Universidad Nacional Autónoma de México
Facultad de Ingeniería
División de Ingeniería Eléctrica
Ingeniería en Computación

Práctica 3.4
“Asociación”

Integrantes:
- Castelan Ramos Carlos
- Reyes Esquivel Ana Karen
- Rocandio Montiel Jesús Alexis
- Romero López Alexis Uriel

Materia: Minería de Datos
Grupo: 03
Semestre: 2024-1

Fecha de entrega: 26 de octubre2023

Práctica 3.4 – Asociación

Instrucciones.
Realizar la práctica 3.4 de Minería de datos utilizando el algoritmo de Asociación APRIORI y los datos de Transacciones de ventas que se enviaron anteriormente.

Desarrollo:
Disponible en:
https://colab.research.google.com/drive/18b69R8bNtiLA7uS9DbRhp2swIm8WGBUQ?usp=sharing

1.-Caso de Estudio:

Optimización de Promociones de Productos

Descripción:
Mejorar sus estrategias de promoción para aumentar las ventas y la satisfacción de los clientes. Para lograrlo, hemos decidido utilizar el algoritmo Apriori para analizar las transacciones de ventas anteriores y descubrir patrones de compra que indiquen qué productos suelen comprarse juntos.
Nuestro objetivo principal es identificar oportunidades para crear promociones específicas que impulsen las ventas cruzadas y maximicen los ingresos.

Objetivos del Análisis:

- Identificar combinaciones de productos que se compran juntos con mayor frecuencia.
- Descubrir reglas de asociación que indiquen qué productos tienen una alta probabilidad de ser comprados juntos.
- Generar recomendaciones para promociones específicas que involucren productos relacionados.
- Evaluar el impacto de las promociones en el aumento de las ventas cruzadas y la lealtad del cliente.

Datos Disponibles:

Los datos de transacciones de ventas contienen información detallada sobre los productos (Transacciones_ventas.csv).

Proceso de Análisis:

- Preparación de Datos: Se carga el conjunto de datos de ventas en un DataFrame de Pandas.
- Agrupación de Datos: Los datos se agrupan por número de pedido y se crea una lista de productos adquiridos en cada pedido.
- Codificación de Transacciones: Las compras se codifican en forma de matriz binaria para aplicar el algoritmo Apriori. Esto significa que cada producto se convierte en una columna y se marca como 1 si se encuentra en una transacción y 0 si no.
- Identificación de Itemsets Frecuentes: Se utiliza el algoritmo Apriori para identificar conjuntos de productos que se compran con cierta frecuencia (definida por un umbral de soporte mínimo).
- Generación de Reglas de Asociación: Se crean reglas de asociación que indican la probabilidad de comprar un producto dado ciertos productos ya en el carrito.

- Selección de Reglas Estratégicas: Se seleccionan reglas estratégicas que pueden utilizarse para tomar decisiones comerciales, como la promoción de productos relacionados.

Acciones Propuestas:

- Crear promociones especiales que ofrezcan descuentos en productos que se compran juntos con frecuencia.
- Diseñar campañas de marketing dirigidas a resaltar los productos relacionados y promocionarlos como un paquete.
- Medir el impacto de las promociones en las ventas cruzadas y la satisfacción del cliente.

Resultados Esperados:

Se espera que este análisis proporcione una comprensión más profunda de los patrones de compra de sus clientes. Esto permitirá la creación de estrategias de promoción más efectivas que aumenten las ventas cruzadas y mejoren la rentabilidad. Además, se espera que las recomendaciones resultantes aumenten la satisfacción de los clientes.

2.-Variables a elegir:

- Pedido
- Producto
- Descripción

Elegimos estas variables para agrupar los datos en función de los números de pedido y crea una lista que contiene la descripción de los productos comprados en cada pedido.

3.-Evaluación de las métricas de asociación:

```
from google.colab import data_table
from vega_datasets import data
import seaborn as sb
from sklearn.cluster import KMeans
from sklearn.metrics import pairwise_distances_argmin_min
from mlxtend.preprocessing import TransactionEncoder

# ----------------------------
# BASE
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import datetime as dt

# ----------------------------
# TRANSACTION ENCODER
from mlxtend.preprocessing import TransactionEncoder

# ----------------------------
# APRIORI FUNCTION
from mlxtend.frequent_patterns import apriori, association_rules
```

```
# ----------------------------
# ITERTOOLS
import itertools

# ----------------------------
# CONFIGURATION
import warnings
warnings.filterwarnings("ignore")
warnings.simplefilter(action='ignore', category=FutureWarning)

pd.set_option('display.max_columns', None)
pd.options.display.float_format = '{:.2f}'.format
```

Ilustración 1

Importamos las bibliotecas necesarias para el análisis de datos, de las bibliotecas más relevantes es ‘mlxtend.preprocessing’ es una biblioteca para preprocesas datos, especialmente para algoritmos de aprendizaje automático. TransactionEncoder se usa para procesar transacciones.

