Market Basket Analysis

Alptuğ Topalhan
Dokuz Eylul University
İzmir,Turkey
alptug.topalhan@ogr.deu.edu.tr

Semih Furkan Karaman
Dokuz Eylul Univeristy
İzmir,Turkey
semihfurkan.karaman@ogr.deu.edu.tr

***Abstract*—This study examines the utilization of market basket analysis techniques and their potential impact within the retail sector. Market basket analysis is a powerful data mining technique used to understand customer purchasing patterns and uncover relationships between associated products. This report specifically explores certain data mining techniques, such as the Apriori algorithm, to discover relationships among items in shopping baskets. The findings of the study highlight the potential implications of these analyses on marketing strategies within retail enterprises. The aim of this report is to provide insights into how market basket analysis techniques can be applied within the retail sector.**

***Keywords*— Market Basket Analysis, Data Mining, Apriori Algorithm, Retail Sector, Customer Purchasing Patterns**

I. INTRODUCTION

Market Basket Analysis is a powerful data mining technique used to uncover patterns and relationships in data. Its purpose is to gain insights into consumer buying behavior. Understanding which products are frequently purchased together is crucial for making decisions in areas such as inventory management, product placement, and discount determination in today's retail environment. This project aims to reveal frequently co-occurring items in shopping baskets using data. This process assists retailers in optimizing product offerings and understanding customer behavior. For example, on a website, after a user selects a product, this data mining technique is employed to recommend another product. Similarly, placing the most commonly purchased items next to each other in stores can increase profits. This data mining technique is also used to determine which product will go on sale when another product is purchased..

II. DATASET

This dataset contains a total of 388,023 instances with 7 features. The "BillNo" feature represents the invoice number, "Itemname" contains product names, "Quantity" indicates product quantity, "Date" includes date and time information of transactions, "Price" specifies product prices, "CustomerID" represents customer identification, and "Country" indicates the country of the customer. Regarding data types, "BillNo" is of type int64; "Itemname," "Date," "Price," and "Country" are of type object; and "Quantity" and "CustomerID" are of type float. "Itemname" and "Country" are nominal categorical variables, each with distinct categories. "BillNo," "Quantity," and "CustomerID" represent numerical data, while "Date" is a time variable.

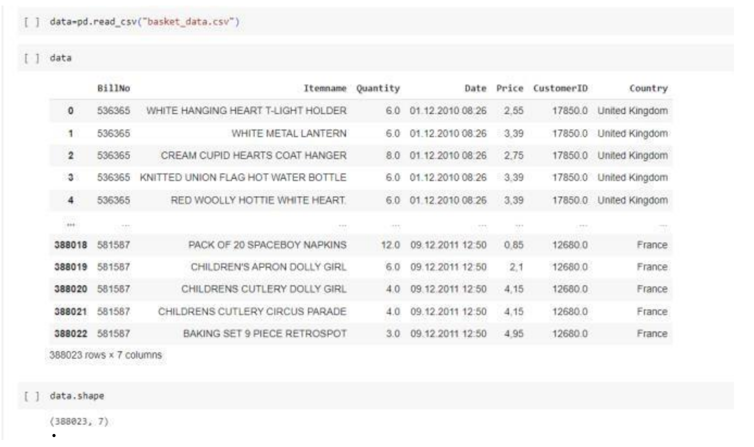
III. PREPROCESSING

Data preprocessing, also known as Data Preparation or Data Cleaning, encompasses the practice of identifying and rectifying erroneous or misleading records within a dataset. Data preprocessing involves a series of steps such as Data Cleaning, Data Integration, Data Reduction, and Data Transformation. In the context of this project, these steps were applied sequentially, utilizing the Python programming language and various libraries. The libraries 'Sklearn,' 'Numpy,' 'Pandas,' 'Seaborn,' and 'Matplotlib' were fundamental tools used in this process.

The initial phase of data preprocessing, known as data cleaning, is a crucial step aimed at addressing potential data inconsistencies or errors. This process includes tasks such as filling in missing values, correcting inaccurate data, identifying and removing outliers, and resolving inconsistencies.

