

## Phase-4

### Market Basket Analysis:

Date	25 Oct 2023
Team ID	Proj-212168-Team-1
Project Name	Market Basket Insights
Maximum Mark	

### Algorithm:

#### Apriori Algorithm:

The Apriori algorithm is a popular data mining technique used for frequent itemset mining and association rule learning. It's primarily used in market basket analysis, which helps identify relationships between items in large datasets. Here's a simplified explanation of how it works:

1. **Support:** The algorithm begins by calculating the support for each item in the dataset. Support is the proportion of transactions that contain a particular item or itemset.
2. **Frequent Itemsets:** Items with support above a specified threshold are considered frequent

itemsets. These are the items that occur frequently enough to be of interest.

3. **Candidate Generation:** The algorithm generates candidate itemsets of larger sizes from the frequent itemsets found in the previous step. It does this by joining pairs of frequent itemsets to create potentially larger itemsets.

4. **Pruning:** Candidate itemsets that do not meet the minimum support threshold are pruned, as they cannot be frequent.

5. **Repeat:** Steps 3 and 4 are repeated iteratively to generate larger and larger frequent itemsets until no more can be found.

6. **Association Rules:** Once the frequent itemsets are identified, association rules are generated. These rules describe relationships between items. For example, "if customers buy A and B, they are likely to buy C."

7. **Confidence:** The strength of association rules is measured by confidence, which is the proportion of transactions containing the antecedent (the items on the left side of the rule) that also contain the consequent (the items on the right side).

8. **Rule Pruning:** Rules with confidence below specified threshold are pruned.

The Apriori algorithm is widely used in retail for market basket analysis and in various fields for discovering associations between items in datasets. It's effective for identifying patterns and making data-driven decisions.

## Data Train & Test:

### #Import package:

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

### Explanation:

- Numpy :(import numpy as np) a library for mathematical operations and handling arrays.
- pandas :(import pandas as pd) a library for data manipulation and analysis.
- mlxtend.frequent\_patterns: a module for performing frequent itemset mining and association rule learning

### #Load Dataset

```
dataset=pd.read_csv('insights (1).csv',encoding="UTF-8")
```

/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `and should\_run\_async(code)`  
<ipython-input-2-7e527f06660a>:1: DtypeWarning: Columns (0) have mixed types. Specify dtype option on import or set low\_resolution=False or high\_resolution=True  
dataset=pd.read\_csv('insights (1).csv',encoding="UTF-8")

This code reads contents of a csv file called "insights.csv" and saves it a variable called "dataset".The "pd" modul is already imported.

## #Data Preprocessing:

```
df=dataset.fillna({'Itemname':'abc'})
df
df1=dataset.fillna(value=dataset['CustomerID'].mean())
df1
```

/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `and should\_run\_async(code)

## Output:

/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should\_run\_async` will not call `tr` and should\_run\_async(code)

	BillNo	Itemname	Quantity	Date	Price	CustomerID	Country
0	536365	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850.0	United Kingdom
1	536365	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
2	536365	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850.0	United Kingdom
3	536365	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
4	536365	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
...	...	...	...	...	...	...	...
522059	581587	PACK OF 20 SPACEBOY NAPKINS	12	12/9/2011 12:50	0.85	12680.0	France
522060	581587	CHILDREN'S APRON DOLLY GIRL	6	12/9/2011 12:50	2.10	12680.0	France
522061	581587	CHILDRENS CUTLERY DOLLY GIRL	4	12/9/2011 12:50	4.15	12680.0	France
522062	581587	CHILDRENS CUTLERY CIRCUS PARADE	4	12/9/2011 12:50	4.15	12680.0	France
522063	581587	BAKING SET 9 PIECE RETROSPOT	3	12/9/2011 12:50	4.95	12680.0	France

522064 rows x 7 columns

✓ 0s completed at 11:28 AM

- `df = dataset.fillna({'Itemname': 'abc'})`:

