Reto: Proceso estocásticos para modelar el comportamiento de los clientes en PiSA

Paulina Martínez López A01639743
Santiago Mora Cruz A01369517
Gabriel Reynoso Escamilla A01643561
Guillermo Villegas A01637169

Septiembre de 2024

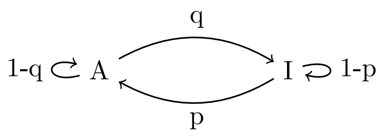
1 Introducción

Medicom es una empresa del grupo PiSA encargada de la distribución de productos a clientes de bajo volumen. Su misión es brindar el mejor servicio en distribución de medicamentos y otro equipo médico a través de su personal. La empresa busca un análisis de la información sobre la desactivación y reactivación de ventas de sus productos, es decir, si un producto se vende o no durante el plazo que se espera que sea comprado, así como el valor de vida de sus clientes. Para lograr esto se usará un modelo estocástico conocido como Cadena de Markov. Esta cadena se refiere a una secuencia de variables aleatorias, cuyos estados dependen únicamente del último estado en un tiempo pasado. [1] Para los fines de este artículo se aprovecharán algunas propiedades de dichas cadenas como lo son las probabilidades de transcición entre estados, y las distribución estática de la cadena.

2 Metodología

Se comenzó con un Análisis Exploratorio de los datos (EDA), se exploraron las variables de la base de datos "fecha", "material", "id_cliente" y "venta". Para explorar las ventas se utilizo un diagrama basado en los cuartiles de los datos, diagrama de Cajas y Bigotes. Para cuantificar las ventas en periodo se realizaron histogramas de número de registros por año. Se identificaron valores negativos en la variable ventas, estos también se cuantificaron y se representaron con histogramas. Para conocer más a fondo este proceso consúltese el **Apéndice a.**.

Para conocer explorar las probabilidades de reactivación y desactivación, tanto de materiales como de clientes, se propuso una cadena que podría usarse en ambos casos.



Esta cuenta con dos estados, Activo (A), e Inactivo (I). La probabilidad de desactivación es q, mientras que la de reactivación es p, esto lo podemos observar con la matriz de transciciones:

$$\mathcal{P} = \begin{pmatrix} 1-p & p \\ q & 1-q \end{pmatrix}$$

Por su estructura la cadena tiene carácterísticas que nos pueden ayudar a dar respuesta a las preguntas del grupo PiSA, ambos estados están comunicados lo que hace su clase de comunicación la misma que su espacio de estados $\mathscr{E} = \mathscr{C} = \{A, I\}$, esto la hace irreducible. El periodo es el máximo común divisor del conjunto de número de pasos en el que es posible el estado regrese a si mismo, en el caso de la cadena propuesta:

$$d(\mathbf{A}) = d(\mathbf{I}) = \text{mcd}\{1, 2, 3, 4, ...\} = 1$$

El periodo de ambos estados es uno esto hace de la cadena una cadena aperiódica. Por último se dice que es recurrente positiva porque si se inicia en cualquier estado, el tiempo esperado para volver al mismo es finito [1].

Estas carácterísticas identifican a la cadena como ergódica, para la cual existe un vector de probabilidades estático, unico π, este vector es útiles porque nos permiten conocer es comportamiento a largo plazo tanto de clientes como materiales [2].

Para lograr un análisis más profundo se decidió analizar por separado los venta de los materiales y las compras de los clientes, sin embargo el proceso es análogo.

Para obtener la matriz de transición asociada a cada material o cliente, se extraen sus observaciones de la base de datos, filtrando el periodo de interés, enero 2021 hasta agosto de 2024. Después se agrupa esta información por mes y año sumando los valores de ”ventas”, en el caso de no existir registros en el periodo el valor es 0, este valor lo llamaremos X_t, siendo t el tiempo. Luego se definen los estados de la siguiente manera:

- Si $X_t > 0$: El estado es activo (A)
- Si $X_t = 0$: El estado es inactivo (I)
- Si $X_t < 0$:
  - Si $X_t < 0$ y $X_{t+1} > 0$: El estado es activo
  - Cualquier otro caso se considera inactivo

Para la matriz de transición P, se calcularon las probabilidades del siguiente modo: Se creó un data frame nuevo a partir de la clasificación de las transiciones mencionada anteriormente. En este data frame, con base en la fecha, se escribió para cada transición, cuantas veces ocurría. Con esto, se creo una matriz que tenía cuántas veces:

$$\begin{vmatrix} \text{Permaneció inactivo} & \text{Se reactivó} \\ \text{Se desactivó} & \text{Permaneció activo} \end{vmatrix}$$

Para calcular las probabilidades, se dividió cada valor entre la suma de su respectiva fila; obteniendo la matriz de transición.

Al ser ergódica podemos obtener la distribución estacionaria de la siguiente manera:

$$\pi = \left(\frac{p}{p+q}, \frac{q}{p+q} \right) \tag{1}$$

Para conocer la frecuencia de venta del producto podemos utilizar el tiempo medio de recurrencia, el número esperado de pasos para volver al estado de activación:

$$\mu_1 = \frac{1}{\pi_{\mathbf{1}}} \tag{2}$$

Por último calculamos la matriz de transición al tiempo n.

Cabe mencionar que este proceso solo es útil para algunos materiales y clientes, existen dos casos en los que su probabilidad será distinta:

- Si el material o cliente, realizaron una solo transacción:

  $$\mathcal{P} = \begin{pmatrix} 1 & 0 \\ N/A & N/A \end{pmatrix}$$

  y,

  $$\pi = (1, 0)$$

- Si el material o cliente, desde la primera transacción no se han deactivado:

  $$\mathcal{P} = \begin{pmatrix} N/A & N/A \\ 0 & 1 \end{pmatrix}$$

  y,

  $$\pi = (0, 1)$$

También era de interés para grupo PiSA el valor de vida de los clientes. El *Customer Lifetime Value* (CLV) mide cuánto ingreso puede aportar considerando la duración de su actividad como cliente y el valor de sus transacciones. Por lo tanto, es una métrica que estima el valor total que un cliente generará para una empresa durante toda su relación con ella. Calculamos el CLV en un proceso escalonado donde primero debemos calcular el CLV monetario. Este nos ayuda a comprender el valor financiero básico que el cliente aporta a la empresa [3]. En un primer acercamiento lo vimos desde un punto de vista de marketing, se calculó utilizando la siguiente fórmula :

$$\textbf{CLV}_{\text{monetario}} = \text{Promedio de ventas} \times \text{Número de transacciones}$$

$$\textbf{CLV}_{\text{ajustado}} = \textbf{CLV}_{\text{monetario}} \times \text{Tiempo de vida del cliente}$$

Donde el tiempo de vida del cliente son los dias desde la primera hasta la última compra. Sin embargo decidimos tomar un acercamiento similar al de C.-J. Cheng y S.W. Chiu et. al. [4] y utilizar la probabilidad de desactivación p_{10} y valor promedio de ventas de cada cliente v y calcularla de la siguiente manera:

$$\text{CLV} = \mu_o v \tag{3}$$

Este valor permitirá conocer qué clientes aportan más a la empresa y por lo tanto es importante mantener buenas relaciones, y aquellos los cuales no aportan y sería conveniente incentivar ventas o cortar relaciones comerciales.

Con la finalidad de utilizar la información de la cadena y proporcionar una base para las estrategias de la empresa se diseñó un sistema de recomendación de productos usando la matriz de co-ocurrencia y la similitud del coseno, una métrica que cuando el valor se acerca al uno implica similaridad y al acercarse al cero disimilitud (en este caso no se han vendido juntos). Si tenemos una matriz de co-ocurrencia M, en la que cada entrada M_{ij} representa cuántas veces los productos i y j han sido comprados juntos, entonces cada fila (o columna) de esa matriz puede ser vista como un vector que describe las relaciones de un producto con los demás.

Así, el vector de cada producto es: P_i es $\vec{P_i} = (M_{i1}, M_{i2}, \ldots, M_{in})$

Con estos vectores, se calcula la similitud del coseno entre dos productos P_i y P_j se define como:

$$\text{sim}(\vec{P_i}, \vec{P_j}) = \frac{\vec{P_i} \cdot \vec{P_j}}{\|\vec{P_i}\| \|\vec{P_j}\|}$$

Donde:

$\vec{P_i} \cdot \vec{P_j} = \sum_{k=1}^{n} M_{ik} M_{jk}$ es el producto punto.

$\|\vec{P_i}\| = \sqrt{\sum_{k=1}^{n} M_{ik}^2}$ es la norma del vector.

Con esto, la matriz de similitud S se construye calculando la similitud del coseno entre todos los productos:

$$S_{ij} = \frac{\sum_{k=1}^{n} M_{ik} M_{jk}}{\sqrt{\sum_{k=1}^{n} M_{ik}^2} \cdot \sqrt{\sum_{k=1}^{n} M_{jk}^2}}$$

Y para recomendar productos basados en uno dado P_i, encontramos los productos con mayor similitud en la fila i de la matriz S:

Recomendaciones(P_i) = Top k productos P_j tal que S_{ij} es máximo, con $P_j \neq P_i$

Es posible observar la implementación de esta metodología en el **Apéndice b.**.

3 Resultados

En el EDA se encontraron dos aspectos de los datos de importancia para el proyecto. El primero son los datos faltantes desde 2015 hasta 2020 (Fig. 1), supusimos que esto se debía a errores en el manejo de los datos, no a la falta de ventas, por lo que solo se realizó el modelo con la información disponible desde 2021. El otro son los valores negativos negativos en ”ventas”, se tuvo la oportunidad tener una sesión con los representantes de PiSA en la que se aclararon este punto, los valores negativos se refieren a devoluciones de materiales. Por último se supuso que la primera vez que se compró un material o un cliente compró coincidía con la primer fecha que se tenía en los datos de dicho cliente o material, a partir de 2021. La base de datos tiene registros diarios, sin embargo consideramos que sería de mayor utilidad agruparlos de manera mensual.

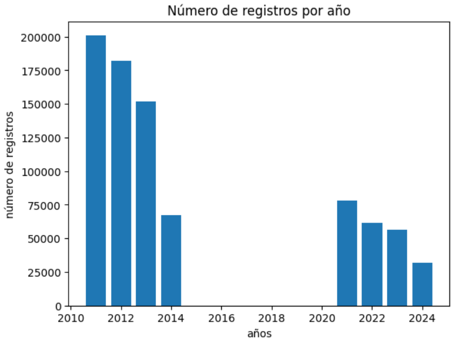


Figure 1: Número de registros por año

3.1 Materiales

Tomaremos como ejemplo del material 768:

- Última fecha en registro del material: 2024-08-06
- La matriz de transición asociada al material es:

$$\mathcal{P} = \begin{pmatrix} 0.9231 & 0.0769 \\ 0.0333 & 0.9666 \end{pmatrix}$$

- El vector de distribución estacionaria es: $\pi = (0.3023, 0.6977)$
- Dado que el material fue comprado, será comprado de nuevo en 1.43 meses

3.2 Clientes

Para el cliente 2222:

- Última fecha en registro para el cliente: 2024-02-29
- La matriz de transición asociada al cliente es:

$$\mathcal{P} = \begin{pmatrix} 0.7059 & 0.2941 \\ 0.2308 & 0.7692 \end{pmatrix}$$

- El vector de distribución estacionaria es: $\pi = (0.4397, 0.5603)$
- Dado que el cliente compró, comprará de nuevo en 1.78 meses

4 Análisis

El análisis realizado tanto con la cadena de Markov como con el CLV permitirán a Medicom: Conocer los productos con ventas más lentas y periodos de inactividad más largos, lo que será útil para idear una estrategia de mayor movilización del producto o retirarlo del mercado. También dará información del comportamiento de los clientes permitiendo confiar el los que tienen un estado estable de activación, sin descuidarlos, y prestar mayor atención en aquellos que tengan un tiempo entre compras más largos.

Para los clientes, teniendo los tiempos medios de recurrencia, el CLV y las probabilidades, se puede hacer una ponderación de estas características para cada cliente y, basado en esto, ofrecer beneficios a los clientes frecuentes y a aquellos que son recurrentes para lograr dos cosas: evitar que pasen a un estado inactivo e incentivar a nuevos clientes a comprar más frecuentemente.

Como estartegia proponemos lo siguiente: Para **Materiales**, segmentarlos es tres casos: Si el material se ha comprado cada mes desde su primer registro, se recomienda mantener el inventario de este bien abastecido, y promocionarlo como *best-seller*. Si el producto tiene ventas intermitentes se recomienda agruparlo en promociones de venta cruzada con productos que se vendan de manera similar, además de ajustar el inventario a su demanda variable. Por último si el producto tiene ventas irregulares, se recomienda ofrecerlo a clientes que lo han comprado antes con ofertas especiales.

Los **Clientes** también se separarán en tres casos: Si el cliente es leal (tiene compras mensuales desde su primera activación), se recomienda ofrecer un programa de fidelidad. Si el cliente tiene compras intermitentes se recomienda implementar recordatorios automáticos ajustados a su tiempo medio de recurrencia. Si las compras del cliente son más espaciadas se recomienda mantener contacto con el cliente, mediante promociones, comunicados, etc., para intentar que compre más frecuentemente.

5 Conclusiones

Con este trabajo fue posible crear una interfaz en la que la empresa, ingresa el material o cliente de su interés y regresa la tasa de desactivación y reactivación, así como sus probabilidades, también el tiempo medio entre compras. Además en el caso que sea un cliente regresa su CLV. Junto con esto proporciona las posibles estrategias comentadas anteriormente. Aunque la aplicación creada es útil es posible utilizar el modelo para clasificación de clientes, predicción de demanda o simplemente otra clase de estrategias de la mano de un equipo de marketing o ventas. Además se podría agregar información para mejorar o aumentar el modelo como los datos faltantes, el número de unidades vendidas, o las categorías de los productos.

References

[1] W. K. Ching and M. K. Ng, *Markov chains : models, algorithms and applications.* Springer, 2006.

[2] R. G. Gallager, *Discrete Stochastic Processes.* New York, NY: Springer, 1996.

[3] Qualtrics, "Customer lifetime value (clv): qué es y cómo calcularlo." `https://www.qualtrics.com/es-es/gestion-de-la-experiencia/cliente/customer-lifetime-value/`. Accessed: 2024-08-24.

[4] C.-J. Cheng, S. Chiu, C.-B. Cheng, and J.-Y. Wu, "Customer lifetime value prediction by a markov chain based data mining model: Application to an auto repair and maintenance company in taiwan," *Scientia Iranica*, vol. 19, no. 3, pp. 849–855, 2012.

