MARKET BASKET INSIGHTS

TEAM MEMBER

963521106027: DHANEESH.A

PHASE 3

PROJECT DEVELOPEMENT PART -1

TITLE: Market Basket Insights

Abstract:

Market basket insights (MBIs) is a data mining technique that identifies patterns and relationships between items purchased together in transactions. MBIs can be used to improve a variety of business processes, such as product placement, cross-selling, and targeted marketing.

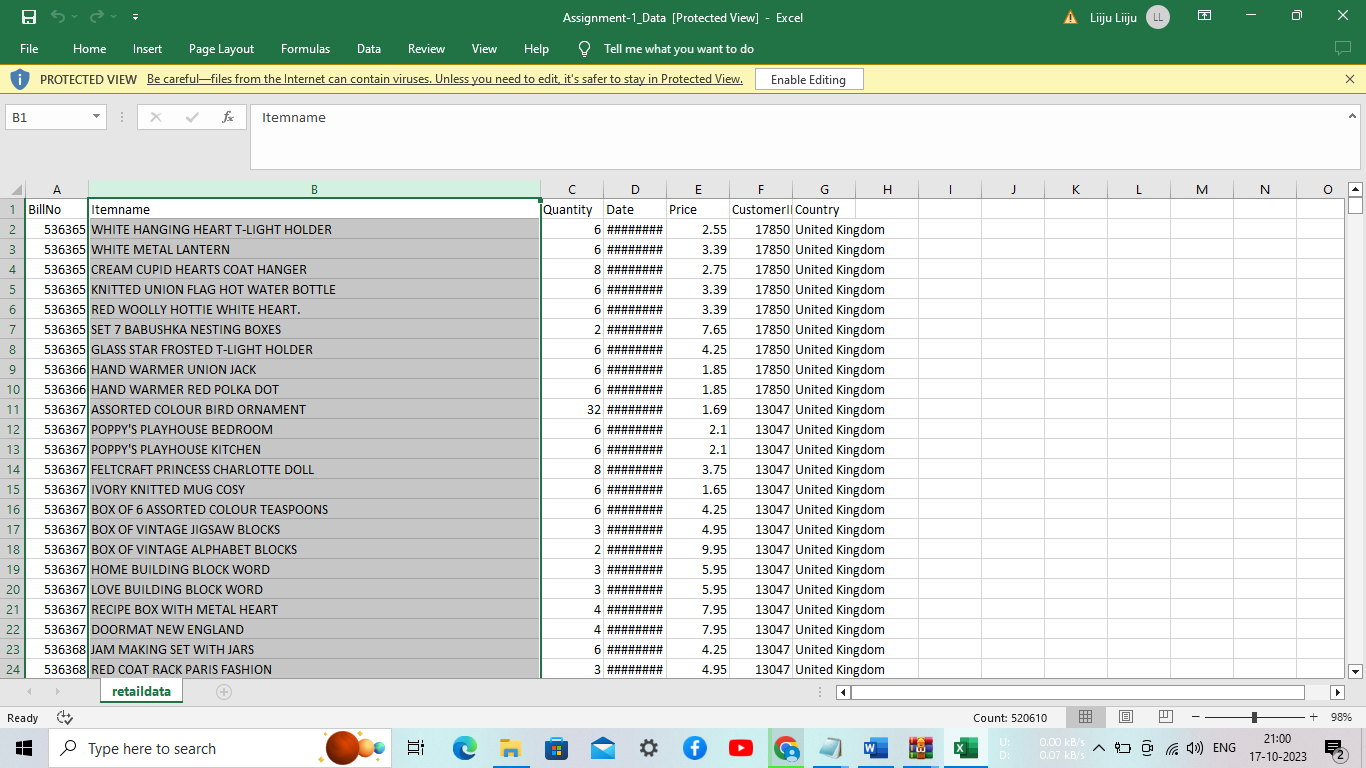


**Problem definition:**

Here we Start the market basket insights project by loading and preprocessing the transaction data. Load the transaction dataset and preprocess the data for association analysis.