- - This line fills missing values in the 'Itemname' column of the `dataset` DataFrame with the string 'abc'. It replaces any NaN values in the 'Itemname' column with 'abc'.
- - The resulting DataFrame is assigned to the variable `df`.
- `df1 = dataset.fillna(value=dataset['CustomerID'].mean())`:
- - This line fills missing values in the entire `dataset` DataFrame with the mean of the 'CustomerID' column.
- - `dataset['CustomerID'].mean()` calculates the mean of the 'CustomerID' column.
- - The resulting DataFrame is assigned to the variable `df1`.

### #Replace missing value& count the missing value:

```

1s df1.isnull().sum()
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `show` and `should_run_async` will not call `ipython.show` by default.
BillNo      0
Itemname     0
Quantity     0
Date         0
Price        0
CustomerID   0
Country      0
dtype: int64

```

The code `df1.isnull().sum()` is used to count the number of missing (NaN) values in each column of the DataFrame `df1`.

### #Group Columns:

```

0s [5] basket = (dataset[dataset['Country'] == 'Germany' ].groupby(['BillNo', 'Itemname'])['Quantity'].sum().unstack().fillna(
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `ipython.show` by default.
and should_run_async(code)

[6] basket

```

## Output:

```
[6] /usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell` automatically in the future. Please pass the
and should_run_async(code)
```

Itemname	10 COLOUR SPACEBOY PEN	12 COLOURED PARTY BALLOONS	12 ROSE PEG PLACE SETTINGS	12 IVORY MESSAGE CARDS WITH ENVELOPES	12 PENCIL SMALL TUBE WOODLAND	12 PENCILS SMALL TUBE RETROSPOT	12 PENCILS SMALL TUBE SKULL	12 PENCILS TALL TUBE POSY	12 PENCILS TALL TUBE RETROSPOT	12 PENCILS TALL TUBE SKULLS	...	YULETIDE IMAGES GIFT WRAP SET	ZINC HEART T- LIGHT HOLDER	ZINC STAR T- LIGHT HOLDER	ZINC BOX SIGN HOME	ZINC FOLKART SLEIGH BELLS	ZINC HEART LATTICE T-LIGHT HOLDER	ZINC METAL HEART DECORATION	ZINC T- LIGHT HOLDER STAR LARGE
BillNo																			
536527	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
536840	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
536861	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
536967	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
536983	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
571729	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
571739	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
571824	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
571904	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
572054	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

457 rows × 1695 columns

The code you provided appears to be creating a dataset `basket` based on filtering and aggregating data from another dataset. Here's a step-by-step explanation:

- `(dataset[dataset['Country'] == 'Germany'])`:`
- - This part filters the original `dataset` to select only the rows where the 'Country' column has the value 'Germany'. It retrieves all the data related to transactions in Germany.
- `.groupby(['BillNo', 'Itemname'])['Quantity'].sum()`:`
- - After filtering, the data is grouped by the 'BillNo' and 'Itemname' columns.
- - The 'Quantity' column is selected for aggregation.
- - `.sum()` is used to calculate the sum of quantities for each unique combination of 'BillNo' and 'Itemname'. This effectively aggregates the total quantity of each item per bill.`
- `.unstack()`:`

- - This step is used to unstack the grouped data. It transforms the data from a hierarchical structure into a more tabular format. It essentially pivots the data to make 'Itemname' values the columns.
- `.fillna(0)`:
- - This step fills in any missing values in the pivoted dataset with zeros. It's common to fill missing values with zeros when dealing with transaction data, assuming that a missing quantity means no items of that type were purchased on a particular bill.

