# Instituto Politécnico Nacional

**Escuela Superior de Cómputo**

## Proyecto Reglas de Asociación - Python

**Unidad de aprendizaje: Data Mining**

**Grupo: 3CV6**

***Alumno(a): Ramos Diaz Enrique***

***Profesor(a): Ocampo Botello Fabiola***

## 1.- Enunciado del problema

Este es un conjunto de datos transnacionales que contiene todas las transacciones que ocurren entre el 01/12/2010 y el 09/12/2011 para un comercio minorista en línea registrado en el Reino Unido y sin tienda. La compañía vende principalmente regalos únicos para toda ocasión. Muchos clientes de la empresa son mayoristas.

**El objetivo es encontrar qué artículos compran los clientes franceses a partir de otros previamente adquiridos**, con ayuda de los registros históricos del comercio.

## 2.- Diccionario de datos

| Atributo | Significado | Tipo de datos y dominio |
|---|---|---|
| **InvoiceNo** | Número de factura. Un número integral de 6 dígitos asignado exclusivamente a cada transacción. Si este código comienza con la letra 'c', indica una cancelación. | Tipo de dato **nominal (números)** con un rango de [536365, 581587] |
| **StockCode** | Código del producto o artículo. Un número de 5 dígitos asignado exclusivamente a cada producto distinto. | Tipo de dato **nominal (números)** con un rango de [10002, 90208] |
| **Description** | Nombre del producto o artículo. | Tipo de dato **nominal (cadena de texto)** |
| **Quantity** | Las cantidades de productos o artículos por transacción. | Tipo de dato **numérico** con un rango de [-80995, 80995] |
| **InvoiceDate** | Fecha y hora de la factura. El día y la hora en que se generó cada transacción. | Tipo de dato **nominal (fecha y hora)** con un rango de [01/12/2010 08:26:00 a. m., 09/12/2011 12:50:00 p. m.] |
| **UnitPrice** | Precio unitario. Precio del producto por unidad en libras esterlinas. | Tipo de dato **numérico** con un rango de [-11062.06, 38970.00] |
| **CustomerID** | Número del cliente. Un número de 5 dígitos único asignado a cada cliente. | Tipo de dato **nominal (números)** con un rango de [12346, 18287] |

| | | |
|---|---|---|
| **Country** | Nombre del país del cliente al momento de la transacción. | Es un tipo de dato **nominal (cadena de texto)** con un dominio de:<br>`array(['United Kingdom', 'France', 'Australia', 'Netherlands', 'Germany', 'Norway', 'EIRE', 'Switzerland', 'Spain', 'Poland', 'Portugal', 'Italy', 'Belgium', 'Lithuania', 'Japan', 'Iceland', 'Channel Islands', 'Denmark', 'Cyprus', 'Sweden', 'Austria', 'Israel', 'Finland', 'Bahrain', 'Greece', 'Hong Kong', 'Singapore', 'Lebanon', 'United Arab Emirates', 'Saudi Arabia', 'Czech Republic', 'Canada', 'Unspecified', 'Brazil', 'USA', 'European Community', 'Malta', 'RSA'], dtype=object)` |

## 3.- Importar las librerias

Primero es necesario importar algunas librerías que nos proporcionaran métodos o funciones para construir modelos utilizando reglas de asociación. Las principales son ***pandas*** y ***numpy***, que se utilizan para todo el preprocesamiento de los datos; y ***mlxtend***, que integra las herramientas para aplicar técnicas de reglas de asociación y encontrar itemsets frequentes en un conjunto de datos.