**Dataset:**

**Dataset Link:**[**https://www.kaggle.com/datasets/aslanahmedov/market-basket-analysis**](https://www.kaggle.com/datasets/aslanahmedov/market-basket-analysis)



**Loading dataset:**

Loading the dataset in market basket insights is the first step in conducting an analysis. The dataset can be loaded from a variety of sources, such as a relational database, CSV file, or XML file.

Here are some steps for loading the dataset in market basket insights:

1. Choose a data mining tool. There are a variety of data mining tools available that support market basket analysis. Some popular options include Python, R, and SAS.
2. Load the dataset into the data mining tool. The specific steps for loading the dataset will vary depending on the data mining tool that you are using. However, most data mining tools will have a built-in function for loading datasets from different sources.
3. Inspect the dataset. Once the dataset has been loaded, it is important to inspect it to identify any errors or inconsistencies. This can be done by looking at the data types of the variables, the number of transactions, and the distribution of the values.
4. Clean and preprocess the dataset. Once any errors or inconsistencies have been identified, the dataset should be cleaned and preprocessed. This may involve removing duplicate transactions, handling missing values, and converting categorical variables to numerical variables.
5. **1. | Loading and Cleaning data**
6. **1-1. | Loading data**
7. Out[2]:

|  | BillNo | Itemname | Quantity | Date | Price | CustomerID | Country |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 536365 | WHITE HANGING HEART T-LIGHT HOLDER | 6 | 01.12.2010 08:26 | 2,55 | 17850.0 | United Kingdom |
| 1 | 536365 | WHITE METAL LANTERN | 6 | 01.12.2010 08:26 | 3,39 | 17850.0 | United Kingdom |
| 2 | 536365 | CREAM CUPID HEARTS COAT HANGER | 8 | 01.12.2010 08:26 | 2,75 | 17850.0 | United Kingdom |
| 3 | 536365 | KNITTED UNION FLAG HOT WATER BOTTLE | 6 | 01.12.2010 08:26 | 3,39 | 17850.0 | United Kingdom |
| 4 | 536365 | RED WOOLLY HOTTIE WHITE HEART. | 6 | 01.12.2010 08:26 | 3,39 | 17850.0 | United Kingdom |

1. **<class 'pandas.core.frame.DataFrame'>**
2. **RangeIndex: 522064 entries, 0 to 522063**
3. **Data columns (total 7 columns):**
4. **# Column Non-Null Count Dtype**
5. **--- ------ -------------- -----**
6. **0 BillNo 522064 non-null object**
7. **1 Itemname 520609 non-null object**
8. **2 Quantity 522064 non-null int64**
9. **3 Date 522064 non-null object**
10. **4 Price 522064 non-null object**
11. **5 CustomerID 388023 non-null float64**
12. **6 Country 522064 non-null object**
13. **dtypes: float64(1), int64(1), object(5)**
14. **memory usage: 27.9+ MB**
15. Out[4]:
16. **BillNo 0**
17. **Itemname 1455**
18. **Quantity 0**
19. **Date 0**
20. **Price 0**
21. **CustomerID 134041**
22. **Country 0**
23. **dtype: int64**
24. **1-2. | Dropping data with negative or zero quantity**
25. In [6]:
26. **df**=**df**.**loc[df['Quantity']**>**0]**
27. **1-3. | Dropping data with zero price**
28. In [8]:
29. **df**=**df**.**loc[df['Price']**>**'0']**
30. **1-4. | Dropping Non-product data.**
31. In [10]:
32. **df**=**df**.**loc[(df['Itemname']**!=**'POSTAGE')**&**(df['Itemname']**!=**'DOTCOM POSTAGE')**&**(df['Itemname']**!=**'Adjust bad debt')**&**(df['Itemname']**!=**'Manual')]**
33. **1-5. | Filling null data**
34. In [12]:
35. **df**=**df**.**fillna('-')**
36. **df**.**isnull()**.**sum()**
37. Out[12]:
38. **BillNo 0**
39. **Itemname 0**
40. **Quantity 0**
41. **Date 0**
42. **Price 0**
43. **CustomerID 0**
44. **Country 0**
45. **dtype: int64**
46. **1-6. | Splitting data into year and month**
47. In [13]:
48. **df['Year']**=**df['Date']**.**apply(**lambda **x:x**.**split('.')[2])**
49. **df['Year']**=**df['Year']**.**apply(**lambda **x:x**.**split(' ')[0])**
50. **df['Month']**=**df['Date']**.**apply(**lambda **x:x**.**split('.')[1])**
51. **df**.**head()**
52. Out[13]:

|  | BillNo | Itemname | Quantity | Date | Price | CustomerID | Country | Year | Month |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 536365 | WHITE HANGING HEART T-LIGHT HOLDER | 6 | 01.12.2010 08:26 | 2,55 | 17850.0 | United Kingdom | 2010 | 12 |
| 1 | 536365 | WHITE METAL LANTERN | 6 | 01.12.2010 08:26 | 3,39 | 17850.0 | United Kingdom | 2010 | 12 |
| 2 | 536365 | CREAM CUPID HEARTS COAT HANGER | 8 | 01.12.2010 08:26 | 2,75 | 17850.0 | United Kingdom | 2010 | 12 |
| 3 | 536365 | KNITTED UNION FLAG HOT WATER BOTTLE | 6 | 01.12.2010 08:26 | 3,39 | 17850.0 | United Kingdom | 2010 | 12 |
| 4 | 536365 | RED WOOLLY HOTTIE WHITE HEART. | 6 | 01.12.2010 08:26 | 3,39 | 17850.0 | United Kingdom | 2010 | 12 |

1. **1-7. | Creating a Total price column**
2. In [14]:
3. **df['Price']**=**df['Price']**.**str**.**replace(',','.')**.**astype('float64')**
4. **df['Total price']**=**df**.**Quantity**\***df**.**Price**
5. **df**.**head()**
6. Out[14]:

|  | BillNo | Itemname | Quantity | Date | Price | CustomerID | Country | Year | Month | Total price |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 536365 | WHITE HANGING HEART T-LIGHT HOLDER | 6 | 01.12.2010 08:26 | 2.55 | 17850.0 | United Kingdom | 2010 | 12 | 15.30 |
| 1 | 536365 | WHITE METAL LANTERN | 6 | 01.12.2010 08:26 | 3.39 | 17850.0 | United Kingdom | 2010 | 12 | 20.34 |
| 2 | 536365 | CREAM CUPID HEARTS COAT HANGER | 8 | 01.12.2010 08:26 | 2.75 | 17850.0 | United Kingdom | 2010 | 12 | 22.00 |
| 3 | 536365 | KNITTED UNION FLAG HOT WATER BOTTLE | 6 | 01.12.2010 08:26 | 3.39 | 17850.0 | United Kingdom | 2010 | 12 | 20.34 |
| 4 | 536365 | RED WOOLLY HOTTIE WHITE HEART. | 6 | 01.12.2010 08:26 | 3.39 | 17850.0 | United Kingdom | 2010 | 12 | 20.34 |

1. **1-8. | Checking the Total price in each month.**
2. In [15]:
3. **df**.**groupby(['Year','Month'])['Total price']**.**sum()**
4. Out[15]:
5. **Year Month**
6. **2010 12 778386.780**
7. **2011 01 648311.120**
8. **02 490058.230**
9. **03 659979.660**
10. **04 507366.971**
11. **05 721789.800**
12. **06 710158.020**
13. **07 642528.481**
14. **08 701411.420**
15. **09 981408.102**
16. **10 1072317.070**
17. **11 1421055.630**
18. **12 606953.650**
19. **Name: Total price, dtype: float64**
20. **It is appropriate to look at 12-month increments to implement data analytics properly, so I'll drop the data for 2020 Dec.**
21. In [16]:
22. **linkcode**
23. **df**=**df**.**loc[df['Year']**!=**'2010']**

**Preprocessing dataset:**

Preprocessing the dataset in market basket insights is an important step in preparing the data for analysis. This involves cleaning the data to remove any errors or inconsistencies, and converting the data into a format that is suitable for market basket insights algorithms.