## # Encoding:

```
def encode(x):  
    if x <= 0:  
        return 0  
    if x >= 1:  
        return 1  
basket = basket.applymap(encode)  
basket
```

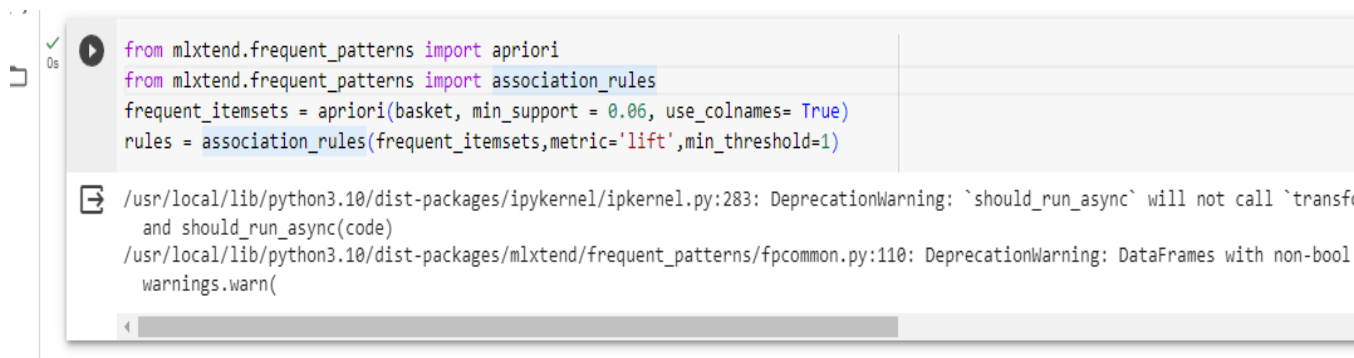
## Output:

[illegible]

- The provided code defines a function `encode(x)` and then applies this function to the `basket` DataFrame. Here's a step-by-step explanation of what this code does:
- **1. `def encode(x)`:**

- - This is the definition of a custom function called ``encode`` that takes a single argument ``x``.
- **2. ``if x <= 0:``**
- - This is an ``if`` statement inside the ``encode`` function. It checks if the value of ``x`` is less than or equal to 0.
- **3. ``return 0``:**
- - If the condition in the first ``if`` statement is met (i.e., if ``x`` is less than or equal to 0), the function returns 0.
- **4. ``if x >= 1:``**
- - This is another ``if`` statement inside the ``encode`` function. It checks if the value of ``x`` is greater than or equal to 1.
- **5. ``return 1``:**
- - If the condition in the second ``if`` statement is met (i.e., if ``x`` is greater than or equal to 1), the function returns 1.
- **6. ``basket = basket.applymap(encode)``:**
  - This line applies the ``encode`` function to every element in the ``basket`` DataFrame using the ``applymap`` method. It essentially replaces the values in the DataFrame with the results of applying the ``encode`` function to each value.

### #Train&Test Using Apriori Algorithm:



```
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
frequent_itemsets = apriori(basket, min_support = 0.06, use_colnames= True)
rules = association_rules(frequent_itemsets, metric='lift', min_threshold=1)
```

/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should\_run\_async` will not call `transform` and `should\_run\_async(code)`  
/usr/local/lib/python3.10/dist-packages/mlxtend/frequent\_patterns/fpcommon.py:110: DeprecationWarning: DataFrames with non-boolean values are deprecated. Please use `warnings.warn` to suppress this warning.

The provided code is using the ``mlxtend`` library to perform Association



Rule Mining using the Apriori algorithm. Here's an explanation of each part of the code:

- 1. **``from mlxtend.frequent_patterns import apriori` and `from mlxtend.frequent_patterns import association_rules`:`**
  - - These lines import the necessary functions from the ``mlxtend.frequent_patterns`` module, which is part of the MLxtend library. MLxtend is a Python library for data analysis and preprocessing.
- 2. **``frequent_itemsets = apriori(basket, min_support=0.06, use_colnames=True)``:**
  - - ``apriori`` is used to find frequent itemsets in the ``basket`` DataFrame.
  - - ``basket`` is the dataset you want to analyze for frequent itemsets, which typically represents transaction data with items.
  - - ``min_support`` is set to 0.06, which is the minimum support threshold. It specifies the minimum fraction of transactions that must contain a particular itemset for it to be considered "frequent."
  - - ``use_colnames=True`` is used to indicate that you want to use the column names from the DataFrame as item names in the results.
- 3. **``rules = association_rules(frequent_itemsets, metric='lift', min_threshold=1)``:**
  - - ``association_rules`` is used to generate association rules from the frequent itemsets found by the Apriori algorithm.
  - - ``frequent_itemsets`` is the result of the Apriori algorithm, containing frequent itemsets with their support values.
  - - ``metric='lift`` specifies that you want to evaluate the generated association rules based on the lift metric, which measures the strength of association between itemsets.

- - `min_threshold=1` sets a minimum threshold for the lift metric. This means that only rules with a lift value greater than or equal to 1 will be considered. It's a way to filter out less significant rules.

## #Print First Five rules:

- 

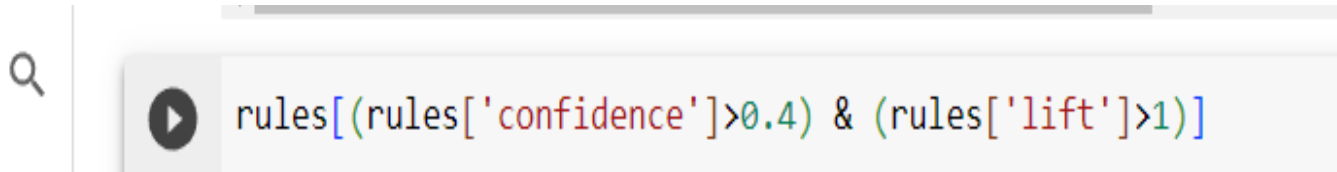
✓ 0s [33] `rules.head()`

/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should\_run\_async` will not call `gather` to force a sync point  
and should\_run\_async(code)

	antecedents	consequents	antecedent support	consequent support	support	confidence
0	(POSTAGE)	(6 RIBBONS RUSTIC CHARM)	0.818381	0.102845	0.091904	0.11229
1	(6 RIBBONS RUSTIC CHARM)	(POSTAGE)	0.102845	0.818381	0.091904	0.89361
2	(CHARLOTTE BAG APPLES DESIGN)	(POSTAGE)	0.065646	0.818381	0.061269	0.93333
3	(POSTAGE)	(CHARLOTTE BAG APPLES DESIGN)	0.818381	0.065646	0.061269	0.07486
4	(JUMBO BAG WOODLAND ANIMALS)	(POSTAGE)	0.100656	0.818381	0.087527	0.86956

The `rules.head()` code is used to display the first few rows of the DataFrame containing the association rules generated using the Apriori algorithm.