Para las funciones Apriori importamos dos funciones de la biblioteca ‘mlxtend.frequent_patterns’. La función ‘aprori’ se utiliza para realizar el análisis de reglas de asociación utilizando el algoritmo Apriori, y ‘association_rules’ se usa para generar reglas de asociación a partir de los resultados de Apriori.

En esta sección la última parte del código es de configuración en donde se suprimen algunas advertencias y configuramos el formato de visualización.

Es importante mencionar que uno de los primeros análisis que se conviene hacer es explorar su **contenido y tamaño.**

```
df = pd.read_csv(r"Transacciones_ventas.csv", index_col=0, encoding='latin-1')
dataframe = df
print(dataframe)
```

```
        Producto                Descripcion
Pedido
70972         12  PECHUGA DE PAVO 250GR SRF
71013         12  PECHUGA DE PAVO 250GR SRF
71018         12  PECHUGA DE PAVO 250GR SRF
71029         12  PECHUGA DE PAVO 250GR SRF
71100         12  PECHUGA DE PAVO 250GR SRF
...          ...                       ...
94962      17676       JAMON YORK 250GR BER
95280      17676       JAMON YORK 250GR BER
95447      17676       JAMON YORK 250GR BER
95469      17676       JAMON YORK 250GR BER
95501      17676       JAMON YORK 250GR BER

[20807 rows x 2 columns]
```

Ilustración 2

En este fragmento de código se hace la carga de nuestro archivo CSV “Transacciones_ventas.csv” con el que hemos venido trabajando pero para este proyecto modificamos el archivo y solo dejamos las tres columnas necesarias para que algoritmo fuera más rápido y también evitar problemas con las asociaciones, la carga se realizó mediante un DataFrame de Panda y luego lo imprimos para ver que la caga de datos se haya realizado de forma correcta.

```
# Agrupación por producto y descripción
df.groupby('Producto')['Descripcion'].apply(list)
```

```
Producto
12       [PECHUGA DE PAVO 250GR SRF, PECHUGA DE PAVO 25...
25       [DELI FRESH PECHUGA DE PAVO HORNEADA, DELI FRE...
26       [DELI FRESH PECHUGA DE PAVO AHUMADA, DELI FRES...
30       [JAMON REAL DE PAVO 300 G MARCA HID, JAMON REA...
116      [PECHUG DE PAVO VIRGINIA 250GR MARCA JAL, PECH...
                              ...
17612    [SALCHICHA SCHUBLING 340 GR BER, SALCHICHA SCH...
17618    [JAMON ESTILO HOLANDES 300GR BER, JAMON ESTILO...
17626    [PECHUGA PAV COCIDA/AHUM 300GR BER, PECHUGA PA...
17675    [PECHUGA SIN SAL 200 GR BER, PECHUGA SIN SAL 2...
17676    [JAMON YORK 250GR BER, JAMON YORK 250GR BER, J...
Name: Descripcion, Length: 133, dtype: object
```

Ilustración 3

Aquí mediante la función ‘groupby’ de Pandas agrupamos datos en un DataFrame por la columna ‘Producto’ y luego aplicamos la función ‘list’ a los valores de la columna ‘Descripcion’ dentro de cada grupo. Esto generará una lista de descripciones de productos para cada producto en el DataFrame.

En resumen, identifica cada producto único en el Data Feame y lista todas las descripciones asociadas a ese producto en particular. Esto es útil para conocer las diversas descripciones que pueden estar vinculadas a un mismo producto en los datos.

```
# Agrupación por pedido y productos adquiridos
df.groupby('Pedido')['Producto'].apply(list)
```

```
Pedido
70855                          [26, 943, 985]
70869                             [583, 909]
70872     [30, 585, 606, 5152, 5154, 5157]
70874                              [30, 909]
70877                             [159, 294]
                          ...
117328                                [234]
117333                   [135, 196, 5170]
117335                   [196, 234, 5170]
117338                                [763]
117442                                [234]
Name: Producto, Length: 8860, dtype: object
```

Ilustración 4

En esta sección se agrupan los datos por pedidos y crea una lista que contiene los productos adquiridos en cada pedido.

```
# Agrupación por pedido y descripcioón
df.groupby('Pedido')['Descripcion'].apply(list)
```

```
Pedido
70855     [DELI FRESH PECHUGA DE PAVO AHUMADA, SALCHICHA...
70869     [JAMON DE PAVO VIRGINIA MARCA JAL 290 GR, SALC...
70872     [JAMON REAL DE PAVO 300 G MARCA HID, CHORIZO P...
70874     [JAMON REAL DE PAVO 300 G MARCA HID, SALCHICHA...
70877     [PECHUGA DE PAVO BALANCE 250GR, SALCHICHA PAVO...
                                ...
117328                [PIERNA AHUMADA EDICION ESPECIAL SRF]
117333    [PIERNA DE CERDO AHUMADA CHMX 3.5KG, PIERNA AH...
117335    [PIERNA AHUMADA NAVIDADSRF, PIERNA AHUMADA EDI...
117338                          [CHORIZO PARA ASAR 400GR CHX]
117442                [PIERNA AHUMADA EDICION ESPECIAL SRF]
Name: Descripcion, Length: 8860, dtype: object
```

Ilustración 5

Se agrupan los datos por pedidos y se crea una lista que contiene las descripciones correspondientes de los productos adquiridos en cada pedido.

```
# Codificar las compras en forma de matriz binaria
transacciones = df.groupby('Pedido')['Descripcion'].apply(list).to_list()

# Entrenar el objeto TransactionEncoder y transformar los datos
encoder = TransactionEncoder()
transacciones_encoded = encoder.fit(transacciones).transform(transacciones)
transacciones_encoded = pd.DataFrame(transacciones_encoded, columns=encoder.columns_)
transacciones_encoded.head(3)
```

	BOLA DE CERVEZA REB 200 GRS TGM	BRESAOLA REB 100G TGM	CHISTORRA 300G TGM	CHORIZO CERDO 150GR MARCA JAL	CHORIZO CERDO RANCHERO CHX	CHORIZO CERDO RANCHERO PACIFICO 250 GR CHX	CHORIZO DE CERDO MARCA JAL 200G	CHORIZO DE POLLO 240GR SRF	CHORIZO ESPAÃOL TGM 100GR	CHORIZO MARCA MON 40G	CHORIZO MARCA SON 100G	CHORIZO PARA ASAR 400GR CHX	CHORIZO PAVO 200GR MARCA JAL	CHORIZO RANCHERO MARCA PUE 100 GR	CHULETA AHUMADA CHX	DELI FRESH PECHUGA DE PAVO AHUMADA	DELI FRESH PECHUGA DE PAVO HORNEADA
0	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	True	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
2	True	False	False	False	False	False	False	False	False	False	False	False	True	False	False	False	False

Ilustración 6

El siguiente análisis básico consiste en identificar cuáles son los ítems más frecuentes para eso obtenemos la matriz binaria. Primero, se agrupan las descripciones de productos por pedido y se convierten en una lista de listas. Cada lista interna representa las descripciones de productos en un pedido específico.
Después transformamos las listas de descripciones en una matriz binaria donde cada fila representa un pedido y cada columna representa un producto. Si un producto está presente en un pedido, la celda correspondiente tendrá un valor ‘True’; de lo contrario será ‘False’.

```
# Porcentaje de transacciones en las que aparece cada producto (top 5)
transacciones_encoded.mean(axis = 0).sort_values(ascending = False).head(5)
```

```
PECHUGA DE PAVO BALANCE 250GR            0.20
CHORIZO PAVO 200GR MARCA JAL             0.15
JAMON DE PAVO VIRGINIA MARCA JAL 290 GR  0.12
PECHUG DE PAVO VIRGINIA 250GR MARCA JAL  0.11
SALCHICHA DE PAVO 500G BALANCE SRF       0.11
dtype: float64
```

Ilustración 7

Aquí obtenemos los 5 productos que aparecen con más frecuencia en las cestas de compra pero esto no significa que sean los productos con mayor volumen de ventas, ya que no se está tomando en cuenta cuántas unidades se venden, sino con qué frecuencia.

```
# Distribución del número de ítems por compra
display(df.groupby('Pedido')['Producto'].size().describe(percentiles=[.25, .5, .75, .9]))

fig, ax = plt.subplots(figsize=(7, 3))
df.groupby('Pedido')['Producto'].size().plot.hist(ax=ax)
ax.set_title('Distribución del tamaño de las compras');
ax.set_xlabel('Número de productos');
```

```
count   8860.00
mean       2.35
std        1.41
min        1.00
25%        1.00
50%        2.00
75%        3.00
90%        4.00
max       13.00
Name: Producto, dtype: float64
```

