Market Basket Analysis in Pthon using Apriori Algorithm

Task 2 Rotem Cohen

Loading the packages

In [2]:
```
import pandas as pd
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
pd.set_option('display.max_rows',None)
pd.set_option('display.max_columns',None)
```

Loading data: We will ube using the encoding as latin 1 to read the few special characters mentioned in the file

In [4]:
```
dataset= pd.read_csv("D:/internship/marketdata.csv",encoding ='latin-1')
dataset.head()
```

Out[4]:

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

In [6]:
```
dataset.shape
```

Out[6]:
```
(541909, 8)
```

In [7]:
```
dataset.dtypes
```

Out[7]:
```
InvoiceNo       object
StockCode       object
Description     object
Quantity         int64
InvoiceDate     object
UnitPrice      float64
CustomerID     float64
Country         object
dtype: object
```

In [8]:
```
dataset.describe()
```

Out[8]:

	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

Data Cleaning

In [9]:
```
dataset['Description'] = dataset['Description'].str.strip()#removing the spaces
```

In [11]:
```
dataset.dropna(axis=0, subset=['InvoiceNo'], inplace=True)#dropping rows that dont hav
dataset['InvoiceNo'] = dataset['InvoiceNo'].astype('str')#conerting InvoiceNo column t
dataset=dataset[~dataset['InvoiceNo'].str.contains('C')]#removing InvoiceNo which cont
```

In [12]:
```
dataset.shape
```

Out[12]:
```
(532621, 8)
```

After the cleaning of the data, wewill have to consolidate the items into 1 transaction per row with each product 1 hot encoded. we will be looking at the data of country France

In [14]:
```
basket = (dataset[dataset['Country']=="France"]#getting the data with the country frar
          .groupby(['InvoiceNo','Description'])['Quantity']#grouping them by Invoice Nu
          .sum().unstack().reset_index().fillna(0)#sum the quamtity, unstack them and f
          .set_index('InvoiceNo'))#make the InvoiceNo as Index
```

In [15]:
```
basket.head()
```

Out[15]:

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	PEN T P
InvoiceNo								
536370	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	
536974	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	
537463	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

In [16]:
```
basket.shape
```

Out[16]: (392, 1563)

In [19]:
```
#function to convert values <0 to 0 and values > 1 to 1
def encode_units(x):
    if x<=0:
        return 0
    if x>=1:
        return 1
    #apply function to data using applymap which is looping through the function
basket_sets= basket.applymap(encode_units)
```

In [20]:
```
basket_sets.head()
```

Out[20]:

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	PEN T P
InvoiceNo								
536370	0	0	0	0	0	0	0	
536852	0	0	0	0	0	0	0	
536974	0	0	0	0	0	0	0	
537065	0	0	0	0	0	0	0	
537463	0	0	0	0	0	0	0	

In [21]:
```
frequent_itemsets = apriori(basket_sets, min_support=0.07, use_colnames= True)
```

```
C:\Users\Rotem Cohen\anaconda3\Lib\site-packages\mlxtend\frequent_patterns\fpcommon.py:110: DeprecationWarning: DataFrames with non-bool types result in worse computationalperformance and their support might be discontinued in the future.Please use a DataFrame with bool type
  warnings.warn(
```

the final step is generate the asscociation rules with their corresponding support, confidence and lift

In [22]:
```
rules= association_rules(frequent_itemsets, metric ="lift", min_threshold=1)
```

In [23]:
```
rules.head()
```

Out[23]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conv
0	(ALARM CLOCK BAKELIKE GREEN)	(ALARM CLOCK BAKELIKE PINK)	0.096939	0.102041	0.073980	0.763158	7.478947	0.064088	3.7
1	(ALARM CLOCK BAKELIKE PINK)	(ALARM CLOCK BAKELIKE GREEN)	0.102041	0.096939	0.073980	0.725000	7.478947	0.064088	3.2
2	(ALARM CLOCK BAKELIKE GREEN)	(ALARM CLOCK BAKELIKE RED)	0.096939	0.094388	0.079082	0.815789	8.642959	0.069932	4.9
3	(ALARM CLOCK BAKELIKE RED)	(ALARM CLOCK BAKELIKE GREEN)	0.094388	0.096939	0.079082	0.837838	8.642959	0.069932	5.5
4	(ALARM CLOCK BAKELIKE GREEN)	(POSTAGE)	0.096939	0.765306	0.084184	0.868421	1.134737	0.009996	1.7

