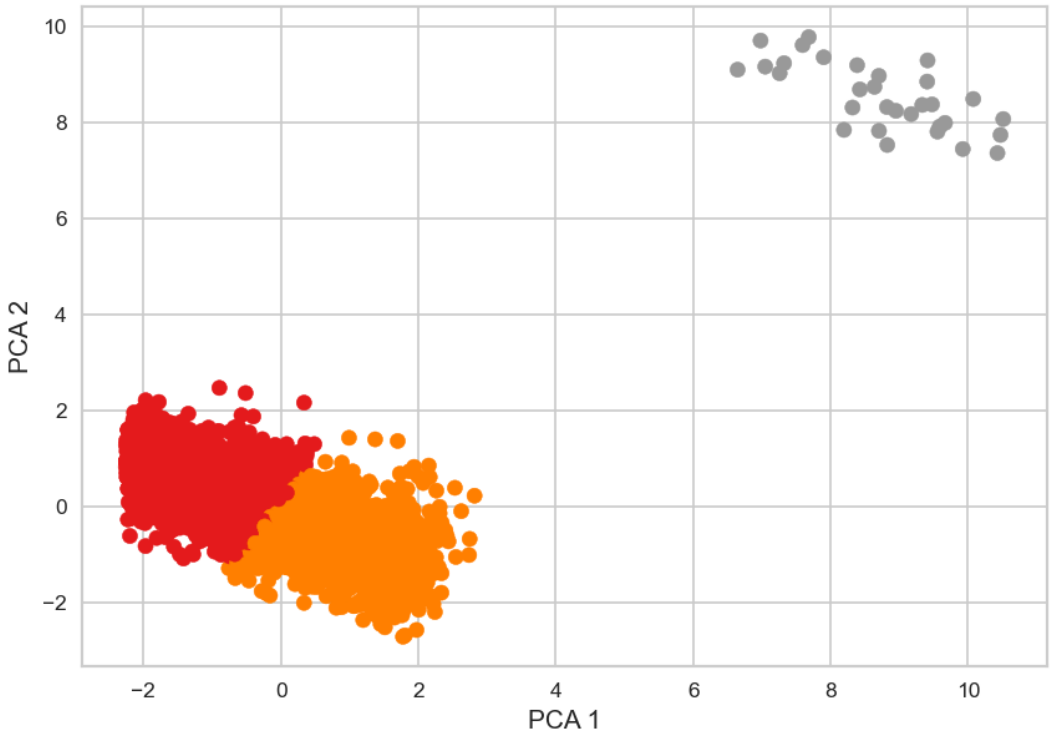
Manual de documentación:

“Clusterización de clientes para un eCommerce”



Guillermo Alvarez Bacame

ga.bacame@gmail.com

Hermosillo, Sonora, 19 febrero 2023

1. INTRODUCCIÓN

La *clusterización* de clientes es una técnica fundamental para cualquier negocio que quiera optimizar sus estrategias de marketing, ofrecer productos y servicios personalizados y mejorar la satisfacción del cliente. Al segmentar a los clientes en diferentes grupos basados en características comunes, las empresas pueden entender mejor las necesidades y preferencias de cada grupo, lo que les permite adaptar sus ofertas y mensajes de marketing a cada segmento de manera efectiva. Esto puede mejorar la eficiencia y efectividad del marketing, reducir costos y aumentar la lealtad del cliente. Además, la clusterización también puede ayudar a las empresas a descubrir nuevos patrones de comportamiento de los clientes y oportunidades de crecimiento que de otra manera no se habrían detectado. En el caso específico del e-commerce, la clusterización de los clientes puede ser especialmente importante debido al gran volumen de datos que se generan y a la gran cantidad de clientes que pueden tener, lo que hace que la segmentación manual sea difícil o imposible.