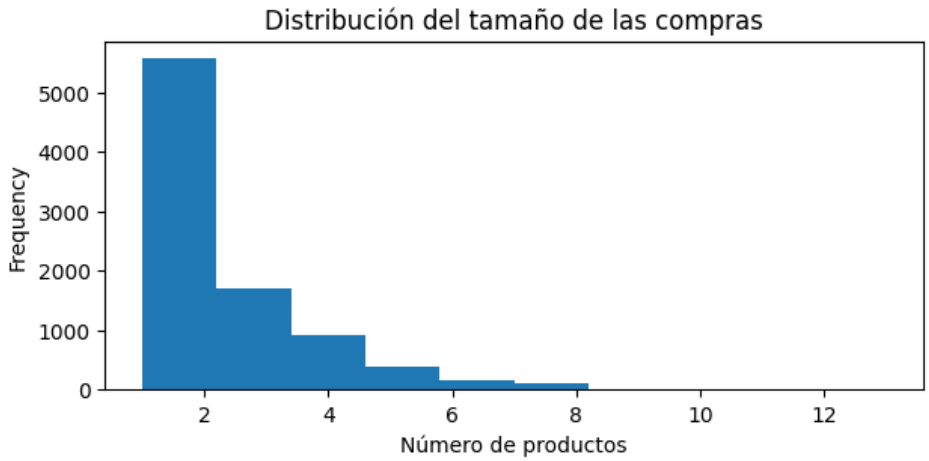

Ilustración 8

Es importante analizar la frecuencia de la distribución del tamaño de las compras, de esta gráfica podemos observar que la gran mayoría de los clientes compra entre 1 y 3 productos.

4.-Impacto asociativo obtenido en las variables como resultado de la prueba

```
# Identificación de itemsets frecuentes
soporte = 30 / transacciones_encoded.shape[0]
itemsets_frecuentes = apriori(transacciones_encoded, min_support=soporte, use_colnames=True)

# Crear reglas de asociación (confianza mínima del 70%)
confianza = 0.7 # confianza mínima del 70% para que una regla sea selecionada
reglas = association_rules(itemsets_frecuentes, metric="confidence", min_threshold=0.7)
print(f"Número de reglas generadas: {len(reglas)}")
reglas.sort_values(by='confidence').head(5)
```

	antecedents	consequents	antecedent support	consequent support	support	confidence
1	(SALCHICHA TURKEY LINE 1KG MARCA JAL, TOCINO 2...	(JAMON DE PAVO VIRGINIA MARCA JAL 290 GR)	0.01	0.12	0.00	0.73
0	(SALCHICHA TURKEY LINE 1KG MARCA JAL, CHORIZO ...	(JAMON DE PAVO VIRGINIA MARCA JAL 290 GR)	0.01	0.12	0.01	0.86
2	(SALCHICHA TURKEY LINE 1KG MARCA JAL, CHORIZO ...	(JAMON DE PAVO VIRGINIA MARCA JAL 290 GR)	0.00	0.12	0.00	0.97

Ilustración 9

Para generar las reglas de asociación hicimos uso de los items frecuentes mediante el uso de un soporte de 30, es decir aquellos productos que fueron comprados al menos 30 veces, pasando este valor a la función 'generate_rules' que es la que se encarga de generar reglas de asociación y selecciona aquellas que alcanzan un valor mínimo de confianza.
Es entonces que observamos que se generan tres reglas de asociación, donde para la regla 1 la confianza de 0.73 indica que en el 73% de los casos en que se compran los antecedentes, también se compra el consecuente. El valor de Lift (6.32) sugiere que esta asociación es más fuerte de lo que se esperaría por azar. En la regla 0 existe un 83% de confianza, pero la más interesante es la Regla 2, ya que se observa una fuerte asociación entre los antecedentes y el consecuente. La confianza de 0.97 indica que en el 97% de los casos en que se compran los antecedentes, también se compra el consecuente. El valor de Lift (8.34) sugiere una asociación significativa y positiva entre los productos. Esto podría ser útil para estrategias de ventas o recomendaciones de productos.
Por otra parte, observamos que la relación consecuente siempre es 'Jamón de Pavo Virginia Marca JAL 290 gr', es decir las tres reglas de asociación se prevé y se propone que siempre se va a comprar este producto.

