## Market Basket Analysis with Apriori Algorithm

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

from mlxtend.frequent_patterns import apriori, association_rules
from pandas.plotting import parallel_coordinates

pd.set_option('display.max_columns', None)
pd.set_option('display.float_format', lambda x: '%.3f' % x
              )
import warnings
warnings.filterwarnings("ignore")
```

```
df = pd.read_excel('/content/drive/MyDrive/Colab Notebooks/online_retail_II.xlsx')
```

```
sheet_name = 'Year 2010-2011'
df_new = pd.read_excel('/content/drive/MyDrive/Colab Notebooks/online_retail_II.xlsx', sheet_name = 'Year 2010-2011')
print(df_new)
```

```
       Invoice StockCode                          Description  Quantity  \
0       536365    85123A   WHITE HANGING HEART T-LIGHT HOLDER         6
1       536365     71053                  WHITE METAL LANTERN         6
2       536365    84406B       CREAM CUPID HEARTS COAT HANGER         8
3       536365    84029G  KNITTED UNION FLAG HOT WATER BOTTLE         6
4       536365    84029E       RED WOOLLY HOTTIE WHITE HEART.         6
...        ...       ...                                  ...       ...
541905  581587     22899         CHILDREN'S APRON DOLLY GIRL          6
541906  581587     23254        CHILDRENS CUTLERY DOLLY GIRL          4
541907  581587     23255      CHILDRENS CUTLERY CIRCUS PARADE         4
541908  581587     22138        BAKING SET 9 PIECE RETROSPOT          3
541909  581587      POST                              POSTAGE         1

               InvoiceDate  Price  Customer ID         Country
0      2010-12-01 08:26:00  2.550    17850.000  United Kingdom
1      2010-12-01 08:26:00  3.390    17850.000  United Kingdom
2      2010-12-01 08:26:00  2.750    17850.000  United Kingdom
3      2010-12-01 08:26:00  3.390    17850.000  United Kingdom
4      2010-12-01 08:26:00  3.390    17850.000  United Kingdom
...                    ...    ...          ...             ...
541905 2011-12-09 12:50:00  2.100    12680.000          France
541906 2011-12-09 12:50:00  4.150    12680.000          France
541907 2011-12-09 12:50:00  4.150    12680.000          France
541908 2011-12-09 12:50:00  4.950    12680.000          France
541909 2011-12-09 12:50:00 18.000    12680.000          France

[541910 rows x 8 columns]
```

```
df_new
```

| | Invoice | StockCode | Description | Quantity | InvoiceDate | Price | Customer ID | Countr |
|---|---|---|---|---|---|---|---|---|
| **0** | 536365 | 85123A | WHITE HANGING HEART T-LIGHT HOLDER | 6 | 2010-12-01 08:26:00 | 2.550 | 17850.000 | Unit Kingdo |

```
df_new.describe()
```

| | Quantity | Price | Customer ID |
|---|---|---|---|
| **count** | 541910.000 | 541910.000 | 406830.000 |
| **mean** | 9.552 | 4.611 | 15287.684 |
| **std** | 218.081 | 96.760 | 1713.603 |
| **min** | -80995.000 | -11062.060 | 12346.000 |
| **25%** | 1.000 | 1.250 | 13953.000 |
| **50%** | 3.000 | 2.080 | 15152.000 |
| **75%** | 10.000 | 4.130 | 16791.000 |
| **max** | 80995.000 | 38970.000 | 18287.000 |

```
df_new.head()
```

| | Invoice | StockCode | Description | Quantity | InvoiceDate | Price | Customer ID | Country |
|---|---|---|---|---|---|---|---|---|
| **0** | 536365 | 85123A | WHITE HANGING HEART T-LIGHT HOLDER | 6 | 2010-12-01 08:26:00 | 2.550 | 17850.000 | United Kingdom |
| **1** | 536365 | 71053 | WHITE METAL LANTERN | 6 | 2010-12-01 08:26:00 | 3.390 | 17850.000 | United Kingdom |

```
df_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541910 entries, 0 to 541909
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   Invoice      541910 non-null  object
 1   StockCode    541910 non-null  object
 2   Description  540456 non-null  object
 3   Quantity     541910 non-null  int64
 4   InvoiceDate  541910 non-null  datetime64[ns]
 5   Price        541910 non-null  float64
 6   Customer ID  406830 non-null  float64
 7   Country      541910 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(1), object(4)
memory usage: 33.1+ MB
```

```
df_new.shape
```

```
(541910, 8)
```

```
df_new['Description'] = df['Description'].str.strip() ## remove empty spaces
```

```
df_new.dropna(axis=0, subset=['Invoice'], inplace=True)  ### drop rows that dont invoice no
```

```
df_new['Invoice'] = df['Invoice'].astype('str')   ## convert invoice no to str
```

```
df_new.shape
```

```
(541910, 8)
```

```
df_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541910 entries, 0 to 541909
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   Invoice      525461 non-null  object
 1   StockCode    541910 non-null  object
 2   Description  522530 non-null  object
 3   Quantity     541910 non-null  int64
 4   InvoiceDate  541910 non-null  datetime64[ns]
 5   Price        541910 non-null  float64
 6   Customer ID  406830 non-null  float64
 7   Country      541910 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(1), object(4)
memory usage: 33.1+ MB
```

```
df_new['Invoice'].unique()
```

```
array(['489434', '489435', '489436', ..., '538170', '538171', nan],
      dtype=object)
```

```
df_new
```

| | Invoice | StockCode | Description | Quantity | InvoiceDate | Price | Customer ID | Coun |
|---|---|---|---|---|---|---|---|---|
| **0** | 489434 | 85123A | 15CM CHRISTMAS GLASS BALL 20 LIGHTS | 6 | 2010-12-01 08:26:00 | 2.550 | 17850.000 | Un Kingc |
| **1** | 489434 | 71053 | PINK CHERRY LIGHTS | 6 | 2010-12-01 08:26:00 | 3.390 | 17850.000 | Un Kingc |
| **2** | 489434 | 84406B | WHITE CHERRY LIGHTS | 8 | 2010-12-01 08:26:00 | 2.750 | 17850.000 | Un Kingc |
| **3** | 489434 | 84029G | RECORD FRAME 7" SINGLE SIZE | 6 | 2010-12-01 08:26:00 | 3.390 | 17850.000 | Un Kingc |
| **4** | 489434 | 84029E | STRAWBERRY CERAMIC TRINKET BOX | 6 | 2010-12-01 08:26:00 | 3.390 | 17850.000 | Un Kingc |

```
df_new.isnull().sum()
```

```
Invoice         16449
StockCode           0
Description     19380
Quantity            0
InvoiceDate         0
Price               0
Customer ID    135080
Country             0
dtype: int64
```

```
df_new.head()
```

| | Invoice | StockCode | Description | Quantity | InvoiceDate | Price | Customer ID | Country |
|---|---|---|---|---|---|---|---|---|
| **0** | 489434 | 85123A | 15CM CHRISTMAS GLASS BALL 20 LIGHTS | 6 | 2010-12-01 08:26:00 | 2.550 | 17850.000 | United Kingdom |
| **1** | 489434 | 71053 | PINK CHERRY LIGHTS | 6 | 2010-12-01 08:26:00 | 3.390 | 17850.000 | United Kingdom |
| | | | WHITE | | | | | |

```
df_new['Description'].fillna("No", inplace = True)
```

```
df_new.isnull().sum()
```

```
Invoice         16449
StockCode           0
Description         0
Quantity            0
InvoiceDate         0
Price               0
Customer ID    135080
Country             0
dtype: int64
```

```
df_new['Invoice'].fillna("No", inplace = True)
```

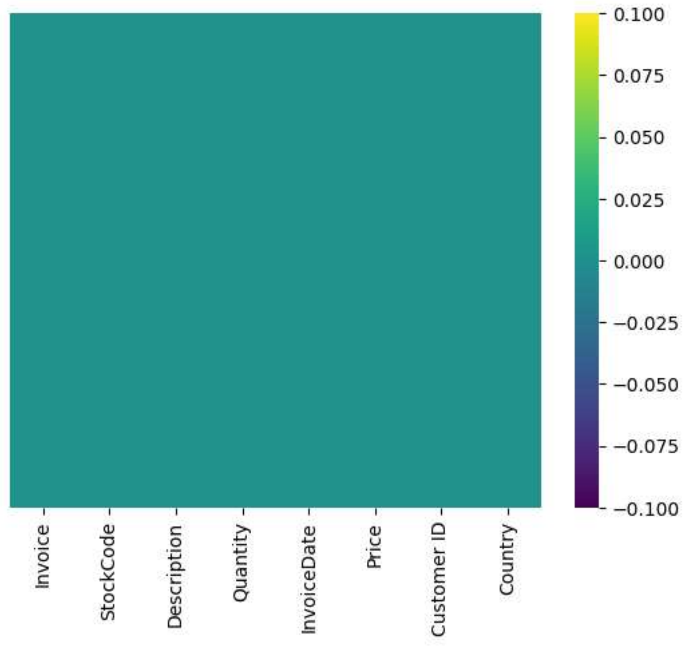
```
df_new['Customer ID'].fillna("No", inplace = True)
```

```
df_new.isnull().sum()
```

```
Invoice        0
StockCode      0
Description    0
Quantity       0
InvoiceDate    0
Price          0
Customer ID    0
Country        0
dtype: int64
```

```
sns.heatmap(df_new.isnull(),yticklabels= False , cmap = 'viridis')
```

```
<Axes: >
```



after clean up ,we needd to consolidate items into 1 transaction per row with each prroduct 1 hot encoded for sake of keeping data set small, i'm only looking at sales for UK

```
basket = (df_new[df_new['Country'] == "United Kingdom"]
          .groupby(['Invoice', 'Description'])['Quantity']
          .sum().unstack().reset_index().fillna(0)
          .set_index('Invoice'))
```

```
basket.head()
```

| Description | *Boombox Ipod Classic | *USB Office Glitter Lamp | *USB Office Mirror Ball | 10 COLOUR SPACEBOY PEN | 11 PC CERAMIC TEA SET POLKADOT | 12 ASS ZINC CHRISTMAS DECORATIONS | 12 COLOURED PARTY BALLOONS | 12 DAISY PEGS IN WOOD BOX |
|---|---|---|---|---|---|---|---|---|
| **Invoice** | | | | | | | | |
| **489434** | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| **489435** | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| **489436** | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| **489437** | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| **489438** | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |

```
basket.shape
```

```
(24536, 4618)
```

```
## below function convert a values < 0 to 0 and value greater than equal 1 to 1
def encode_units(x):
    if x <= 0:
        return 0
    if x >= 1:
        return 1

# Apply the function to the data using applymap   ## one-hot-encoded
basket_sets = basket.applymap(encode_units)
```

```
frequent_itemsets = apriori(basket_sets, min_support=0.01, use_colnames=True)
```

## Association rules

```
from mlxtend.frequent_patterns import association_rules

rules = association_rules(frequent_itemsets, metric="lift", min_threshold=10)
```

```
rules[ (rules['leverage'] >= 0) &
       (rules['confidence'] >= 0.01)]
```

| antecedents | consequents | antecedent support | consequent support | support | confidence | lift |
|---|---|---|---|---|---|---|

```
sorted_rules = rules.sort_values(by=['lift'], ascending=False)
filtered_rules = sorted_rules[sorted_rules['confidence'] > 0.5]
```

(72

```
recommendations = filtered_rules[['antecedents', 'consequents']]
```

CASES)

## product recommendations

(60 TEATIME (PACK OF 60

```
for index, row in recommendations.iterrows():
    antecedents = list(row['antecedents'])
    consequents = list(row['consequents'])
    print(f"If the customer buys {antecedents}, recommend {consequents}")
```

```
If the customer buys ['ROSES REGENCY TEACUP AND SAUCER'], recommend ['GREEN REGENCY TEACUP AND SAUCER']
If the customer buys ['GREEN REGENCY TEACUP AND SAUCER'], recommend ['ROSES REGENCY TEACUP AND SAUCER']
If the customer buys ['SET/6 RED SPOTTY PAPER PLATES'], recommend ['SET/6 RED SPOTTY PAPER CUPS']
If the customer buys ['SET/6 RED SPOTTY PAPER CUPS'], recommend ['SET/6 RED SPOTTY PAPER PLATES']
If the customer buys ['PINK 3 PIECE MINI DOTS CUTLERY SET'], recommend ['BLUE 3 PIECE MINI DOTS CUTLERY SET']
If the customer buys ['BLUE 3 PIECE MINI DOTS CUTLERY SET'], recommend ['PINK 3 PIECE MINI DOTS CUTLERY SET']
If the customer buys ['FELTCRAFT CUSHION BUTTERFLY'], recommend ['FELTCRAFT CUSHION RABBIT']
If the customer buys ['FELTCRAFT CUSHION RABBIT'], recommend ['FELTCRAFT CUSHION BUTTERFLY']
If the customer buys ['BLUE 3 PIECE MINI DOTS CUTLERY SET'], recommend ['RED 3 PIECE MINI DOTS CUTLERY SET']
If the customer buys ['RED 3 PIECE MINI DOTS CUTLERY SET'], recommend ['BLUE 3 PIECE MINI DOTS CUTLERY SET']
If the customer buys ['SPACEBOY LUNCH BOX'], recommend ['DOLLY GIRL LUNCH BOX']
If the customer buys ['DOLLY GIRL LUNCH BOX'], recommend ['SPACEBOY LUNCH BOX']
If the customer buys ['HAND WARMER OWL DESIGN'], recommend ['HAND WARMER BIRD DESIGN']
If the customer buys ['HAND WARMER BIRD DESIGN'], recommend ['HAND WARMER OWL DESIGN']
If the customer buys ['HAND WARMER OWL DESIGN'], recommend ['HAND WARMER SCOTTY DOG DESIGN']
If the customer buys ['HAND WARMER SCOTTY DOG DESIGN'], recommend ['HAND WARMER OWL DESIGN']
If the customer buys ['KITCHEN METAL SIGN'], recommend ['BATHROOM METAL SIGN']
If the customer buys ['TOILET METAL SIGN'], recommend ['BATHROOM METAL SIGN']
If the customer buys ['HAND WARMER SCOTTY DOG DESIGN'], recommend ['HAND WARMER BIRD DESIGN']
If the customer buys ['HAND WARMER BIRD DESIGN'], recommend ['HAND WARMER SCOTTY DOG DESIGN']
If the customer buys ['WOOD 2 DRAWER CABINET WHITE FINISH'], recommend ['WOOD S/3 CABINET ANT WHITE FINISH']
If the customer buys ['LARGE POPCORN HOLDER'], recommend ['SMALL POPCORN HOLDER']
If the customer buys ['PINK BLUE FELT CRAFT TRINKET BOX'], recommend ['PINK CREAM FELT CRAFT TRINKET BOX']
If the customer buys ['SINGLE HEART ZINC T-LIGHT HOLDER'], recommend ['HANGING HEART ZINC T-LIGHT HOLDER']
If the customer buys ['WOODEN FRAME ANTIQUE WHITE', 'WHITE HANGING HEART T-LIGHT HOLDER'], recommend ['WOODEN PICTURE FRAME WHITE FINISH
If the customer buys ['WOODEN PICTURE FRAME WHITE FINISH', 'WHITE HANGING HEART T-LIGHT HOLDER'], recommend ['WOODEN FRAME ANTIQUE WHITE
If the customer buys ['VINTAGE HEADS AND TAILS CARD GAME'], recommend ['VINTAGE SNAP CARDS']
If the customer buys ['WOODEN FRAME ANTIQUE WHITE'], recommend ['WOODEN PICTURE FRAME WHITE FINISH']
If the customer buys ['WOODEN PICTURE FRAME WHITE FINISH'], recommend ['WOODEN FRAME ANTIQUE WHITE']
If the customer buys ['72 SWEETHEART FAIRY CAKE CASES', 'PACK OF 60 PINK PAISLEY CAKE CASES'], recommend ['60 TEATIME FAIRY CAKE CASES']
If the customer buys ['STRAWBERRY CERAMIC TRINKET BOX', 'WHITE HANGING HEART T-LIGHT HOLDER'], recommend ['SWEETHEART CERAMIC TRINKET BC
If the customer buys ['72 SWEETHEART FAIRY CAKE CASES', '60 TEATIME FAIRY CAKE CASES'], recommend ['PACK OF 60 PINK PAISLEY CAKE CASES']
If the customer buys ['CHOCOLATE HOT WATER BOTTLE'], recommend ['HOT WATER BOTTLE TEA AND SYMPATHY']
If the customer buys ['PAINTED METAL PEARS ASSORTED'], recommend ['ASSORTED COLOUR BIRD ORNAMENT']
If the customer buys ['HEART OF WICKER SMALL'], recommend ['HEART OF WICKER LARGE']
If the customer buys ['SWEETHEART CERAMIC TRINKET BOX', 'WHITE HANGING HEART T-LIGHT HOLDER'], recommend ['STRAWBERRY CERAMIC TRINKET BC
If the customer buys ['PACK OF 60 MUSHROOM CAKE CASES'], recommend ['PACK OF 60 PINK PAISLEY CAKE CASES']
If the customer buys ['PACK OF 72 RETRO SPOT CAKE CASES', '60 TEATIME FAIRY CAKE CASES'], recommend ['PACK OF 60 PINK PAISLEY CAKE CASES
If the customer buys ['SWEETHEART CERAMIC TRINKET BOX'], recommend ['STRAWBERRY CERAMIC TRINKET BOX']
If the customer buys ['JUMBO STORAGE BAG SKULLS'], recommend ['JUMBO STORAGE BAG SUKI']
If the customer buys ['PACK OF 72 RETRO SPOT CAKE CASES', 'PACK OF 60 PINK PAISLEY CAKE CASES'], recommend ['60 TEATIME FAIRY CAKE CASES
If the customer buys ['LUNCH BAG PINK RETROSPOT'], recommend ['LUNCH BAG RED SPOTTY']
If the customer buys ['PACK OF 60 DINOSAUR CAKE CASES'], recommend ['60 TEATIME FAIRY CAKE CASES']
If the customer buys ['72 SWEETHEART FAIRY CAKE CASES'], recommend ['60 TEATIME FAIRY CAKE CASES']
If the customer buys ['SET OF 72 RETRO SPOT PAPER  DOILIES'], recommend ['PACK OF 72 RETRO SPOT CAKE CASES']
If the customer buys ['LOVE BUILDING BLOCK WORD'], recommend ['HOME BUILDING BLOCK WORD']
```

## functionalization

```
def suggest_product_offers(df):
    basket = (df[df['Country'] == "United Kingdom"]
              .groupby(['Invoice', 'Description'])['Quantity']
              .sum().unstack().reset_index().fillna(0)
              .set_index('Invoice'))

    def encode_units(x):
        if x <= 0:
            return 0
        if x >= 1:
            return 1

    basket_sets = basket.applymap(encode_units)

    frequent_itemsets = apriori(basket_sets, min_support=0.01, use_colnames=True)

    rules = association_rules(frequent_itemsets, metric="lift", min_threshold=10)

    sorted_rules = rules.sort_values(by=['lift'], ascending=False)
    filtered_rules = sorted_rules[sorted_rules['confidence'] > 0.5]

    recommendations = filtered_rules[['antecedents', 'consequents']]

    product_offers = []
    for index, row in recommendations.iterrows():
        antecedents = list(row['antecedents'])
        consequents = list(row['consequents'])
        product_offers.append((antecedents, consequents))

    return product_offers



product_offers = suggest_product_offers(df_new)
print(product_offers)
```

```
[(['SET/6 RED SPOTTY PAPER CUPS'], ['SET/6 RED SPOTTY PAPER PLATES']), (['SET/6 RED SPOTTY PAPER PLATES'], ['SET/6 RED SPOTTY PAPER CUPS
```