IV. ANALYSIS OF DATA

Before starting the preprocessing stage, it's essential to examine our dataset. We used the data.shape method to analyze how many rows and columns exist in this dataset. Additionally, we analyzed which columns in this dataset would be useful for our purposes.



```
[ ] data=pd.read_csv("basket_data.csv")

[ ] data
```

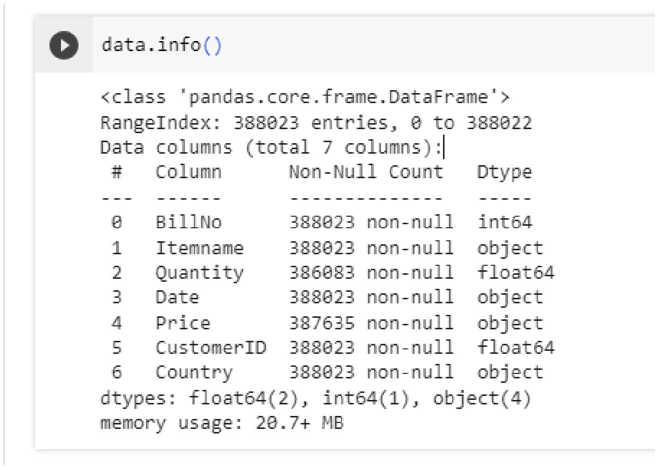
	BillNo	Itemname	Quantity	Date	Price	CustomerID	Country
0	536365	WHITE HANGING HEART T-LIGHT HOLDER	6.0	01.12.2010 08:26	2,55	17850.0	United Kingdom
1	536365	WHITE METAL LANTERN	6.0	01.12.2010 08:26	3,39	17850.0	United Kingdom
2	536365	CREAM CUPID HEARTS COAT HANGER	8.0	01.12.2010 08:26	2,75	17850.0	United Kingdom
3	536365	KNITTED UNION FLAG HOT WATER BOTTLE	6.0	01.12.2010 08:26	3,39	17850.0	United Kingdom
4	536365	RED WOOLLY HOTTIE WHITE HEART.	6.0	01.12.2010 08:26	3,39	17850.0	United Kingdom
...	...	...	...	...	...	...	...
388018	581587	PACK OF 20 SPACEBOY NAPKINS	12.0	09.12.2011 12:50	0,85	12680.0	France
388019	581587	CHILDREN'S APRON DOLLY GIRL	6.0	09.12.2011 12:50	2,1	12680.0	France
388020	581587	CHILDRENS CUTLERY DOLLY GIRL	4.0	09.12.2011 12:50	4,15	12680.0	France
388021	581587	CHILDRENS CUTLERY CIRCUS PARADE	4.0	09.12.2011 12:50	4,15	12680.0	France
388022	581587	BAKING SET 9 PIECE RETROSPOT	3.0	09.12.2011 12:50	4,95	12680.0	France

388023 rows × 7 columns

```
[ ] data.shape

(388023, 7)
```

Figure 1: Dataframe

We used the data.info method to view summary information about the data, aiming to conduct a more logical analysis. Additionally, we analyzed which columns would be useful for our market basket analysis.

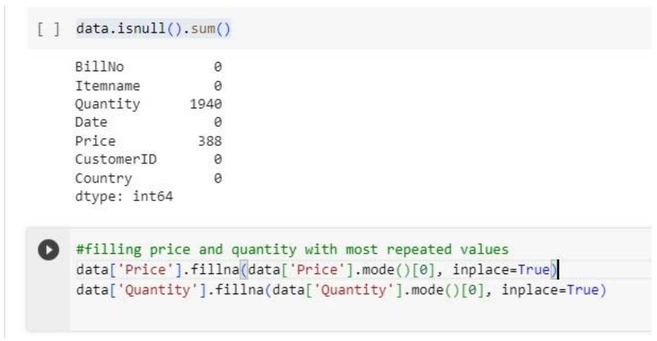


```
data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 388023 entries, 0 to 388022
Data columns (total 7 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   BillNo      388023 non-null  int64
 1   Itemname    388023 non-null  object
 2   Quantity    386083 non-null  float64
 3   Date        388023 non-null  object
 4   Price       387635 non-null  object
 5   CustomerID  388023 non-null  float64
 6   Country     388023 non-null  object
dtypes: float64(2), int64(1), object(4)
memory usage: 20.7+ MB
```

Figure 2: Infos about the data

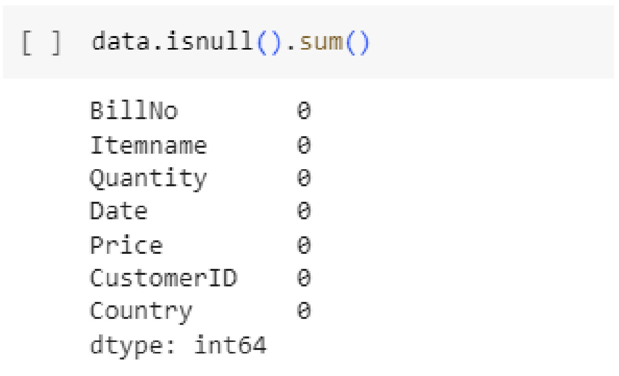
V. MISSING VALUES

We used the data.isnull().sum() function to identify missing values, and subsequently filled the missing values in columns with the most frequent value (mode) of each respective column where missing values were found.



```
[ ] data.isnull().sum()

BillNo          0
Itemname        0
Quantity     1940
Date            0
Price         388
CustomerID      0
Country         0
dtype: int64

#filling price and quantity with most repeated values
data['Price'].fillna(data['Price'].mode()[0], inplace=True)
data['Quantity'].fillna(data['Quantity'].mode()[0], inplace=True)
```

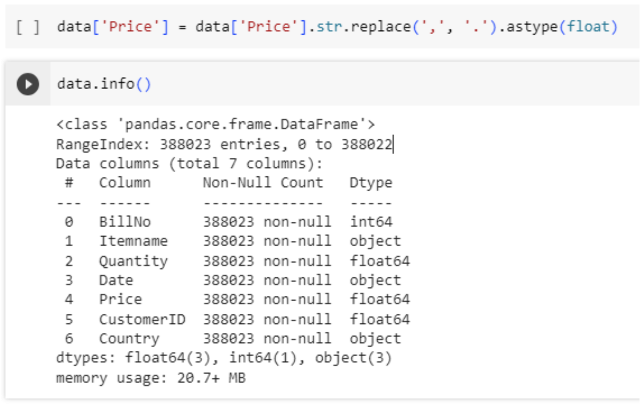
Figure 3: Showing Missing Values



```
[ ] data.isnull().sum()

BillNo        0
Itemname      0
Quantity      0
Date          0
Price         0
CustomerID    0
Country       0
dtype: int64
```

Figure 4: After filling

This function standardizes the format of values within the 'Price' column. It first replaces any commas used as decimal separators with periods, ensuring consistency. Then, it converts these corrected price values into floating-point numbers. Essentially, it transforms comma-separated price values into a format compatible with numerical operations, enabling easier mathematical computations and ensuring uniformity in the 'Price' column.

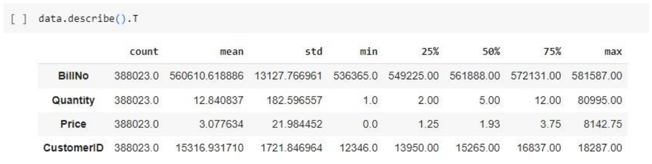


```
[ ] data['Price'] = data['Price'].str.replace(',', '.').astype(float)

data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 388023 entries, 0 to 388022
Data columns (total 7 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   BillNo      388023 non-null  int64
 1   Itemname    388023 non-null  object
 2   Quantity    388023 non-null  float64
 3   Date        388023 non-null  object
 4   Price       388023 non-null  float64
 5   CustomerID  388023 non-null  float64
 6   Country     388023 non-null  object
dtypes: float64(3), int64(1), object(3)
memory usage: 20.7+ MB
```

Figure 5: Types of columns

VI. VISUALIZATIONS

We used the data.describe().T function to obtain the statistical summary of the dataset using a tabular structure. This function provides key statistical characteristics of numerical values within the dataset, such as mean, standard deviation, minimum and maximum values. It offers a quick overview of the data's distribution and central tendencies. The transpose operation enhances the readability of these statistics by presenting the features in rows and the statistical measures in columns, making it easier to interpret.



```
[ ] data.describe().T
```

