**Market Basket Insights**

**Artificial Intelligence - Group 1**

**Market Basket Insights**

**Problem Definition:** The problem is to perform market basket analysis on a provided dataset to unveil hidden patterns and associations between products. The goal is to understand customer purchasing behavior and identify potential cross-selling opportunities for a retail business. This project involves using association analysis techniques, such as Apriori algorithm, to find frequently co-occurring products and generate insights for business optimization.

**Design Thinking:**

**Step: 1 Data Source Selection**

* - Choose a dataset that represents real- world sale data, similar as deals records from a retail store,e-commerce platform, or point- of- trade system.
* - insure that the dataset contains applicable information, including sale IDs, timestamps, and lists of bought products.
* - Consider the size of the dataset, as a larger dataset may give further comprehensive perceptivity but may also bear further computational coffers.
* - corroborate the quality and integrity of the data, addressing issues like missing values, outliers, and data thickness.
* - Understand the source of the data, its update frequence, and whether it aligns with the objects of the request handbasket analysis.

**Step: 2 Data Preprocessing**

* - Remove indistinguishable deals to insure that each sale is counted only formerly.
* - Handle missing product information by attributing missing values or banning deals with missing data if they're negligible.
* - Render categorical variables, similar as product names, into a format suitable for the Apriori algorithm, like double or one-hot encoding.
* - Explore and fantasize introductory statistics and patterns in the data, similar as sale frequence, popular products, and seasonality.
* - homogenize or gauge the data if necessary to alleviate the impact of varying sale sizes.

**Step: 3 Association Analysis( Apriori Algorithm)**

* - Configure the minimal support threshold, specifying the minimum frequence that an itemset must meet to be considered" frequent."
* - Set the minimal confidence threshold, indicating the position of certainty needed for an association rule to be considered significant.
* - induce frequent itemsets by applying the Apriori algorithm, which employs alevel-wise hunt approach.
* - pare occasional itemsets to optimize computational effectiveness and concentrate on applicable associations.
* - induce association rules that detail product connections grounded on support, confidence, and lift values.

**Step: 4 perceptivity Generation**

* - dissect association rules to identify meaningful patterns and product associations.
* - Focus on high- confidence rules, which indicate strong associations between products.
* - probe lift values to distinguish between arbitrary associations and those with factual significance.
* - Consider the directionality of association rules(e.g., A-> B and B-> A) to understand the inflow of product purchases.
* - Explore the size and imbrication of itemsets to gain perceptivity into the complexity of client geste .

**Step: 5 Visualization**

* - produce bar maps or histograms to fantasize the frequence of product combinations.
* - figure heatmaps to represent the strength and direction of associations between products.
* - Construct network plates to show interconnections between products in a visually intuitive way.
* - Use pie maps or word shadows to punctuate the most common product dyads or associations.
* - Consider interactive dashboards for stakeholders to explore the data and perceptivity stoutly.

**Step: 6 Business Recommendations**

* - Knitter recommendations to specific business objects, similar as adding deals, optimizing force, or perfecting marketing strategies.
* - give practicable perceptivity on which products to rush together forcross-selling openings.
* - Suggest strategies for product placement within the store or one-commerce platforms grounded on association findings.
* - Recommend substantiated marketing juggernauts targeting guests who constantly buy certain product combinations.
* - estimate the implicit impact of enforcing these recommendations and estimate ROI( Return on Investment) when possible.

**Introduction:**

Market Basket Analysis (MBA) is a powerful data mining technique widely used in retail and various industries to uncover hidden patterns and relationships within transactional data. It enables businesses to gain valuable insights into customer behavior by identifying associations between products or items frequently purchased together. By understanding these purchase patterns, companies can make informed decisions to improve sales, optimize inventory, enhance marketing strategies, and ultimately boost profitability.

One of the most popular and effective algorithms for performing Market Basket Analysis is the Apriori algorithm. Apriori is designed to efficiently extract frequent itemsets from large datasets, making it an essential tool for retailers and businesses aiming to uncover valuable associations in their transactional data.

In this guide, we will delve into the essential steps of Market Basket Analysis with the Apriori algorithm. We will explore how to prepare your data, set meaningful support and confidence thresholds, generate frequent itemsets, create association rules, and interpret the results. Additionally, we will provide practical examples and code snippets to help you apply Apriori-based Market Basket Analysis in your own business context.

By the end of this guide, you will have a comprehensive understanding of how Market Basket Analysis with the Apriori algorithm can empower your organization to make data-driven decisions, optimize product placements, design effective marketing campaigns, and ultimately provide a better shopping experience for your customers.

**Program:**

*# This Python 3 environment comes with many helpful analytics libraries installed*

*# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python*

*# For example, here's several helpful packages to load*

import numpy as np *# linear algebra*

import pandas as pd *# data processing, CSV file I/O (e.g. pd.read\_csv)*

*# Input data files are available in the read-only "../input/" directory*

*# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory*

import os

for dirname, \_, filenames **in** os.walk('/kaggle/input'):

for filename **in** filenames:

print(os.path.join(dirname, filename))

*# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"*

*# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session*

/kaggle/input/market-basket-analysis/Assignment-1\_Data.xlsx

/kaggle/input/market-basket-analysis/Assignment-1\_Data.csv

In [2]: from matplotlib import pyplot as plt

df=pd.read\_excel("/kaggle/input/market-basket-analysis/Assignment-1\_Data.xlsx")

In [3]: df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 522064 entries, 0 to 522063

Data columns (total 7 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 BillNo 522064 non-null object

1 Itemname 520609 non-null object

2 Quantity 522064 non-null int64

3 Date 522064 non-null datetime64[ns]

4 Price 522064 non-null float64

5 CustomerID 388023 non-null float64

6 Country 522064 non-null object

dtypes: datetime64[ns](1), float64(2), int64(1), object(3)

memory usage: 27.9+ MB

**Step 1: Data Hygiene** We're going to do the following steps:

1. Drop any rows where item name column is null.
2. Drop any rows where item quantity sold is 0 or less.
3. Fill missing customer IDs with a placeholder ID (99999)
4. Create a new column, Sumprice, that tells us total sales revenue (Quantity \* Price) of the item

In [4]: df.isnull().sum()

Out[4]:

BillNo 0

Itemname 1455

Quantity 0

Date 0

Price 0

CustomerID 134041

Country 0

dtype: int64

In [5]:

*#Dropping rows where ItemName isn't available*

df.dropna(subset=["Itemname"],inplace=True)

*#Dropping rows where Quantity <=0*

df = df[df["Quantity"]>0]

df.isnull().sum()

Out[5]:

BillNo 0

Itemname 0

Quantity 0

Date 0

Price 0

CustomerID 132113

Country 0

dtype: int64

In [6]:

*Fill missing customer IDs*

df['CustomerID'].fillna(99999, inplace=True)

*#Create SumPrice column*

df["SumPrice"]=df["Quantity"]\*df["Price"]

**Step 2: EDA** Let's explore the data for any insights. Let's find which countries sell the most items, and what items are the most popular in each country.