## #FliterAssociationrule:



## Output:

9	(PLASTERS IN TIN CIRCUS PARADE)	(PLASTERS IN TIN WOODLAND ANIMALS)	0.115974	0.137036	0.067034	0.504900	4.242001	0.031040	2.010904	0.004300
10	(PLASTERS IN TIN CIRCUS PARADE)	(POSTAGE)	0.115974	0.818381	0.100656	0.867925	1.060539	0.005746	1.375117	0.064572
12	(PLASTERS IN TIN WOODLAND ANIMALS)	(PLASTERS IN TIN SPACEBOY)	0.137856	0.107221	0.061269	0.444444	4.145125	0.046488	1.607002	0.880076
13	(PLASTERS IN TIN SPACEBOY)	(PLASTERS IN TIN WOODLAND ANIMALS)	0.107221	0.137856	0.061269	0.571429	4.145125	0.046488	2.011670	0.849877
14	(PLASTERS IN TIN SPACEBOY)	(POSTAGE)	0.107221	0.818381	0.100656	0.938776	1.147113	0.012909	2.966448	0.143649
17	(PLASTERS IN TIN STRONGMAN)	(POSTAGE)	0.070022	0.818381	0.067834	0.968750	1.183740	0.010529	5.811816	0.166907
18	(PLASTERS IN TIN WOODLAND ANIMALS)	(POSTAGE)	0.137856	0.818381	0.118162	0.857143	1.047364	0.005344	1.271335	0.052453
20	(PLASTERS IN TIN WOODLAND ANIMALS)	(ROUND SNACK BOXES SET OF 4 WOODLAND)	0.137856	0.245077	0.074398	0.539683	2.202098	0.040613	1.640006	0.633174
23	(RED RETROSPOT CHARLOTTE BAG)	(POSTAGE)	0.070022	0.818381	0.065646	0.937500	1.145555	0.008341	2.905908	0.136627
24	(RED TOADSTOOL LED NIGHT LIGHT)	(POSTAGE)	0.096280	0.818381	0.080963	0.840909	1.027528	0.002169	1.141607	0.029645
27	(REGENCY CAKESTAND 3 TIER)	(POSTAGE)	0.137856	0.818381	0.120350	0.873016	1.066760	0.007532	1.430252	0.072589
29	(ROUND SNACK BOXES SET OF 4 FRUITS)	(POSTAGE)	0.157549	0.818381	0.150985	0.958333	1.171012	0.022049	4.358862	0.173348
31	(ROUND SNACK BOXES SET OF 4 WOODLAND)	(POSTAGE)	0.245077	0.818381	0.225383	0.919643	1.123735	0.024817	2.260151	0.145856
33	(SPACEBOY LUNCH BOX)	(POSTAGE)	0.102845	0.818381	0.091904	0.893617	1.091933	0.007738	1.707221	0.093844
34	(STRAWBERRY LUNCH BOX WITH CUTLERY)	(POSTAGE)	0.078775	0.818381	0.067834	0.861111	1.052213	0.003366	1.307659	0.053866
36	(WOODLAND CHARLOTTE BAG)	(POSTAGE)	0.126915	0.818381	0.115974	0.913793	1.116587	0.012109	2.106783	0.119591
39	(WOODLAND PARTY BAG + STICKER SET)	(POSTAGE)	0.067834	0.818381	0.061269	0.903226	1.103674	0.005755	1.876732	0.100771
40	(ROUND SNACK BOXES SET OF 4 WOODLAND)	(ROUND SNACK BOXES SET OF 4 FRUITS)	0.245077	0.157549	0.131291	0.535714	3.400298	0.092679	1.814509	0.935072
41	(ROUND SNACK BOXES SET OF 4 FRUITS)	(ROUND SNACK BOXES SET OF 4 WOODLAND)	0.157549	0.245077	0.131291	0.833333	3.400298	0.092679	4.529540	0.837922
43	(SPACEBOY LUNCH BOX)	(ROUND SNACK BOXES SET OF 4 WOODLAND)	0.102845	0.245077	0.070022	0.680851	2.778116	0.044817	2.365427	0.713415
44	(WOODLAND CHARLOTTE BAG)	(ROUND SNACK BOXES SET OF 4 WOODLAND)	0.126915	0.245077	0.063457	0.500000	2.040179	0.032354	1.508847	0.583960

The code `rules[(rules['confidence'] > 0.4) & (rules['lift'] > 1)]` is used to filter association rules in a DataFrame based on two conditions:

1. `rules['confidence'] > 0.4`: This condition filters for rules with a confidence greater than 0.4. Confidence measures how likely the consequent item is given the antecedent item and is an indicator of the strength of the rule.
2. `rules['lift'] > 1`: This condition filters for rules with a lift greater than 1. Lift measures the strength of association between antecedent and consequent items. A lift greater than 1 indicates a positive association between items.