Reporte2

October 4, 2023

Autores: Alejandra Velasco Zárate A01635453, José Antonio Juárez Pacheco A00572186, Jose Carlos Yamuni Contreras A01740285, Juan Manuel Hernández Solano A00572208 y Mayra Sarahí De Luna Castillo A01635774

1 Implementación del primer modelo usando cadenas de Markov

1.1 Importación librerías necesarias

[1]:
```
# Warnings ----------------------------------------------

import warnings
warnings.filterwarnings("ignore")

# Lectura y manipulación de datos -----------------------

import pandas as pd

# Descomposición espectral ------------------------------

import numpy as np
from numpy.linalg import eig, inv
```

1.2 Lectura de la base de datos

[13]:
```
data = pd.read_parquet('../data/tec_estocasticos.parquet', engine='pyarrow')
data.head()
```

[13]:

	periodo	cliente_id	material_id	tipo_cliente
0	05-2022	4894.0	22.0	Distribuidor
1	05-2022	4769.0	17.0	Distribuidor
2	05-2022	4823.0	227.0	Distribuidor
3	08-2022	4816.0	340.0	Distribuidor
4	08-2022	4888.0	270.0	Distribuidor

1.3 Análisis premilinar de los datos

[6]:
```
# Tipos de cliente

print(f"Los tipos de cliente en la base de datos son {len(data.tipo_cliente.
 ↪unique())}: {data.tipo_cliente.unique()}")
```

```
Los tipos de cliente en la base de datos son 4: ['Distribuidor' 'Hospital'
'Farmacia' 'Otro']
```

[8]:
```
# Número de clientes únicos

print(f"El número de clientes únicos en la base de datos son: {len(data.
 ↪cliente_id.unique())}")
```

```
El número de clientes en la base de datos son: 4774
```

[9]:
```
# Número de productos únicos

print(f"El número de productos únicos en la base de datos son: {len(data.
 ↪material_id.unique())}")
```

```
El número de productos únicos en la base de datos son: 1720
```

[16]:
```
# Conversión de la variable periodo de string a timestamp

data['periodo'] = pd.to_datetime(data['periodo'])
data.sort_values(by='periodo', inplace=True)
data.dropna(inplace=True)
data.reset_index(inplace=True)
data.drop('index', axis = 1,inplace=True)
print(f"Los datos del periodo en la base de datos son: {data.periodo.dtype}")
```

```
Los datos del periodo en la base de datos son: datetime64[ns]
```

1.4 Ejemplo de cadena de Markov: cliente_id = 800, material_id = 317.0

Primero, se necesita hacer un nuevo subset agrupando por cliente y producto.

[28]:
```
# Agrupación de la base de datos por cliente_id -> nuevo subset

subset_cliente = data.loc[data['cliente_id'] == 800.0]
subset_cliente_producto = subset_cliente.loc[subset_cliente['material_id'] ==␣
 ↪317.0]

subset_cliente_producto.reset_index(inplace = True)
subset_cliente_producto.drop('index', axis = 1, inplace = True)
print("El subset de la base de datos para el cliente 800 y el producto 317 es␣
 ↪el siguiente. ")
subset_cliente_producto.head()
```

El subset de la base de datos para el cliente 800 y el producto 317 es el siguiente.

[28]:
```
     periodo  cliente_id  material_id tipo_cliente
0 2021-01-01       800.0        317.0     Hospital
1 2021-07-01       800.0        317.0     Hospital
2 2021-09-01       800.0        317.0     Hospital
3 2021-10-01       800.0        317.0     Hospital
4 2021-11-01       800.0        317.0     Hospital
```

Como en la base de datos solo aparecen registros en el mes que compraron los clientes ese producto, se tiene que realizar un registro donde aparezcan todos los meses, sin importar si compró o no.

Para hacer esto, se crea una lista t_i donde su valor es 0 si el cliente compró el producto y 1 en caso contrario.

