Universidad Nacional Autónoma de México

Facultad de Ingeniería

PRÁCTICA 11

REGLAS DE ASOCIACIÓN

Minería de Datos

Profesor:

Dr. Molero Castillo Guillermo Gilberto

Grupo 1

Alumna:

Monroy Velázquez Alejandra Sarahí

No. Cuenta: 314000417

OBJETIVO

Analizar las transacciones y obtener reglas significativas (patrones) de los productos vendidos en un comercio minorista en Francia. Los datos son transacciones de un comercio de un periodo de una semana (7 días).

DESARROLLO

El conjunto de datos con el que se trabajará corresponde a registros varios de datos de productos vendidos.

Primero comenzamos la importación de bibliotecas correspondientes que nos ayudarán para la realización del código, las cuales son *pandas* para la manipulación y análisis de datos, *numpy* para crear vectores y matrices, *matplotlib* para la generación de gráficas, así como *seaborn* para la visualización de datos. En esta práctica agregamos *apyori* el cual es un paquete que se instala mediante *pip*. Por último, la biblioteca files para subir el archivo *csv*.

Una vez importadas, el *dataframe* se lee y se despliega en pantalla:

```
[1] !pip install apyori  # pip es un administrador de paquetes de Python. Se instala el paquete Apyori
```

```
Collecting apyori
  Downloading apyori-1.1.2.tar.gz (8.6 kB)
Building wheels for collected packages: apyori
  Building wheel for apyori (setup.py) ... done
  Created wheel for apyori: filename=apyori-1.1.2-py3-none-any.whl size=5974 sha256=85d679fd577ef917c47c5ef8
  Stored in directory: /root/.cache/pip/wheels/cb/f6/e1/57973c631d27efd1a2f375bd6a83b2a616c4021f24aab84080
Successfully built apyori
Installing collected packages: apyori
Successfully installed apyori-1.1.2
```

```
[2] import pandas as pd                # Para la manipulación y análisis de los datos
    import numpy as np                 # Para crear vectores y matrices n dimensionales
    import matplotlib.pyplot as plt    # Para la generación de gráficas a partir de los datos
    from apyori import apriori
```

```
[3] from google.colab import files
    files.upload()
```

Elegir archivos store_data.csv

- **store_data.csv**(application/vnd.ms-excel) - 302908 bytes, last modified: 28/10/2021 - 100% done

```
Saving store_data.csv to store_data.csv
{'store_data.csv': b'shrimp,almonds,avocado,vegetables mix,green grapes,whole weat flour,yams,cottage cheese,energy drink,t
```

```
[4] DatosTransacciones = pd.read_csv('store_data.csv')
    DatosTransacciones
```

	shrimp	almonds	avocado	vegetables mix	green grapes	whole weat flour	yams	cottage cheese	energy drink	tomato juice	low fat yogurt	green tea	honey	salad
0	burgers	meatballs	eggs	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	chutney	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	turkey	avocado	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	mineral water	milk	energy bar	whole wheat rice	green tea	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	low fat	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Ahora que el *dataframe* está cargado, comenzamos con los pasos vistos en clase.

En este paso observamos dos cosas muy importantes:

- Que el encabezado es la primera transacción, lo cual no es ideal.
- 'NaN' indica que ese producto no fue comprado en esa transacción.

Por lo que lo siguiente será remover el encabezado para que los datos de las transacciones sean uniformes:

```
[5] DatosTransacciones = pd.read_csv('store_data.csv', header=None)
    DatosTransacciones.head(5)
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
0	shrimp	almonds	avocado	vegetables mix	green grapes	whole weat flour	yams	cottage cheese	energy drink	tomato juice	low fat yogurt	green tea	honey	salad	mineral water	s
1	burgers	meatballs	eggs	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2	chutney	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
3	turkey	avocado	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
4	mineral water	milk	energy bar	whole wheat rice	green tea	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

Procesamiento de los datos

Antes de ejecutar el algoritmo, es recomendable observar la distribución de la frecuencia de los elementos. Crearemos una sola lista, incluyendo todas las transacciones en ella, luego crearemos una matriz usando la lista y

agregaremos una columna extra de nombre “Frecuencia”, esta será llenada con ceros, y luego agruparemos los elementos de modo que podamos ver en orden ascendente las películas y la frecuencia de estas.