In [7]:

*#Find the best selling items in each country*

best\_selling\_items = df.groupby(['Country', 'Itemname']).agg({'Quantity': 'sum'}).reset\_index()

best\_selling\_items = best\_selling\_items.groupby('Country').apply(lambda x: x[x['Quantity'] == x['Quantity'].max()]).reset\_index(drop=True)

best\_selling\_items.sort\_values("Quantity",ascending=False)

Out[7]:

|  | Country | Itemname | Quantity |
| --- | --- | --- | --- |
| 47 | United Kingdom | PAPER CRAFT , LITTLE BIRDIE | 80995 |
| 25 | Netherlands | RABBIT NIGHT LIGHT | 4801 |
| 12 | France | RABBIT NIGHT LIGHT | 4024 |
| 20 | Japan | RABBIT NIGHT LIGHT | 3408 |
| 0 | Australia | MINI PAINT SET VINTAGE | 2952 |
| 42 | Sweden | MINI PAINT SET VINTAGE | 2916 |
| 13 | Germany | ROUND SNACK BOXES SET OF4 WOODLAND | 1233 |
| 41 | Spain | CHILDRENS CUTLERY POLKADOT PINK | 729 |
| 43 | Switzerland | PLASTERS IN TIN WOODLAND ANIMALS | 639 |
| 26 | Norway | SMALL FOLDING SCISSOR(POINTED EDGE) | 576 |
| 3 | Belgium | PACK OF 72 RETROSPOT CAKE CASES | 480 |
| 40 | Singapore | CHRISTMAS TREE PAINTED ZINC | 384 |
| 1 | Austria | SET 12 KIDS COLOUR CHALK STICKS | 288 |
| 17 | Iceland | ICE CREAM SUNDAE LIP GLOSS | 240 |
| 19 | Italy | FEATHER PEN,HOT PINK | 240 |
| 29 | Portugal | POLKADOT PEN | 240 |
| 16 | Hong Kong | ROUND SNACK BOXES SET OF4 WOODLAND | 150 |
| 28 | Poland | STRAWBERRY CERAMIC TRINKET BOX | 144 |
| 27 | Poland | CERAMIC CAKE DESIGN SPOTTED MUG | 144 |
| 18 | Israel | WOODLAND CHARLOTTE BAG | 130 |
| 48 | Unspecified | WORLD WAR 2 GLIDERS ASSTD DESIGNS | 96 |
| 2 | Bahrain | ICE CREAM SUNDAE LIP GLOSS | 96 |
| 44 | USA | SET 12 COLOURING PENCILS DOILY | 88 |
| 24 | Malta | GRAND CHOCOLATECANDLE | 81 |
| 46 | United Arab Emirates | BIG DOUGHNUT FRIDGE MAGNETS | 72 |
| 45 | United Arab Emirates | ASSORTED CHEESE FRIDGE MAGNETS | 72 |
| 22 | Lithuania | FELTCRAFT DOLL ROSIE | 48 |
| 23 | Lithuania | RED HARMONICA IN BOX | 48 |
| 15 | Greece | 4 PEAR BOTANICAL DINNER CANDLES | 48 |
| 14 | Greece | 4 LAVENDER BOTANICAL DINNER CANDLES | 48 |
| 4 | Brazil | DOLLY GIRL LUNCH BOX | 24 |
| 5 | Brazil | GREEN REGENCY TEACUP AND SAUCER | 24 |
| 6 | Brazil | PINK REGENCY TEACUP AND SAUCER | 24 |
| 7 | Brazil | ROSES REGENCY TEACUP AND SAUCER | 24 |
| 21 | Lebanon | ASSTD FRUIT+FLOWERS FRIDGE MAGNETS | 24 |
| 8 | Brazil | SET OF 4 PANTRY JELLY MOULDS | 24 |
| 9 | Brazil | SET OF 6 SPICE TINS PANTRY DESIGN | 24 |
| 10 | Brazil | SET/3 RED GINGHAM ROSE STORAGE BOX | 24 |
| 11 | Brazil | SMALL HEART FLOWERS HOOK | 24 |
| 34 | RSA | WOODEN BOX OF DOMINOES | 12 |
| 36 | Saudi Arabia | HOMEMADE JAM SCENTED CANDLES | 12 |
| 37 | Saudi Arabia | PLASTERS IN TIN CIRCUS PARADE | 12 |
| 38 | Saudi Arabia | PLASTERS IN TIN SKULLS | 12 |
| 39 | Saudi Arabia | PLASTERS IN TIN STRONGMAN | 12 |
| 33 | RSA | SET OF 20 KIDS COOKIE CUTTERS | 12 |
| 32 | RSA | PACK OF 6 BIRDY GIFT TAGS | 12 |
| 31 | RSA | ASSORTED BOTTLE TOP MAGNETS | 12 |
| 30 | RSA | 4 TRADITIONAL SPINNING TOPS | 12 |
| 35 | Saudi Arabia | ASSORTED BOTTLE TOP MAGNETS | 12 |

In [8]:

*#Find the total sales by country.*

total\_sales\_country = df.groupby(['Country']).agg({'SumPrice': 'sum'}).reset\_index()

total\_sales\_country = total\_sales\_country.sort\_values('SumPrice', ascending=False).reset\_index(drop=True)

total\_sales\_country

Out[8]:

|  | Country | SumPrice |
| --- | --- | --- |
| 0 | United Kingdom | 9003097.964 |
| 1 | Netherlands | 285446.340 |
| 2 | Germany | 228867.140 |
| 3 | France | 209715.110 |
| 4 | Australia | 138521.310 |
| 5 | Spain | 61577.110 |
| 6 | Switzerland | 57089.900 |
| 7 | Belgium | 41196.340 |
| 8 | Sweden | 38378.330 |
| 9 | Japan | 37416.370 |
| 10 | Norway | 36165.440 |
| 11 | Portugal | 33747.100 |
| 12 | Singapore | 21279.290 |
| 13 | Italy | 17483.240 |
| 14 | Hong Kong | 15691.800 |
| 15 | Austria | 10198.680 |
| 16 | Israel | 8135.260 |
| 17 | Poland | 7334.650 |
| 18 | Greece | 4760.520 |
| 19 | Unspecified | 4749.790 |
| 20 | Iceland | 4310.000 |
| 21 | USA | 3580.390 |
| 22 | Malta | 2725.590 |
| 23 | United Arab Emirates | 1902.280 |
| 24 | Lebanon | 1693.880 |
| 25 | Lithuania | 1661.060 |
| 26 | Brazil | 1143.600 |
| 27 | RSA | 1002.310 |
| 28 | Bahrain | 754.140 |
| 29 | Saudi Arabia | 145.920 |

In [9]:

*#Visualizing Total sales by country.*

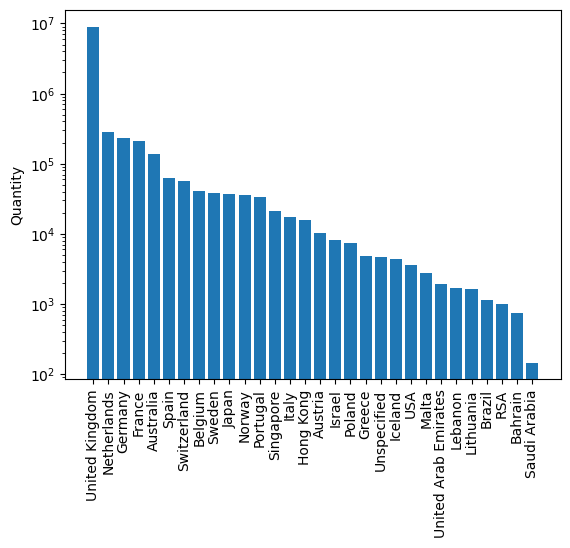
plt.bar(total\_sales\_country["Country"],total\_sales\_country["SumPrice"])

plt.yscale('log')

plt.ylabel('Quantity')

plt.xticks(rotation=90)

plt.show()



So far we've noticed that the UK has the most amount of sales and the most popular item sold in UK is 'PAPER CRAFT, LITTLE BIRDIE'. However, this outsells the most popular items in other countries by a large magnitude. Let's dig in by only looking at UK's grocery store data.