Apéndice a

Análisis de datos exploratorio

23 de agosto de 2024

Santiago Mora Cruz

Gabriel Reynoso Escamilla

Paulina Martinez Lopez

Guillermo Villegas Morales

```
In [ ]: import pandas as pd
        import matplotlib.pyplot as plt
```

```
In [ ]: db = pd.read_parquet('24o_medicom.parquet', engine='pyarrow')
        db['ventas'] = db['ventas'].astype('float')
        db.head()
```

Out[ ]:

	fecha	id_material	id_cliente	ventas
0	2013-05-06	768	7939	384.0
1	2011-09-20	768	7939	384.0
2	2014-01-08	768	7939	384.0
3	2011-04-19	768	7939	384.0
4	2013-03-21	768	7805	384.0

```
In [ ]: # db.to_csv('df.csv', index = False)
```

```
In [ ]: db.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 830517 entries, 0 to 830516
Data columns (total 4 columns):
 #   Column       Non-Null Count   Dtype  
---  ------       --------------   -----  
 0   fecha        830517 non-null  object 
 1   id_material  830517 non-null  int64  
 2   id_cliente   830517 non-null  int64  
 3   ventas       830517 non-null  float64
dtypes: float64(1), int64(2), object(1)
memory usage: 25.3+ MB
```

La base de datos se observa completa, sin valores nulos

```
In [ ]: db['ventas'].describe()
```

Out[ ]:

	ventas
count	8.305170e+05
mean	2.852943e+03
std	1.359341e+04
min	-1.149758e+06
25%	4.231500e+02
50%	9.525000e+02
75%	2.316800e+03
max	2.022000e+06

dtype: float64

Podemos observar que contamos con valores de ventas negativos que representan pérdidas. Además, vemos que el mínimo y el máximo distan demasiado de los valores donde caen los cuartiles.

```
In [ ]: print('clientes: ', len(db['id_cliente'].unique()))
        print('materiales: ', len(db['id_material'].unique()))
```

```
clientes:  454
materiales:  1471
```

```
In [ ]: 
```

```
In [ ]: # Clientes que más compran
        clientes_mas_compran = db.groupby('id_cliente')['ventas'].sum().sort_values(ascending=False)
        print("Clientes que más compran:")
        print(clientes_mas_compran.head())

        # Clientes que menos compran
        clientes_menos_compran = db.groupby('id_cliente')['ventas'].sum().sort_values(ascending=True)
        print("\nClientes que menos compran:")
        print(clientes_menos_compran.head())
```

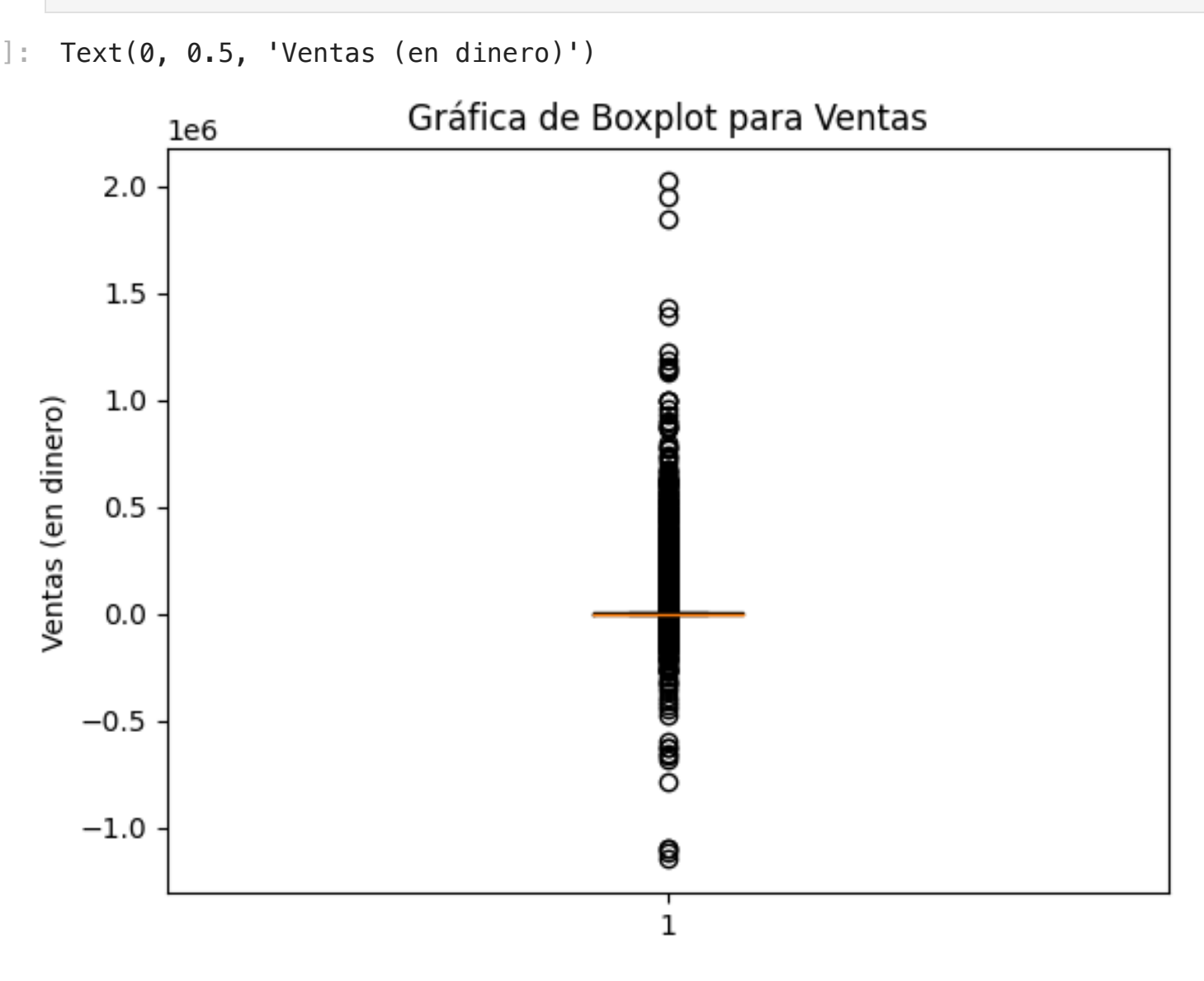
```
Clientes que más compran:
id_cliente
8342    3.419803e+08
8635    1.095702e+08
8318    7.813064e+07
7713    7.060284e+07
7806    5.166043e+07
Name: ventas, dtype: float64

Clientes que menos compran:
id_cliente
8182   -48987.90
308    -38823.60
8233   -10610.00
8202    -6869.85
7698    -1033.80
Name: ventas, dtype: float64
```

```
In [ ]: # Materiales que más se venden
        materiales_mas_venden = db.groupby('id_material')['ventas'].sum().sort_values(ascending=False)
        print("\nMateriales que más se venden:")
        print(materiales_mas_venden.head())

        # Materiales que menos se venden
        materiales_menos_venden = db.groupby('id_material')['ventas'].sum().sort_values(ascending=True)
        print("\nMateriales que menos se venden:")
        print(materiales_menos_venden.head())
```

```
Materiales que más se venden:
id_material
591    1.343663e+08
893    1.236979e+08
601    1.207977e+08
590    1.065122e+08
772    9.313992e+07
Name: ventas, dtype: float64

Materiales que menos se venden:
id_material
2643   -2485.12
1585    -722.50
6929    -310.60
3488    -115.60
1586     -62.50
Name: ventas, dtype: float64
```

```
In [ ]: plt.boxplot(db['ventas']);
        plt.title("Gráfica de Boxplot para Ventas")
        plt.ylabel('Ventas (en dinero)')
```

```
Out[ ]: Text(0, 0.5, 'Ventas (en dinero)')
```



Observamos que existen muchos outliers

Convertimos `fecha` a variable tipo `datetime`

```
In [ ]: db['fecha'] = pd.to_datetime(db['fecha'])
```

```
In [ ]: years = list(range(2011, 2025, 1))
        count4year = []
        for item in years:
            count4year.append(db[db['fecha'].dt.year == item]['fecha'].count())
```