```
[6]  #Se incluyen todas las transacciones en una sola lista
     Transacciones = DatosTransacciones.values.reshape(-1).tolist() #-1 significa 'dimensión desconocida'

     #Se crea una matriz (dataframe) usando la lista y se incluye una columna 'Frecuencia'
     Lista = pd.DataFrame(Transacciones)
     Lista['Frecuencia'] = 1

     #Se agrupa los elementos
     Lista = Lista.groupby(by=[0], as_index=False).count().sort_values(by=['Frecuencia'], ascending=True) #Conteo
     Lista['Porcentaje'] = (Lista['Frecuencia'] / Lista['Frecuencia'].sum()) #Porcentaje
     Lista = Lista.rename(columns={0 : 'Item'})

     #Se muestra la lista
     Lista
```

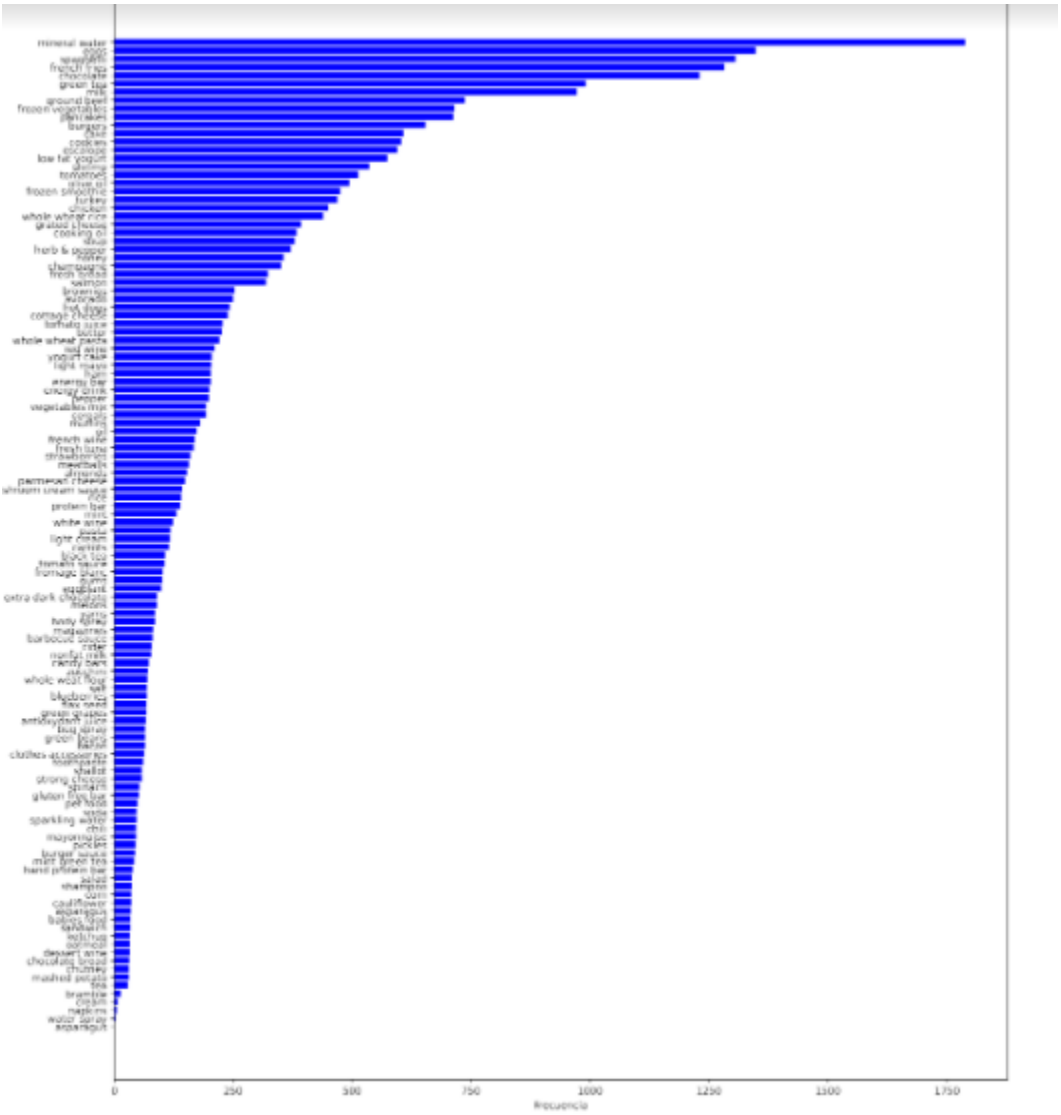
	Item	Frecuencia	Porcentaje
0	asparagus	1	0.000034
112	water spray	3	0.000102
77	napkins	5	0.000170
34	cream	7	0.000238
11	bramble	14	0.000477
...	...	...	...
25	chocolate	1230	0.041889
43	french fries	1282	0.043660
100	spaghetti	1306	0.044478
37	eggs	1348	0.045908
72	mineral water	1788	0.060893

120 rows × 3 columns

Como vemos *asparagus* solo tiene un 1 de frecuencia, lo que quiere decir que solo aparece en una transacción, lo que la convierte en el producto menos comprado de todos, a diferencia de *mineral wáter* el cual tiene 1788 de frecuencia, lo que lo convierte en el producto más comprado de todos.

Otra opción para observar las frecuencias de los productos es generar un gráfico de barras con base a la lista anterior:

```
[7]  # Se genera un gráfico de barras
     plt.figure(figsize=(16,20), dpi=300)
     plt.ylabel('Item')
     plt.xlabel('Frecuencia')
     plt.barh(Lista['Item'], width=Lista['Frecuencia'], color='blue')
     plt.show()
```



Dado que los datos actuales están en un *dataframe* de Pandas, los convertiremos a una lista ya que la función de Apriori de Python requiere que el conjunto de datos tenga la forma de una lista de listas, donde cada transacción es una lista interna dentro de una gran lista.

```
[8]  #Se crea una lista de listas a partir del dataframe y se remueven los 'NaN'
     #level=0 especifica desde el primer índice
     TransaccionesLista = DatosTransacciones.stack().groupby(level=0).apply(list).tolist()
     TransaccionesLista
```

```
['green tea'],
['cookies'],
['french fries', 'cookies'],
['milk', 'butter', 'eggs'],
['eggs', 'mushroom cream sauce', 'low fat yogurt'],
['eggs', 'green tea'],
['mineral water', 'whole wheat rice'],
['shrimp',
 'frozen vegetables',
 'parmesan cheese',
 'mineral water',
 'whole wheat rice',
 'cake'],
['whole wheat rice', 'pancakes'],
['frozen vegetables',
 'spaghetti',
 'olive oil',
 'butter',
 'salmon',
 'oil',
 'cooking oil',
 'frozen smoothie',
 'cauliflower'],
['green tea'],
['fresh tuna', 'eggs', 'escalope', 'strawberries'],
```

Aplicación del algoritmo Apriori

Configuración 1

Obtenemos las reglas para aquellos artículos que se compran al menos 5 veces al día, entonces, 5 x 7 = 35 veces en una semana, entonces:

- El soporte mínimo se calcula de 35/7500 = 0.0045 (0.45%).
- La confianza mínima para las reglas de 20%
- La elevación de 3.

```
[9]  ReglasC1 = apriori(TransaccionesLista,
                        min_support=0.0045,
                        min_confidence=0.2,
                        min_lift=3)
```

Convertimos las reglas a una lista para una mejor observación de estas. En total se encuentran 24 reglas.

```
[10] ResultadosC1 = list(ReglasC1)
     print(len(ResultadosC1)) #Total de reglas encontradas
```