2. METODOLOGÍA

En este programa se generó un conjunto de datos de compras de clientes a partir de un catálogo de productos. Luego, se aplicó un modelo de clustering con el algoritmo *K-means* para segmentar el mercado en función del valor total de las compras, el número total de compras y las categorías de los productos comprados. Se utilizó la técnica de reducción de dimensionalidad *PCA* para visualizar los datos en dos dimensiones y se graficaron los clusters en diferentes colores. Además, se utilizó el *Elbow method* para determinar el número óptimo de clusters.

1 Generación de datos

Se hizo *Web Scrapping* a la página web de *ClaroShop*® para poder extraer parte de catálogo, con el fin de recopilar datos de productos, se extrajeron los datos de 3829 productos, cuyas columnas incluyen: Nombre, Proveedor, Precio, Descuento y Categoría. Las categorías los productos son las siguientes: Animales, Bebes, Celulares, Deportes, Electrónicos, Ferretería, Hogar, Joyería, Juguetes, Libros, Ropa, Salud, Videojuegos. Esto contenido en el archivo “productos.csv”.

Con el siguiente código lo que se hizo fue asignarle un ID único a cada producto, se eliminaron el resto de las secciones del dataset y solo se dejaron las secciones de: ID_Prod, Precio y Categoria. Estos fueron guardados en el documento “productos_seleccionados.csv”.

```
import pandas as pd
import uuid

# Cargar el archivo CSV en un DataFrame
df = pd.read_csv('productos.csv')

# Agregar una nueva columna con un ID único para cada fila
df['ID_Prod'] = [str(uuid.uuid4()) for _ in range(len(df))]

# Seleccionar solo las columnas que se quieren mostrar
df2 = df[['ID_Prod', 'Precio', 'Categoria']]

# Guardar el nuevo DataFrame en un archivo CSV
df2.to_csv('productos_seleccionados.csv', index=False)
```

Con el uso de la librería *Faker* de Python, se generó un dataset de 3000 clientes, con los siguientes datos: id, edad y sexo. Guardados en el archivo “clientes.csv”.

```python
from faker import Faker
import csv

# Crear una instancia de Faker
fake = Faker()

# Generar datos ficticios de clientes con identificadores únicos
clientes = []
for _ in range(3000):
    id_unico = fake.uuid4()
    edad = fake.random_int(min=18, max=70)
    sexo = fake.random_element(elements=('M', 'F'))
    clientes.append({"id": id_unico, "edad": edad, "sexo": sexo})

# Crear un archivo CSV
with open('clientes.csv', 'w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(["ID", "Edad", "Sexo"])
    for cliente in clientes:
        writer.writerow([cliente["id"], cliente["edad"], cliente["sexo"]])
```

A continuación, se simularon compras de estos clientes (clientes.csv) de los productos (productos_seleccionados.csv) y se generó un nuevo dataset llamado “compras.csv”, en el cual:

- ID_Cliente: El ID del cliente que realizó la compra.
- Val_compras: El costo total de las compras realizadas por el cliente.
- Num_compras: El número total de compras realizadas por el cliente.
- Categorias: Una lista separada por comas de las categorías de los productos comprados por el cliente.

Cada cliente hace un numero aleatorio de compras entre el rango de 1 a 104, esto con el fin de simular todas las compras realizadas por un cliente a lo largo del año, siendo el mínimo una y el máximo dos compras por semana (valor tomado de forma arbitraria).

```python
import pandas as pd
import numpy as np

# Cargar los archivos CSV con los datos de clientes y productos
df_clientes = pd.read_csv('clientes.csv')
df_productos = pd.read_csv('productos_seleccionados.csv')

# Definir el número máximo y mínimo de compras que se generarán para cada cliente
num_compras_min = 1
num_compras_max = 104

# Inicializar una lista para almacenar las compras
compras = []
```

```
# Generar compras aleatorias para cada cliente
for i, row in df_clientes.iterrows():
    num_compras = np.random.randint(num_compras_min, num_compras_max + 1)
    val_compras = 0
    categorias = []
    for j in range(num_compras):
        # Seleccionar productos aleatorios del catálogo de productos
        producto = df_productos.sample().iloc[0]

        # Eliminar todos los caracteres no numéricos del precio del producto
        precio_limpio = ''.join(filter(str.isdigit, producto['Precio']))

        # Convertir el precio a un número entero y sumarlo al valor total de las
compras
        val_compras += int(precio_limpio)

        # Agregar la categoría del producto a la lista de categorías
        categorias.append(producto['Categoria'])

    # Agregar una nueva fila al DataFrame de compras
    compras.append({
        'ID_Cliente': row['ID'],
        'Val_compras': val_compras,
        'Num_compras': num_compras,
        'Categorias': ','.join(categorias)
    })

# Crear un DataFrame a partir de la lista de compras
df_compras = pd.DataFrame(compras)

# Guardar el DataFrame en un nuevo archivo CSV
df_compras.to_csv('compras.csv', index=False)
```