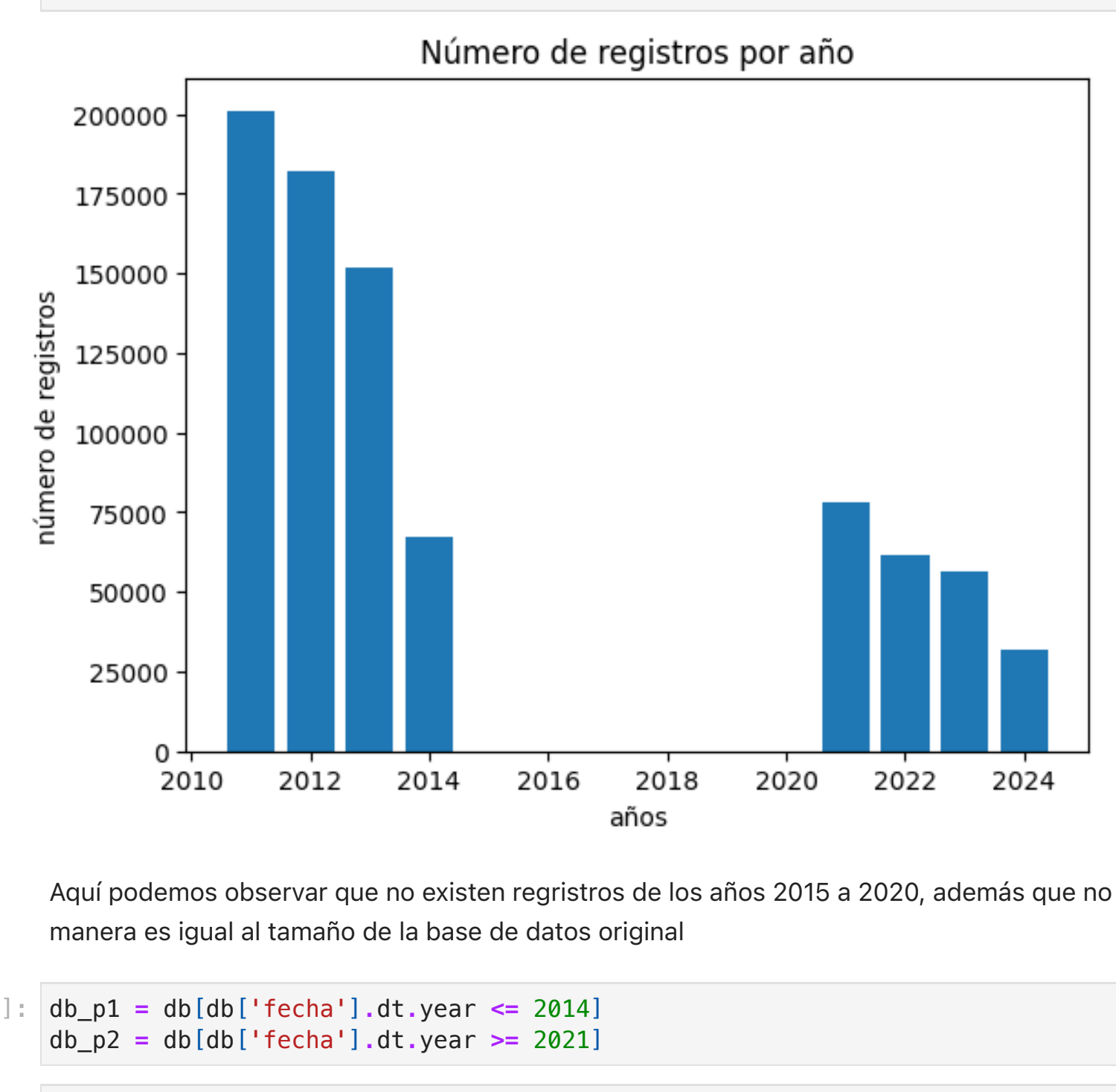
```
In [ ]: import numpy as np
        np.sum(count4year)
```

```
Out[ ]: 830517
```

```
In [ ]: count4year
```

```
Out[ ]: [201111, 182082, 152053, 67476, 0, 0, 0, 0, 0, 0, 78186, 61282, 56684, 31643]
```

```
In [ ]: plt.figure()
        plt.bar(years, count4year)
        plt.title('Número de registros por año')
        plt.xlabel('años')
        plt.ylabel('número de registros')
        plt.savefig('Barplot_registros.jpeg')
```



Aquí podemos observar que no existen regristros de los años 2015 a 2020, además que no es un error de formato ya que la suma de todos los registros que obtuvimos de esta manera es igual al tamaño de la base de datos original

```
In [ ]: db_p1 = db[db['fecha'].dt.year <= 2014]
        db_p2 = db[db['fecha'].dt.year >= 2021]
```

```
In [ ]: len(db_p1['id_cliente'].unique())
```

```
Out[ ]: 417
```

```
In [ ]: len(db_p2['id_cliente'].unique())
```

```
Out[ ]: 158
```

Con este análisis anterior, se encontró que de 2014 para atrás, se tenían 417 clientes, mientras que de 2021 en adelante, solo están 158. Por esto mismo, despreciaremos los registros que se situen de 2011-2014 ya que la mayoría de los clientes no continuan después de 2021. Además esto hará un mejor modelo ya que la brecha en los tiempos y la inconsistencia en los clientes sesgaría el proceso

```
In [ ]: # Clientes que más compran
        clientes_mas_compran = db_p2.groupby('id_cliente')['ventas'].sum().sort_values(ascending=False)
        print("Clientes que más compran:")
        print(clientes_mas_compran.head())

        # Clientes que menos compran
        clientes_menos_compran = db_p2.groupby('id_cliente')['ventas'].sum().sort_values(ascending=True)
        print("\nClientes que menos compran:")
        print(clientes_menos_compran.head())
```

```
Clientes que más compran:
id_cliente
8342    3.022364e+08
8635    8.345564e+07
7713    6.060637e+07
9066    4.142447e+07
7806    3.916261e+07
Name: ventas, dtype: float64

Clientes que menos compran:
id_cliente
9213     3502.77
8107     4600.20
3452     8184.24
3451    21436.34
8363    23151.80
Name: ventas, dtype: float64
```

```
In [ ]: # Materiales que más se venden
        materiales_mas_venden = db_p2.groupby('id_material')['ventas'].sum().sort_values(ascending=False)
        print("\nMateriales que más se venden:")
        print(materiales_mas_venden.head())

        # Materiales que menos se venden
        materiales_menos_venden = db_p2.groupby('id_material')['ventas'].sum().sort_values(ascending=True)
        print("\nMateriales que menos se venden:")
        print(materiales_menos_venden.head())
```

```
Materiales que más se venden:
id_material
591    97903712.02
601    84710470.41
893    84654422.81
590    78186308.77
772    77445106.93
Name: ventas, dtype: float64

Materiales que menos se venden:
id_material
6929   -310.6
3488   -115.6
7207      0.0
6056      0.0
2934      0.0
Name: ventas, dtype: float64
```

Obtenemos nuevamente los clientes que más y menos han comprado en cuanto a ventas así como los materiales que más y menos se han vendido, pero solo para la segunda mitad de los datos (2021-2024). De este punto en adelante solo trabajaremos con los datos mencionados