[32]:
```
# Lista t_i

t = [0]

for x in range(0, len(subset_cliente_producto['periodo'])-2):
    if (subset_cliente_producto['periodo'][x+1] -␣
 ↪subset_cliente_producto['periodo'][x]).days <= 31:
        t.append(0)
    else:
        for _ in range(((subset_cliente_producto['periodo'][x+1] -␣
 ↪subset_cliente_producto['periodo'][x]).days // 30)-1):
            t.append(1)
        t.append(0)
print(f"La lista para representar los estados (compró o no compró) en el subset␣
 ↪es: \n{t}")
```

```
La lista para representar los estados (compró o no compró) en el subset es:
[0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1, 1, 0, 0]
```

Por otro lado, se necesita el tiempo $t_i + 1$, es decir, la lista t_i desplazada por un mes. Esto es, debido a que las cadenas de Markov tienen 'pérdida de memoria' y solo se necesita un paso antes.

[34]:
```
# Lista t_i+1

t_1 = []

for x in range(0, len(subset_cliente_producto['periodo'])-1):
    if (subset_cliente_producto['periodo'][x+1] -␣
 ↪subset_cliente_producto['periodo'][x]).days <= 31:
        t_1.append(0)
    else:
```

```
        for _ in range(((subset_cliente_producto['periodo'][x+1] -␣
 ↪subset_cliente_producto['periodo'][x]).days // 30)-1):
            t_1.append(1)
        t_1.append(0)

print(f"La lista para representar los estados (compró o no compró) en el subset␣
 ↪desplazado un mes después es: \n{t_1}")
```

```
La lista para representar los estados (compró o no compró) en el subset
desplazado un mes después es:
[1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 1, 1, 0, 0, 0]
```

Después estas dos listas con los estados se hacen una base de datos.

[35]:
```
# Nueva base de datos para hacer la matriz de transición

estados = pd.DataFrame()

estados['t'] = t
estados['t_1'] = t_1

Xt = estados['t'][0:-1].reset_index(drop=True).rename('X_t')
Xt_1 = estados['t_1'][1::].reset_index(drop=True).rename('X_t+1')

new_data=pd.concat((Xt, Xt_1), axis=1)
new_data.head()
```

[35]:

	X_t	X_t+1
0	0	1
1	1	1
2	1	1
3	1	1
4	1	0

Utilizando esta nueva base de datos se crea la matriz de transición, ya que es lo único necesario para poder hacerla.

[36]:
```
matriz_transicion = new_data.groupby('X_t').value_counts(normalize=True).
 ↪unstack(level='X_t+1')
matriz_transicion= matriz_transicion.fillna(0)
print(f"La matriz de transición para el cliente 800 y el producto 317 es: ")
matriz_transicion
```

```
La matriz de transición para el cliente 800 y el producto 317 es:
```

[36]:

X_t+1	0	1
X_t		
0	0.818182	0.181818

```
1       0.555556  0.444444
```

Esta matriz indica lo siguiente:

- La probabilidad de que el cliente 800 compre el producto 317 dado que ya lo compró es de 0.8181 (81.81%)
- La probabilidad de que el cliente 800 compre el producto 317 dado que no lo compró es de 0.1818 (18.18%)
- La probabilidad de que el cliente 800 compre el producto 317 dado que no lo compró es de 0.5556 (55.56%)
- La probabilidad de que el cliente 800 no compre el producto 317 dado que no lo compró es de 0.4444 (44.44%)

Estas probabilidades son exactamente las que necesitamos para responder las hipótesis planteadas en el primer reporte, que representan la información que el socio formador quiere saber.

1.5 Automatización del proceso

Se planea hacer una cadena de Markov por cliente por producto y esto resultaría en $8,211,280$ cadenas de Markov. Esto podría ser imposible si no se automatiza el proceso. Afortunadamente, el procedimiento y las herramientas computacionales dan paso a la automatización de la creación de las cadenas de Markov.