2 Clusterización

Una vez teniendo el dataset que me muestra, el gasto de los clientes en compras, la cantidad de veces que compraron y las categorías en las que lo hicieron, ya podemos buscar una correlación entre estos datos para segmentarlos.

Primero se hizo una reducción de dimensionalidad, esto utilizando *PCA* (Principal Components Analysis), esto para poder visualizar los datos en un plano a futuro. Después buscamos el numero optimo de clusters o segmentos de nuestros datos, esto usando el *Elbow method* o método del codo en español. Ya con esto podemos entrenar el modelo, en este caso se opto por el algoritmo *K-Means*, después se evaluó el modelo para ver que tan bien esta separando a los clientes. Finalmente se visualizaron los resultados de la clusterización mediante un scatter plot, asignándole colores distintos a cada cluster.

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from yellowbrick.cluster import KElbowVisualizer

# Cargar datos
compras = pd.read_csv('compras.csv')

# Seleccionar variables de interés
X = compras[['Val_compras', 'Num_compras', 'Categorias']]

# Transformar la variable categórica a numérica con one-hot encoding
X = pd.get_dummies(X)

# Normalizar datos
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Reducir dimensionalidad con PCA
pca = PCA()
pipeline = make_pipeline(scaler, pca)
pipeline.fit(X_scaled)
explained_variances = pca.explained_variance_ratio_
plt.plot(np.cumsum(explained_variances))
plt.xlabel('Número de componentes')
plt.ylabel('Varianza explicada acumulada')
plt.show()

# Elegir número de componentes
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

# Encontrar número óptimo de clusters con el método del codo
model = KMeans()
visualizer = KElbowVisualizer(model, k=(1,10))
visualizer.fit(X_pca)
visualizer.show()

# Entrenar modelo con número óptimo de clusters
k = visualizer.elbow_value_
model = KMeans(n_clusters=k)
```

```
model.fit(X_pca)