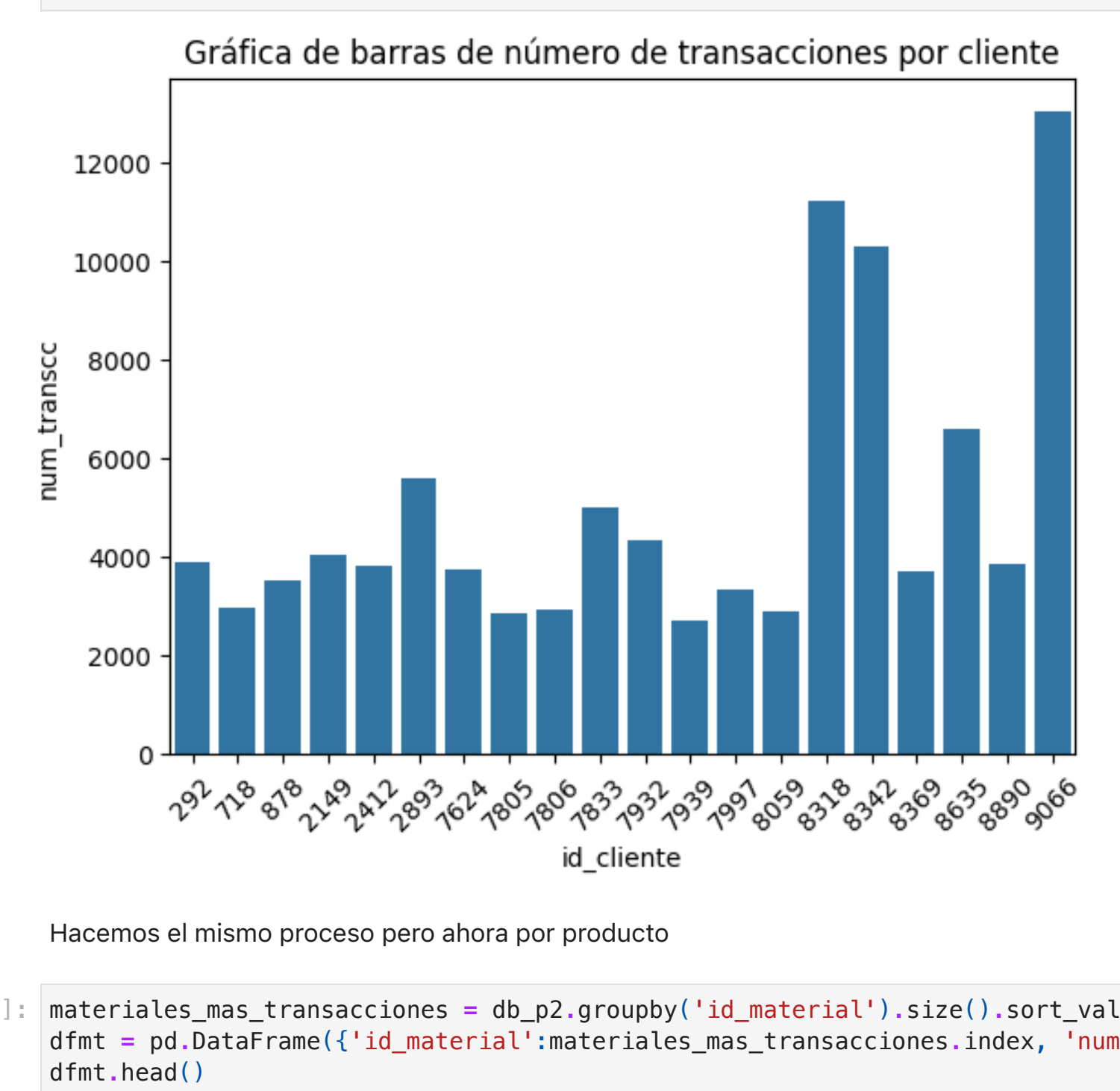
Buscamos los clientes que cuentan con más regsitros y los graficamos en un barplot

```
In [ ]: clientes_mas_transacciones = db_p2.groupby('id_cliente').size().sort_values(ascending=False).head(20)
        dfct = pd.DataFrame({'id_cliente':clientes_mas_transacciones.index, 'num_transcc':clientes_mas_transacciones.values})
        dfct.head()
```

Out[ ]:

	id_cliente	num_transcc
0	9066	13051
1	8318	11240
2	8342	10284
3	8635	6601
4	2893	5576

```
In [ ]: import seaborn as sns
        sns.barplot(data = dfct, x = 'id_cliente', y = 'num_transcc')
        plt.xticks(rotation = 45)
        plt.title('Gráfica de barras de número de transacciones por cliente')
        plt.show();
```



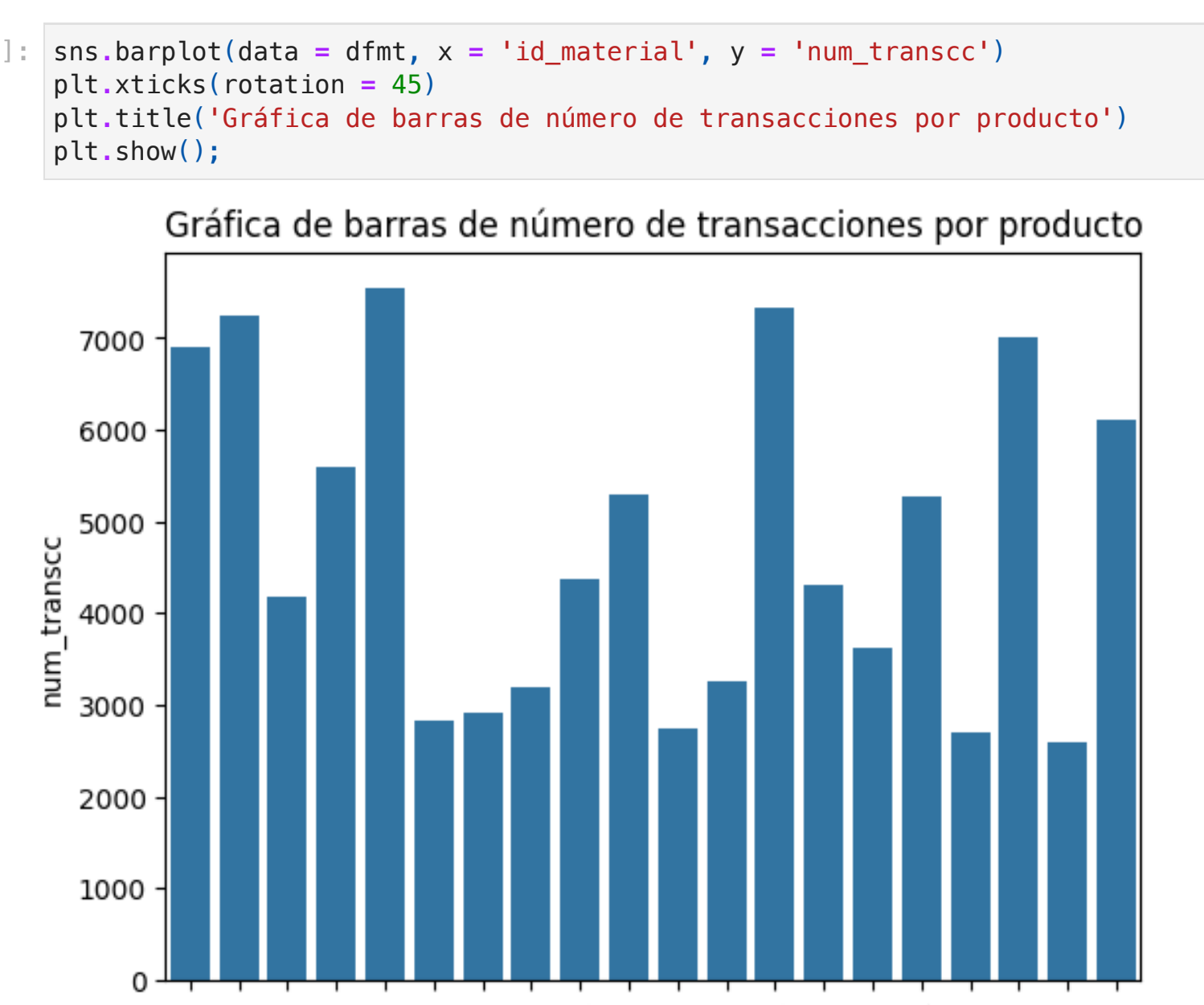
Hacemos el mismo proceso pero ahora por producto

```
In [ ]: materiales_mas_transacciones = db_p2.groupby('id_material').size().sort_values(ascending=False).head(20)
        dfmt = pd.DataFrame({'id_material':materiales_mas_transacciones.index, 'num_transcc':materiales_mas_transacciones.values})
        dfmt.head()
```