```
24
```

```
[11] ResultadosC1
```

```
[RelationRecord(items=frozenset({'light cream', 'chicken'}), support=0.004532728969470737, ordered_statistics=[OrderedStatistic(items
 RelationRecord(items=frozenset({'mushroom cream sauce', 'escalope'}), support=0.005732568990801226, ordered_statistics=[OrderedStati
 RelationRecord(items=frozenset({'escalope', 'pasta'}), support=0.005865884548726837, ordered_statistics=[OrderedStatistic(items_base
 RelationRecord(items=frozenset({'ground beef', 'herb & pepper'}), support=0.015997866951073192, ordered_statistics=[OrderedStatistic
 RelationRecord(items=frozenset({'ground beef', 'tomato sauce'}), support=0.005332622317024397, ordered_statistics=[OrderedStatistic(
 RelationRecord(items=frozenset({'whole wheat pasta', 'olive oil'}), support=0.007998933475536596, ordered_statistics=[OrderedStatist
 RelationRecord(items=frozenset({'shrimp', 'pasta'}), support=0.005065991201173177, ordered_statistics=[OrderedStatistic(items_base=f
 RelationRecord(items=frozenset({'frozen vegetables', 'chocolate', 'shrimp'}), support=0.005332622317024397, ordered_statistics=[Orde
 RelationRecord(items=frozenset({'ground beef', 'spaghetti', 'cooking oil'}), support=0.004799360085321957, ordered_statistics=[Order
 RelationRecord(items=frozenset({'frozen vegetables', 'ground beef', 'spaghetti'}), support=0.008665511265164644, ordered_statistics=
 RelationRecord(items=frozenset({'frozen vegetables', 'milk', 'olive oil'}), support=0.004799360085321957, ordered_statistics=[Ordere
 RelationRecord(items=frozenset({'frozen vegetables', 'mineral water', 'shrimp'}), support=0.007199040127982935, ordered_statistics=[
```

Podemos imprimir regla por regla para tener mayor detalle de cada una y poder analizarlas por separado, por ejemplo, imprimiremos la primer regla:

```
[12] print(ResultadosC1[0])
```

```
RelationRecord(items=frozenset({'light cream', 'chicken'}), support=0.004532728969470737, ordered_statistics=[OrderedStat
```

La primera regla contiene dos elementos *chicken* y *light cream* que exclusivamente se compran juntos. Esto tiene sentido, las personas que compran crema ligera tienen cuidado con lo que comen, por lo que, es probable que compren pollo, en lugar de carne roja.

El soporte es de 0.0045, la confianza de 0.2905, la elevación de 4.84, esto es, 4.84 veces más posibilidad de que compren crema ligera.

A pesar de que imprimimos una sola regla, la observación de esta es un poco difícil, por lo que podemos generar código que nos ayude a una mejor visualización de todas las reglas:

```
[15] for item in ResultadosC1:
        #El primer índice de la lista
        Emparejar = item[0]
        items = [x for x in Emparejar]
        print("Regla: " + str(item[0]))

        #El segundo índice de la lista
        print("Soporte: " + str(item[1]))

        #El tercer índice de la lista
        print("Confianza: " + str(item[2][0][2]))
        print("Lift: " + str(item[2][0][3]))
        print("====================================")
```

```
Regla: frozenset({'light cream', 'chicken'})
Soporte: 0.004532728969470737
Confianza: 0.29059829059829057
Lift: 4.84395061728395
====================================
Regla: frozenset({'mushroom cream sauce', 'escalope'})
Soporte: 0.005732568990801226
Confianza: 0.3006993006993007
Lift: 3.790832696715049
====================================
Regla: frozenset({'escalope', 'pasta'})
Soporte: 0.005865884548726837
Confianza: 0.3728813559322034
Lift: 4.700811850163794
====================================
Regla: frozenset({'ground beef', 'herb & pepper'})
Soporte: 0.015997866951073192
Confianza: 0.3234501347708895
Lift: 3.2919938411349285
====================================
Regla: frozenset({'ground beef', 'tomato sauce'})
Soporte: 0.005332622317024397
Confianza: 0.3773584905660377
Lift: 3.840659481324083
====================================
Regla: frozenset({'whole wheat pasta', 'olive oil'})
Soporte: 0.007998933475536596
Confianza: 0.2714932126696833
Lift: 4.122410097642296
====================================
Regla: frozenset({'shrimp', 'pasta'})
Soporte: 0.005065991201173177
Confianza: 0.3220338983050847
Lift: 4.506672147735896
====================================
Regla: frozenset({'frozen vegetables', 'chocolate', 'shrimp'})
Soporte: 0.005332622317024397
Confianza: 0.23255813953488375
Lift: 3.2545123221103784
====================================
Regla: frozenset({'ground beef', 'spaghetti', 'cooking oil'})
Soporte: 0.004799360085321957
Confianza: 0.5714285714285714
Lift: 3.2819951870487856
====================================
Regla: frozenset({'frozen vegetables', 'ground beef', 'spaghetti'})
Soporte: 0.008665511265164644
Confianza: 0.31100478468899523
Lift: 3.165328208890303
====================================
Regla: frozenset({'frozen vegetables', 'milk', 'olive oil'})
Soporte: 0.004799360085321957
Confianza: 0.20338983050847456
Lift: 3.088314005352364
====================================
Regla: frozenset({'frozen vegetables', 'mineral water', 'shrimp'})
Soporte: 0.007199040127982935
Confianza: 0.30508474576271183
Lift: 3.200616332819722
====================================
```