# Graficar resultados
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=model.labels_, cmap='Set1')
plt.xlabel('PCA 1')
plt.ylabel('PCA 2')
plt.show()
```

En general, la metodología se enfoca en obtener una segmentación de los clientes en grupos homogéneos en función de sus comportamientos de compra y preferencias de categorías.

3. RESULTADOS Y CONCLUSIONES

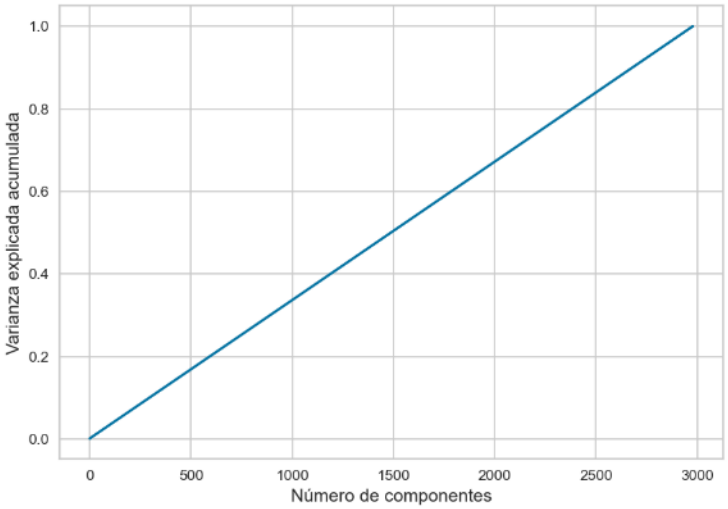


Fig 1. Varianza explicada acumulada vs Número de componentes

Como se puede observar en la gráfica, esta es una función lineal, lo que significa que la mayoría de la varianza puede ser explicada en un numero reducido de componente principales. En este caso, utilizador dos componentes puede ser suficiente para representar la mayoría de los datos, a su vez siendo practico para su representación en un grafico de dos dimensiones.

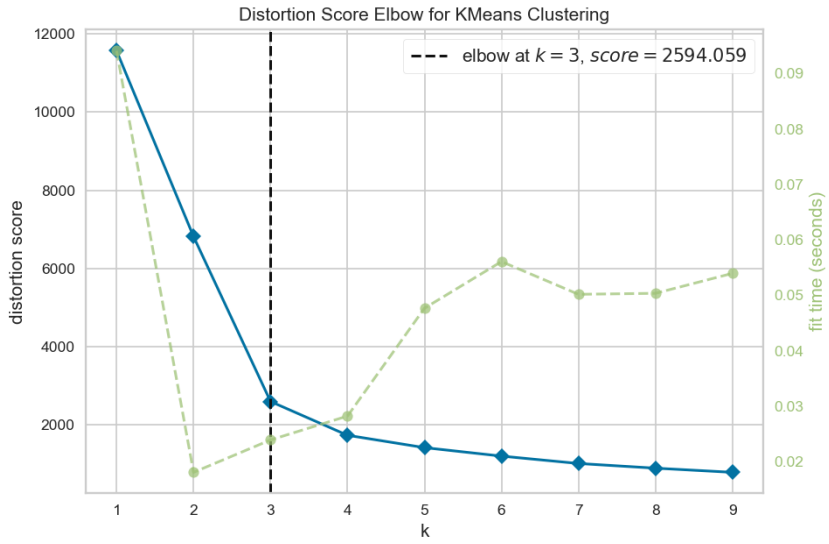


Fig 2. Distorsión de puntuación de codo

El gráfico muestra la puntuación de distorsión en el eje y y el número de clusters en el eje x. Al aumentar el número de clusters, la puntuación de distorsión disminuye. El gráfico del codo obtiene su nombre porque la curva a menudo tiene una forma de codo. El codo representa el punto en el gráfico donde la tasa de disminución de la puntuación de distorsión comienza a disminuir. En otras palabras, el codo es el punto donde se obtiene el mayor beneficio de agregar un cluster adicional. Por lo tanto, el codo es una forma de determinar el número óptimo de clusters para la clusterización K-means, siendo en este caso el 3.

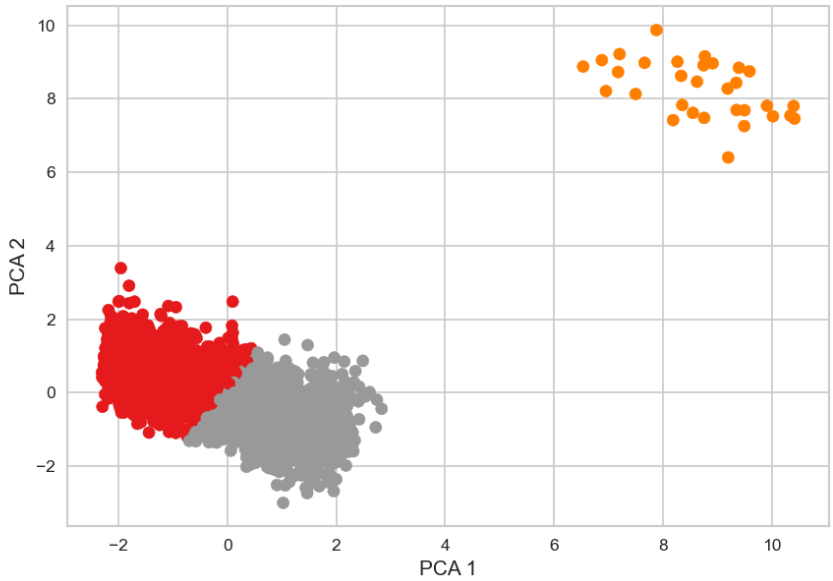


Fig 3. Clusterización

El siguiente diagrama de dispersión representa la segmentación de los clientes en función de su comportamiento de compra y categorías en las cuales compran, cada color representa cada uno de los segmentos de mercado por separado.

En este caso específico es evidente el traslape del cluster rojo con el gris, esto podría ser debido a que se necesitan más variables relevantes, recordemos que este dataset fue generado de forma artificial, siendo este no representativo en el comportamiento, debido a que este esta basado en un modelo totalmente aleatorio.

Aunque este es solo un ejercicio, el uso de esta técnica en una plataforma real podría mejorar la experiencia del usuario al proporcionar recomendaciones personalizadas de productos y ofertas que se ajusten a sus intereses y necesidades. Además, la segmentación de los clientes en diferentes grupos también podría ayudar a los minoristas en línea a comprender mejor las tendencias de compra y mejorar su oferta de productos y servicios para satisfacer las necesidades de cada grupo. En resumen, la clusterización de los clientes de un e-commerce es una técnica útil para mejorar la experiencia del usuario y optimizar las operaciones comerciales.

4. IMPLEMENTACIONES

Si se tuvieran más datos, se podrían buscar correlaciones entre diferentes variables y usarlas para mejorar la experiencia del usuario en la plataforma. Algunos ejemplos de correlaciones que se podrían buscar incluyen:

- Correlación entre los productos comprados y el género del cliente: Si hay productos que son más populares entre los clientes de un género en particular, se podría personalizar la experiencia de compra para cada cliente. Por ejemplo, podrías mostrar productos relacionados con las compras previas del cliente y con los intereses del género al que pertenece.

- Correlación entre los productos comprados y la ubicación geográfica del cliente: Si hay productos que son más populares en ciertas regiones, se podría personalizar la experiencia de compra para los clientes de cada región. Por ejemplo, se podría ofrecer promociones especiales para los productos populares en una región determinada.

- Correlación entre la frecuencia de compra y el gasto total: Si hay clientes que compran con frecuencia, pero gastan poco, podrías ofrecer promociones especiales para incentivar a estos clientes a gastar más. Por otro lado, si hay clientes que compran con poca frecuencia, pero gastan mucho, podrías ofrecer promociones especiales para incentivar a estos clientes a comprar con más frecuencia.

Estos son solo algunos ejemplos de correlaciones que podrían ser implementadas si se tuvieran más datos. La idea es usar esta información para personalizar la experiencia de compra de cada cliente y mejorar la satisfacción del usuario en la plataforma.

5. ANEXOS

1 PCA

El PCA (Principal Component Analysis) es una técnica de reducción de dimensionalidad que se utiliza para analizar y representar datos en un espacio de menor dimensión. El objetivo es encontrar una representación de los datos que conserve la mayor cantidad posible de la información original. Esto se logra mediante la transformación de las variables originales en un conjunto reducido de variables no correlacionadas, denominadas componentes principales, que explican la mayor cantidad de varianza en los datos. El PCA se utiliza comúnmente en campos como la estadística, el aprendizaje automático y el análisis de datos para reducir la complejidad de los conjuntos de datos y facilitar su análisis y visualización.

La gráfica de varianza explicada acumulada contra número de componentes es una representación gráfica de la cantidad de información que se mantiene después de reducir la dimensionalidad de un conjunto de datos utilizando PCA. En general, la idea detrás de reducir la dimensionalidad de los datos es eliminar la redundancia y la información no relevante en los datos, lo que puede mejorar la eficiencia y precisión de los algoritmos de aprendizaje automático que se utilizan posteriormente. El objetivo de la gráfica es ayudar a determinar el número óptimo de componentes que se deben mantener en el análisis de reducción de dimensionalidad. En general, se desea retener la mayor cantidad posible de información (varianza explicada) mientras se reduce la dimensionalidad.

2 Clusterización

La clusterización es una técnica de aprendizaje no supervisado utilizada para agrupar objetos similares en conjuntos llamados clusters. El objetivo es separar los datos en grupos que sean coherentes y diferentes entre sí, para poder comprender mejor la estructura y las relaciones entre los datos. La técnica busca maximizar la similitud dentro de cada grupo y minimizar la similitud entre grupos diferentes. Esta técnica es utilizada en diversas áreas, como la biología, la medicina, la ciencia de datos, la minería de datos y la inteligencia artificial, entre otras.