Out[ ]:

	id_material	num_transcc
0	601	7551
1	772	7341
2	591	7256
3	893	7007
4	590	6902

```
In [ ]: sns.barplot(data = dfmt, x = 'id_material', y = 'num_transcc')
        plt.xticks(rotation = 45)
        plt.title('Gráfica de barras de número de transacciones por producto')
        plt.show();
```



Buscamos a qué se deben los valores negativos y qué clientes o productos los generan

```
In [ ]: db_minus = db_p2[db_p2['ventas'] <=0].reset_index()
        db_minus
```

Out[ ]:

	index	fecha	id_material	id_cliente	ventas
0	10	2023-01-24	768	9066	-484.92
1	103	2024-05-13	768	9066	-183.38
2	108	2023-01-20	768	9066	-161.64
3	195	2023-06-22	768	9066	-519.00
4	196	2023-06-02	768	9066	-519.00
...	...	...	...	...	...
6546	830188	2023-07-31	767	7805	-1170.00
6547	830273	2023-07-06	767	8635	-2925.00
6548	830278	2023-07-14	767	7805	-1462.50
6549	830279	2023-07-17	767	7805	-1462.50
6550	830481	2021-01-07	3071	7842	-5265.46

6551 rows × 5 columns

existen 6,551 registros con ventas negativas

```
In [ ]: db_minus.shape[0]/db_p2.shape[0]
```

```
Out[ ]: 0.028758313395816415
```

Aquí podemos ver que los valores negativos solo corresponden a poco menos de 3% de la información, por lo que rechazarla es opción. Primero buscaremos los materiales y clientes que generan estos valores

```
In [ ]: cliente_minus = db_minus['id_cliente'].value_counts()
```

```
In [ ]: minus_total = []
        for i in cliente_minus.index:
          minus_total.append(db_p2[db_p2['id_cliente'] == i].shape[0])
        db_clientes_minus = pd.DataFrame({'cliente': cliente_minus.index,
                                          'count_minus': cliente_minus,
                                          'count_total':minus_total})
        db_clientes_minus['difference'] = round(db_clientes_minus['count_minus']/db_clientes_minus['count_total'],3)
        db_clientes_minus
```

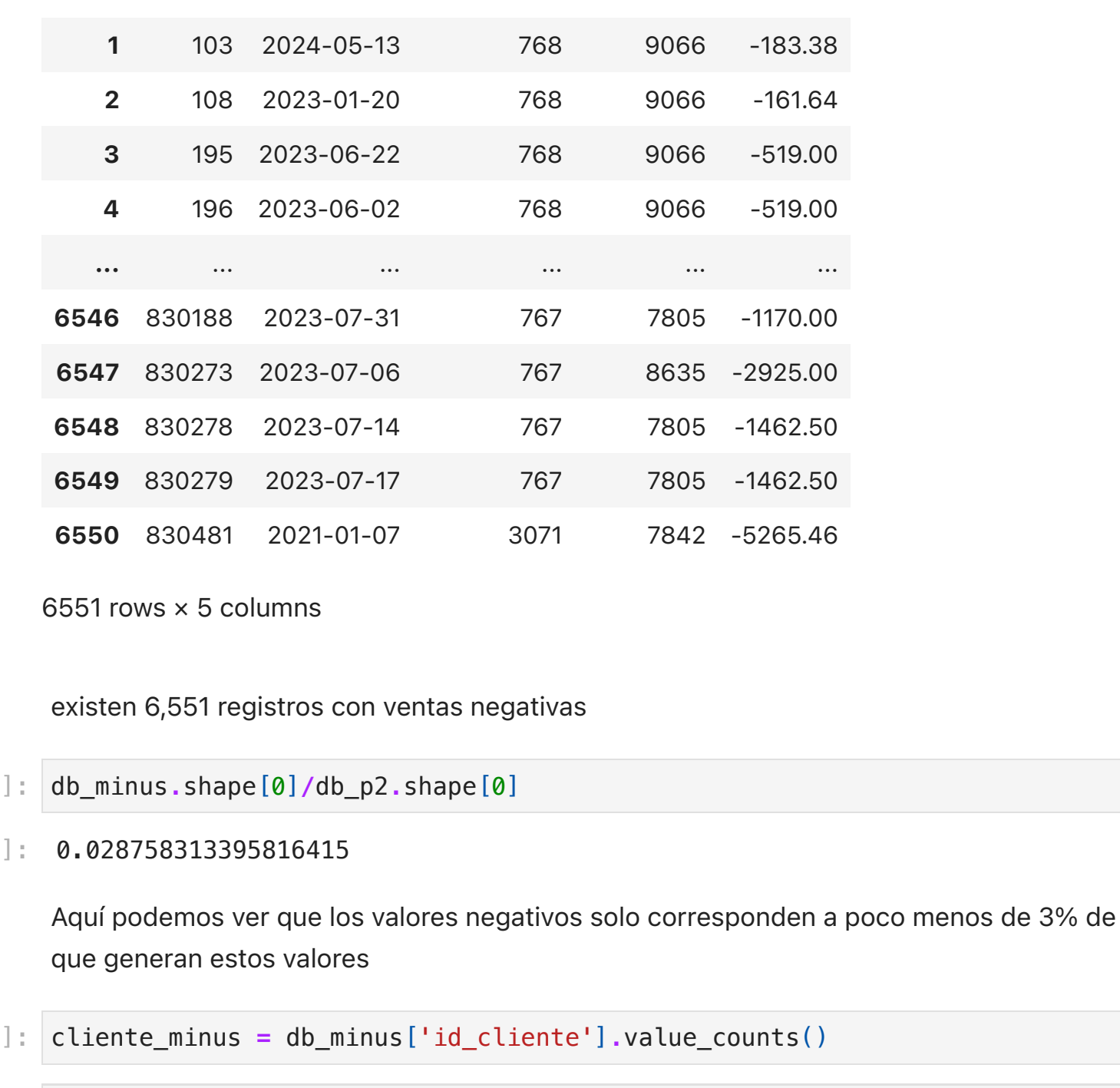
Out[ ]:

	cliente	count_minus	count_total	difference
id_cliente				
8342	8342	681	10284	0.066
8635	8635	540	6601	0.082
9066	9066	287	13051	0.022
7806	7806	251	2927	0.086
7648	7648	210	2199	0.095
...	...	...	...	...
8682	8682	1	600	0.002
1103	1103	1	50	0.020
8209	8209	1	378	0.003
8099	8099	1	1125	0.001
8598	8598	1	43	0.023

136 rows × 4 columns

```
In [ ]: sns.histplot(db_clientes_minus['difference'])
        plt.title('Gráfica de número de clientes con porcentaje de valores negativos')
```

```
Out[ ]: Text(0.5, 1.0, 'Gráfica de número de clientes con porcentaje de valores negativos')
```