Here are some of the most common steps involved in preprocessing the dataset in market basket insights:

1. Identify and remove duplicate transactions. Duplicate transactions can occur for a variety of reasons, such as when a customer places multiple orders for the same items, or when a transaction is accidentally recorded twice.
2. Handle missing values. Missing values can occur when customers do not purchase all of the items in their basket, or when the data is not recorded correctly. There are a variety of ways to handle missing values, such as imputing the missing values with the most common values, or removing the transactions with missing values.
3. Convert categorical variables to numerical variables. Market basket insights algorithms typically work with numerical data. Therefore, any categorical variables in the dataset, such as product category or customer location, need to be converted to numerical variables. This can be done using a variety of methods, such as one-hot encoding or label encoding.
4. Bin continuous variables. Continuous variables, such as product price or customer age, can be binned into a smaller number of categories. This can help to improve the performance of market basket insights algorithms.

Once the dataset has been preprocessed, it is ready to be analyzed using market basket insights algorithms. These algorithms will identify frequent itemsets, which are sets of items that occur together frequently in transactions. The frequent itemsets can then be used to generate association rules, which are statements of the form "if A is purchased, then B is also likely to be purchased."

# **Data Pre-processing:**

In [2]:

df\_ = pd.read\_csv("../input/market-basket-analysis/Assignment-1\_Data.csv", sep = ";")

df = df\_.copy()

/opt/conda/lib/python3.7/site-packages/IPython/core/interactiveshell.py:3444: DtypeWarning: Columns (0) have mixed types.Specify dtype option on import or set low\_memory=False.

exec(code\_obj, self.user\_global\_ns, self.user\_ns)

In [3]:

df.head(10)

Out[3]:

|  | BillNo | Itemname | Quantity | Date | Price | CustomerID | Country |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 536365 | WHITE HANGING HEART T-LIGHT HOLDER | 6 | 01.12.2010 08:26 | 2,55 | 17850.0 | United Kingdom |
| 1 | 536365 | WHITE METAL LANTERN | 6 | 01.12.2010 08:26 | 3,39 | 17850.0 | United Kingdom |
| 2 | 536365 | CREAM CUPID HEARTS COAT HANGER | 8 | 01.12.2010 08:26 | 2,75 | 17850.0 | United Kingdom |
| 3 | 536365 | KNITTED UNION FLAG HOT WATER BOTTLE | 6 | 01.12.2010 08:26 | 3,39 | 17850.0 | United Kingdom |
| 4 | 536365 | RED WOOLLY HOTTIE WHITE HEART. | 6 | 01.12.2010 08:26 | 3,39 | 17850.0 | United Kingdom |
| 5 | 536365 | SET 7 BABUSHKA NESTING BOXES | 2 | 01.12.2010 08:26 | 7,65 | 17850.0 | United Kingdom |
| 6 | 536365 | GLASS STAR FROSTED T-LIGHT HOLDER | 6 | 01.12.2010 08:26 | 4,25 | 17850.0 | United Kingdom |
| 7 | 536366 | HAND WARMER UNION JACK | 6 | 01.12.2010 08:28 | 1,85 | 17850.0 | United Kingdom |
| 8 | 536366 | HAND WARMER RED POLKA DOT | 6 | 01.12.2010 08:28 | 1,85 | 17850.0 | United Kingdom |
| 9 | 536367 | ASSORTED COLOUR BIRD ORNAMENT | 32 | 01.12.2010 08:34 | 1,69 | 13047.0 | United Kingdom |

In [4]:

def check\_df(dataframe, head=5):

print("##################### Shape #####################")

print(dataframe.shape)

print("##################### Types #####################")

print(dataframe.dtypes)

print("##################### Head #####################")

print(dataframe.head(head))

print("##################### Tail #####################")

print(dataframe.tail(head))

print("##################### NA #####################")

print(dataframe.isnull().sum())

In [5]:

check\_df(df)

##################### Shape #####################

(522064, 7)