5.- Selección de reglas significativas.

```
# Seleccionar reglas que tienen "JAMON DE PAVO VIRGINIA MARCA JAL 290 GR" en el consecuente
mask = reglas['consequents'].map(lambda x: 'JAMON DE PAVO VIRGINIA MARCA JAL 290 GR' in x)
reglas.loc[mask]
```

Ilustración

	antecedents	consequents	antecedent support	consequent support	support	confidence
0	(SALCHICHA TURKEY LINE 1KG MARCA JAL, CHORIZO ...	(JAMON DE PAVO VIRGINIA MARCA JAL 290 GR)	0.01	0.12	0.01	0.86
1	(SALCHICHA TURKEY LINE 1KG MARCA JAL, TOCINO 2...	(JAMON DE PAVO VIRGINIA MARCA JAL 290 GR)	0.01	0.12	0.00	0.73
2	(SALCHICHA TURKEY LINE 1KG MARCA JAL, CHORIZO ...	(JAMON DE PAVO VIRGINIA MARCA JAL 290 GR)	0.00	0.12	0.00	0.97

Ilustración 10

Dada la cantidad de reglas obtenidas donde el ítem consecuente de asociación siempre es el mismo Jamón de Pavo Virginia Marca JAL 290 gr' podemos decir que el filtrado queda ya marcado de forma inmediata.

```
antecedents = reglas.loc[mask, 'antecedents'].to_list()
set().union(*antecedents)
```

```
{'CHORIZO PAVO 200GR MARCA JAL',
 'SALCHICHA TURKEY LINE 1KG MARCA JAL',
 'TOCINO 250GR MARCA JAL'}
```

Ilustración 11

Para esto, obtenemos los ítems antecedentes que recomiendan esta regla.

6.- Conclusiones de prueba:

Basándonos en las reglas de asociación a priori generadas a partir de las transacciones de su tienda, donde se observa una asociación entre los productos como salchicha, chorizo y tocino en los antecedentes como productos frecuentes, y jamón en los consecuentes, podemos ofrecer las siguientes recomendaciones para su tienda:

- Promoción Cruzada: Dado que ha habido una asociación significativa entre los productos como salchicha, chorizo y tocino, y el jamón en las compras de sus clientes, sería una buena estrategia realizar promociones cruzadas. Por ejemplo, puede considerar ofrecer descuentos especiales o paquetes que incluyan estos productos juntos, como un "Desayuno Gourmet" que contenga jamón, salchicha, chorizo y tocino. Esto podría incentivar a los clientes a comprar más de estos productos.
- Ubicación Estratégica en la Tienda: Coloque estos productos en ubicaciones estratégicas dentro de su tienda. Por ejemplo, agrupe estos productos en una sección de "Desayuno" o "Ingredientes para Sándwiches" para facilitar la compra conjunta. También, asegúrese de que el jamón esté cerca de los otros productos mencionados para que los clientes puedan encontrarlos fácilmente.
- Ofertas Especiales: Ofrezca ofertas especiales, descuentos o promociones temporales en productos como jamón cuando se compren en conjunto con salchicha, chorizo o tocino. Esto puede incentivar a los clientes a llevar más productos y aumentar sus ventas.
- Programas de Fidelización: Considere la implementación de un programa de fidelización donde los clientes que compren estos productos en conjunto con el jamón acumulen puntos o reciban recompensas especiales. Esto podría aumentar la lealtad de los clientes y promover compras repetidas.
- Monitoreo de Inventario: Dado que estos productos están asociados, asegúrese de mantener un inventario adecuado de jamón, salchicha, chorizo y tocino para satisfacer la demanda cuando se promocionen juntos. Evite situaciones de falta de stock, ya que esto podría afectar negativamente la experiencia del cliente.
- Personalización en Marketing: Utilice los datos de las transacciones para personalizar su estrategia de marketing. Puede enviar correos electrónicos o anuncios a los clientes que hayan comprado productos como jamón en el pasado, sugiriéndoles productos relacionados o promociones.

Referencias.

- *Reglas de asociación con Python*. (s. f.). https://cienciadedatos.net/documentos/py50-reglas-de-asociacion-python
- Kotsiantis, S., & Kanellopoulos, D. (2006). Association rules mining: A recent overview. *GESTS International Transactions on Computer Science and Engineering*, *32*(1), 71-82.
- Hegland, M. (2007). The apriori algorithm–a tutorial. *Mathematics and computation in imaging science and information processing*, 209-262.
- Bodon, F. (2003, November). A fast APRIORI implementation. In *FIMI* (Vol. 3, p. 63).