Python con Excel II Análisis Cluster, Caso Supermercado

Introducción al Análisis Cluster con Python en Excel, uso de las principales librerías. Marco conceptual básico. Perfil de los clientes del supermercado. Está dirigido a profesionales que buscan ir más allá de las capacidades estándar de Excel.

Jose Ignacio González Gómez
Departamento de Economía, Contabilidad y Finanzas - Universidad de La Laguna
www.jggomez.eu
V. 2.5
Ejercicio adaptado de:

Contenido

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

Asi **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. Asi 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 numero 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, asi 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 asi tenemos la tabla ya preparada para el analisis 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 David Langer, perfil del cliente, analisis cluster

Fuente: Finally! Powerful Cluster Analysis with Microsoft Excel is HERE!

Presentacion

Objetivo

En este caso vamos explorar la aplicación de **análisis de clústeres** utilizando **Python en Excel**, destacando la importancia de que los datos sean **numéricos** para un clustering efectivo. El contenido se centra en un **conjunto de datos de comportamiento de clientes de supermercados**, donde cada fila representa un cliente y las columnas contienen **mediciones numéricas** de su comportamiento de compra.

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. Un paso crucial es **escalar los datos numéricos** con un StandardScaler para asegurar que todas las variables estén en una escala comparable, lo cual es esencial para el **algoritmo K-means**, una técnica popular de clustering. Después de aplicar K-means para agrupar clientes en clústeres (en este caso, tres), las **asignaciones de clústeres** se reincorporan al dataframe original.

Finalmente, se detalla **métodos para analizar los clústeres resultantes**, comenzando con una **tabla dinámica** en Excel para examinar promedios de variables clave por clúster (como edad o ingresos). Una técnica más efectiva para la **exploración de datos** es el uso de **visualizaciones de datos**, lo que permite una comprensión más profunda de las distribuciones dentro de cada clúster. Como conclusión se señala que la **combinación de análisis de clústeres con aprendizaje automático** es la mejor forma de análisis.

Información disponible

Vamos a trabajar con un fichero correspondiente a un informe del CRM de una empresa de alimentación que contiene unos registros básicos de una muestra de clientes relacionadas con su comportamiento de compra (2205 registros).

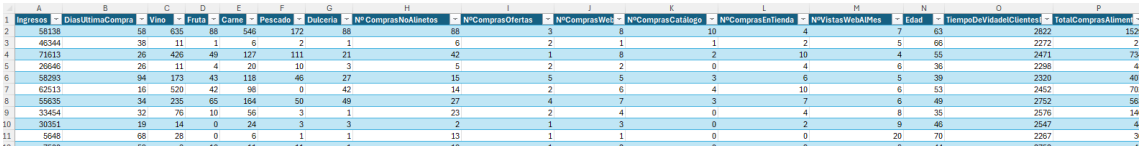


Ilustración 1

Los campos o columnas disponibles son 16 y que se corresponden a:

	A	B	C	D	E	F	G	H	I
1	Ingresos	DiasUltimaCompra	Vino	Fruta	Carne	Pescado	Dulceria	Nº ComprasNoAlinetos	NºComprasOfertas
2	58138	58	635	88	546	172	88	88	3
3	46344	38	11	1	6	2	1	6	2
4	71613	26	426	49	127	111	21	42	1
5	26646	26	11	4	20	10	3	5	2

	J	K	L	M	N	O	P
1	NºComprasWeb	NºComprasCatálogo	NºComprasEnTienda	NºVistasWebAlMes	Edad	TiempoDeVidadelClientes	TotalComprasAliment
2	8	10	4	7	63	2822	1529
3	1	1	2	5	66	2272	21
4	8	2	10	4	55	2471	734
5	2	0	4	6	36	2298	48

Ilustración 2

1	Ingresos	Total ingresos	9	NºComprasOfertas	Nº de Compras de Ofertas
2	DiasUltimaCompra	Dias transcurridos desde ultima compra	10	NºComprasWeb	Nº de Compras en Web
3	Vino	Compra vino (1)	11	NºComprasCatálogo	Nº de Compras de Catálogo
4	Fruta	Compra fruta (1)	12	NºComprasEnTienda	Nº de Compras en Tienda
5	Carne	Compra Carne (1)	13	NºVistasWebAlMes	Nº de Visitas Web al Mes
6	Pescado	Compra Pescado (1)	14	Edad	Edad
7	Dulcería	Compra Dulcería (1)	15	TiempoDeVidadelClientesEnDias	Tiempo de alta de clientes en dias
8	Nº ComprasNoAlimentos	Nº de Compras de No Alimentos	16	TotalComprasAlimentos	Total de compras de alimentos (3+4+5+6+7)

(1) Hace referencia al nº de veces que se compra

Tabla 1

En este caso, todos los valores disponibles son numéricos.