##################### Types #####################

BillNo object

Itemname object

Quantity int64

Date object

Price object

CustomerID float64

Country object

dtype: object

##################### Head #####################

BillNo Itemname Quantity Date \

0 536365 WHITE HANGING HEART T-LIGHT HOLDER 6 01.12.2010 08:26

1 536365 WHITE METAL LANTERN 6 01.12.2010 08:26

2 536365 CREAM CUPID HEARTS COAT HANGER 8 01.12.2010 08:26

3 536365 KNITTED UNION FLAG HOT WATER BOTTLE 6 01.12.2010 08:26

4 536365 RED WOOLLY HOTTIE WHITE HEART. 6 01.12.2010 08:26

Price CustomerID Country

0 2,55 17850.0 United Kingdom

1 3,39 17850.0 United Kingdom

2 2,75 17850.0 United Kingdom

3 3,39 17850.0 United Kingdom

4 3,39 17850.0 United Kingdom

##################### Tail #####################

BillNo Itemname Quantity Date \

522059 581587 PACK OF 20 SPACEBOY NAPKINS 12 09.12.2011 12:50

522060 581587 CHILDREN'S APRON DOLLY GIRL 6 09.12.2011 12:50

522061 581587 CHILDRENS CUTLERY DOLLY GIRL 4 09.12.2011 12:50

522062 581587 CHILDRENS CUTLERY CIRCUS PARADE 4 09.12.2011 12:50

522063 581587 BAKING SET 9 PIECE RETROSPOT 3 09.12.2011 12:50

Price CustomerID Country

522059 0,85 12680.0 France

522060 2,1 12680.0 France

522061 4,15 12680.0 France

522062 4,15 12680.0 France

522063 4,95 12680.0 France

##################### NA #####################

BillNo 0

Itemname 1455

Quantity 0

Date 0

Price 0

CustomerID 134041

Country 0

dtype: int64

In [6]:

*# Drop na values*

df.dropna(inplace=True)

*# Quantity and Price should be greater than 0*

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

*# We have to change the price column datatype as a numeric*

df ['Price'] = pd.to\_numeric(df['Price'], errors='coerce')

df = df[df["Price"] > 0]

In [7]:

check\_df(df)

##################### Shape #####################

(1537, 7)

##################### Types #####################

BillNo object

Itemname object

Quantity int64

Date object

Price float64

CustomerID float64

Country object

dtype: object

##################### Head #####################

BillNo Itemname Quantity Date \

45 536370 POSTAGE 3 01.12.2010 08:45

237 536392 RUSTIC SEVENTEEN DRAWER SIDEBOARD 1 01.12.2010 10:29

377 536403 POSTAGE 1 01.12.2010 11:27

1113 536527 POSTAGE 1 01.12.2010 13:04

4348 536779 Bank Charges 1 02.12.2010 15:08

Price CustomerID Country

45 18.0 12583.0 France

237 165.0 13705.0 United Kingdom

377 15.0 12791.0 Netherlands

1113 18.0 12662.0 Germany

4348 15.0 15823.0 United Kingdom

##################### Tail #####################

BillNo Itemname Quantity Date Price CustomerID \

521357 581493 POSTAGE 1 09.12.2011 10:10 15.0 12423.0

521375 581494 POSTAGE 2 09.12.2011 10:13 18.0 12518.0

521885 581570 POSTAGE 1 09.12.2011 11:59 18.0 12662.0

521922 581574 POSTAGE 2 09.12.2011 12:09 18.0 12526.0

521923 581578 POSTAGE 3 09.12.2011 12:16 18.0 12713.0

Country

521357 Belgium

521375 Germany

521885 Germany

521922 Germany

521923 Germany

##################### NA #####################

BillNo 0

Itemname 0

Quantity 0

Date 0

Price 0

CustomerID 0

Country 0

dtype: int64

164 rows × 9 columns

# **CONCLUSION**

linkcode

**Look at the confidences, it indicates the possibility that customers buying the X product will buy the Y product. We need to make a decision for them. Maybe in our website, when the customer click on first one, we need to show them the other item**