```
Regla: frozenset({'frozen vegetables', 'spaghetti', 'olive oil'})
Soporte: 0.005732568990801226
Confianza: 0.20574162679425836
Lift: 3.1240241752707125
====================================
Regla: frozenset({'frozen vegetables', 'spaghetti', 'shrimp'})
Soporte: 0.005999200106652446
Confianza: 0.21531100478468898
Lift: 3.0131489680782684
====================================
Regla: frozenset({'frozen vegetables', 'spaghetti', 'tomatoes'})
Soporte: 0.006665777896280496
Confianza: 0.23923444976076558
Lift: 3.4980460188216425
====================================
Regla: frozenset({'ground beef', 'spaghetti', 'grated cheese'})
Soporte: 0.005332622317024397
Confianza: 0.3225806451612903
Lift: 3.283144395325426
====================================
Regla: frozenset({'ground beef', 'mineral water', 'herb & pepper'})
Soporte: 0.006665777896280496
Confianza: 0.39062500000000006
Lift: 3.975682666214383
====================================
Regla: frozenset({'ground beef', 'spaghetti', 'herb & pepper'})
Soporte: 0.006399146780429276
Confianza: 0.3934426229508197
Lift: 4.004359721511667
====================================
 Regla: frozenset({'ground beef', 'milk', 'olive oil'})
 Soporte: 0.004932675643247567
 Confianza: 0.22424242424242427
 Lift: 3.40494417862839
 ====================================
 Regla: frozenset({'ground beef', 'spaghetti', 'shrimp'})
 Soporte: 0.005999200106652446
 Confianza: 0.5232558139534884
 Lift: 3.005315360233627
 ====================================
 Regla: frozenset({'spaghetti', 'milk', 'olive oil'})
 Soporte: 0.007199040127982935
 Confianza: 0.20300751879699247
 Lift: 3.0825089038385434
 ====================================
 Regla: frozenset({'soup', 'mineral water', 'olive oil'})
 Soporte: 0.005199306759098787
 Confianza: 0.22543352601156072
 Lift: 3.4230301186492245
 ====================================
 Regla: frozenset({'spaghetti', 'pancakes', 'olive oil'})
 Soporte: 0.005065991201173177
 Confianza: 0.20105820105820105
 Lift: 3.0529100529100526
 ====================================
 Regla: frozenset({'frozen vegetables', 'mineral water', 'spaghetti', 'milk'})
 Soporte: 0.004532728969470737
 Confianza: 0.28813559322033894
 Lift: 3.0228043143297376
 ====================================
```

Configuración 2

Obtenemos las reglas para aquellos artículos que se compran al menos 30 veces al día, entonces, 30 x 7 = 210 veces en una semana, entonces:

- El soporte mínimo se calcula de 210/7500 = 0.028 (2.8%).
- La confianza mínima para las reglas de 25%.
- La elevación mayor a 1.

```
[16] ReglasC2 = apriori(TransaccionesLista,
                        min_support=0.028,
                        min_confidence=0.25,
                        min_lift = 1.01)
```

Convertimos las reglas a una lista para una mejor observación de estas. En total se encuentran diez reglas.