Queremos realizar un análisis de cluster para agrupar a los clientes 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.

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

Análisis con Python para Excel

Fase I. Cargar Datos. Crear el Dataframe

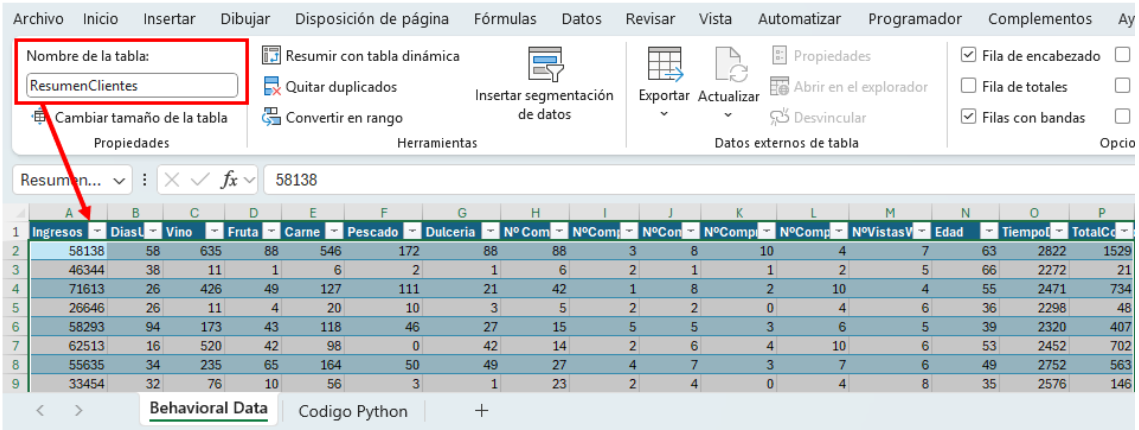


Ilustración 3

Como podemos observar de la Ilustración 3 disponemos de 2 hojas, la primera con el nombre de la pestaña Behavioral Data contiene la tabla de datos con el nombre “ResumenClientes”.

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

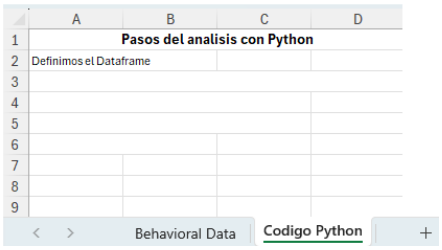


Ilustración 4

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 encabezaos.

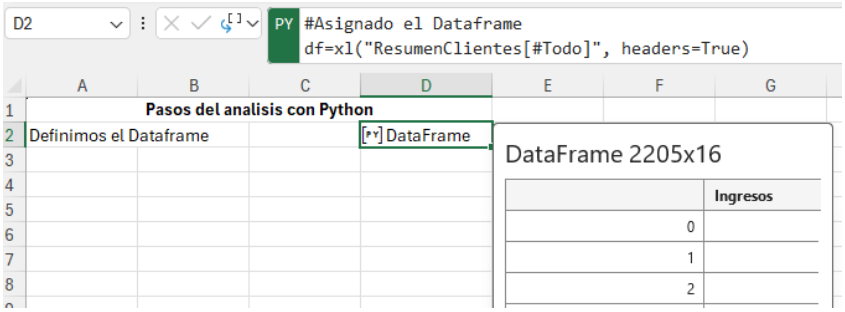


Ilustración 5

El script Python:

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

Fase II. Escalado de Datos: StandardScaler

El siguiente script nos va a permitir escalar los datos, en concreto estandarizar los datos de un DataFrame, en nuestro caso el nuestro que hemos llamado como df usando StandardScaler de scikit-learn.