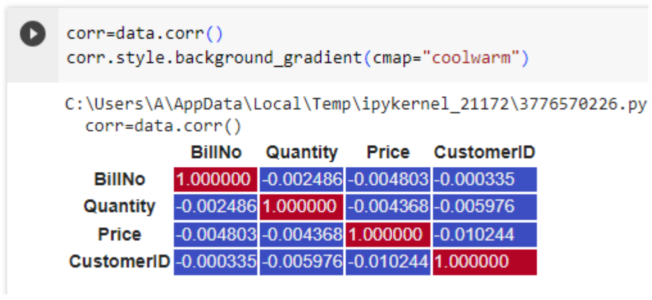
	count	mean	std	min	25%	50%	75%	max
BillNo	388023.0	560610.618886	13127.766961	536365.0	549225.00	561888.00	572131.00	581587.00
Quantity	388023.0	12.840837	182.596557	1.0	2.00	5.00	12.00	80995.00
Price	388023.0	3.077634	21.984452	0.0	1.25	1.93	3.75	8142.75
CustomerID	388023.0	15316.931710	1721.846964	12346.0	13950.00	15265.00	16837.00	18287.00

Figure 6: Statistical Infos About The Data

This code segment creates and visualizes a correlation matrix to understand relationships between variables in the dataset. Firstly, using data.corr(), it computes a correlation matrix that measures the relationship between each variable. This matrix is a table where values range from -1 to 1. Values close to 1 indicate a strong positive correlation, values close to -1 suggest a strong negative correlation, while values around 0 imply no correlation.

Next, corr.style.background_gradient(cmap="coolwarm") visualizes this correlation matrix in color. Each cell is represented by a color indicating the strength of the relationship. Shades of blue signify weak or negative correlations, while shades of red depict strong or positive correlations.

This visualization aids in comprehending relationships between variables within the dataset, facilitating the identification of variables that tend to change together. Essentially, it helps in grasping which variables rise or fall in relation to each other visually, offering insights into the dataset's inter-variable dynamics.



```
corr=data.corr()
corr.style.background_gradient(cmap="coolwarm")
```

```
C:\Users\A\AppData\Local\Temp\ipykernel_21172\3776570226.py
  corr=data.corr()
```

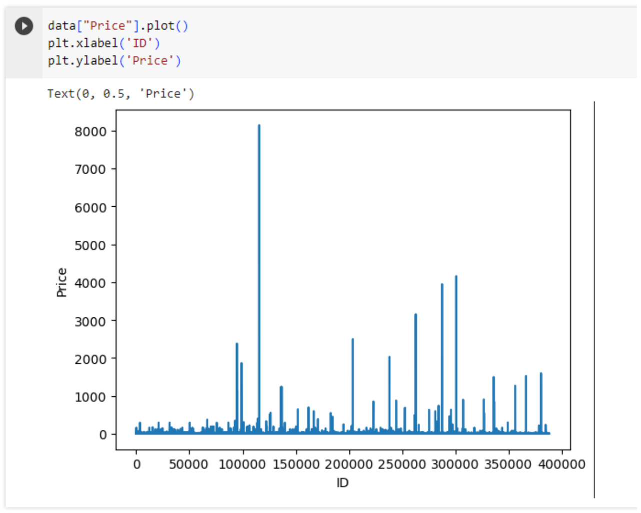
	BillNo	Quantity	Price	CustomerID
BillNo	1.000000	-0.002486	-0.004803	-0.000335
Quantity	-0.002486	1.000000	-0.004368	-0.005976
Price	-0.004803	-0.004368	1.000000	-0.010244
CustomerID	-0.000335	-0.005976	-0.010244	1.000000

Figure 7: Correlation Matrix With Coolwarm

This code plots the values in the "Price" column of the dataset. Using data["Price"].plot(), it represents the data in the "Price" column graphically, where each data point corresponds to a specific observation or ID.

The statements plt.xlabel('ID') and plt.ylabel('Price') label the axes of the graph. The X-axis is labeled as "ID," while the Y-axis represents "Price." This makes it clearer for readers to understand which axis represents which information.

Ultimately, this code visually represents the distribution or variation of price data in the dataset, aiding in understanding the overall trends or patterns in prices.

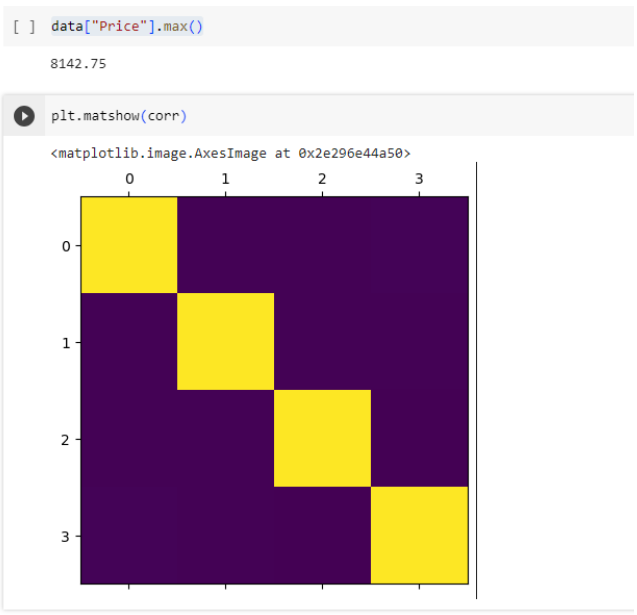


```
data["Price"].plot()
plt.xlabel('ID')
plt.ylabel('Price')
```

```
Text(0, 0.5, 'Price')
```

Figure 8: Plot of Price Column

These codes are used to represent the highest value of prices in the dataset and visualize the relationships between variables. The first one allows us to determine the maximum price within the dataset. The second one provides a visualization to understand the relationships between variables. It utilizes a correlation matrix visually, highlighting relationships between variables through different colors, with higher correlations emphasized using distinct shades. These codes serve the purpose of both identifying the maximum price and providing a visual representation to comprehend inter-variable relationships.



```
data["Price"].max()
```

```
8142.75
```

```
plt.matshow(corr)
```

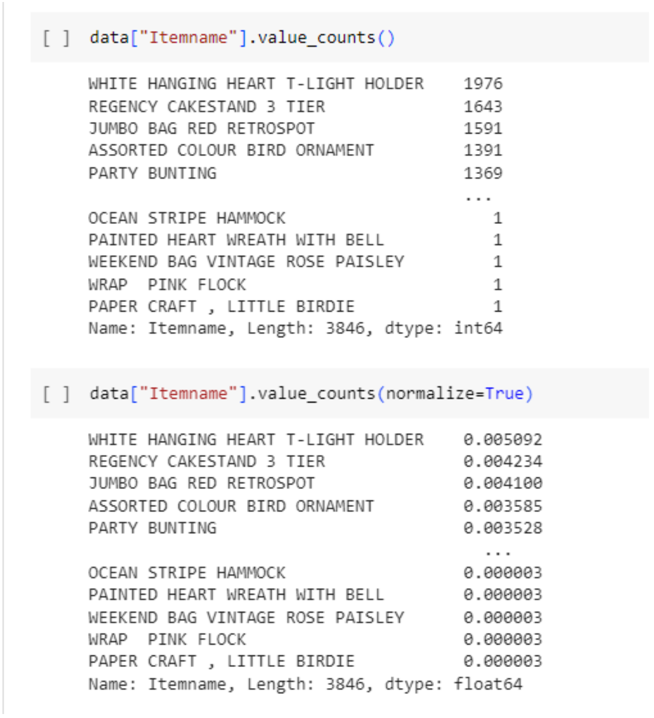
```
<matplotlib.image.AxesImage at 0x2e296e44a50>
```

Figure 9: Matshow of Correlation Matrix

VII. ANALYSIS OF ITEM DISTRIBUTION AND PERCENTAGE RATES IN THE DATASET

The data["Itemname"].value_counts() function retrieves the count of unique items in the "Itemname" column, displaying how many times each item appears. On the other hand, data["Itemname"].value_counts(normalize=True) normalizes the count of unique items in the "Itemname" column, showing the percentage of each item in the total.

Together, these functions offer insights into the frequency distribution of items in the dataset, providing both the count and the proportional representation of each unique item.

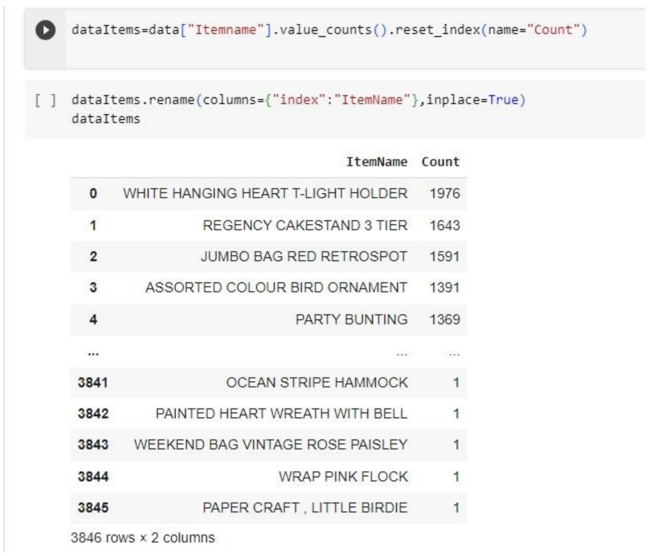


```
[ ] data["Itemname"].value_counts()

    WHITE HANGING HEART T-LIGHT HOLDER    1976
    REGENCY CAKESTAND 3 TIER              1643
    JUMBO BAG RED RETROSPOT               1591
    ASSORTED COLOUR BIRD ORNAMENT         1391
    PARTY BUNTING                         1369
                                          ...
    OCEAN STRIPE HAMMOCK                     1
    PAINTED HEART WREATH WITH BELL           1
    WEEKEND BAG VINTAGE ROSE PAISLEY         1
    WRAP  PINK FLOCK                         1
    PAPER CRAFT , LITTLE BIRDIE              1
    Name: Itemname, Length: 3846, dtype: int64

[ ] data["Itemname"].value_counts(normalize=True)

    WHITE HANGING HEART T-LIGHT HOLDER    0.005092
    REGENCY CAKESTAND 3 TIER              0.004234
    JUMBO BAG RED RETROSPOT               0.004100
    ASSORTED COLOUR BIRD ORNAMENT         0.003585
    PARTY BUNTING                         0.003528
                                            ...
    OCEAN STRIPE HAMMOCK                  0.000003
    PAINTED HEART WREATH WITH BELL        0.000003
    WEEKEND BAG VINTAGE ROSE PAISLEY      0.000003
    WRAP  PINK FLOCK                      0.000003
    PAPER CRAFT , LITTLE BIRDIE           0.000003
    Name: Itemname, Length: 3846, dtype: float64
```

Figure 10: Infos About Value Counts Column

dataItems=data["Itemname"].value_counts().reset_index(name="Count"): This operation computes the count of items, creating a new column named "Count" to display how many times each item appears.
dataItems.rename(columns={"index":"ItemName"}, inplace=True): Renames the column names in the generated dataframe. Here, it renames the "index" column to "ItemName".

These steps generate a dataframe with item counts and modify the column names within the dataframe.



```
dataItems=data["Itemname"].value_counts().reset_index(name="Count")
```

```
[ ] dataItems.rename(columns={"index":"ItemName"},inplace=True)
    dataItems
```

	ItemName	Count
0	WHITE HANGING HEART T-LIGHT HOLDER	1976
1	REGENCY CAKESTAND 3 TIER	1643
2	JUMBO BAG RED RETROSPOT	1591
3	ASSORTED COLOUR BIRD ORNAMENT	1391
4	PARTY BUNTING	1369
...	...	...
3841	OCEAN STRIPE HAMMOCK	1
3842	PAINTED HEART WREATH WITH BELL	1
3843	WEEKEND BAG VINTAGE ROSE PAISLEY	1
3844	WRAP PINK FLOCK	1
3845	PAPER CRAFT , LITTLE BIRDIE	1