```
[17] ResultadosC2 = list(ReglasC2)
     print(len(ResultadosC2))

     10
```

```
[18] ResultadosC2

     [RelationRecord(items=frozenset({'eggs', 'burgers'}), support=0.02879616051193174, ordered_statistics=[Order
      RelationRecord(items=frozenset({'chocolate', 'mineral water'}), support=0.05265964538061592, ordered_statis
      RelationRecord(items=frozenset({'mineral water', 'eggs'}), support=0.05092654312758299, ordered_statistics=
      RelationRecord(items=frozenset({'frozen vegetables', 'mineral water'}), support=0.03572856952406346, ordere
      RelationRecord(items=frozenset({'ground beef', 'mineral water'}), support=0.040927876283162246, ordered_sta
      RelationRecord(items=frozenset({'ground beef', 'spaghetti'}), support=0.03919477403012932, ordered_statisti
```

Podemos imprimir regla por regla para tener mayor detalle de cada una y poder analizarlas por separado, por ejemplo, imprimiremos la primer regla:

```
[19] print(ResultadosC2[0])

     RelationRecord(items=frozenset({'eggs', 'burgers'}), support=0.02879616051193174, ordered_statistics=[OrderedSt
```

La primera regla contiene dos elementos: hamburguesas y huevos que comúnmente se compran juntos. Tiene sentido, algunas personas que compran hamburguesas consumen también huevos, como comida de preparación rápida.

El soporte es de 0.028 (2.8%), la confianza de 0.33 (33%), la elevación de 1.83, esto es, hay casi 2 veces más probabilidades de que cuando se compre posibilidades se compre también huevos.

A pesar de que imprimimos una sola regla, la observación de esta es un poco difícil, por lo que, como en la configuración anterior, podemos generar código que nos ayude a una mejor visualización de todas las reglas:

```
[21] for item in ResultadosC2:
       #El primer índice de la lista
       Emparejar = item[0]
       items = [x for x in Emparejar]
       print("Regla: " + str(item[0]))

       #El segundo índice de la lista
       print("Soporte: " + str(item[1]))

       #El tercer índice de la lista
       print("Confianza: " + str(item[2][0][2]))
       print("Lift: " + str(item[2][0][3]))
       print("====================================")
```

```
Regla: frozenset({'eggs', 'burgers'})
Soporte: 0.02879616051193174
Confianza: 0.33027522935779813
Lift: 1.8378297443715457
====================================
Regla: frozenset({'chocolate', 'mineral water'})
Soporte: 0.05265964538061592
Confianza: 0.3213995117982099
Lift: 1.3483320682317521
====================================
Regla: frozenset({'mineral water', 'eggs'})
Soporte: 0.05092654312758299
Confianza: 0.28338278931750743
Lift: 1.188844688294532
====================================
Regla: frozenset({'frozen vegetables', 'mineral water'})
Soporte: 0.03572856952406346
Confianza: 0.37482517482517486
Lift: 1.57246288387228
====================================
Regla: frozenset({'ground beef', 'mineral water'})
Soporte: 0.040927876283162246
Confianza: 0.41655359565807326
Lift: 1.7475215442008991
====================================
Regla: frozenset({'ground beef', 'spaghetti'})
Soporte: 0.03919477403012932
Confianza: 0.3989145183175034
Lift: 2.291162176033379
====================================
```

```
====================================
Regla: frozenset({'mineral water', 'milk'})
Soporte: 0.04799360085321957
Confianza: 0.3703703703703704
Lift: 1.5537741320739085
====================================
Regla: frozenset({'spaghetti', 'milk'})
Soporte: 0.03546193840821224
Confianza: 0.27366255144032925
Lift: 1.5717785592296398
====================================
Regla: frozenset({'mineral water', 'pancakes'})
Soporte: 0.03372883615517931
Confianza: 0.3548387096774194
Lift: 1.4886158620191963
====================================
Regla: frozenset({'mineral water', 'spaghetti'})
Soporte: 0.05972536995067324
Confianza: 0.2505592841163311
Lift: 1.4390851379453289
====================================
```

CONCLUSIÓN

Así como en la práctica anterior, pusimos en practica los pasos requeridos para generar las reglas de asociación con el algoritmo Apriori, en este caso ahora sobre el conjunto de datos de productos vendidos de una tienda. Esta herramienta puede ser de gran ayuda para implementar estrategias de venta, como por ejemplo cuando ya se encontraron las reglas de asociación se puede saber que productos poner juntos dentro de un supermercado para vender más y para que el cliente tenga la comodidad de encontrar los productos que usualmente más compra juntos.