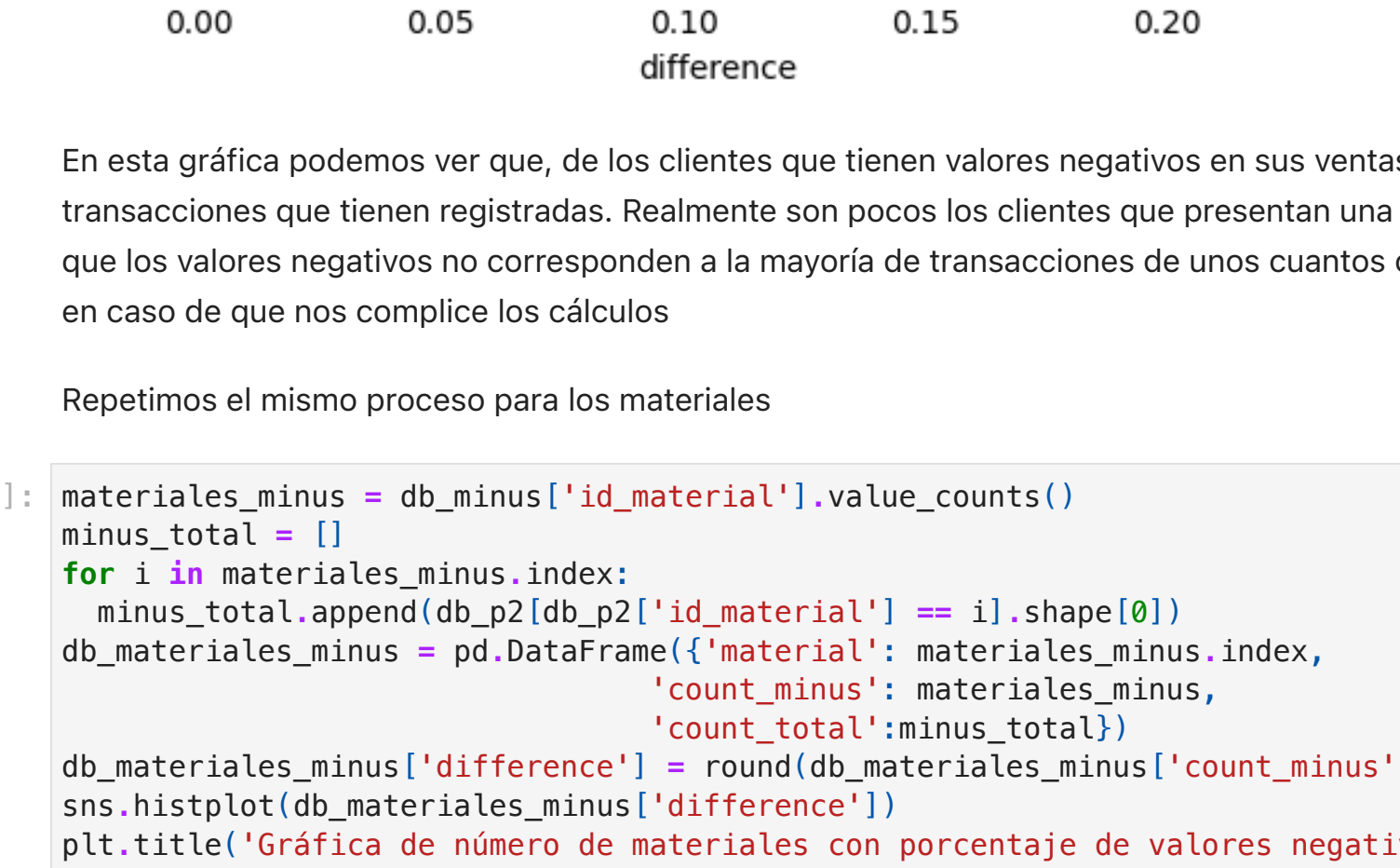


En esta gráfica podemos ver que, de los clientes que tienen valores negativos en sus ventas, la mayoría tiene una proporción muy chica de valores negativos con respecto al total de transacciones que tienen registradas. Realmente son pocos los clientes que presentan una proporción mayor a una décima parte y no superan el 25% de los registros. Ahora sabemos que los valores negativos no corresponden a la mayoría de transacciones de unos cuantos clientes, y que tampoco son muchas. Por lo tanto podemos descpreciar esta información en caso de que nos complice los cálculos

Repetimos el mismo proceso para los materiales

```
In [ ]: materiales_minus = db_minus['id_material'].value_counts()
        minus_total = []
        for i in materiales_minus.index:
          minus_total.append(db_p2[db_p2['id_material'] == i].shape[0])
        db_materiales_minus = pd.DataFrame({'material': materiales_minus.index,
                                            'count_minus': materiales_minus,
                                            'count_total':minus_total})
        db_materiales_minus['difference'] = round(db_materiales_minus['count_minus']/db_materiales_minus['count_total'],3)
        sns.histplot(db_materiales_minus['difference'])
        plt.title('Gráfica de número de materiales con porcentaje de valores negativos')
```

```
Out[ ]: Text(0.5, 1.0, 'Gráfica de número de materiales con porcentaje de valores negativos')
```



Observamos un patrón muy similar en la disitrbución de las proporciones de los valores negativos cuando los contamos por mateiral, aunque son más los materiales que suelen presentar pérdidas. Expecionalmente aparecen valores que siempre a representado pérdidas

```
In [ ]: db_materiales_minus[db_materiales_minus['difference'] == 1]
```

Out[ ]:

	material	count_minus	count_total	difference
id_material				
6929	6929	1	1	1.0
3488	3488	1	1	1.0

Como vemos que son materiales que se vendieron una única vez y presentaron pérdidas, podemos despreciarlos.

Apéndice b

Modelación cadena de Markov

Septiembre de 2024

Santiago Mora Cruz

Gabriel Reynoso Escamilla

Paulina Martinez Lopez

Guillermo Villegas Morales

In [ ]:

```python
import pandas as pd
import numpy as np
from numpy.linalg import eig, inv
import seaborn as sns
import matplotlib.pyplot as plt
```