In [24]:
```
rules[(rules['lift']>=6)&
     (rules['confidence']>=0.8)]
```

Out[24]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	co
2	(ALARM CLOCK BAKELIKE GREEN)	(ALARM CLOCK BAKELIKE RED)	0.096939	0.094388	0.079082	0.815789	8.642959	0.069932	
3	(ALARM CLOCK BAKELIKE RED)	(ALARM CLOCK BAKELIKE GREEN)	0.094388	0.096939	0.079082	0.837838	8.642959	0.069932	
75	(SET/6 RED SPOTTY PAPER PLATES)	(SET/20 RED RETROSPOT PAPER NAPKINS)	0.127551	0.132653	0.102041	0.800000	6.030769	0.085121	
76	(SET/6 RED SPOTTY PAPER CUPS)	(SET/6 RED SPOTTY PAPER PLATES)	0.137755	0.127551	0.122449	0.888889	6.968889	0.104878	
77	(SET/6 RED SPOTTY PAPER PLATES)	(SET/6 RED SPOTTY PAPER CUPS)	0.127551	0.137755	0.122449	0.960000	6.968889	0.104878	2
79	(ALARM CLOCK BAKELIKE GREEN, POSTAGE)	(ALARM CLOCK BAKELIKE RED)	0.084184	0.094388	0.071429	0.848485	8.989353	0.063483	
80	(ALARM CLOCK BAKELIKE RED, POSTAGE)	(ALARM CLOCK BAKELIKE GREEN)	0.086735	0.096939	0.071429	0.823529	8.495356	0.063021	
115	(SET/6 RED SPOTTY PAPER CUPS, POSTAGE)	(SET/6 RED SPOTTY PAPER PLATES)	0.117347	0.127551	0.102041	0.869565	6.817391	0.087073	
116	(SET/6 RED SPOTTY PAPER PLATES, POSTAGE)	(SET/6 RED SPOTTY PAPER CUPS)	0.107143	0.137755	0.102041	0.952381	6.913580	0.087281	1
118	(SET/6 RED SPOTTY PAPER PLATES)	(SET/6 RED SPOTTY PAPER CUPS, POSTAGE)	0.127551	0.117347	0.102041	0.800000	6.817391	0.087073	
120	(SET/6 RED SPOTTY PAPER CUPS, SET/20 RED RETRO...	(SET/6 RED SPOTTY PAPER PLATES)	0.102041	0.127551	0.099490	0.975000	7.644000	0.086474	3

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	co
121	(SET/6 RED SPOTTY PAPER CUPS, SET/6 RED SPOTTY...	(SET/20 RED RETROSPOT PAPER NAPKINS)	0.122449	0.132653	0.099490	0.812500	6.125000	0.083247	
122	(SET/20 RED RETROSPOT PAPER NAPKINS, SET/6 RED...	(SET/6 RED SPOTTY PAPER CUPS)	0.102041	0.137755	0.099490	0.975000	7.077778	0.085433	3
127	(SET/6 RED SPOTTY PAPER CUPS, SET/20 RED RETRO...	(SET/6 RED SPOTTY PAPER PLATES)	0.084184	0.127551	0.081633	0.969697	7.602424	0.070895	2
129	(SET/20 RED RETROSPOT PAPER NAPKINS,	(SET/6 RED SPOTTY PAPER CUPS)	0.084184	0.137755	0.081633	0.969697	7.039282	0.070036	2

In [ ]: