In [1]:
```python
import numpy as np
import pandas as pd
import math

from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
from sklearn.metrics.pairwise import cosine_similarity
```

Problem 1

In [2]:
```python
df = pd.read_excel('http://archive.ics.uci.edu/ml/machine-learning-databases/00352/Online%20Retail.xls
x')
df.head()
```

Out[2]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom

Let us do some clean up first. This include:

1. Stripping spaces in the description column
2. Dropping rows that doesn't contain involice numbers
3. Remove credit transactions

In [3]:
```python
df['Description'] = df['Description'].str.strip()
df.dropna(axis = 0, subset=['InvoiceNo'], inplace = True)
df['InvoiceNo'] = df['InvoiceNo'].astype('str')
df = df[~df['InvoiceNo'].str.contains('C')]
```

Before proceeding, let us understand the data distribution by country:

In [4]:
```python
df.groupby('Country').count().reset_index().sort_values('InvoiceNo', ascending = False).head()
```

Out[4]:

	Country	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID
36	United Kingdom	487622	487622	486167	487622	487622	487622	354345
14	Germany	9042	9042	9042	9042	9042	9042	9042
13	France	8408	8408	8408	8408	8408	8408	8342
10	EIRE	7894	7894	7894	7894	7894	7894	7238
31	Spain	2485	2485	2485	2485	2485	2485	2485

In [5]:
```python
Basket = (df[df['Country']=="France"]
          .groupby(['InvoiceNo', 'Description'])['Quantity']
          .sum().unstack().reset_index().fillna(0)
          .set_index('InvoiceNo'))

Basket.head()
```

Out[5]:

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
InvoiceNo										
536370	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
536974	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
537463	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 1563 columns

In-order to complete the one-hot encoding process, we need to replace all values of quantity >=1 by 1.

In [6]:
```python
def sum_to_boolean(x):
    if x<=0:
        return 0
    else:
        return 1

Basket_Final = Basket.applymap(sum_to_boolean)
```

Dropping the postage column, and the final one-hot codded matrix.

In [7]:
```python
Basket_Final.drop('POSTAGE', inplace=True, axis=1)
Basket_Final.head()
```

Out[7]:

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
InvoiceNo										
536370	0	0	0	0	0	0	0	0	0	0
536852	0	0	0	0	0	0	0	0	0	0
536974	0	0	0	0	0	0	0	0	0	0
537065	0	0	0	0	0	0	0	0	0	0
537463	0	0	0	0	0	0	0	0	0	0

5 rows × 1562 columns

Apriori:

To start with and have sufficient data, let us look at frequent itemsets that have a support of atleast 5%.

In [8]:
```python
# Apriori to select the most important itemsets
Frequent_itemsets = apriori(Basket_Final, min_support = 0.05, use_colnames = True)
Frequent_itemsets.sort_values('support', ascending = False).head()
```

Out[8]:

	support	itemsets
46	0.188776	(RABBIT NIGHT LIGHT)
52	0.181122	(RED TOADSTOOL LED NIGHT LIGHT)
44	0.170918	(PLASTERS IN TIN WOODLAND ANIMALS)
40	0.168367	(PLASTERS IN TIN CIRCUS PARADE)
59	0.158163	(ROUND SNACK BOXES SET OF4 WOODLAND)

In [9]:
```python
Frequent_itemsets.sort_values('support', ascending = False).head(1)
```

Out[9]:

	support	itemsets
46	0.188776	(RABBIT NIGHT LIGHT)

(RABBIT NIGHT LIGHT) itemset have a highest suppport value.

Association Rules:

Now since we have identified the key itemsets, let us apply the association rules to learn the purchase behaviours.

In [10]:
```python
# extract the association rules with the highest values using metric="lift"
Asso_Rules = association_rules(Frequent_itemsets, metric = "lift")
Asso_Rules.sort_values('lift',ascending = False).head(1)
```

Out[10]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
39	(PACK OF 6 SKULL PAPER CUPS)	(PACK OF 6 SKULL PAPER PLATES)	0.063776	0.056122	0.05102	0.8	14.254545	0.047441	4.719388

In [11]:
```python
print('Antecedents: {0}'.format(Asso_Rules.loc[39,'antecedents']))
print('Consequents: {0}'.format(Asso_Rules.loc[39,'consequents']))
```

```
Antecedents: frozenset({'PACK OF 6 SKULL PAPER CUPS'})
Consequents: frozenset({'PACK OF 6 SKULL PAPER PLATES'})
```

In [12]:
```python
# extract the association rules with the highest values using metric="confidence"
Asso_Rules = association_rules(Frequent_itemsets, metric = "confidence")
Asso_Rules.sort_values('confidence',ascending = False).head(1)
```

Out[12]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
12	(SET/6 RED SPOTTY PAPER PLATES, SET/20 RED RET...	(SET/6 RED SPOTTY PAPER CUPS)	0.102041	0.137755	0.09949	0.975	7.077778	0.085433	34.489796

In [13]:
```python
print('Antecedents: {0}'.format(Asso_Rules.loc[12,'antecedents']))
print('Consequents: {0}'.format(Asso_Rules.loc[12,'consequents']))
```

```
Antecedents: frozenset({'SET/6 RED SPOTTY PAPER PLATES', 'SET/20 RED RETROSPOT PAPER NAPKINS'})
Consequents: frozenset({'SET/6 RED SPOTTY PAPER CUPS'})
```

- Is the rule with the highest confidence the same as the rule with the highest lift?

Answer: No

because

Confidence is the ratio of the number of transactions that include all items in the consequent, as well as the antecedent (the support) to the number of transactions that include all items in the antecedent.

Lift is nothing but the ratio of Confidence to Expected Confidence.

Problem - 2

In [14]:
```python
data = pd.read_csv('75000-out2-binary.csv')
```

In [15]:
```python
data.head()
```

Out[15]:

	Transaction Number	Chocolate Cake	Lemon Cake	Casino Cake	Opera Cake	Strawberry Cake	Truffle Cake	Chocolate Eclair	Coffee Eclair	Vanilla Eclair	...	Lemon Lemonade	Raspberry Lemonade	Orange Juice	G
0	1	0	0	0	0	0	0	0	0	0	...	0	0	0	
1	2	0	0	0	0	0	0	0	1	0	...	0	0	0	
2	3	0	0	0	1	0	0	0	0	0	...	0	0	1	
3	4	0	0	0	0	0	1	0	0	0	...	0	0	0	
4	5	0	0	0	0	0	0	1	0	0	...	0	0	1	

5 rows × 51 columns

In [16]:
```python
item1_name = 'Chocolate Coffee'
item2_name = 'Chocolate Cake'

selection = data[[item1_name,
                  item2_name]]
```

In [17]:
```python
item1_count = selection[item1_name] == 1
item2_count = selection[item2_name] == 1
```

In [18]:
```python
f = selection.groupby([item1_count,
                       item2_count]).count()
print(f)
```

```
                                 Chocolate Coffee  Chocolate Cake
Chocolate Coffee Chocolate Cake
False            False                      65802           65802
                 True                        2962            2962
True             False                       2933            2933
                 True                        3303            3303
```

In [19]:
```python
f00 = f[item1_name][0][0]
f01 = f[item1_name][0][1]
f10 = f[item2_name][1][0]
f11 = f[item2_name][1][1]

print(f00)
print(f01)
print(f10)
print(f11)
```

```
65802
2962
2933
3303
```

In [20]:
```python
f1p = f11 + f10
fp1 = f11 + f01
f0p = f01 + f00
fp0 = f10 + f00

print(f1p)
print(fp1)
print(f0p)
print(fp0)
```

```
6236
6265
68764
68735
```

In [21]:
```python
N = f00+f01+f10+f11
N
```

Out[21]: 75000

In [22]:
```python
# Find Correlation coefficient manually
Correlation = ((N*f11) - (f1p*fp1))/math.sqrt(f1p*fp1*f0p*fp0)
print('Correlation coefficient for Chocolate Coffee and Chocolate Cake items : {0}'.format(Correlation
))
```

```
Correlation coefficient for Chocolate Coffee and Chocolate Cake items : 0.4855664925278768
```

In [23]:
```python
# Find Correlation coefficient programmatically
correlation = selection['Chocolate Coffee'].corr(selection['Chocolate Cake'])
print('Correlation coefficient for Chocolate Coffee and Chocolate Cake items : {0}'.format(Correlation
))
```

```
Correlation coefficient for Chocolate Coffee and Chocolate Cake items : 0.4855664925278768
```

In [24]:
```python
from sklearn.metrics.pairwise import euclidean_distances
```

In [25]:
```python
# Find euclidean_distances d(X,Y)
X = [selection['Chocolate Coffee']]
Y = [selection['Chocolate Cake']]
euclidean_distances(X,Y)
```

Out[25]: array([[76.77890335]])

In [26]:
```python
# Find euclidean_distances d(Y,X)
print(euclidean_distances(Y,X))
```

```
[[76.77890335]]
```

- Both the distance d(X,Y) and d(Y,X) are same.
- d(X,Y) = d(Y,X)
- So, both the itmes "Chocolate Coffee" and "Chocolate Cake"' are symmetric

In [27]:
```python
r = np.corrcoef(selection['Chocolate Coffee'], selection['Chocolate Cake'])
r
```

Out[27]:
```
array([[1.        , 0.48556649],
       [0.48556649, 1.        ]])
```

In [28]:
```python
#Correlation {Chocolate Coffee} -> {Chocolate Cake}
r[0,1]
```

Out[28]: 0.48556649252787826

In [29]:
```python
#Correlation {Chocolate Cake} -> {Chocolate Coffee}
r[1,0]
```

Out[29]: 0.48556649252787837

Would the association rules {Chocolate Coffee} => {Chocolate Cake} have the same value for coefficient as {Chocolate Cake} => {Chocolate Coffeeg} ?

- Answer is Yes (If we consider up to 14 decimal place)

In [ ]: