120A3051

Shreya Idate

Batch: E3

Experiment 4

Aim: To study and implement Apriori algorithm in python.

Theory:

- Apriori algorithm refers to the algorithm which is used to calculate the association rules between objects.
- It means how two or more objects are related to one another. In other words, we can say that the apriori algorithm is an association rule leaning that analyzes that people who bought product A also bought product B.
- The primary objective of the apriori algorithm is to create the association rule between different objects.
- The association rule describes how two or more objects are related to one another.
- Apriori algorithm is also called frequent pattern mining. Generally, you operate the Apriori algorithm on a database that consists of a huge number of transactions.
- For example: suppose you go to Big Bazar and buy different products. It helps the customers buy their products with ease and increases the sales performance of the Big Bazar.

Output:

```
[1] import numpy as np
    import pandas as pd
    from mlxtend.frequent_patterns import apriori, association_rules
```

```
[2] from google.colab import drive

    drive.mount('/content/drive')

    Mounted at /content/drive
```

```
[9] path = '/content/Online_Retail.xlsx'
    df = pd.read_excel(path)
```

```
df.head()
```

	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

+ Code + Text

```
[12] df.columns
```

```
Index(['InvoiceNo', 'StockCode', 'Description', 'Quantity', 'InvoiceDate',
       'UnitPrice', 'CustomerID', 'Country'],
      dtype='object')
```

```
[13] df.Country.unique()
```

```
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)
```

```
[14] #Stripping extra spaces in the description
     df['Description'] = df['Description'].str.strip()

     # Dropping the rows without any invoice number
     df.dropna(axis = 0, subset =['InvoiceNo'], inplace = True)
     df['InvoiceNo'] = df['InvoiceNo'].astype('str')

     # Dropping all transactions which were done on credit
     data = df[~df['InvoiceNo'].str.contains('C')]
```

```
# Transactions done in France
basket_France = (data[data['Country'] =="France"]
          .groupby(['InvoiceNo', 'Description'])['Quantity']
          .sum().unstack().reset_index().fillna(0)
          .set_index('InvoiceNo'))

# Transactions done in the United Kingdom
basket_UK = (data[data['Country'] =="United Kingdom"]
          .groupby(['InvoiceNo', 'Description'])['Quantity']
          .sum().unstack().reset_index().fillna(0)
          .set_index('InvoiceNo'))

# Transactions done in Portugal
basket_Por = (data[data['Country'] =="Portugal"]
          .groupby(['InvoiceNo', 'Description'])['Quantity']
          .sum().unstack().reset_index().fillna(0)
          .set_index('InvoiceNo'))

basket_Sweden = (data[data['Country'] =="Sweden"]
          .groupby(['InvoiceNo', 'Description'])['Quantity']
          .sum().unstack().reset_index().fillna(0)
          .set_index('InvoiceNo'))
```

```
# Defining the hot encoding function to make the data suitable
# for the concerned libraries
def hot_encode(x):
    if(x<= 0):
        return 0
    if(x>= 1):
        return 1

# Encoding the datasets
basket_encoded = basket_France.applymap(hot_encode)
basket_France = basket_encoded

basket_encoded = basket_UK.applymap(hot_encode)
basket_UK = basket_encoded

basket_encoded = basket_Por.applymap(hot_encode)
basket_Por = basket_encoded

basket_encoded = basket_Sweden.applymap(hot_encode)
basket_Sweden = basket_encoded
```

```
# Building the model
frq_items = apriori(basket_France, min_support = 0.05, use_colnames = True)

# Collecting the inferred rules in a dataframe
rules = association_rules(frq_items, metric ="lift", min_threshold = 1)
rules = rules.sort_values(['confidence', 'lift'], ascending =[False, False])
print(rules.head())
```

```
                                    antecedents  \
44                    (JUMBO BAG WOODLAND ANIMALS)
258  (RED TOADSTOOL LED NIGHT LIGHT, PLASTERS IN TI...
271  (RED TOADSTOOL LED NIGHT LIGHT, PLASTERS IN TI...
302  (SET/20 RED RETROSPOT PAPER NAPKINS, SET/6 RED...
301  (SET/6 RED SPOTTY PAPER PLATES, SET/20 RED RET...

                       consequents  antecedent support  consequent support  \
44                       (POSTAGE)            0.076531            0.765306
258                      (POSTAGE)            0.051020            0.765306
271                      (POSTAGE)            0.053571            0.765306
302  (SET/6 RED SPOTTY PAPER PLATES)            0.102041            0.127551
301    (SET/6 RED SPOTTY PAPER CUPS)            0.102041            0.137755

      support  confidence      lift  leverage  conviction
44   0.076531       1.000  1.306667  0.017961         inf
258  0.051020       1.000  1.306667  0.011974         inf
271  0.053571       1.000  1.306667  0.012573         inf
302  0.099490       0.975  7.644000  0.086474   34.897959
301  0.099490       0.975  7.077778  0.085433   34.489796
```


```
# Building the model
frq_items = apriori(basket_France, min_support = 0.05, use_colnames = True)

# Collecting the inferred rules in a dataframe
rules = association_rules(frq_items, metric ="lift", min_threshold = 1)
rules = rules.sort_values(['confidence', 'lift'], ascending =[False, False])
print(rules.head())
```


```
                                    antecedents  \
44                    (JUMBO BAG WOODLAND ANIMALS)
258  (RED TOADSTOOL LED NIGHT LIGHT, PLASTERS IN TI...
271  (RED TOADSTOOL LED NIGHT LIGHT, PLASTERS IN TI...
302  (SET/20 RED RETROSPOT PAPER NAPKINS, SET/6 RED...
301  (SET/6 RED SPOTTY PAPER PLATES, SET/20 RED RET...
```

```
frq_items = apriori(basket_UK, min_support = 0.01, use_colnames = True)
rules = association_rules(frq_items, metric ="lift", min_threshold = 1)
rules = rules.sort_values(['confidence', 'lift'], ascending =[False, False])
print(rules.head())
```

```
                                    antecedents            consequents  \
116           (BEADED CRYSTAL HEART PINK ON STICK)        (DOTCOM POSTAGE)
2018  (SUKI  SHOULDER BAG, JAM MAKING SET PRINTED)        (DOTCOM POSTAGE)
2294         (HERB MARKER MINT, HERB MARKER THYME)  (HERB MARKER ROSEMARY)
2301   (HERB MARKER PARSLEY, HERB MARKER ROSEMARY)     (HERB MARKER THYME)
2300      (HERB MARKER PARSLEY, HERB MARKER THYME)  (HERB MARKER ROSEMARY)
```

```
frq_items = apriori(basket_UK, min_support = 0.01, use_colnames = True)
rules = association_rules(frq_items, metric ="lift", min_threshold = 1)
rules = rules.sort_values(['confidence', 'lift'], ascending =[False, False])
print(rules.head())
```

```
                                    antecedents            consequents  \
116          (BEADED CRYSTAL HEART PINK ON STICK)        (DOTCOM POSTAGE)
2018  (SUKI  SHOULDER BAG, JAM MAKING SET PRINTED)        (DOTCOM POSTAGE)
2294        (HERB MARKER MINT, HERB MARKER THYME)  (HERB MARKER ROSEMARY)
2301   (HERB MARKER PARSLEY, HERB MARKER ROSEMARY)     (HERB MARKER THYME)
2300      (HERB MARKER PARSLEY, HERB MARKER THYME)  (HERB MARKER ROSEMARY)