In [1]:
```
!pip install mlxtend
import numpy as np
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules
```

```
Requirement already satisfied: mlxtend in c:\programdata\anaconda3\lib\site-packages (0.17.0)
Requirement already satisfied: scikit-learn>=0.20.3 in c:\programdata\anaconda3\lib\site-packages (from mlxtend) (0.21.3)
Requirement already satisfied: pandas>=0.24.2 in c:\programdata\anaconda3\lib\site-packages (from mlxtend) (0.25.2)
Requirement already satisfied: matplotlib>=3.0.0 in c:\programdata\anaconda3\lib\site-packages (from mlxtend) (3.1.1)
Requirement already satisfied: joblib>=0.13.2 in c:\programdata\anaconda3\lib\site-packages (from mlxtend) (0.13.2)
Requirement already satisfied: numpy>=1.16.2 in c:\programdata\anaconda3\lib\site-packages (from mlxtend) (1.16.5)
Requirement already satisfied: scipy>=1.2.1 in c:\programdata\anaconda3\lib\site-packages (from mlxtend) (1.3.1)
Requirement already satisfied: setuptools in c:\programdata\anaconda3\lib\site-packages (from mlxtend) (41.6.0.post20191030)
Requirement already satisfied: pytz>=2017.2 in c:\programdata\anaconda3\lib\site-packages (from pandas>=0.24.2->mlxtend) (2019.3
)
Requirement already satisfied: python-dateutil>=2.6.1 in c:\programdata\anaconda3\lib\site-packages (from pandas>=0.24.2->mlxten
d) (2.8.0)
Requirement already satisfied: cycler>=0.10 in c:\programdata\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (0.1
0.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\programdata\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend)
 (1.1.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in c:\programdata\anaconda3\lib\site-packages (from matp
lotlib>=3.0.0->mlxtend) (2.4.2)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\lib\site-packages (from python-dateutil>=2.6.1->pandas>=0.24
.2->mlxtend) (1.12.0)
```

# 4.- Desarrollo: Proceso KDD

## 4.1.- Carga de la base de datos

La base de datos a utilizar tiene como nombre *groceries.csv*. Utilizamos el método de pandas llamado **read_csv()** y mostramos la información y descripción de los atributos con **info()** y **describe()**.

In [2]:
```
data=pd.read_csv("groceries.csv")
data.head()
```

Out[2]:

| | InvoiceNo | StockCode | Description | Quantity | InvoiceDate | UnitPrice | CustomerID | Country |
|---|---|---|---|---|---|---|---|---|
| 0 | 536365 | 85123A | WHITE HANGING HEART T-LIGHT HOLDER | 6 | 01/12/2010 08:26 | 2.55 | 17850.0 | United Kingdom |
| 1 | 536365 | 71053 | WHITE METAL LANTERN | 6 | 01/12/2010 08:26 | 3.39 | 17850.0 | United Kingdom |
| 2 | 536365 | 84406B | CREAM CUPID HEARTS COAT HANGER | 8 | 01/12/2010 08:26 | 2.75 | 17850.0 | United Kingdom |
| 3 | 536365 | 84029G | KNITTED UNION FLAG HOT WATER BOTTLE | 6 | 01/12/2010 08:26 | 3.39 | 17850.0 | United Kingdom |
| 4 | 536365 | 84029E | RED WOOLLY HOTTIE WHITE HEART. | 6 | 01/12/2010 08:26 | 3.39 | 17850.0 | United Kingdom |

### 4.1.1.- Información y descripción de los datos

In [3]:
```
data.info()
data.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
InvoiceNo      541909 non-null object
StockCode      541909 non-null object
Description    540455 non-null object
Quantity       541909 non-null int64
InvoiceDate    541909 non-null object
UnitPrice      541909 non-null float64
CustomerID     406829 non-null float64
Country        541909 non-null object
dtypes: float64(2), int64(1), object(5)
memory usage: 33.1+ MB
```

Out[3]:

| | Quantity | UnitPrice | CustomerID |
|---|---|---|---|
| count | 541909.000000 | 541909.000000 | 406829.000000 |
| mean | 9.552250 | 4.611114 | 15287.690570 |
| std | 218.081158 | 96.759853 | 1713.600303 |
| min | -80995.000000 | -11062.060000 | 12346.000000 |
| 25% | 1.000000 | 1.250000 | 13953.000000 |
| 50% | 3.000000 | 2.080000 | 15152.000000 |
| 75% | 10.000000 | 4.130000 | 16791.000000 |
| max | 80995.000000 | 38970.000000 | 18287.000000 |

## 4.2.- Limpieza de datos

Si revisamos la existencia de datos nulos con el método **isnull().sum()**, notaremos que existen datos nulos en los atributos *Description* y *CustomerID*. Para este proyecto, los eliminaremos.

In [4]:
```
data.isnull().sum()
```

Out[4]:
```
InvoiceNo          0
StockCode          0
Description     1454
Quantity           0
InvoiceDate        0
UnitPrice          0
CustomerID     135080
Country            0
dtype: int64
```

In [5]:
```
data = data.dropna()
```

Según el diccionario de datos, en los registros del atributo *InvoiceNo* existen transacciones que fueron canceladas y otros más que no poseen número de factura, por los que los eliminamos igualmente.

In [6]:
```
# Quitar transacciones sin número de factura
data['InvoiceNo'] = data['InvoiceNo'].astype('str')

# Quitar todas las transacciones canceladas
data = data[~data['InvoiceNo'].str.contains('C')]
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 397924 entries, 0 to 541908
Data columns (total 8 columns):
InvoiceNo      397924 non-null object
StockCode      397924 non-null object
Description    397924 non-null object
Quantity       397924 non-null int64
InvoiceDate    397924 non-null object
UnitPrice      397924 non-null float64
CustomerID     397924 non-null float64
Country        397924 non-null object
dtypes: float64(2), int64(1), object(5)
memory usage: 27.3+ MB
```

Por último, eliminamos los registros de los campos *UnitPrice* y *Quantity* cuyo valor sea negativo.

In [7]:
```
data = data[data["UnitPrice"] >= 0]
data = data[data["Quantity"] >= 0]
data.info()
data.head()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 397924 entries, 0 to 541908
Data columns (total 8 columns):
InvoiceNo      397924 non-null object
StockCode      397924 non-null object
Description    397924 non-null object
Quantity       397924 non-null int64
InvoiceDate    397924 non-null object
UnitPrice      397924 non-null float64
CustomerID     397924 non-null float64
Country        397924 non-null object
dtypes: float64(2), int64(1), object(5)
memory usage: 27.3+ MB
```

Out[7]:

| | InvoiceNo | StockCode | Description | Quantity | InvoiceDate | UnitPrice | CustomerID | Country |
|---|---|---|---|---|---|---|---|---|
| 0 | 536365 | 85123A | WHITE HANGING HEART T-LIGHT HOLDER | 6 | 01/12/2010 08:26 | 2.55 | 17850.0 | United Kingdom |
| 1 | 536365 | 71053 | WHITE METAL LANTERN | 6 | 01/12/2010 08:26 | 3.39 | 17850.0 | United Kingdom |
| 2 | 536365 | 84406B | CREAM CUPID HEARTS COAT HANGER | 8 | 01/12/2010 08:26 | 2.75 | 17850.0 | United Kingdom |
| 3 | 536365 | 84029G | KNITTED UNION FLAG HOT WATER BOTTLE | 6 | 01/12/2010 08:26 | 3.39 | 17850.0 | United Kingdom |
| 4 | 536365 | 84029E | RED WOOLLY HOTTIE WHITE HEART. | 6 | 01/12/2010 08:26 | 3.39 | 17850.0 | United Kingdom |

## 4.3.- Integración y selección de los datos

Para el objetivo de este proyecto, solamente se van a tomar las transacciones cuyos clientes residen en Francia, por lo que aplicamos ese filtro por medio del atributo *Country*.

In [8]:
```
basket = data[data["Country"] == "France"]
basket = basket.drop("Country", axis=1)
basket.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8342 entries, 26 to 541908
Data columns (total 7 columns):
InvoiceNo      8342 non-null object
StockCode      8342 non-null object
Description    8342 non-null object
Quantity       8342 non-null int64
InvoiceDate    8342 non-null object
UnitPrice      8342 non-null float64
CustomerID     8342 non-null float64
dtypes: float64(2), int64(1), object(4)
memory usage: 521.4+ KB
```

## 4.4.- Transformación de los datos

Eliminamos los espacios en blanco extras de los registros del atributo *Description*.