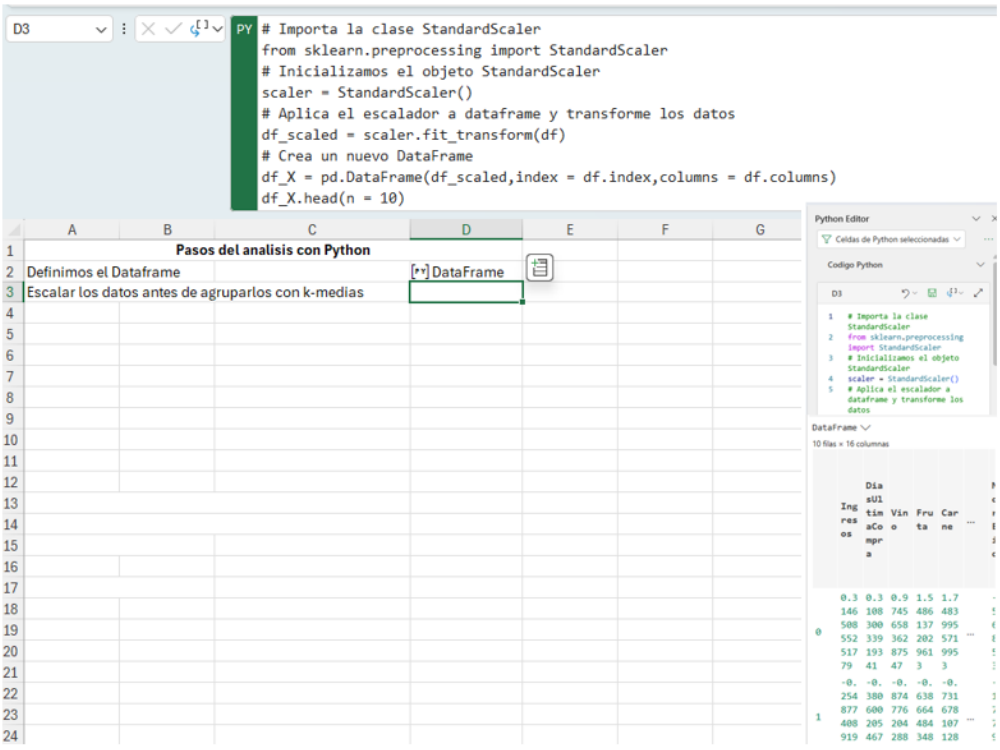


Ilustración 6

```
# Importa la clase StandardScaler
```

```
from sklearn.preprocessing import StandardScaler
# Inicializamos el objeto StandardScaler
scaler = StandardScaler()
# Aplica el escalador a dataframe y transforme los datos
df_scaled = scaler.fit_transform(df)
# Crea un nuevo DataFrame
df_X = pd.DataFrame(df_scaled,index = df.index,columns = df.columns)
df_X.head(n = 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

```
# Inicializamos el objeto StandardScaler
scaler = StandardScaler()
```

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

3. Aplica el escalador al dataframe y transforma los datos

```
# Aplica el escalador al dataframe y transforma los datos
df_scaled = scaler.fit_transform(df)
```

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

```
# Crea un nuevo DataFrame
df_X = pd.DataFrame(df_scaled, index=df.index, columns=df.columns)
df_X.head(n = 10)
```

Se convierte el array escalado de nuevo en un DataFrame de pandas, con los mismos índices y nombres de columnas que el original.

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

DataFrame
2205 filas × 16 columnas

	Ingresos	DiasUltimaCompra	Vino	Fruta	Carne	...	NºComprasEnTienda	NºVistasWebAlMes	Edad	TiempoDeVidadelClientesEnDias	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

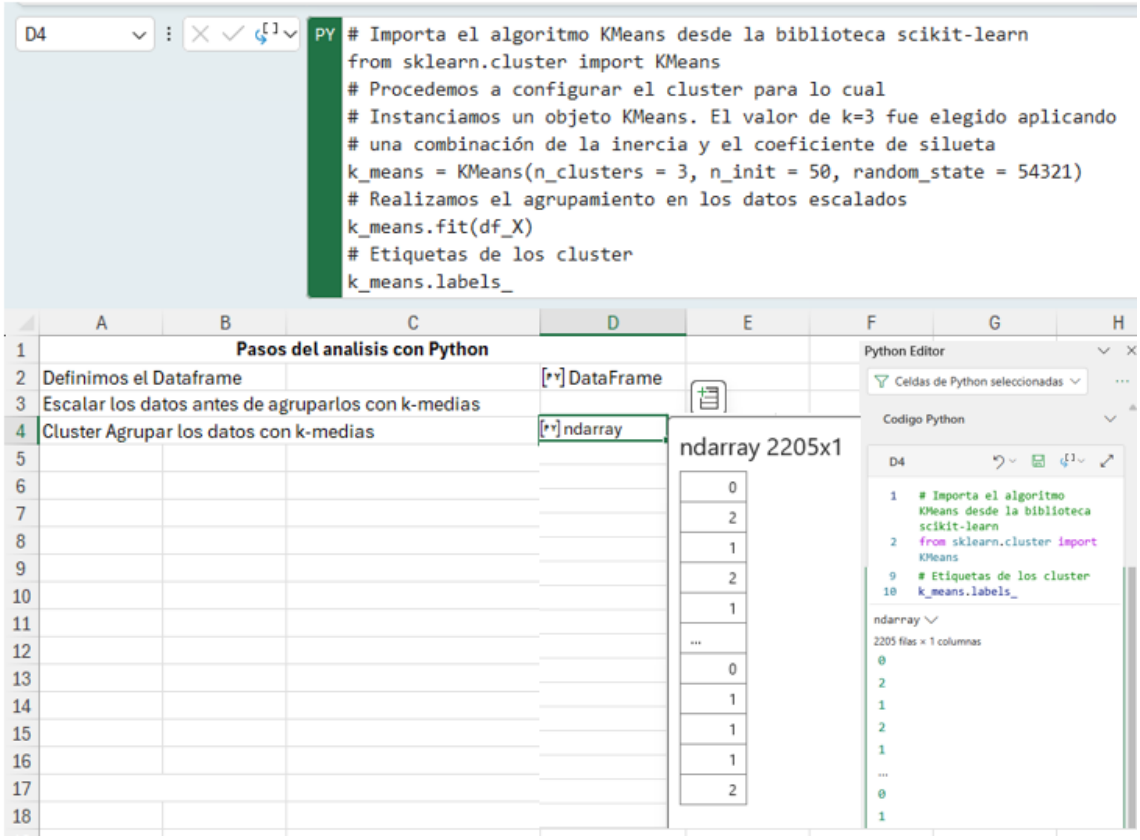


Ilustración 8

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

```
# 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_
```

Básicamente, consiste en:

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

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

Explicacion del codigo 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 tenemos100 filas, obtendremos 100 etiquetas (0, 1 o 2 en este caso).

Fase IV Añadir columna cluster al Dataframe

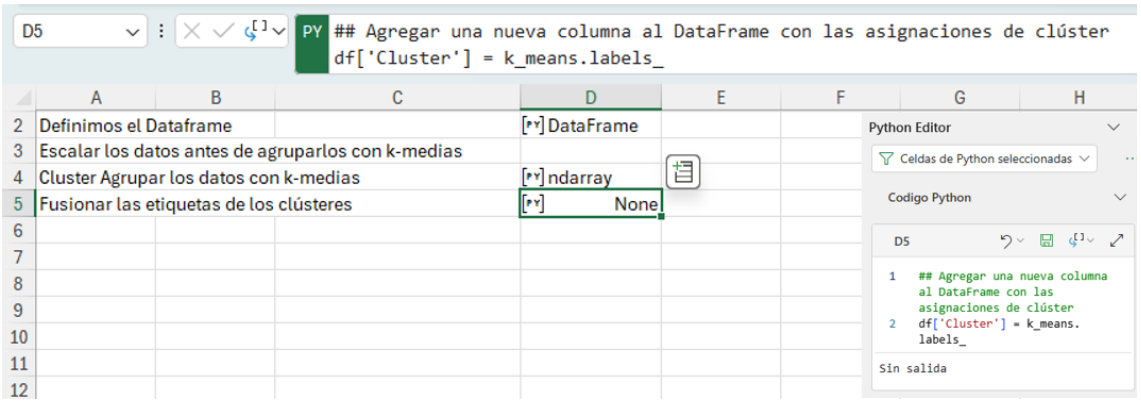


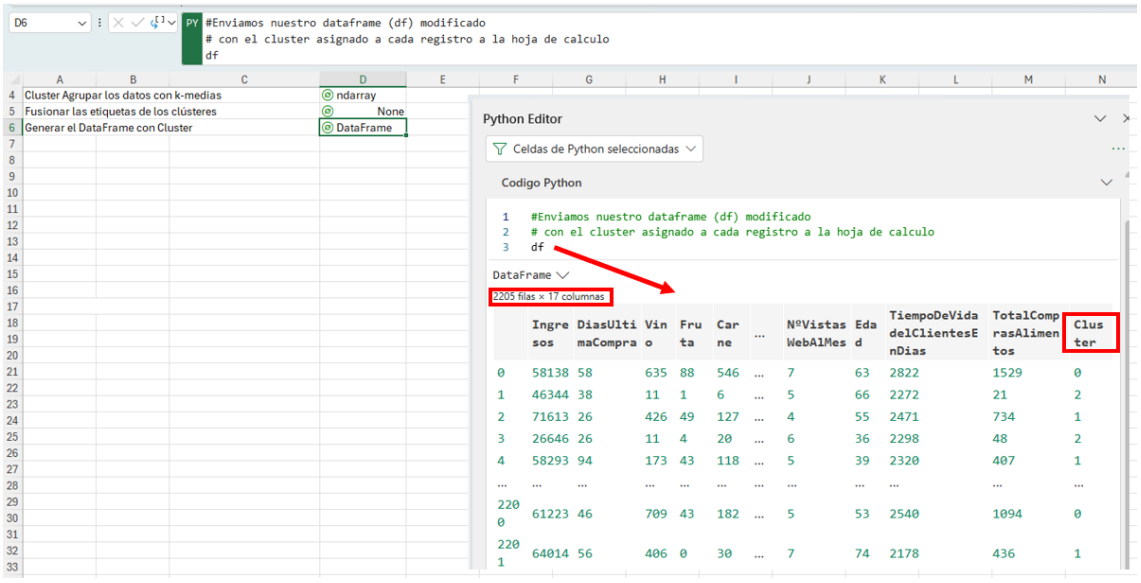
Ilustración 9

```
## Agregar una nueva columna al DataFrame con las asignaciones de clúster
df['Cluster'] = k_means.labels_
```

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

Fase V Análisis de los Clústeres Resultantes

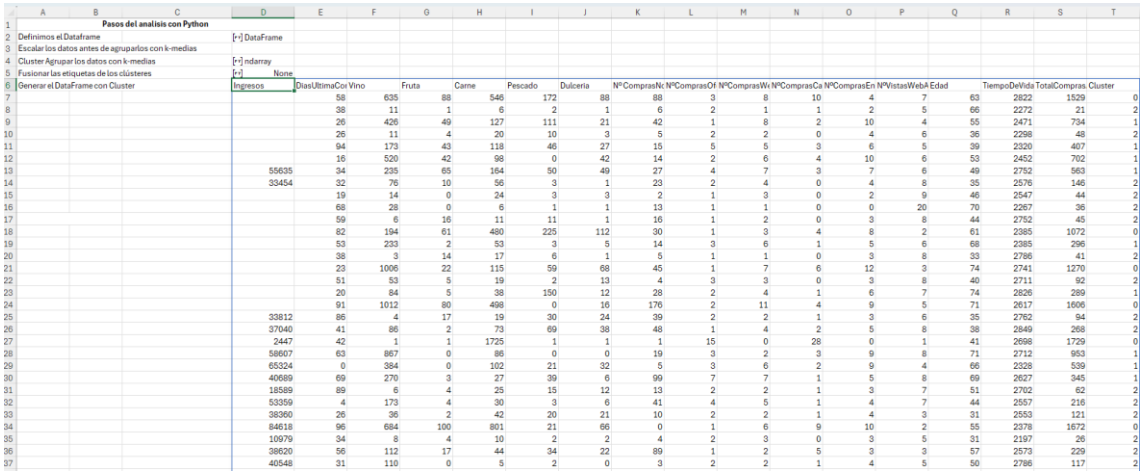
Finalmente disponemos del dataframe con el cluster tal y como se muestra en la siguiente ilustración



	Ingresos	DiasUltimaCompra	Vino	Fruta	Carne	...	NºVistasWebAlMes	Edad	TiempoDeVidadelClientesEnDias	TotalComprasAlimentos	Cluster
0	58138	58	635	88	546	...	7	63	2822	1529	0
1	46344	38	11	1	6	...	5	66	2272	21	2
2	71613	26	426	49	127	...	4	55	2471	734	1
3	26646	26	11	4	20	...	6	36	2298	48	2
4	58293	94	173	43	118	...	5	39	2320	407	1
...	...	...	...	...	...	...	...	...	...	...	...
2200	61223	46	709	43	182	...	5	53	2540	1094	0
2201	64014	56	406	0	30	...	7	74	2178	436	1

Ilustración 10

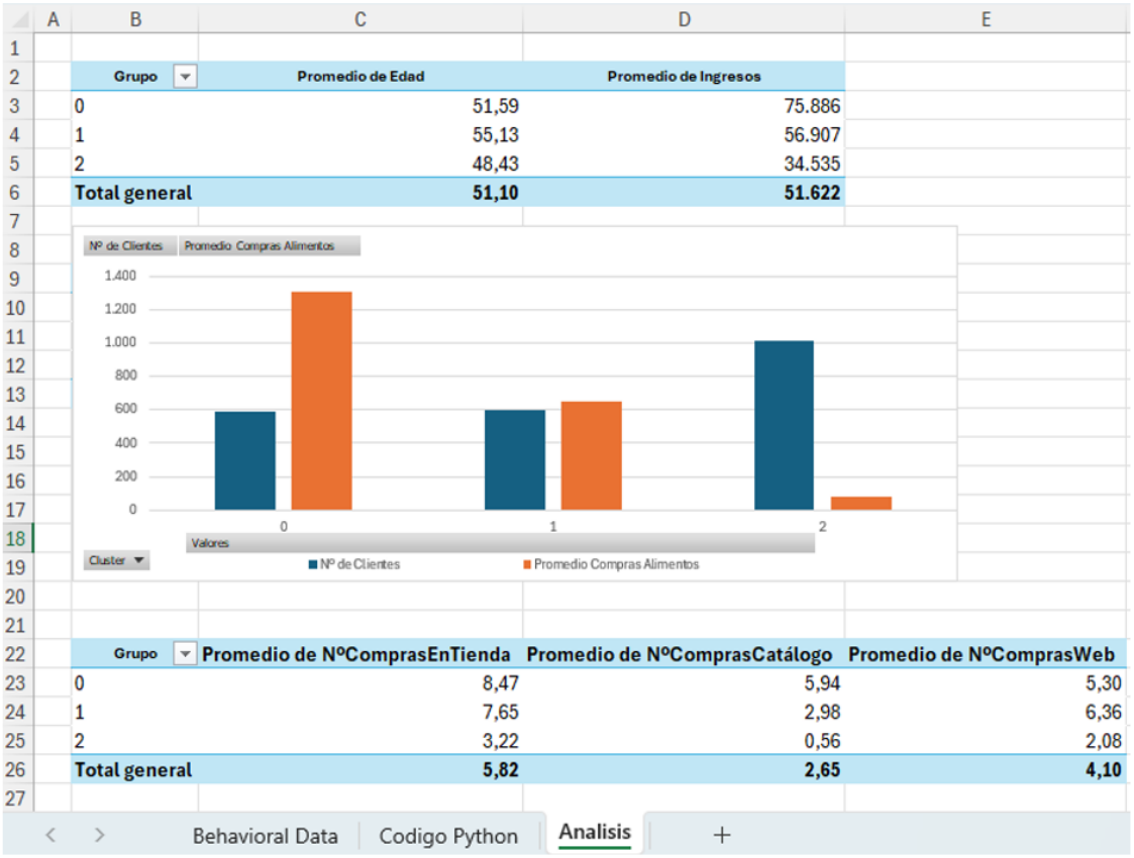
Si lo desplegamos disponemos de la siguiente tabla generada y sobre la cual podemos basar tablas dinámicas, graficos teniendo disponible como novedad el cluster asignado a cada registro o cliente.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	Pasos del analisis con Python																			
2	Definimos el Dataframe			[+] DataFrame																
3	Escalar los datos antes de agruparlos con k-medias																			
4	Cluster Agrupar los datos con k-medias			[+] ndarray																
5	Fusionar las etiquetas de los clústeres			[+] None																
6	Generar el DataFrame con Cluster			Ingresos	DiasUltimaCor	Vino	Fruta	Carne	Pescado	Dulceria	NºComprasNo	NºComprasOf	NºComprasWe	NºComprasCa	NºComprasEn	NºVistasWeb	Edad	TiempoDeVida	TotalCompras	Cluster
7					58	635	88	546	172	88	88	3	8	10	4	7	63	2822	1529	0
8					38	11	1	6	2	1	6	2	1	1	2	5	66	2272	21	2
9					26	426	49	127	111	21	42	1	8	2	10	4	55	2471	734	1
10					26	11	4	20	10	3	5	2	2	0	4	6	36	2298	48	2
11					94	173	43	118	46	27	15	5	5	3	6	5	39	2320	407	1
12					16	520	42	98	0	42	14	2	6	4	10	6	53	2452	702	1
13				55635	34	235	65	164	50	49	27	4	7	3	7	6	49	2752	563	1
14				33454	32	76	10	56	3	1	23	2	4	0	4	8	35	2576	146	2
15					19	14	0	24	3	3	2	1	3	0	2	9	46	2547	44	2
16					68	28	0	6	1	1	13	1	1	0	0	20	70	2267	36	2
17					59	6	16	11	11	1	16	1	2	0	3	8	44	2752	45	2
18					82	194	61	480	225	112	30	1	3	4	8	2	61	2385	1072	0
19					53	233	2	53	3	5	14	3	6	1	5	6	68	2385	296	1
20					38	3	14	17	6	1	5	1	1	0	3	8	33	2786	41	2
21					23	1006	22	115	59	68	45	1	7	6	12	3	74	2741	1270	0
22					51	53	5	19	2	13	4	3	3	0	3	8	40	2711	92	2
23					20	84	5	38	150	12	28	2	4	1	6	7	74	2826	289	1
24					91	1012	80	498	0	16	176	2	11	4	9	5	71	2617	1606	0
25				33812	86	4	17	19	30	24	39	2	2	1	3	6	35	2762	94	2
26				37040	41	86	2	73	69	38	48	1	4	2	5	8	38	2849	268	2
27				2447	42	1	1	1725	1	1	1	15	0	28	0	1	41	2698	1729	0
28				58607	63	867	0	86	0	0	19	3	2	3	9	8	71	2712	953	1
29				65324	0	384	0	102	21	32	5	3	6	2	9	4	66	2328	539	1
30				40689	69	270	3	27	39	6	99	7	7	1	5	8	69	2627	345	1
31				18589	89	6	4	25	15	12	13	2	2	1	3	7	51	2702	62	2
32				53359	4	173	4	30	3	6	41	4	5	1	4	7	44	2557	216	2
33				38360	26	36	2	42	20	21	10	2	2	1	4	3	31	2553	121	2
34				84618	96	684	100	801	21	66	0	1	6	9	10	2	55	2378	1672	0
35				10979	34	8	4	10	2	2	4	2	3	0	3	5	31	2197	26	2
36				38620	56	112	17	44	34	22	89	1	2	5	3	3	57	2573	229	2
37				40548	31	110	0	5	2	0	3	2	2	1	4	5	50	2786	117	2

Ilustración 11

Como ejemplo presentamos el siguiente analisis basado en tablas dinámicas.



Grupo	Promedio de Edad	Promedio de Ingresos
0	51,59	75.886
1	55,13	56.907
2	48,43	34.535
Total general	**51,10**	**51.622**

Grupo	Promedio de NºComprasEnTienda	Promedio de NºComprasCatálogo	Promedio de NºComprasWeb
0	8,47	5,94	5,30
1	7,65	2,98	6,36
2	3,22	0,56	2,08
Total general	**5,82**	**2,65**	**4,10**

Ilustración 12