      antecedent support  consequent support   support  confidence       lift  \
116             0.011036            0.037928  0.010768    0.975728  25.725872
2018            0.011625            0.037928  0.011196    0.963134  25.393807
2294            0.010714            0.012375  0.010232    0.955000  77.173095
2301            0.011089            0.012321  0.010553    0.951691  77.240055
2300            0.011089            0.012375  0.010553    0.951691  76.905682

      leverage  conviction
116   0.010349   39.637371
2018  0.010755   26.096206
2294  0.010099   21.947227
2301  0.010417   20.444951
2300  0.010416   20.443842
```

```
frq_items = apriori(basket_Por, min_support = 0.05, use_colnames = True)
rules = association_rules(frq_items, metric ="lift", min_threshold = 1)
rules = rules.sort_values(['confidence', 'lift'], ascending =[False, False])
print(rules.head())
```

```
                              antecedents                          consequents  \
1170  (SET 12 COLOUR PENCILS DOLLY GIRL)     (SET 12 COLOUR PENCILS SPACEBOY)
1171    (SET 12 COLOUR PENCILS SPACEBOY)   (SET 12 COLOUR PENCILS DOLLY GIRL)
1172  (SET 12 COLOUR PENCILS DOLLY GIRL)   (SET OF 4 KNICK KNACK TINS LONDON)
1173  (SET OF 4 KNICK KNACK TINS LONDON)   (SET 12 COLOUR PENCILS DOLLY GIRL)
1174  (SET 12 COLOUR PENCILS DOLLY GIRL)  (SET OF 4 KNICK KNACK TINS POPPIES)
```

```
frq_items = apriori(basket_Por, min_support = 0.05, use_colnames = True)
rules = association_rules(frq_items, metric ="lift", min_threshold = 1)
rules = rules.sort_values(['confidence', 'lift'], ascending =[False, False])
print(rules.head())
```

```
                              antecedents                          consequents  \
1170  (SET 12 COLOUR PENCILS DOLLY GIRL)     (SET 12 COLOUR PENCILS SPACEBOY)
1171    (SET 12 COLOUR PENCILS SPACEBOY)   (SET 12 COLOUR PENCILS DOLLY GIRL)
1172  (SET 12 COLOUR PENCILS DOLLY GIRL)   (SET OF 4 KNICK KNACK TINS LONDON)
1173  (SET OF 4 KNICK KNACK TINS LONDON)   (SET 12 COLOUR PENCILS DOLLY GIRL)
1174  (SET 12 COLOUR PENCILS DOLLY GIRL)  (SET OF 4 KNICK KNACK TINS POPPIES)

      antecedent support  consequent support   support  confidence       lift  \
1170            0.051724            0.051724  0.051724         1.0  19.333333
1171            0.051724            0.051724  0.051724         1.0  19.333333
1172            0.051724            0.051724  0.051724         1.0  19.333333
1173            0.051724            0.051724  0.051724         1.0  19.333333
1174            0.051724            0.051724  0.051724         1.0  19.333333

      leverage  conviction
1170  0.049049         inf
1171  0.049049         inf
1172  0.049049         inf
1173  0.049049         inf
1174  0.049049         inf
```

```
frq_items = apriori(basket_Sweden, min_support = 0.05, use_colnames = True)
rules = association_rules(frq_items, metric ="lift", min_threshold = 1)
rules = rules.sort_values(['confidence', 'lift'], ascending =[False, False])
print(rules.head())
```

```
                          antecedents                    consequents  \
0        (PACK OF 72 SKULL CAKE CASES)   (12 PENCILS SMALL TUBE SKULL)
1        (12 PENCILS SMALL TUBE SKULL)   (PACK OF 72 SKULL CAKE CASES)
4              (36 DOILIES DOLLY GIRL)  (ASSORTED BOTTLE TOP  MAGNETS)
5       (ASSORTED BOTTLE TOP  MAGNETS)         (36 DOILIES DOLLY GIRL)
180  (CHILDRENS CUTLERY CIRCUS PARADE)  (CHILDRENS CUTLERY DOLLY GIRL)
```

Conclusion: Successfully studied and implemented Apriori algorithm in python.