In [9]:
```
# Quitando espacios en blanco extras
basket['Description'] = basket['Description'].str.strip()
basket.info()
basket.head()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8342 entries, 26 to 541908
Data columns (total 7 columns):
InvoiceNo      8342 non-null object
StockCode      8342 non-null object
Description    8342 non-null object
Quantity       8342 non-null int64
InvoiceDate    8342 non-null object
UnitPrice      8342 non-null float64
CustomerID     8342 non-null float64
dtypes: float64(2), int64(1), object(4)
memory usage: 521.4+ KB
```

Out[9]:

| | InvoiceNo | StockCode | Description | Quantity | InvoiceDate | UnitPrice | CustomerID |
|---|---|---|---|---|---|---|---|
| **26** | 536370 | 22728 | ALARM CLOCK BAKELIKE PINK | 24 | 01/12/2010 08:45 | 3.75 | 12583.0 |
| **27** | 536370 | 22727 | ALARM CLOCK BAKELIKE RED | 24 | 01/12/2010 08:45 | 3.75 | 12583.0 |
| **28** | 536370 | 22726 | ALARM CLOCK BAKELIKE GREEN | 12 | 01/12/2010 08:45 | 3.75 | 12583.0 |
| **29** | 536370 | 21724 | PANDA AND BUNNIES STICKER SHEET | 12 | 01/12/2010 08:45 | 0.85 | 12583.0 |
| **30** | 536370 | 21883 | STARS GIFT TAPE | 24 | 01/12/2010 08:45 | 0.65 | 12583.0 |

Es necesario cambiar el formato del conjunto de datos para poder aplicar los algoritmos de reglas de asociación. En primer lugar, hay que agrupar los datos con base al ***número de factura***, y en cada factura se agrupan los ***productos o artículos*** existentes.

Luego, rellenamos el nuevo DataFrame con la ***cantidad de productos*** por cada transaccion, y nos olvidamos del resto de los atributos.

De esta forma, podemos tener una mejor ***descripción de los productos*** y la ***cantidad*** de ellos en cada transacción por separado (los productos son los que nos interesan para generar los itemsets y reglas de asociación).

In [10]:
```
basket = (basket.groupby(['InvoiceNo', 'Description'])['Quantity']
          .sum().unstack().reset_index().fillna(0)
          .set_index('InvoiceNo'))
basket.info()
basket.head()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 389 entries, 536370 to 581587
Columns: 1543 entries, 10 COLOUR SPACEBOY PEN to ZINC T-LIGHT HOLDER STARS SMALL
dtypes: float64(1543)
memory usage: 4.6+ MB
```

Out[10]:

| Description | 10 COLOUR SPACEBOY PEN | 12 COLOURED PARTY BALLOONS | 12 EGG HOUSE PAINTED WOOD | 12 MESSAGE CARDS WITH ENVELOPES | 12 PENCIL SMALL TUBE WOODLAND | 12 PENCILS SMALL TUBE RED RETROSPOT | 12 PENCILS SMALL TUBE SKULL | 12 PENCILS TALL TUBE POSY | 12 PENCILS TALL TUBE RED RETROSPOT | 12 PENCILS TALL TUBE WOODLAND | ... | WRAP SUKI AND FRIENDS | WRAP VINTAGE PETALS DESIGN | YELLOW COAT RACK PARIS FASHION |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **InvoiceNo** | | | | | | | | | | | | | | |
| **536370** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 |
| **536852** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 |
| **536974** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 |
| **537065** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 |
| **537463** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 |

5 rows × 1543 columns

Adicionalmente a esto, programamos un pequeño método que identifica qué productos estan presentes en cada transacción, según la ***cantidad*** de estos presentes en ellas, codificandolos como parte de estas o no y así tener una **respresentación binaria de la base de datos**:

0 = El producto no forma parte de la transacción.

1 = El producto forma parte de la transacción.

In [11]:
```
def bin_encode(x):
    if(x<= 0):
        return 0
    if(x>= 1):
        return 1

# Codificando los productos de cada transacción
basket = basket.applymap(bin_encode)
basket.head()
```

Out[11]:

| Description | 10 COLOUR SPACEBOY PEN | 12 COLOURED PARTY BALLOONS | 12 EGG HOUSE PAINTED WOOD | 12 MESSAGE CARDS WITH ENVELOPES | 12 PENCIL SMALL TUBE WOODLAND | 12 PENCILS SMALL TUBE RED RETROSPOT | 12 PENCILS SMALL TUBE SKULL | 12 PENCILS TALL TUBE POSY | 12 PENCILS TALL TUBE RED RETROSPOT | 12 PENCILS TALL TUBE WOODLAND | ... | WRAP SUKI AND FRIENDS | WRAP VINTAGE PETALS DESIGN | YELLOW COAT RACK PARIS FASHION |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **InvoiceNo** | | | | | | | | | | | | | | |
| **536370** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 |
| **536852** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 |
| **536974** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 |
| **537065** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 |
| **537463** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 |

5 rows × 1543 columns

## 4.5.- Minería de datos

### 4.5.1.- Extracción de itemsets frecuentes (2 - Itemset)

En primer lugar, vamos a hacer uso del **algoritmo Apriori** para identificar y generar conjuntos candidatos que se encuentren en la base de datos, de tamaño 2.

**NOTA: Al generar conjuntos de items de tamaño 1 (1 - Itemset), no se lograba obtener ninguna regla de asociación, por lo que se descartó y se trabajo directamente con tamaños 2 y 3.**

Utilizamos el método ***apriori()***, que recibe como parámetros: el dataframe con la base de datos **basket**, un soporte de umbral mínimo que debe cumplir cada conjunto de items **min_support**, un booleano para indicarle que conserve el nombre de los productos o artículos **use_colnames**, y el tamaño máximo de cada conjunto **max_len**.

In [12]:
```
# Construir el modelo
frq_items2 = apriori(basket, min_support = 0.02, use_colnames = True, max_len=2)
frq_items2
```

Out[12]:

| | support | itemsets |
|---|---|---|
| **0** | 0.030848 | (10 COLOUR SPACEBOY PEN) |
| **1** | 0.023136 | (12 PENCILS TALL TUBE RED RETROSPOT) |
| **2** | 0.035990 | (3 PIECE SPACEBOY COOKIE CUTTER SET) |
| **3** | 0.046272 | (36 PENCILS TUBE RED RETROSPOT) |
| **4** | 0.023136 | (36 PENCILS TUBE WOODLAND) |
| **...** | ... | ... |
| **1206** | 0.028278 | (STRAWBERRY LUNCH BOX WITH CUTLERY, SPACEBOY L... |
| **1207** | 0.030848 | (SPACEBOY MINI BACKPACK, WOODLAND MINI BACKPACK) |
| **1208** | 0.020566 | (STRAWBERRY LUNCH BOX WITH CUTLERY, TEA PARTY ... |
| **1209** | 0.020566 | (STRAWBERRY LUNCH BOX WITH CUTLERY, WATERING C... |
| **1210** | 0.028278 | (WATERING CAN PINK BUNNY, WATERING CAN BLUE EL... |

1211 rows × 2 columns

### 4.5.2.- Aplicación de reglas de asociacion (2 - Itemset)

Ahora solo queda darle formato a las reglas de asociación que los conjuntos de items calculados poseen.

Utilizando el método ***association_rules()*** que recibe como parámetros: un dataframe con los conjuntos de items (junto a su soporte) **frq_items2**, una métrica para evaluar si una regla es interesante o útil **metric**, y un valor de umbral mínimo para esta métrica **min_threshold**.

Se utiliza la métrica **lift (carga)** de entre todas ya que cuantifica la utilidad de una regla de asociación. Además, podemos afirmar que las reglas de asociación son más interesantes con base a esta métrica si su valor distinto a 1.

Las de menos utilidad son aquellas cuya carga es cercana a 1.

El valor de umbral mínimo para la carga es 0, ya que le decimos al algoritmo que obtenga todas las reglas de asociación. En el análisis de resultados solamente tomaremos aquellas distintas o lejanas a 1.

Ya obtenidas las reglas, se ordenan de manera ascendente según su **carga** y se muestran. Por cada regla, se tiene su **antecedente, su consecuencia, así como los valores de soporte, confianza, carga, influencia y convicción**.

In [13]:
```
# Insertar las reglas inferidas a un dataframe
itemset2 = association_rules(frq_items2, metric = "lift", min_threshold = 0)
itemset2 = itemset2.sort_values(['lift', 'support', 'confidence'], ascending =[True, False, False])
itemset2
```