In [10]:

*#Isolate the UK data and let's sort the most popular items in UK by quantity sold.*

only\_uk = df[df["Country"]=="United Kingdom"]

only\_uk.groupby("Itemname")["Quantity"].sum().sort\_values(ascending=False)

Out[10]:

Itemname

PAPER CRAFT , LITTLE BIRDIE 80995

MEDIUM CERAMIC TOP STORAGE JAR 77036

WORLD WAR 2 GLIDERS ASSTD DESIGNS 49526

JUMBO BAG RED RETROSPOT 44268

WHITE HANGING HEART T-LIGHT HOLDER 35744

...

HEN HOUSE W CHICK IN NEST 1

BLACKCHRISTMAS TREE 30CM 1

GOLD COSMETICS BAG WITH BUTTERFLY 1

WATERING CAN SINGLE HOOK PISTACHIO 1

\*Boombox Ipod Classic 1

Name: Quantity, Length: 4046, dtype: int64

In[11]

total\_sales\_item = df.groupby(['Itemname']).agg({'Price': 'mean', 'Quantity': 'sum', 'SumPrice': 'sum'}).reset\_index()

*# Create a new column with the count of rows for each group*

total\_sales\_item['Count'] = df.groupby(['Itemname']).size().values

*# Sort the dataframe by 'SumPrice' column in descending order*

total\_sales\_item = total\_sales\_item.sort\_values("SumPrice", ascending=False)

total\_sales\_item

Out[11]:

|  | Itemname | Price | Quantity | SumPrice | Count |
| --- | --- | --- | --- | --- | --- |
| 1060 | DOTCOM POSTAGE | 291.311822 | 708 | 206248.77 | 708 |
| 2386 | PAPER CRAFT , LITTLE BIRDIE | 2.080000 | 80995 | 168469.60 | 1 |
| 2848 | REGENCY CAKESTAND 3 TIER | 14.043347 | 13119 | 165689.19 | 1930 |
| 3840 | WHITE HANGING HEART T-LIGHT HOLDER | 3.220569 | 36527 | 102588.37 | 2269 |
| 2411 | PARTY BUNTING | 5.808664 | 17812 | 97367.48 | 1677 |
| ... | ... | ... | ... | ... | ... |
| 4025 | allocate stock for dotcom orders ta | 0.000000 | 4 | 0.00 | 1 |
| 4026 | amazon | 0.000000 | 161 | 0.00 | 8 |
| 4027 | amazon adjust | 0.000000 | 10 | 0.00 | 1 |
| 4028 | amazon sales | 0.000000 | 20 | 0.00 | 1 |
| 255 | Adjust bad debt | -3687.353333 | 3 | -11062.06 | 3 |

Interesting. We find out that the most sold item globally, 'PAPER CRAFT, LITTLE BIRDIE' was sold in just one transaction. Perhaps this was a large corporate order. If we were to ever do a marketing or promotional push in the future, that required us to analyse our most popular products, this would be an anomaly that we would need to adjust for.

**Step 3: EDA** Market Basket Analysis using Apriori Algorithm and Association Rule Mining

1. Convert the Dataset into transactional format (Each row is one bill number with every item sold in that bill in a list)
2. Create a one-hot matrix of the products (Product sold = 1, Not sold = 0)
3. Merge the transactional matrix and the one hot matrix
4. Import the mlxtend library and perform association mining and generate association rules

In [12]:

*#Convert the dataset into transactional format*

transactions = df.groupby(['BillNo'])['Itemname'].apply(list)

transactions

Out[12]:

BillNo

536365 [WHITE HANGING HEART T-LIGHT HOLDER, WHITE MET...

536366 [HAND WARMER UNION JACK, HAND WARMER RED POLKA...

536367 [ASSORTED COLOUR BIRD ORNAMENT, POPPY'S PLAYHO...

536368 [JAM MAKING SET WITH JARS, RED COAT RACK PARIS...

536369 [BATH BUILDING BLOCK WORD]

...

581586 [LARGE CAKE STAND HANGING STRAWBERY, SET OF 3...

581587 [CIRCUS PARADE LUNCH BOX, PLASTERS IN TIN CIRC...

A563185 [Adjust bad debt]

A563186 [Adjust bad debt]

A563187 [Adjust bad debt]

Name: Itemname, Length: 19735, dtype: object

In [13]

*#Create a one-hot matrix of the products*

one\_hot = pd.get\_dummies(df['Itemname'])

one\_hot

Out [13]

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **\*Boombox Ipod Classic** | **\*USB Office Mirror Ball** | **10 COLOUR SPACEBOY PEN** | **12 COLOURED PARTY BALLOONS** | **12 DAISY PEGS IN WOOD BOX** | **12 EGG HOUSE PAINTED WOOD** | **12 HANGING EGGS HAND PAINTED** | **12 IVORY ROSE PEG PLACE SETTINGS** | **12 MESSAGE CARDS WITH ENVELOPES** | **12 PENCIL SMALL TUBE WOODLAND** | **...** |
| **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **1** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **2** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **3** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **4** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **...** | **...** | **...** | **...** | **...** | **...** | **...** | **...** | **...** | **...** | **...** |
| **522059** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **522060** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **522061** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **522062** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **522063** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **taig adjust** | **taig adjust** | **taig adjust** | **taig adjust** | **taig adjust** | **taig adjust** | **taig adjust** | **taig adjust** | **taig adjust** | **taig adjust** | **taig adjust** |
| **...** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **536365** |
| **...** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **536365** |
| **...** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **536365** |
| **...** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **536365** |
| **...** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **536365** |
| **...** | **...** | **...** | **...** | **...** | **...** | **...** | **...** | **...** | **...** | **...** |
| **...** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **581587** |
| **...** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **581587** |
| **...** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **581587** |
| **...** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **581587** |
| **...** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **581587** |

In [14]

*#Add the BillNo column back to the one-hot encoded matrix*

one\_hot['BillNo']=df['BillNo']

one\_hot

Out [14]

