resultados (esto es opcional)
5. print("Columnas numéricas:")
6. print(df_numericas.columns.tolist())
7. print("\nColumnas de texto:")
8. print(df_texto.columns.tolist())

Vemos por tanto que en la línea 2 y 3 se han creado dos dataframe una para cada tipo de valor, numero y texto. Señalar que a partir del punto 4 es opcional, simplemente para comprobar los resultados de la separacion.

Escalando las variables o columnas numéricas (df_numericas)

El paso a continuación es escalar sobre el dataframe que contiene los valores numéricos de las columnas (df_numericas). El siguiente script es el que nos va a permitir escalar los valores.

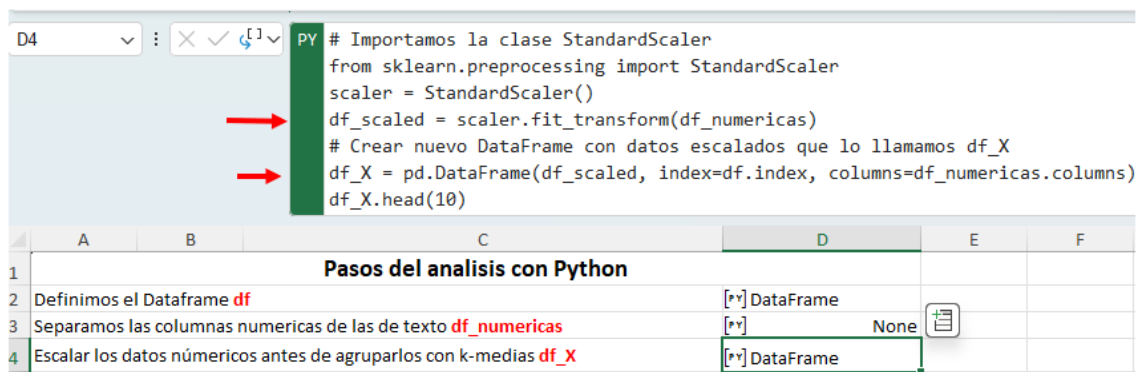


Ilustración 6

```
1. # Importamos la clase StandardScaler
2. from sklearn.preprocessing import StandardScaler
3. scaler = StandardScaler()
4. df_scaled = scaler.fit_transform(df_numericas)
5. # Crear nuevo DataFrame con datos escalados que lo llamamos df_X
6. df_X = pd.DataFrame(df_scaled, index=df.index, columns=df_numericas.columns)
7. df_X.head(10)
```

Explicacion del codigo de Ilustración 6

1. Importa la clase StandardScaler

```
from sklearn.preprocessing import StandardScaler
```

Importa la clase StandardScaler desde el módulo preprocessing de la biblioteca scikit-learn (abreviada como sklearn), que es muy utilizada para **machine learning** y **análisis de datos**.

Se importa la clase StandardScaler, que sirve para **escalar** (normalizar) los datos. Este escalado transforma los datos para que tengan **media 0 y desviación estándar 1**.

StandardScaler se usa para **escalar** (normalizar) los datos. Este escalado transforma los datos para que tengan **media 0 y desviación estándar 1**.

Esto se conoce como **estandarización** o **z-score normalization**.

2. Inicializamos el objeto StandardScaler, normalizador.

```
scaler = StandardScaler()
```

Se crea una instancia (variable) del escalador. Aún no se ha aplicado a los datos.

3. Aplica el escalador al dataframe a las columnas numericas y transforma los datos normalizandolos

```
df_scaled = scaler.fit_transform(df_numericas)
```

Aquí se aplica el escalador a un DataFrame llamado df_numericas, que contiene solo las columnas numéricas del DataFrame original.

fit_transform(df) hace dos cosas:

1. **Calcula** la media y desviación estándar de cada columna numérica del DataFrame df.
2. **Transforma** los datos usando la fórmula:

$z = \frac{x - \mu}{\sigma}$ donde	• x es el valor original, • μ es la media de la columna, • σ es la desviación estándar.
---	---

El resultado (df_scaled) es un **array de NumPy**, no un DataFrame

4. Crea un nuevo dataframe con los valores normalizados

```
df_X = pd.DataFrame(df_scaled, index=df.index, columns=df_numericas.columns)
```

Se convierte el array escalado anterior (df_scaled) en un nuevo DataFrame llamado df_X, manteniendo los mismos índices y nombres de columnas que el original df_numericas.

```
df_X.head(10)
```

Esto muestra las primeras 10 filas del DataFrame escalado.

df_X.head(n = 10), muestra las **primeras 10 filas** del DataFrame escalado.

	Ingresos	DiasUltimaCompra	Vin	Fruta	Carne	...	NºComprasEnTienda	NºVistasWebAlMes	Edad	TiempoDeVidaDelClienteEnDias	TotalComprasAlimentos
0	58138	58	635	88	546	...	4	7	63	2822	1529
1	46344	38	11	1	6	...	2	5	66	2272	21
2	71613	26	426	49	127	...	10	4	55	2471	734
3	26646	26	11	4	20	...	4	6	36	2298	48
4	58293	94	173	43	118	...	6	5	39	2320	407
...	...	...	...	...	...	...	...	...	...	...	...
2200	61223	46	709	43	182	...	4	5	53	2540	1094

Ilustración 7

Fase III. Creación de los Cluster, agrupar los datos con k-medias

```

PY # Importa el algoritmo KMeans desde la biblioteca scikit-learn
from sklearn.cluster import KMeans
# Procedemos a configurar el cluster para lo cual
# Instanciamos un objeto KMeans. El valor de k=3 fue elegido aplicando
# una combinación de la inercia y el coeficiente de silueta
k_means = KMeans(n_clusters = 3, n_init = 50, random_state = 54321)
# Realizamos el agrupamiento en los datos escalados
k_means.fit(df_X)
# Etiquetas de los cluster
k_means.labels_

```

	A	B	C	D	E	F
1	Pasos del analisis con Python					
2	Definimos el Dataframe df			[*] DataFrame		
3	Separamos las columnas numericas de las de texto df_numericas			[*] None		
4	Escalar los datos numéricos antes de agruparlos con k-medias df_X			[*] DataFrame		
5	Cluster Agrupar los datos con k-medias obtenemos k_means.labels_			[*] ndarray		
6						

Ilustración 8

Con este script vamos a crear los cluster, es decir agrupar los datos según su similitud aplicando el algoritmo K-Means (agrupamiento no supervisado) aplicandolo evidentemente a los datos normalizados anteriorente (df_X)

1. # Importa el algoritmo KMeans desde la biblioteca scikit-learn
2. from sklearn.cluster import KMeans
3. # Procedemos a configurar el cluster para lo cual
4. # Instanciamos un objeto KMeans. El valor de k=3 fue elegido aplicando
5. # una combinación de la inercia y el coeficiente de silueta
6. k_means = KMeans(n_clusters = 3, n_init = 50, random_state = 54321)
7. # Realizamos el agrupamiento en los datos escalados
8. k_means.fit(df_X)
9. # Etiquetas de los cluster
10. k_means.labels_

Básicamente, consiste en:

- a) Defines cuántos grupos queremos agrupar (3 por ejemplo).
- b) El modelo busca 3 centroides y asigna cada punto al grupo más cercano.

- c) Guarda las etiquetas (labels_) para saber a qué cluster pertenece cada observación.

Explicación del código de Ilustración 8

1. Importa el algoritmo KMeans

```
from sklearn.cluster import KMeans
```

Importa el algoritmo KMeans desde la biblioteca scikit-learn, que es muy usada para machine learning en Python.

2. Configuramos el modelo de cluster (grupos) como los deseamos

```
# Procedemos a configurar el cluster para lo cual
# Instanciamos un objeto KMeans. El valor de k=3 fue elegido aplicando
# una combinación de la inercia y el coeficiente de silueta
k_means = KMeans(n_clusters = 3, n_init = 50, random_state = 54321)
```

Aquí se configura el modelo. Crea el modelo de agrupamiento definiéndolo con el nombre de k_means y con la siguientes configuración:

- n_clusters=3: indica que queremos **3 grupos**. Este valor se suele elegir analizando métricas como *inercia* (distancia interna de los puntos al centroide- qué tan compactos son los grupos), y el **coeficiente de silueta** (qué tan separados y consistentes están los grupos).
- n_init=50: el algoritmo se ejecutará **50 veces con diferentes centroides iniciales** y elegirá el mejor resultado (esto mejora la estabilidad del modelo).
- random_state=54321: fija la semilla aleatoria para que los resultados sean **reproducibles**.

3. Aplicamos la configuración del cluster al dataframe escalado

```
# Realizamos el agrupamiento en los datos escalados
k_means.fit(df_X)
```

- fit(df_X): aplica el algoritmo K-Means sobre los datos contenidos en df_X, que deben estar escalados (normalizados) para que todas las variables tengan el mismo peso. Es decir, fit() busca los **centroides de cada cluster** (los puntos que representan a cada grupo) y asigna cada observación al cluster más cercano.
- df_X: es nuestro DataFrame con las características que estás agrupando.

4. Obtenemos las etiquetas de los cluster

```
# Etiquetas de los cluster
k_means.labels_
```

labels_: devuelve un array con la etiqueta de cluster asignada a cada fila de df_X. Por ejemplo, si tenemos 100 filas, obtendremos 100 etiquetas (0, 1 o 2 en este caso).

Fase IV Añadir columna cluster al Dataframe

D6	:	X	✓	123	PY	## Agregar una nueva columna al DataFrame original(df) # o inicial con las asignaciones de clúster (el array) df['Cluster'] = k_means.labels_
	A	B	C	D		
1	Pasos del análisis con Python					
2	Definimos el Dataframe df				[*] DataFrame	
3	Separamos las columnas numericas de las de texto df_numericas				[*] None	
4	Escalar los datos numéricos antes de agruparlos con k-medias df_X				[*] DataFrame	
5	Cluster Agrupar los datos con k-medias obtenemos k_means.labels_				[*] ndarray	
6	Fusionar las etiquetas de los clústeres al dataframe df['Cluster'] = k_means.labels_					None

Ilustración 9

```
df['Cluster'] = k_means.labels_
```

Esa línea de código en Python añade una nueva columna llamada 'Cluster' al DataFrame inicial df, asignando a cada fila el número de clúster correspondiente según el modelo k_means.

De esta forma ya contamos con la información necesaria para crear el análisis

Fase V Análisis de los Clústeres Resultantes

Finalmente disponemos del dataframe con el cluster tal y como se muestra en la siguiente ilustración

D7

:

✓

↺

PY

#Enviamos nuestro dataframe (df) modificado

con el cluster asignado a cada registro a la hoja de calculo

df

	A	B	C	D	E
1	Pasos del analisis con Python				
2	Definimos el Dataframe df			[*] DataFrame	
3	Separamos las columnas numericas de las de texto df_numericas			[*] None	
4	Escalar los datos numéricos antes de agruparlos con k-medias df_X			[*] DataFrame	
5	Cluster Agrupar los datos con k-medias obtenemos k_means.labels_			[*] ndarray	
6	Fusionar las etiquetas de los clústeres al dataframe df['Cluster'] = k_means.labels_				None
7	Generar el DataFrame con Cluster df			[*] DataFrame	
8					

Ilustración 10

Si lo desplegamos tendremos todos los registros de la tabla inicial junto con el cluester asignado pudiendo basar nuestro analisis con tablas y graficos dinámicos sobre este dataframe ampliado.

	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	[*] DataFrame																			
2	[*] None																			
3	[*] DataFrame																			
4	[*] ndarray																			
5	None																			
6																				
7	Film	categoria	recaudación % del presupuesto	puntuación	desviación	presupuesto	recaudación	recaudación primer fin	segundo fin	Caida entre e % de la reca	% de la reca	% de la reca	% del presupuesto	Año	Fuente	Cluster				
8	Ant-Man: El hombre hor	Ant-Man	518	3,9846154	0,83	0,85	-0,02	130	180	338	57	24	-0,5789474	0,318	0,3474903	0,6525097	0,4384615	2015	https://www	0
9	Ant-Man y la Avispa	Ant-Man	623	4,7923077	0,87	0,8	0,07	130	216	406	75,8	29	-0,6174142	0,35	0,3467095	0,6516854	0,5830769	2018	https://www	0
10	Vengadores: La era de U Vengadores	Vengadores	1395	3,8219178	0,76	0,82	-0,06	365	459	936	191	77	-0,5968586	0,417	0,3290323	0,6709677	0,5232877	2015	https://www	0
11	Vengadores: End game	Vengadores	2797	6,9925	0,94	0,9	0,04	400	858	1939	357	147	-0,5882353	0,416	0,3067572	0,6932428	0,8925	2019	https://www	1
12	Vengadores: Infinity Wa Vengadores	Vengadores	2048	6,8266667	0,85	0,91	-0,06	300	678	1369	257	114	-0,5564202	0,38	0,3310547	0,668457	0,8566667	2018	https://www	1
13	Black Panther: Pantera I Black Panthe	Black Panther	1336	6,68	0,96	0,79	0,17	200	700	636	202	111	-0,450495	0,289	0,5239521	0,4760479	1,01	2018	https://www	1
14	Black Panther: Wakandi Black Panthe	Black Panther	855	3,42	0,84	0,94	-0,1	250	453	401	181	66	-0,6353591	0,486	0,5298246	0,4690058	0,724	2022	https://www	2
15	Viuda Negra	Única	379	1,895	0,79	0,8	-0,01	200	183	196	80,3	25,8	-0,6787049	0,438	0,4828496	0,5171504	0,4015	2021	https://www	2

Ilustración 11

Como ejemplo presentamos el siguiente analisis basado en tablas dinámicas.

	A	B	C	D	E	F
1						
2		Cluster	Promedio recaudación (M\$)	Promedio puntuación de los críticos (%)	Promedio puntuación del público (%)	Cuenta de film
3		0	867,53	84,53%	82,27%	15
4		1	1.921,40	91,80%	89,40%	5
5		2	560,60	75,00%	78,60%	10
6		Total general	940,87	82,57%	82,23%	30
7						
8		categoria Ant-Man Black Panther Capitán América Doctor Strange Guardianes Iron Man Los Vengadores Spider-Man Thor Única Vengadores				
9						
10						
11						
12						
13						
14						

Ilustración 12