Práctica 5. Reglas de asociación

Dataset: Store_data

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from apyori import apriori
```

In [2]:
```
tienda = pd.read_csv('store_data.csv', header=None)
tienda.head(3)
```

Out[2]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
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	salmon	antiox
1	burgers	meatballs	eggs	NaN	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	NaN	

Utilizando APRIORI

In [3]:
```
listas = []
for i in range(len(tienda)):
    #if tienda.values[i,j]!="NaN":
    #for j in range(len(tienda.columns)):
    listas.append([str(tienda.values[i,j]) for j in range(len(tienda.columns)) if str(tienda.values[i,j]) != "nan
```

In [4]:
```
reglasAsociacion = apriori(listas, min_support=0.0053, min_confidence=0.2, min_lift=3, min_length=2,max_length=3)
resultadosAsociacion = list(reglasAsociacion)
print(len(resultadosAsociacion))
```

```
16
```

In [5]:
```
data=[]
columnas=["Rule","Support","Confidence","Lift"]
consolidado=pd.DataFrame()
for item in resultadosAsociacion:
    par = item[0]
    items = [x for x in par]
    if len(items)==3:
        Rule= items[0] +" , "+ items[1]+ " -> "   + items[2]
    else:
        Rule= items[0] + " -> " + items[1]
    data.append({"Rule":Rule,"Support":item[1],"Confidence":item[2][0][2],"Lift":item[2][0][3]})
    consolidado=pd.DataFrame(data,columns=columnas)
consolidado.sort_values(by="Lift", ascending=False).head(10
```

Out[5]:

	Rule	Support	Confidence	Lift
1	escalope -> pasta	0.005866	0.372881	4.700812
4	whole wheat pasta -> olive oil	0.007999	0.271493	4.122410
13	spaghetti , herb & pepper -> ground beef	0.006399	0.393443	4.004360
12	mineral water , herb & pepper -> ground beef	0.006666	0.390625	3.975683
3	tomato sauce -> ground beef	0.005333	0.377358	3.840659
0	escalope -> mushroom cream sauce	0.005733	0.300699	3.790833
10	spaghetti , tomatoes -> frozen vegetables	0.006666	0.239234	3.498046
2	herb & pepper -> ground beef	0.015998	0.323450	3.291994
11	grated cheese , spaghetti -> ground beef	0.005333	0.322581	3.283144
5	shrimp , chocolate -> frozen vegetables	0.005333	0.232558	3.254512

Interpretación

Interpretación
Considerando los parámetros de la función apriori, se evidencia que los productos con mayor asociación son:
Pasta y Escalope, Whole wheat pasta y Olive Oil, y Ground beef, herb&pepper y spaghetti
Destacar que se ha modificado el código para eliminar los datos NaN
Vamos a comparar estos mismos resultados con la técnica Mlxtend

Utilizando MLXTEND

In [6]:
```
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
```

In [7]:
```
tienda1 = tienda.stack()
tienda1 = tienda1.reset_index()
del tienda1["level_1"]
tienda1.rename(columns={0:'items'}, inplace=True)
tienda1.rename(columns={'level_0':'nro'}, inplace=True)
tienda1.head(3)
```

Out[7]:

	nro	items
0	0	shrimp
1	0	almonds
2	0	avocado

In [8]:
```
tienda1 = tienda1.groupby(['nro', 'items']).size()
aux = tienda1.unstack(level=-1)
productos = aux.reset_index().fillna(0).set_index('nro')
#productosTienda = lambda x: 0 if == 0 else 1
def codificar(x):
    if x <= 0:
        return 0
    if x >= 1:
        return 1
productosTienda = productos.applymap(codificar)
productosTienda.head(3)
```

Out[8]:

items	asparagus	almonds	antioxydant juice	asparagus	avocado	babies food	bacon	barbecue sauce	black tea	blueberries	...	turkey	vegetables mix	water spray
nro														
0	0	1	1	0	1	0	0	0	0	0	...	0	1	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0

3 rows × 120 columns

In [9]:
```
frequent_itemsets = apriori(productosTienda, min_support=0.0053,use_colnames=True)
reglas = association_rules(frequent_itemsets, metric="lift", min_threshold=1)
reglas.sort_values(by = "lift", ascending = False).head(3)
```

Out[9]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
353	(pasta)	(escalope)	0.015731	0.079323	0.005866	0.372881	4.700812	0.004618	1.468107
352	(escalope)	(pasta)	0.079323	0.015731	0.005866	0.073950	4.700812	0.004618	1.062867
702	(whole wheat pasta)	(olive oil)	0.029463	0.065858	0.007999	0.271493	4.122410	0.006059	1.282270

In [10]:
```
reglas[(reglas['confidence']>0.2) & (reglas['support']>0.0053)].sort_values(by = "lift", ascending = False).head(
```

Out[10]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
353	(pasta)	(escalope)	0.015731	0.079323	0.005866	0.372881	4.700812	0.004618	1.468107
702	(whole wheat pasta)	(olive oil)	0.029463	0.065858	0.007999	0.271493	4.122410	0.006059	1.282270
1472	(spaghetti, herb & pepper)	(ground beef)	0.016264	0.098254	0.006399	0.393443	4.004360	0.004801	1.486663
1466	(mineral water, herb & pepper)	(ground beef)	0.017064	0.098254	0.006666	0.390625	3.975683	0.004989	1.479789
554	(tomato sauce)	(ground beef)	0.014131	0.098254	0.005333	0.377358	3.840659	0.003944	1.448259
351	(mushroom cream sauce)	(escalope)	0.019064	0.079323	0.005733	0.300699	3.790833	0.004220	1.316568
1427	(spaghetti, frozen vegetables)	(tomatoes)	0.027863	0.068391	0.006666	0.239234	3.498046	0.004760	1.224568
1426	(spaghetti, tomatoes)	(frozen vegetables)	0.020931	0.095321	0.006666	0.318471	3.341054	0.004671	1.327427
530	(herb & pepper)	(ground beef)	0.049460	0.098254	0.015998	0.323450	3.291994	0.011138	1.332860
1432	(grated cheese, spaghetti)	(ground beef)	0.016531	0.098254	0.005333	0.322581	3.283144	0.003708	1.331149

Interpretación

Interpretación
Aplicando la técnica de asociación de la librería Mlxtend se llegó a los mismos resultados citados en la interpretación anterior.