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **\*Boombox Ipod Classic** | **\*USB Office Mirror Ball** | **10 COLOUR SPACEBOY PEN** | **12 COLOURED PARTY BALLOONS** | **12 DAISY PEGS IN WOOD BOX** | **12 EGG HOUSE PAINTED WOOD** | **12 HANGING EGGS HAND PAINTED** | **12 IVORY ROSE PEG PLACE SETTINGS** | **12 MESSAGE CARDS WITH ENVELOPES** | **12 PENCIL SMALL TUBE WOODLAND** | **...** |
| **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **1** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **2** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **3** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **4** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **...** | **...** | **...** | **...** | **...** | **...** | **...** | **...** | **...** | **...** | **...** |
| **522059** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **522060** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **522061** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **522062** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **522063** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **taig adjust** | **taig adjust** | **taig adjust** | **taig adjust** | **taig adjust** | **taig adjust** | **taig adjust** | **taig adjust** | **taig adjust** | **taig adjust** | **taig adjust** |
| **...** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **536365** |
| **...** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **536365** |
| **...** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **536365** |
| **...** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **536365** |
| **...** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **536365** |
| **...** | **...** | **...** | **...** | **...** | **...** | **...** | **...** | **...** | **...** | **...** |
| **...** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **581587** |
| **...** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **581587** |
| **...** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **581587** |
| **...** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **581587** |
| **...** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **581587** |

In[15]

*#Now, we group the One-Hot Matrix by BillNo and sum the values*

one\_hot = one\_hot.groupby('BillNo').sum()

one\_hot

Out[15]

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| \*Boombox Ipod Classic | \*USB Office Mirror Ball | 10 COLOUR SPACEBOY PEN | 12 COLOURED PARTY BALLOONS | 12 DAISY PEGS IN WOOD BOX | 12 EGG HOUSE PAINTED WOOD | 12 HANGING EGGS HAND PAINTED | 12 IVORY ROSE PEG PLACE SETTINGS | 12 MESSAGE CARDS WITH ENVELOPES | 12 PENCIL SMALL TUBE WOODLAND | ... |
| BillNo |  |  |  |  |  |  |  |  |  |  |
| 536365 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 536366 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 536367 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 536368 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 536369 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 581586 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 581587 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| A563185 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| A563186 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| A563187 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| returned | taig adjust | test | to push order througha s stock was | website fixed | wrongly coded 20713 | wrongly coded 23343 | wrongly marked | wrongly marked 23343 | wrongly sold (22719) barcode |  |
|  |  |  |  |  |  |  |  |  |  |  |
| ... | 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 | 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 | 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 | 0 | 0 |
| ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

In[16]

*#Now, we merge the one-hot encoded matrix, with the transactional data*

transaction\_matrix = pd.merge(transactions, one\_hot, on='BillNo')

transaction\_matrix

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Itemname | \*Boombox Ipod Classic | \*USB Office Mirror Ball | 10 COLOUR SPACEBOY PEN | 12 COLOURED PARTY BALLOONS | 12 DAISY PEGS IN WOOD BOX | 12 EGG HOUSE PAINTED WOOD | 12 HANGING EGGS HAND PAINTED | 12 IVORY ROSE PEG PLACE SETTINGS | 12 MESSAGE CARDS WITH ENVELOPES | ... |
| BillNo |  |  |  |  |  |  |  |  |  |  |
| 536365 | [WHITE HANGING HEART T-LIGHT HOLDER, WHITE MET... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 536366 | [HAND WARMER UNION JACK, HAND WARMER RED POLKA... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 536367 | [ASSORTED COLOUR BIRD ORNAMENT, POPPY'S PLAYHO... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 536368 | [JAM MAKING SET WITH JARS, RED COAT RACK PARIS... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 536369 | [BATH BUILDING BLOCK WORD] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

In [17]:

*#Now we have to convert the product columns to 0s and 1s. We are converting sum values to binary as number doesn't matter*

transaction\_matrix[one\_hot.columns[:-1]] = (transaction\_matrix[one\_hot.columns[:-1]] >= 1).astype(int)

transaction\_matrix

Out [17]

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| emname | \*Boombox Ipod Classic | \*USB Office Mirror Ball | 10 COLOUR SPACEBOY PEN | 12 COLOURED PARTY BALLOONS | 12 DAISY PEGS IN WOOD BOX | 12 EGG HOUSE PAINTED WOOD | 12 HANGING EGGS HAND PAINTED | 12 IVORY ROSE PEG PLACE SETTINGS | 12 MESSAGE CARDS WITH ENVELOPES | ... |
| BillNo |  |  |  |  |  |  |  |  |  |  |
| 536365 | [WHITE HANGING HEART T-LIGHT HOLDER, WHITE MET... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 536366 | [HAND WARMER UNION JACK, HAND WARMER RED POLKA... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 536367 | [ASSORTED COLOUR BIRD ORNAMENT, POPPY'S PLAYHO... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 536368 | [JAM MAKING SET WITH JARS, RED COAT RACK PARIS... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

In [18]

from mlxtend.frequent\_patterns import apriori

from mlxtend.frequent\_patterns import association\_rules

print(transaction\_matrix.dtypes)

Out [18]

Itemname object

\*Boombox Ipod Classic int64

\*USB Office Mirror Ball int64

10 COLOUR SPACEBOY PEN int64

12 COLOURED PARTY BALLOONS int64

...

wrongly coded 20713 int64

wrongly coded 23343 int64

wrongly marked int64

wrongly marked 23343 int64

wrongly sold (22719) barcode uint8

Length: 4057, dtype: object

In [19]

transaction\_matrix.iloc[:, 1:] = transaction\_matrix.iloc[:, 1:].astype(bool)

*#Perform frequent itemset mining*

frequent\_itemsets = apriori(transaction\_matrix.iloc[:, 1:], min\_support=0.01, use\_colnames=True)

frequent\_itemsets

Out [19]

| support | itemsets |
| --- | --- |
| 0 | 0.015809 | (10 COLOUR SPACEBOY PEN) |
| 1 | 0.012567 | (12 MESSAGE CARDS WITH ENVELOPES) |
| 2 | 0.017887 | (12 PENCIL SMALL TUBE WOODLAND) |
| 3 | 0.018242 | (12 PENCILS SMALL TUBE RED RETROSPOT) |
| 4 | 0.017887 | (12 PENCILS SMALL TUBE SKULL) |
| ... | ... | ... |
| 1891 | 0.011249 | (JUMBO BAG RED RETROSPOT, JUMBO SHOPPER VINTAG... |
| 1892 | 0.011249 | (LUNCH BAG CARS BLUE, LUNCH BAG BLACK SKULL.,... |
| 1893 | 0.010388 | (LUNCH BAG CARS BLUE, LUNCH BAG BLACK SKULL.,... |
| 1894 | 0.010286 | (LUNCH BAG SUKI DESIGN, LUNCH BAG BLACK SKULL... |
| 1895 | 0.010286 | (CHARLOTTE BAG PINK POLKADOT, CHARLOTTE BAG SU.. |

In [20]

*# generate association rules*

rules = association\_rules(frequent\_itemsets, metric="lift", min\_threshold=1)

rules

Out [20]

| antecedents | consequents | antecedent support | consequent support | support | confidence | lift | leverage | conviction | zhangs\_metric |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | (DOTCOM POSTAGE) | (6 RIBBONS RUSTIC CHARM) | 0.035875 | 0.047732 | 0.010236 | 0.285311 | 5.977290 | 0.008523 | 1.332422 | 0.863685 |
| 1 | (6 RIBBONS RUSTIC CHARM) | (DOTCOM POSTAGE) | 0.047732 | 0.035875 | 0.010236 | 0.214437 | 5.977290 | 0.008523 | 1.227305 | 0.874439 |
| 2 | (JAM MAKING SET PRINTED) | (6 RIBBONS RUSTIC CHARM) | 0.056549 | 0.047732 | 0.011806 | 0.208781 | 4.373992 | 0.009107 | 1.203545 | 0.817611 |
| 3 | (6 RIBBONS RUSTIC CHARM) | (JAM MAKING SET PRINTED) | 0.047732 | 0.056549 | 0.011806 | 0.247346 | 4.373992 | 0.009107 | 1.253499 | 0.810041 |
| 4 | (6 RIBBONS RUSTIC CHARM) | (JAM MAKING SET WITH JARS) | 0.047732 | 0.055181 | 0.010337 | 0.216561 | 3.924538 | 0.007703 | 1.205988 | 0.782546 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 3337 | (CHARLOTTE BAG PINK POLKADOT) | (RED RETROSPOT CHARLOTTE BAG, STRAWBERRY CHARL... | 0.037395 | 0.013073 | 0.010286 | 0.275068 | 21.040551 | 0.009797 | 1.361406 | 0.989475 |
| 3338 | (CHARLOTTE BAG SUKI DESIGN) | (STRAWBERRY CHARLOTTE BAG, CHARLOTTE BAG PINK ... | 0.044337 | 0.012212 | 0.010286 | 0.232000 | 18.998008 | 0.009745 | 1.286183 | 0.991315 |
| 3339 | (STRAWBERRY CHARLOTTE BAG) | (RED RETROSPOT CHARLOTTE BAG, CHARLOTTE BAG PI... | 0.036281 | 0.012668 | 0.010286 | 0.283520 | 22.381034 | 0.009827 | 1.378031 | 0.991284 |
| 3340 | (WOODLAND CHARLOTTE BAG) | (STRAWBERRY CHARLOTTE BAG, CHARLOTTE BAG PINK ... | 0.041905 | 0.012364 | 0.010286 | 0.245466 | 19.853534 | 0.009768 | 1.308934 | 0.991166 |
| 3341 | (RED RETROSPOT CHARLOTTE BAG) | (STRAWBERRY CHARLOTTE BAG, CHARLOTTE BAG PINK ... | 0.052090 | 0.011198 | 0.010286 | 0.197471 | 17.633876 | 0.009703 | 1.232107 | 0.995127 |

In [21]

*#Let's see the top 10 rules by lift*

rules.sort\_values('lift', ascending=False).head(10)

Out [21]

| antecedents | consequents | antecedent support | consequent support | support | confidence | lift | leverage | conviction | zhangs\_metric |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2080 | (HERB MARKER THYME) | (HERB MARKER PARSLEY, HERB MARKER ROSEMARY) | 0.011806 | 0.010641 | 0.010134 | 0.858369 | 80.666258 | 0.010009 | 6.985474 | 0.999403 |
| 2077 | (HERB MARKER PARSLEY, HERB MARKER ROSEMARY) | (HERB MARKER THYME) | 0.010641 | 0.011806 | 0.010134 | 0.952381 | 80.666258 | 0.010009 | 20.752065 | 0.998225 |
| 2081 | (HERB MARKER ROSEMARY) | (HERB MARKER PARSLEY, HERB MARKER THYME) | 0.011857 | 0.010641 | 0.010134 | 0.854701 | 80.321530 | 0.010008 | 6.809118 | 0.999400 |
| 2076 | (HERB MARKER PARSLEY, HERB MARKER THYME) | (HERB MARKER ROSEMARY) | 0.010641 | 0.011857 | 0.010134 | 0.952381 | 80.321530 | 0.010008 | 20.751001 | 0.998172 |
| 534 | (HERB MARKER THYME) | (HERB MARKER ROSEMARY) | 0.011806 | 0.011857 | 0.010996 | 0.931330 | 78.546183 | 0.010856 | 14.389831 | 0.999064 |
| 535 | (HERB MARKER ROSEMARY) | (HERB MARKER THYME) | 0.011857 | 0.011806 | 0.010996 | 0.927350 | 78.546183 | 0.010856 | 13.602194 | 0.999115 |
| 2079 | (HERB MARKER PARSLEY) | (HERB MARKER THYME, HERB MARKER ROSEMARY) | 0.011756 | 0.010996 | 0.010134 | 0.862069 | 78.400604 | 0.010005 | 7.170281 | 0.998989 |
| 2078 | (HERB MARKER THYME, HERB MARKER ROSEMARY) | (HERB MARKER PARSLEY) | 0.010996 | 0.011756 | 0.010134 | 0.921659 | 78.400604 | 0.010005 | 12.614647 | 0.998221 |

In [22]

import mpld3

*# create scatter plot with x and y as lift and confidence values*

fig, ax = plt.subplots()

scatter = ax.scatter(rules['lift'], rules['confidence'], alpha=0.5)

*# Define tooltips*

tooltips = []

for i **in** range(len(rules)):

rule = rules.iloc[i]

tooltip = f"Rule: **{**rule['antecedents']**}** -> **{**rule['consequents']**}\n**Support: **{**rule['support']**:**.3f**}\n**Confidence: **{**rule['confidence']**:**.3f**}\n**Lift: **{**rule['lift']**:**.3f**}**"

tooltips.append(tooltip)

*# Add tooltips to scatter plot using mpld3*

mpld3.plugins.connect(fig, mpld3.plugins.PointHTMLTooltip(scatter, tooltips))

*# Set axis labels and title*

ax.set\_xlabel("Lift")

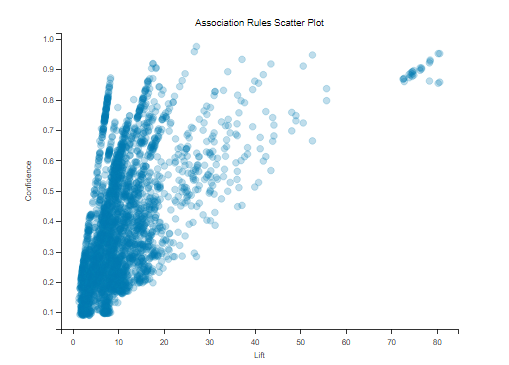
ax.set\_ylabel("Confidence")

ax.set\_title("Association Rules Scatter Plot")

*# Show the plot*

mpld3.display()

Out [22]



In [23]

rules[(rules['lift'] > 40) & (rules['lift'] < 50)]

Out [23]

| antecedents | consequents | antecedent support | consequent support | support | confidence | lift | leverage | conviction | zhangs\_metric |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 124 | (BLUE POLKADOT CUP) | (PINK POLKADOT CUP) | 0.016418 | 0.015505 | 0.010489 | 0.638889 | 41.204158 | 0.010234 | 2.726293 | 0.992017 |
| 125 | (PINK POLKADOT CUP) | (BLUE POLKADOT CUP) | 0.015505 | 0.016418 | 0.010489 | 0.676471 | 41.204158 | 0.010234 | 3.040164 | 0.991098 |
| 264 | (CHILDRENS CUTLERY SPACEBOY) | (CHILDRENS CUTLERY DOLLY GIRL) | 0.017938 | 0.014441 | 0.010996 | 0.612994 | 42.447170 | 0.010737 | 2.546626 | 0.994276 |
| 265 | (CHILDRENS CUTLERY DOLLY GIRL) | (CHILDRENS CUTLERY SPACEBOY) | 0.014441 | 0.017938 | 0.010996 | 0.761404 | 42.447170 | 0.010737 | 4.115997 | 0.990749 |
| 552 | (JAM JAR WITH PINK LID) | (JAM JAR WITH GREEN LID) | 0.016874 | 0.015100 | 0.011198 | 0.663664 | 43.951015 | 0.010944 | 2.928319 | 0.994020 |
| 553 | (JAM JAR WITH GREEN LID) | (JAM JAR WITH PINK LID) | 0.015100 | 0.016874 | 0.011198 | 0.741611 | 43.951015 | 0.010944 | 3.804827 | 0.992230 |
| 1556 | (REGENCY SUGAR BOWL GREEN) | (REGENCY MILK JUG PINK) | 0.014897 | 0.015252 | 0.011148 | 0.748299 | 49.062083 | 0.010920 | 3.912377 | 0.994432 |
| 1557 | (REGENCY MILK JUG PINK) | (REGENCY SUGAR BOWL GREEN) | 0.015252 | 0.014897 | 0.011148 | 0.730897 | 49.062083 | 0.010920 | 3.660690 | 0.994790 |
| 1568 | (REGENCY TEA PLATE ROSES) | (REGENCY TEA PLATE PINK) | 0.021079 | 0.014289 | 0.012617 | 0.598558 | 41.888426 | 0.012316 | 2.455423 | 0.997146 |
| 1569 | (REGENCY TEA PLATE PINK) | (REGENCY TEA PLATE ROSES) | 0.014289 | 0.021079 | 0.012617 | 0.882979 | 41.888426 | 0.012316 | 8.365322 | 0.990277 |
| 1616 | (SET OF 3 WOODEN TREE DECORATIONS) | (SET OF 3 WOODEN STOCKING DECORATION) | 0.014492 | 0.015759 | 0.010996 | 0.758741 | 48.147134 | 0.010767 | 4.079608 | 0.993630 |
| 1617 | (SET OF 3 WOODEN STOCKING DECORATION) | (SET OF 3 WOODEN TREE DECORATIONS) | 0.015759 | 0.014492 | 0.010996 | 0.697749 | 48.147134 | 0.010767 | 3.260564 | 0.994909 |
| 3024 | (POPPY'S PLAYHOUSE LIVINGROOM, POPPY'S PLAYHOU... | (POPPY'S PLAYHOUSE BEDROOM) | 0.012921 | 0.020927 | 0.011046 | 0.854902 | 40.851066 | 0.010776 | 6.747663 | 0.988291 |
| 3025 | (POPPY'S PLAYHOUSE LIVINGROOM, POPPY'S PLAYHOU... | (POPPY'S PLAYHOUSE KITCHEN) | 0.012820 | 0.021535 | 0.011046 | 0.861660 | 40.011439 | 0.010770 | 7.072902 | 0.987669 |
| 3026 | (POPPY'S PLAYHOUSE KITCHEN, POPPY'S PLAYHOUSE ... | (POPPY'S PLAYHOUSE LIVINGROOM) | 0.015404 | 0.016215 | 0.011046 | 0.717105 | 44.225226 | 0.010797 | 3.477566 | 0.992680 |
| 3027 | (POPPY'S PLAYHOUSE LIVINGROOM) | (POPPY'S PLAYHOUSE KITCHEN, POPPY'S PLAYHOUSE ... | 0.016215 | 0.015404 | 0.011046 | 0.681250 | 44.225226 | 0.010797 | 3.088928 | 0.993498 |

**Conclusion:**

This program outlines the essential steps for conducting Market Basket Analysis (MBA) using the Apriori algorithm. Market Basket Analysis is a valuable data mining technique that helps businesses uncover hidden patterns and relationships within transactional data, enabling them to make data-driven decisions and enhance various aspects of their operations.

Here's a summary of the key steps covered in this program:

1. **Data Hygiene**: The program begins by cleaning and preparing the transactional data. It involves removing rows with missing item names, filtering out rows with zero or negative quantities, filling missing customer IDs with a placeholder, and creating a new column to calculate total sales revenue.
2. **Exploratory Data Analysis (EDA):** The EDA section explores the dataset to gain insights into customer behavior. It identifies the best-selling items in different countries, visualizes total sales by country, and examines the most popular products in the United Kingdom.
3. **Market Basket Analysis:** The main focus of the program is on performing Market Basket Analysis using the Apriori algorithm and association rule mining. The steps include:

* Converting the dataset into a transactional format where each row represents a bill with the items purchased.
* Creating a one-hot matrix of products, indicating whether each item was purchased in a bill.
* Merging the transactional matrix and the one-hot matrix to prepare the data for analysis.
* Applying the Apriori algorithm to find frequent itemsets with a specified minimum support threshold.
* Generating association rules based on these frequent itemsets, considering lift as a metric.

The generated association rules provide insights into item associations and can be used to make strategic decisions in areas such as product placement, marketing campaigns, and customer experience improvement.

In practice, businesses can leverage these insights to optimize their operations, enhance sales strategies, and ultimately provide a better shopping experience for customers. Additionally, the program demonstrates how to use Python and popular libraries like pandas and mlxtend to perform these analyses efficiently.

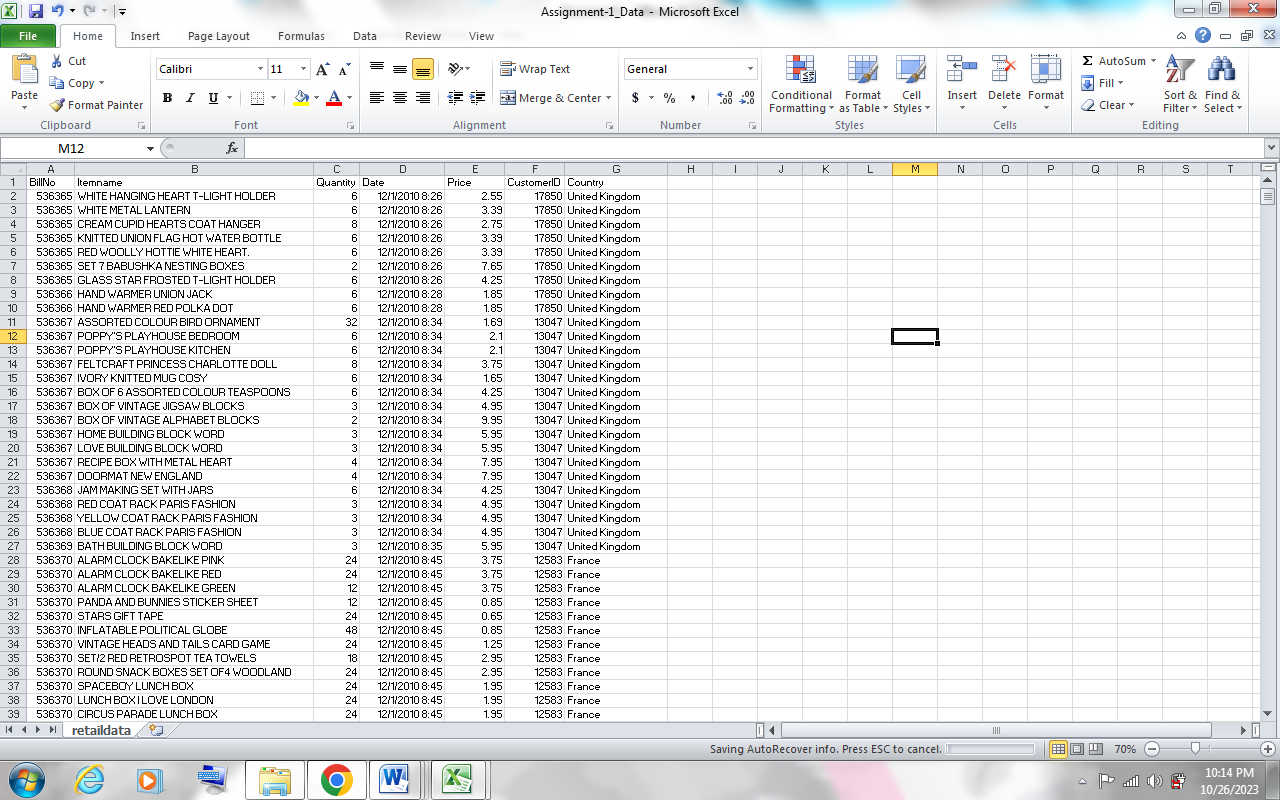
Market Basket Analysis is a powerful tool for retailers and businesses seeking to unlock the hidden potential within their transactional data, enabling them to make informed decisions and drive profitability.

***Development Part 2***

**Topic:** *Start building the Market Basket Analysis*



## Given data set:



The "Assignment-1\_Data" dataset is kept in an Excel file ending in ".xlsx." It includes retail data-related information. Based on the data supplied, here is a description of the dataset:

* File Name: The dataset is contained in an Excel file called "Assignment-1\_Data". All of the retail data is kept in this file.
* List Name: It looks that the dataset in the Excel file is referred to as "retaildata". This could be a label or identifier inside the Excel document that is used to refer to this particular dataset.
* File Format: The ".xlsx" file extension indicates that the dataset is in Excel format. A well-liked spreadsheet tool for handling and storing tabular data is Excel.
* Number of Rows: 522,065 rows make up the dataset's total number of rows. It is presumably the case that each row corresponds to a distinct retail data entry or observation. Information about sales, merchandise, clients, and transactions, among other aspects of retail business, may be included in these entries.
* Number of attributes: This dataset contains seven attributes. A dataset's attributes, sometimes referred to as columns or fields, are the many kinds of data that are gathered for every entry. Specifics like the product ID, the sale date, the price, the quantity, the customer's information, and more could be included in these properties.
* Previously we started building the Market Basket Analysis by loading and pre-processing the dataset
* In this segment we’re going to build, train and evaluate the model.

## Model Building:

Market Basket Analysis (MBA): To find patterns of item co-occurrence in transaction data, Market Basket Analysis is a data mining technique used in retail and online commerce. The objective is to comprehend the connections between products that consumers usually buy in tandem. A number of uses for this data exist, including better inventory control, focused marketing campaigns, and product placement optimisation.

Apriori Algorithm: One popular method for carrying out market basket analysis is the Apriori algorithm. The idea of association rule mining serves as its foundation. The relationships between objects in a dataset are described by association rules, which are if-then expressions. When applied to MBA contexts, these rules usually take the form of "If item A is purchased, then item B is likely to be purchased."

The following are the main ideas underlying the Apriori algorithm:

* Support: The frequency with which an itemset (a group of items) appears in the dataset is the measure of support. It can be calculated by dividing the total number of transactions by the number of transactions that contain the itemset. A high level of support means that people usually purchase the itemset together.
* Confidence: Confidence quantifies the frequency with which an association rule's if-then connection holds true. It is computed by dividing the support of the antecedent (if) item by the support of both items in the rule. A high degree of confidence indicates a robust correlation between the items.
* Lift: The probability that item B will be purchased in conjunction with item A as opposed to being purchased separately is measured by lift. A positive correlation, denoting a higher likelihood of purchasing the two things together, is shown by a lift value larger than 1.

The following is how the Apriori algorithm functions:

* Create Frequent Itemsets: The first step involves locating item combinations that fall under a minimum support threshold, or frequent itemsets. These frequently occurring itemsets are seen to be excellent choices for association rules.
* Create Association Rules: After identifying the frequently occurring item sets, the algorithm creates association rules by taking various item combinations into account within these sets. For every rule, it computes the lift and confidence.
* Pruning: To get rid of candidate itemsets that don't satisfy the minimal support criteria, the algorithm uses a "pruning" step.
* Repeat: Up until no more frequently occurring itemsets can be discovered, the procedure is repeated repeatedly by enlarging the itemsets.

**Model Training**

1. Data Gathering:

* Assemble transactional data: A dataset with transaction records is what you require, with each row denoting a transaction and each column representing an item that was bought during that transaction.]
* Prepare the information: Make that the format of the data is appropriate. Every transaction should typically be shown as a list or collection of elements.

2.Apriori Algorithm

One popular method for carrying out market basket analysis is the Apriori algorithm. The idea of association rule mining serves as its foundation. The relationships between objects in a dataset are described by association rules, which are if-then expressions. When applied to MBA contexts, these rules usually take the form of "If item A is purchased, then item B is likely to be purchased."

The following are the main ideas underlying the Apriori algorithm:

* Support: The frequency with which an itemset (a group of items) appears in the dataset is the measure of support. It can be calculated by dividing the total number of transactions by the number of transactions that contain the itemset. A high level of support means that people usually purchase the itemset together.
* Confidence: Confidence quantifies the frequency with which an association rule's if-then connection holds true. It is computed by dividing the support of the antecedent (if) item by the support of both items in the rule. A high degree of confidence indicates a robust correlation between the items.
* Lift: The probability that item B will be purchased in conjunction with item A as opposed to being purchased separately is measured by lift. A positive correlation, denoting a higher likelihood of purchasing the two things together, is shown by a lift value larger than 1.

3. Training Models:

Establish a minimal support threshold, which is the bare minimum of support required for an itemset to qualify as frequent. Depending on your dataset and desired level of significance, adjust this threshold.

Create often occurring itemsets by utilising the Apriori method to identify itemsets that satisfy the minimal support requirement. Itemsets that appear frequently in transactions are known as frequent itemsets.

4. Formulating Rules

Following the identification of frequently occurring itemsets, association rules based on these itemsets can be created. These guidelines illustrate the relationships between the various elements.

5. Rule Assessment

Assess the created rules using metrics like lift, support, and confidence. Sort the regulations according to your own needs.

6. Explanation

Examine the produced rules to learn more about item correlations. For instance, you might find that buyers of "Item A" are likewise inclined to purchase "Item B."

7. Reporting and Visualisation

To aid in decision-making, visualise the data using lift charts or support-confidence plots.

8. Put into Practise

Utilise the analysis's insights to improve company tactics including targeted marketing, cross-selling, and product positioning.