In [ ]:

```python
df = pd.read_csv('db.csv', usecols = ['fecha', 'material', 'id_cliente', 'ventas'])
df['ventas'] = df['ventas'].astype('float')
df['fecha'] = pd.to_datetime(df['fecha'])

similarity_matrix = pd.read_csv('similarity_matrix.csv', index_col=0)

def monthly_trans_mat_mat(material, df = df):

    # Filter dataframe by material
    df_material = df[df['material'] == material]

    # Determine the global date range across all materials
    material_start_date = df_material['fecha'].min().to_period('M')
    global_end_date = df['fecha'].max().to_period('M')

    # Create a full date range for all months from the global start to end date
    all_months = pd.period_range(start=material_start_date, end=global_end_date, freq='M')

    # Group by year and month, and aggregate 'ventas' per month

    df_material['year_month'] = df_material['fecha'].dt.to_period('M')

    monthly_sales = df_material.groupby('year_month')['ventas'].sum().reset_index()

    # Reindex to include all months in the global date range, filling missing months with 0 ventas
    monthly_sales = monthly_sales.set_index('year_month').reindex(all_months, fill_value=0).reset_index()

    # Initialize activity states: 1 for active, 0 for inactive
    monthly_sales['activity'] = 0

    # Determine activity based on sales
    for i in range(len(monthly_sales)):
        if monthly_sales.loc[i, 'ventas'] > 0:
            monthly_sales.loc[i, 'activity'] = 1
        elif monthly_sales.loc[i, 'ventas'] < 0 and i < len(monthly_sales) - 1:
            # Lookahead: if the next month is positive, mark this month as active
            if monthly_sales.loc[i + 1, 'ventas'] > 0:
                monthly_sales.loc[i, 'activity'] = 1

    # Calculate the transitions
    transitions = monthly_sales['activity'].diff().fillna(0)

    # Initialize the transition matrix
    transition_matrix = np.zeros((2, 2))

    # Count the transitions and fill the transition matrix
    for i in range(1, len(transitions)):
        prev_state = int(monthly_sales['activity'].iloc[i-1])
        current_state = int(monthly_sales['activity'].iloc[i])
        transition_matrix[prev_state, current_state] += 1

    n = transition_matrix[0].sum()
    m = transition_matrix[1].sum()

    transition_matrix[0][0], transition_matrix[0][1] = transition_matrix[0][0]/n, transition_matrix[0][1]/n
    transition_matrix[1][0], transition_matrix[1][1] = transition_matrix[1][0]/m, transition_matrix[1][1]/m

    # Return both the transition matrix and the last active date
    return transition_matrix

def last_date_mat(material, df = df):

    return df[df['material'] == material]['fecha'].max()

def first_date_mat(material, df = df):

    return df[df['material'] == material]['fecha'].min()

def stationary_distr(P):

    p = P[0][1]
    q = P[1][0]

    pi = np.array([q/(p+q), p/(p+q)])

    if P[0][0] == 1 or P[1][1] == 1:
        raise ValueError('Al menos uno de los de la cadena de Markov es absorbente, por lo que la cadena no tiene distribución estacionaria')

    return pi

def t_medio_recurr(pi):

    return 1/pi[1]

def monthly_trans_mat_cli(id_cliente, df = df):

    # Filter dataframe by material
    df_cliente = df[df['id_cliente'] == id_cliente]

    # Determine the global date range across all materials
    client_start_date = df_cliente['fecha'].min().to_period('M')
    global_end_date = df['fecha'].max().to_period('M')

    # Create a full date range for all months from the global start to end date
    all_months = pd.period_range(start=client_start_date, end=global_end_date, freq='M')

    # Group by year and month, and aggregate 'ventas' per month

    df_cliente['year_month'] = df_cliente['fecha'].dt.to_period('M')

    monthly_sales = df_cliente.groupby('year_month')['ventas'].sum().reset_index()

    # Reindex to include all months in the global date range, filling missing months with 0 ventas
    monthly_sales = monthly_sales.set_index('year_month').reindex(all_months, fill_value=0).reset_index()

    # Initialize activity states: 1 for active, 0 for inactive
    monthly_sales['activity'] = 0

    # Determine activity based on sales
    for i in range(len(monthly_sales)):
        if monthly_sales.loc[i, 'ventas'] > 0:
            monthly_sales.loc[i, 'activity'] = 1
        elif monthly_sales.loc[i, 'ventas'] < 0 and i < len(monthly_sales) - 1:
            # Lookahead: if the next month is positive, mark this month as active
            if monthly_sales.loc[i + 1, 'ventas'] > 0:
                monthly_sales.loc[i, 'activity'] = 1

    # Calculate the transitions
    transitions = monthly_sales['activity'].diff().fillna(0)

    # Initialize the transition matrix
    transition_matrix = np.zeros((2, 2))

    # Count the transitions and fill the transition matrix
    for i in range(1, len(transitions)):
        prev_state = int(monthly_sales['activity'].iloc[i-1])
        current_state = int(monthly_sales['activity'].iloc[i])
        transition_matrix[prev_state, current_state] += 1

    n = transition_matrix[0].sum()
    m = transition_matrix[1].sum()

    transition_matrix[0][0], transition_matrix[0][1] = transition_matrix[0][0]/n, transition_matrix[0][1]/n
    transition_matrix[1][0], transition_matrix[1][1] = transition_matrix[1][0]/m, transition_matrix[1][1]/m

    # Return both the transition matrix and the last active date
    return transition_matrix

def last_date_cli(client_id, df = df):

    return df[df['id_cliente'] == client_id]['fecha'].max()

def first_date_cli(client_id, df = df):

    return df[df['id_cliente'] == client_id]['fecha'].min()

def plot_mat(material, df = df):
    df_material = df[df['material'] == material]
    clientes_mas_compran = df_material.groupby('id_cliente')['ventas'].sum().sort_values(ascending=False).head(10)
    order = clientes_mas_compran.index

    fig, _ = plt.subplots()
    graph  = sns.barplot(x = clientes_mas_compran.index, y = list(clientes_mas_compran.values), order = order, palette = 'plasma')

    fig.patch.set_facecolor('#2b2b2b')
    graph.set_facecolor('#1f1f1f')
    graph.tick_params(colors='white')  # Change tick colors
    graph.xaxis.label.set_color('white')  # Change x-axis label color
    graph.yaxis.label.set_color('white')  # Change y-axis label color
    graph.title.set_color('white')

    plt.xticks(rotation = 90)
    title = 'top 10 clientes que compran el material'
    plt.title(title)
    plt.xlabel('id de cliente')
    plt.ylabel('ventas totales')

    return graph

def plot_cli(id_cliente, df = df):
    df_cliente = df[df['id_cliente'] == id_cliente]
    materiales_mas_compran = df_cliente.groupby('material')['ventas'].sum().sort_values(ascending=False).head(10)
    order = materiales_mas_compran.index

    fig, _ = plt.subplots()
    graph  = sns.barplot(x = materiales_mas_compran.index, y = list(materiales_mas_compran.values), order = order, palette = 'plasma')

    fig.patch.set_facecolor('#2b2b2b')
    graph.set_facecolor('#1f1f1f')
    graph.tick_params(colors='white')  # Change tick colors
    graph.xaxis.label.set_color('white')  # Change x-axis label color
    graph.yaxis.label.set_color('white')  # Change y-axis label color
    graph.title.set_color('white')

    plt.xticks(rotation = 90)
    title = 'top 10 materiales que compra el cliente'
    plt.title(title)
    plt.xlabel('material')
    plt.ylabel('ventas totales')


    return graph

def proporcion_negativos_cli(id_cliente, db = df):
    db_cliente = db[db['id_cliente'] == id_cliente]
    return np.round(100*db_cliente[db_cliente['ventas'] < 0].shape[0] / db_cliente.shape[0], 2)

def proporcion_negativos_mat(material, db = df):
    db_material = db[db['material'] == material]
    return np.round(100*db_material[db_material['ventas'] < 0].shape[0] / db_material.shape[0], 2)

# --------------  sistema de recomendaciones -------------------------------------

def recomendar_productos(id_cliente, num_recomendaciones=10, df = df):
    df_cliente = df[df['id_cliente'] == id_cliente]

    # Obtener los top 5 materiales más comprados por el cliente
    materiales_mas_compran = df_cliente.groupby('material')['ventas'].sum().sort_values(ascending=False).head(10)
    top5_cliente = materiales_mas_compran.index.tolist()

    recomendaciones = pd.Series(dtype=float)
    for material in top5_cliente:
        recomendaciones = recomendaciones.add(similarity_matrix[material], fill_value=0)

    recomendaciones = recomendaciones.groupby(recomendaciones.index).mean().sort_values(ascending=False)
    recomendaciones = recomendaciones[~recomendaciones.index.isin(top5_cliente)]
    return recomendaciones.head(num_recomendaciones)

def recomendar_materiales_similares(material, num_recomendaciones=10):

    # Verificar si el material está en la matriz de similitud
    if material not in similarity_matrix.index:
        raise ValueError(f"El material '{material}' no se encuentra en la matriz de similitud.")

    # Obtener las similitudes de ese material con todos los demás
    similitudes = similarity_matrix[material]

    # Ordenar los materiales por similitud de forma descendente y excluir el propio material
    recomendaciones = similitudes.sort_values(ascending=False).drop(material)

    # Retornar los 10 materiales más similares
    return recomendaciones.head(num_recomendaciones)

clientes = df['id_cliente'].unique()
materiales = df['material'].unique()
```