[42]:
```python
def matriz_transicion(tipo_cliente, cliente_id, material_id):

    # Subset

    cliente_tipo = data.loc[data['tipo_cliente'] == tipo_cliente]
    pruducto = cliente_tipo.loc[cliente_tipo['material_id'] == material_id]
    id_cliente = pruducto.loc[pruducto['cliente_id'] == cliente_id]
    id_cliente.reset_index(inplace = True)
    id_cliente.drop('index', axis = 1, inplace = True)

    # Fecha de inicio
    fecha_inicio = pd.Timestamp(2021, 1, 1)
    # Fecha de finalización
    fecha_fin = pd.Timestamp(2023, 9, 1)

    frecuencia = pd.DateOffset(months=1)

    fechas = []

    while fecha_inicio <= fecha_fin:
        fechas.append(fecha_inicio)
        fecha_inicio += frecuencia

    fechas
```

```
    periodos = {}

    for k,v in enumerate(fechas):
        periodos[v] = k

    t = [1 for x in range(len(fechas))]

    indices = [periodos[x] for x in id_cliente['periodo']]

    for i in indices:
        t[i] = 0

    estados = pd.DataFrame()

    estados['t'] = t
    estados['t_1'] = t

    Xt = estados['t'][0:-1].reset_index(drop=True).rename('X_t')
    Xt_1 = estados['t_1'][1::].reset_index(drop=True).rename('X_t+1')

    new_data=pd.concat((Xt, Xt_1), axis=1)

    matriz_transicion = new_data.groupby('X_t').value_counts(normalize=True).
↪unstack(level='X_t+1')
    matriz_transicion= matriz_transicion.fillna(0)

    return matriz_transicion
```

1.6 Ejemplos con automatización

1.6.1 Cliente_id = 4769 y producto_id = 17

[43]:
```
matriz_transicion('Distribuidor', 4769.0, 17.0)
```

[43]:
```
X_t+1         0         1
X_t
0      0.896552  0.103448
1      1.000000  0.000000
```

1.6.2 Cliente_id = 2000 y producto_id = 259

[44]:
```
matriz_transicion('Farmacia', 2000.0 ,259.0)
```

[44]:
```
X_t+1         0         1
X_t
0      0.916667  0.083333
```

```
1      0.000000  1.000000
```

2 ¿Qué propiedades deberían tener estas cadenas para asegurar su convergencia?

Para que una cadena de Markov converga necesita cumplir con las siguientes características:

- **Irreducibilidad**: La cadena de Markov debe ser irreducible, lo que significa que desde cualquier estado es posible llegar a cualquier otro estado con una probabilidad positiva en un número finito de pasos.
- **Aperiodicidad**: La cadena debe ser aperiódica, es decir, que los periodos de cada clase tiene que ser 1. Esto significa que no debe seguir un patrón regular o periódico en la transición entre estados.
- **Recurrente**: Los estados de la cadena de Markov deben ser recurrentes positivos. Los estados recurrentes positivos son aquellos desde los cuales eventualmente regresará a sí mismos. Para asegurar la convergencia, es importante que la cadena tenga al menos un estado recurrente positivo.
- **Ergodicidad**: La cadena debe ser ergódica, lo que significa que debe ser posible alcanzar cualquier estado desde cualquier otro estado en un número finito de pasos.

3 Potenciales recomendaciones para PiSA

Con las cadenas de Markov ya se conocen las probabilidades de la desactivación y activación de los productos. Con esta información se pueden dar recomendaciones valiosas al socio formador para optimizar sus procesos y mejorar las decisiones.

Dependiendo de la probabilidad obtenida para cada caso específico, si la probabilidad de que un cliente vuelva a comprar un producto dado que lo compró anteriormente es muy alta, se recomendaría a PiSA mantener la producción de este producto porque es muy probable que siga activo. Si en cambio la probabilidad de que un cliente no vuelva a comprar un producto dado que lo compró anteriormente es muy baja, se recomendaría a PiSA reducir la producción de este producto porque es muy probable que pase al estado de desactivación. También, se podría recomendar cuando aumentar la producción de cierto producto teniendo las probabilides para la reactivación de productos que se encontraban en estado desactivado.

En el ejemplo visto en el reporte para el cliente_id 800 y el producto_id 317, se podrían dar las siguientes recomendaciones:

Probabilidades:

- La probabilidad de que el cliente 800 compre el producto 317 dado que ya lo compró es de 0.8181 (81.81%)
- La probabilidad de que el cliente 800 compre el producto 317 dado que no lo compró es de 0.1818 (18.18%)
- La probabilidad de que el cliente 800 compre el producto 317 dado que no lo compró es de 0.5556 (55.56%)

- La probabilidad de que el cliente 800 no compre el producto 317 dado que no lo compró es de 0.4444 (44.44%)

Recomendaciones:

- En este caso, se recomendaría a PiSA mantener la producción del producto 317 porque la probabilidad de que el cliente lo siga comprando el próximo mes es de 81%, que indica que el producto seguirá activo.