**Model Evaluation**

Evaluation Metrics for the Model:

* Support: The frequency with which an itemset (a group of items) appears in the dataset is the measure of support. It can be calculated by dividing the total number of transactions by the number of transactions that contain the itemset. A high level of support means that people usually purchase the itemset together.
* Confidence: Confidence quantifies the frequency with which an association rule's if-then connection holds true. It is computed by dividing the support of the antecedent (if) item by the support of both items in the rule. A high degree of confidence indicates a robust correlation between the items.
* Lift: The probability that item B will be purchased in conjunction with item A as opposed to being purchased separately is measured by lift. A positive correlation, denoting a higher likelihood of purchasing the two things together, is shown by a lift value larger than 1.

Qualitative Assessment:

1. Relevance for Business:

Relevance to Business Goals: Determine whether the guidelines you've found are in line with your company's goals. Do they enhance consumer satisfaction, boost revenue, or improve product placement?

Actionability: Ascertain whether the guidelines can be applied to real-world situations. When it comes to formulating suggestions, product bundles, or marketing initiatives, are the guidelines helpful?

2. Comprehensibility:

Rule Complexity: Take a look at how straightforward the rules are. It could be difficult to execute or explain complex regulations to the team. Prefer guidelines that are simple to comprehend.

Common Sense: Determine whether the relationships found make sense. Do the regulations seem reasonable or do they represent typical consumer behaviour?

3. Domain Expertise:

Subject Matter Expertise: Use domain expertise to evaluate how useful the rules are. Are there any particular patterns in your industry or business specialty that the rules should be capturing?

Contextual Understanding: Recognise the environment in which the regulations are applicable. For grocery stores, internet retailers, and other business kinds, there could be different regulations that apply.

4. Assistance from relevant parties:

Involve Stakeholders: Talk about the established guidelines with pertinent parties, like product managers, sales teams, and marketing teams. Ask them for their opinions on the rules' possible usefulness.

User Input: Gather input from clients or end users who have engaged with suggestions made in accordance with these guidelines. Their opinions can offer insightful information about how well the regulations work.

5. Real-World Application:

Technical Feasibility: Determine whether putting the rules into effect is technically feasible. Are the infrastructure and data needed for execution available?

Analyse the costs and benefits of implementing the guidelines, taking into account any potential gains in terms of higher revenue, happier customers, or other KPIs.

6. Trial and error and A/B testing:

Real-World Testing: Do A/B testing and put some of the guidelines into practise in a safe setting. Analyse how the regulations affect revenue, consumer behaviour, or other pertinent data.

7. Moral Lessons to Recall

Ethical Assessment: Confirm that the rules won't have unanticipated consequences such as discriminatory targeting or invasions of privacy. Consider the ethical implications of adhering to the rules.

8. Feedback Loop:

Continuous Improvement: Preserve the MBA process's vitality. Regularly assess the effectiveness of the regulations and make any necessary modifications in light of new knowledge and feedback.

**Program:**

import numpy as np  
import pandas as pd  
import os  
from matplotlib import pyplot as plt  
from mlxtend.frequent\_patterns import apriori  
from mlxtend.frequent\_patterns import association\_rules  
  
# Load the data from an Excel file  
df = pd.read\_excel("Assignment-1\_Data.xlsx")  
  
# Data Preprocessing  
df = df.dropna(subset=["Itemname"])  
df = df[df["Quantity"] > 0]  
df['CustomerID'].fillna(99999, inplace=True)  
df["SumPrice"] = df["Quantity"] \* df["Price"]  
  
# Total sales by Country  
total\_sales\_country = df.groupby(['Country']).agg({'SumPrice': 'sum'}).sort\_values('SumPrice', ascending=False).reset\_index()  
plt.bar(total\_sales\_country["Country"], total\_sales\_country["SumPrice"])  
plt.yscale('log')  
plt.ylabel('Quantity')  
plt.xticks(rotation=90)  
plt.show()  
  
# Filter data for the United Kingdom  
only\_uk = df[df["Country"] == "United Kingdom"]  
  
# Best selling items in the UK  
best\_selling\_uk\_items = only\_uk.groupby("Itemname")["Quantity"].sum().sort\_values(ascending=False)  
  
# Total sales by Item  
total\_sales\_item = df.groupby(['Itemname']).agg({'Price': 'mean', 'Quantity': 'sum', 'SumPrice': 'sum'}).reset\_index()  
total\_sales\_item['Count'] = df.groupby(['Itemname']).size().values  
total\_sales\_item = total\_sales\_item.sort\_values("SumPrice", ascending=False)  
  
# Transaction data preparation  
transactions = df.groupby(['BillNo'])['Itemname'].apply(list)  
one\_hot = pd.get\_dummies(df['Itemname'])  
one\_hot['BillNo'] = df['BillNo']  
one\_hot = one\_hot.groupby('BillNo').sum()  
  
# Create a binary transaction matrix  
transaction\_matrix = pd.merge(transactions, one\_hot, on='BillNo')  
transaction\_matrix[one\_hot.columns[:-1]] = (transaction\_matrix[one\_hot.columns[:-1]] >= 1).astype(int)  
  
# Frequent itemset mining  
frequent\_itemsets = apriori(transaction\_matrix.iloc[:, 1:], min\_support=0.01, use\_colnames=True)  
  
# Association rule generation  
rules = association\_rules(frequent\_itemsets, metric="lift", min\_threshold=1)  
  
# Scatter plot of association rules  
fig, ax = plt.subplots()  
scatter = ax.scatter(rules['lift'], rules['confidence'], alpha=0.5)  
tooltips = []  
for i in range(len(rules)):  
    rule = rules.iloc[i]  
    tooltip = f"Rule: {rule['antecedents']} -> {rule['consequents']}\nSupport: {rule['support']:.3f}\nConfidence: {rule['confidence']:.3f}\nLift: {rule['lift']:.3f}"  
    tooltips.append(tooltip)  
mpld3.plugins.connect(fig, mpld3.plugins.PointHTMLTooltip(scatter, tooltips))  
ax.set\_xlabel("Lift")  
ax.set\_ylabel("Confidence")  
ax.set\_title("Association Rules Scatter Plot")  
plt.show()  
  
# Filter rules based on lift values  
filtered\_rules = rules[(rules['lift'] > 40) & (rules['lift'] < 50)]

* This code outlines the entire process, from data preprocessing to model training. Please ensure you have the necessary libraries installed.

## 

## Conclusion:

* A Python-based Market Basket Insights application provides a data-driven method for analysing customer behaviour and enhancing corporate tactics. This programme makes it possible for businesses to make well-informed decisions about product placement, marketing campaigns, and cross-selling opportunities by revealing important information about which goods are frequently purchased together through data preparation and analysis. Businesses can obtain actionable insights that help them customise their strategies for certain markets and improve their overall operational efficiency by calculating total sales by item and country and visualising trends in a scatter plot.
* The main components of this analysis are the binary transaction matrix that is created and the association rules that are derived by using the Apriori algorithm to determine the probability of adding one product to the basket when another is present. By quantifying the strength of these interactions using metrics like lift evaluation, these rules assist firms in prioritising which product pairings to concentrate on. In conclusion, by carefully matching their products with customer preferences and buying habits, Market Basket Insights programmes in Python enable companies to use transaction data for decision-making, enhancing customer experiences and spurring revenue growth.