Out[13]:

| | antecedents | consequents | antecedent support | consequent support | support | confidence | lift | leverage | conviction |
|---|---|---|---|---|---|---|---|---|---|
| **204** | (ASSORTED COLOUR BIRD ORNAMENT) | (POSTAGE) | 0.043702 | 0.771208 | 0.020566 | 0.470588 | 0.610196 | -0.013138 | 0.432162 |
| **205** | (POSTAGE) | (ASSORTED COLOUR BIRD ORNAMENT) | 0.771208 | 0.043702 | 0.020566 | 0.026667 | 0.610196 | -0.013138 | 0.982498 |
| **9** | (POSTAGE) | (36 PENCILS TUBE RED RETROSPOT) | 0.771208 | 0.046272 | 0.025707 | 0.033333 | 0.720370 | -0.009979 | 0.986615 |
| **8** | (36 PENCILS TUBE RED RETROSPOT) | (POSTAGE) | 0.046272 | 0.771208 | 0.025707 | 0.555556 | 0.720370 | -0.009979 | 0.514781 |
| **479** | (CLOTHES PEGS RETROSPOT PACK 24) | (POSTAGE) | 0.041131 | 0.771208 | 0.023136 | 0.562500 | 0.729375 | -0.008584 | 0.522953 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **1644** | (RECYCLED ACAPULCO MAT TURQUOISE) | (RECYCLED ACAPULCO MAT PINK) | 0.030848 | 0.025707 | 0.020566 | 0.666667 | 25.933333 | 0.019773 | 2.922879 |
| **1641** | (RECYCLED ACAPULCO MAT LAVENDER) | (RECYCLED ACAPULCO MAT RED) | 0.025707 | 0.030848 | 0.023136 | 0.900000 | 29.175000 | 0.022343 | 9.691517 |
| **1643** | (RECYCLED ACAPULCO MAT LAVENDER) | (RECYCLED ACAPULCO MAT TURQUOISE) | 0.025707 | 0.030848 | 0.023136 | 0.900000 | 29.175000 | 0.022343 | 9.691517 |
| **1640** | (RECYCLED ACAPULCO MAT RED) | (RECYCLED ACAPULCO MAT LAVENDER) | 0.030848 | 0.025707 | 0.023136 | 0.750000 | 29.175000 | 0.022343 | 3.897172 |
| **1642** | (RECYCLED ACAPULCO MAT TURQUOISE) | (RECYCLED ACAPULCO MAT LAVENDER) | 0.030848 | 0.025707 | 0.023136 | 0.750000 | 29.175000 | 0.022343 | 3.897172 |

1864 rows × 9 columns

### 4.5.3.- Extracción de itemsets frecuentes (3 - Itemset)

De manera similar al itemset anterior, identificamos y generamos conjuntos candidatos que se encuentren en la base de datos de tamaño 3.

In [14]:
```
# Construir el modelo
frq_items3 = apriori(basket, min_support = 0.02, use_colnames = True, max_len=3)
frq_items3
```

Out[14]:

| | support | itemsets |
|---|---|---|
| **0** | 0.030848 | (10 COLOUR SPACEBOY PEN) |
| **1** | 0.023136 | (12 PENCILS TALL TUBE RED RETROSPOT) |
| **2** | 0.035990 | (3 PIECE SPACEBOY COOKIE CUTTER SET) |
| **3** | 0.046272 | (36 PENCILS TUBE RED RETROSPOT) |
| **4** | 0.023136 | (36 PENCILS TUBE WOODLAND) |
| **...** | ... | ... |
| **2105** | 0.025707 | (SET/6 RED SPOTTY PAPER PLATES, SET/6 RED SPOT... |
| **2106** | 0.100257 | (SET/6 RED SPOTTY PAPER PLATES, SET/6 RED SPOT... |
| **2107** | 0.020566 | (SPACEBOY BIRTHDAY CARD, SET/6 RED SPOTTY PAPE... |
| **2108** | 0.020566 | (SET/6 RED SPOTTY PAPER PLATES, SPACEBOY BIRTH... |
| **2109** | 0.020566 | (STRAWBERRY LUNCH BOX WITH CUTLERY, SET/6 RED ... |

2110 rows × 2 columns

### 4.5.4.- Aplicación de reglas de asociacion (3 - Itemset)

Se le da formato a las reglas de asociación que los conjuntos de items calculados poseen. Los parámetros son los mismos que en el 2 - Itemset.

In [15]:
```
# Insertar las reglas inferidas a un dataframe
itemset3 = association_rules(frq_items3, metric = "lift", min_threshold = 0)
itemset3 = itemset3.sort_values(['lift', 'support', 'confidence'], ascending =[True, False, False])
itemset3
```

Out[15]:

| | antecedents | consequents | antecedent support | consequent support | support | confidence | lift | leverage | conviction |
|---|---|---|---|---|---|---|---|---|---|
| **204** | (ASSORTED COLOUR BIRD ORNAMENT) | (POSTAGE) | 0.043702 | 0.771208 | 0.020566 | 0.470588 | 0.610196 | -0.013138 | 0.432162 |
| **205** | (POSTAGE) | (ASSORTED COLOUR BIRD ORNAMENT) | 0.771208 | 0.043702 | 0.020566 | 0.026667 | 0.610196 | -0.013138 | 0.982498 |
| **4474** | (LUNCH BAG RED RETROSPOT, LUNCH BAG DOLLY GIRL... | (POSTAGE) | 0.043702 | 0.771208 | 0.023136 | 0.529412 | 0.686471 | -0.010567 | 0.486183 |
| **4479** | (POSTAGE) | (LUNCH BAG RED RETROSPOT, LUNCH BAG DOLLY GIRL... | 0.771208 | 0.043702 | 0.023136 | 0.030000 | 0.686471 | -0.010567 | 0.985874 |
| **3566** | (POSTAGE) | (LUNCH BAG RED RETROSPOT, DOLLY GIRL LUNCH BOX) | 0.771208 | 0.038560 | 0.020566 | 0.026667 | 0.691556 | -0.009173 | 0.987780 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **1642** | (RECYCLED ACAPULCO MAT TURQUOISE) | (RECYCLED ACAPULCO MAT LAVENDER) | 0.030848 | 0.025707 | 0.023136 | 0.750000 | 29.175000 | 0.022343 | 3.897172 |
| **6647** | (RECYCLED ACAPULCO MAT RED, POSTAGE) | (RECYCLED ACAPULCO MAT LAVENDER) | 0.023136 | 0.025707 | 0.020566 | 0.888889 | 34.577778 | 0.019971 | 8.768638 |
| **7126** | (RECYCLED ACAPULCO MAT TURQUOISE, RECYCLED ACA... | (RECYCLED ACAPULCO MAT LAVENDER) | 0.023136 | 0.025707 | 0.020566 | 0.888889 | 34.577778 | 0.019971 | 8.768638 |
| **6650** | (RECYCLED ACAPULCO MAT LAVENDER) | (RECYCLED ACAPULCO MAT RED, POSTAGE) | 0.025707 | 0.023136 | 0.020566 | 0.800000 | 34.577778 | 0.019971 | 4.884319 |
| **7131** | (RECYCLED ACAPULCO MAT LAVENDER) | (RECYCLED ACAPULCO MAT TURQUOISE, RECYCLED ACA... | 0.025707 | 0.023136 | 0.020566 | 0.800000 | 34.577778 | 0.019971 | 4.884319 |

7258 rows × 9 columns

# 5.- Resultados encontrados

A grandes rasgos, nos encontramos con gran variedad de casos representados en las reglas de asociación obtenidas.

En el 2 - Itemset podemos decir, por ejemplo, que las personas que compran ornamentos de aves, también comprarán sellos postales con un soporte del 2% y una confianza del 47% aproximadamente. En otro caso, aquellos que compran manteles reciclados de playa de un color, como el turquesa, comprarán otro igual pero de distinto color, como rojo, con un soporte del 2.3% y una confianza del 80%.

En el 3 - Itemset podemos decir que aquellas personas que compraron loncheras rojas y para niñas, tambien comprarán sellos postales con un soporte del 2.3% y una confianza del 25% aproximadamente. En otro caso, retomando el ejemplo de las mantas de playa, si un cliente compra dos de distintos colores, comprará uno extra de otro color distinto con un soporte del 2% y una confianza del 88%.

In [16]:
```
itemset2.to_csv("itemset2.csv")
```

In [17]:
```
itemset3.to_csv("itemset3.csv")
```

# 6.- Bibliografía


Larose, T. Daniel & Larose, D. Chantal. (2015). Data Mining and Predictive Analytics. Second Edition. Wiley.

http://rasbt.github.io/mlxtend/api_subpackages/mlxtend.frequent_patterns/#apriori

http://rasbt.github.io/mlxtend/api_subpackages/mlxtend.frequent_patterns/#association_rules