3 K-Means

K-means es un algoritmo de clustering que agrupa los datos en k clusters, donde k es un valor predefinido. El algoritmo busca iterativamente los centroides de los clusters y asigna cada punto de datos al centroide más cercano. Luego, actualiza los centroides de cada cluster y repite el proceso hasta que se alcanza la convergencia. El objetivo del algoritmo es minimizar la distancia entre los puntos de datos y su centroide correspondiente. K-means es una técnica simple y ampliamente utilizada en el análisis de datos no supervisado y en la segmentación de datos.

Centroide: En el contexto de K-means clustering, un centroide es un punto que representa el centro de un grupo de puntos similares. los centroides son el punto de referencia para cada grupo de puntos similares en el clustering de K-means.

4 Elbow Method

El método del codo es una técnica utilizada en la clusterización para determinar el número óptimo de clusters en un conjunto de datos. El método implica ejecutar el algoritmo de clusterización varias veces para un rango de valores de k (número de clusters) y luego trazar el resultado de la suma de los errores cuadrados (SSE) para cada valor de k. El punto donde la curva SSE comienza a aplanarse se conoce como "codo". Este punto se considera como el número óptimo de clusters para el conjunto de datos.

SSE: SSE (Sum of Squared Errors) es una medida utilizada en técnicas de clustering como K-means que mide la suma de las distancias al cuadrado entre cada punto y el centroide del cluster al que pertenece. El objetivo de la técnica de clustering es minimizar la SSE, lo que significa encontrar clusters que estén más cerca de sus centroides y que tengan una mayor cohesión interna.

5 Web Scrapping

El Web Scrapping es una técnica que consiste en extraer y analizar información de páginas web de forma automatizada, utilizando herramientas o programas especializados. Con el Web Scrapping se pueden obtener datos como texto, imágenes, videos, precios, opiniones, entre otros, para su posterior análisis y utilización en diferentes aplicaciones y procesos de toma de decisiones.

Esta técnica es ampliamente utilizada en diversas áreas, como en el análisis de mercados, la inteligencia empresarial, el análisis de datos y el aprendizaje automático.

6. REFERENCIAS

1 *Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction (2nd ed.). Springer.*

2 *Jain, A. K. (2010). Data clustering: 50 years beyond K-means. Pattern Recognition Letters, 31(8), 651-666.*

3 *Scikit-learn. (n.d.). Clustering. Retrieved from https://scikit-learn.org/stable/modules/clustering.html*

4 *Shlens, J. (2014). A Tutorial on Principal Component Analysis. Retrieved from https://arxiv.org/abs/1404.1100*

5 *Zhang, T., Ramakrishnan, R., & Livny, M. (1996). BIRCH: An efficient data clustering method for very large databases. Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, 103-114.*

6 *"Web Scraping with Python: Collecting More Data from the Modern Web" de Ryan Mitchell.*