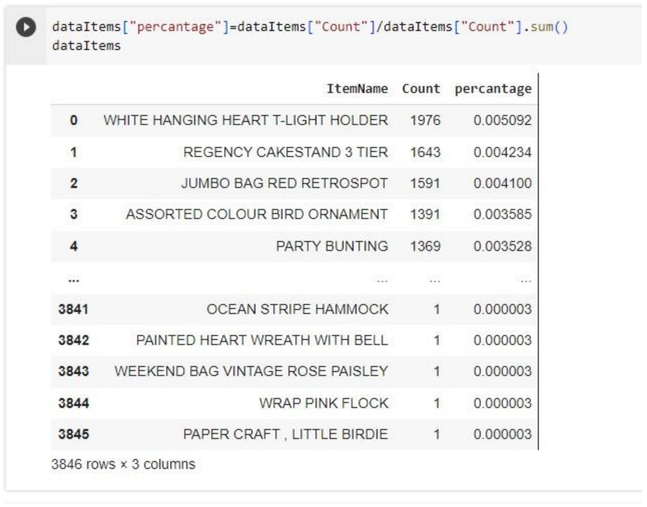
3846 rows × 2 columns

Figure 11: Renaming Columns

dataItems["percantage"]=dataItems["Count"]/dataItems["Count"].sum() calculates the percentage of each item in the total count and creates a new column named "percentage" to store these percentages in the DataFrame.

The resulting DataFrame, dataItems, displays the following:

The name of each item.
The frequency or count of occurrences for each item.
The percentage of each item in relation to the total count of all items in the dataset.
These operations summarize the count, frequency, and percentage distribution of items in the dataset.



```
dataItems["percantage"]=dataItems["Count"]/dataItems["Count"].sum()
dataItems
```

	ItemName	Count	percantage
0	WHITE HANGING HEART T-LIGHT HOLDER	1976	0.005092
1	REGENCY CAKESTAND 3 TIER	1643	0.004234
2	JUMBO BAG RED RETROSPOT	1591	0.004100
3	ASSORTED COLOUR BIRD ORNAMENT	1391	0.003585
4	PARTY BUNTING	1369	0.003528
...	...	...	...
3841	OCEAN STRIPE HAMMOCK	1	0.000003
3842	PAINTED HEART WREATH WITH BELL	1	0.000003
3843	WEEKEND BAG VINTAGE ROSE PAISLEY	1	0.000003
3844	WRAP PINK FLOCK	1	0.000003
3845	PAPER CRAFT , LITTLE BIRDIE	1	0.000003

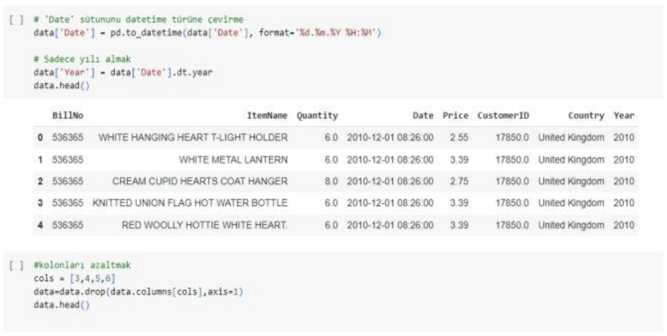
3846 rows × 3 columns

Figure 12: Calculating Percentage of Items

VIII. DATA CLEANING

Converting the 'Date' column to datetime type:
In this step, we convert the 'Date' column in the dataset to a timestamp format using the pd.to_datetime function. It restructures the date and time information into a specific format.

Extracting only the year:
Utilizing the 'Date' column, we extract the year information from each entry in the dataset. This facilitates the analysis to be conducted on a yearly basis.



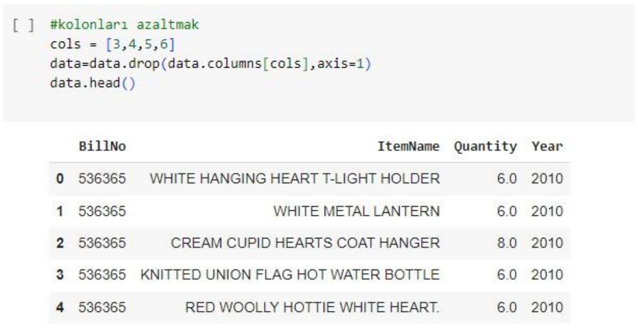
```
[ ] # 'Date' sütununu datetime türüne çevirme
    data['Date'] = pd.to_datetime(data['Date'], format='%d.%m.%Y %H:%M')

    # Sadece yılı almak
    data['Year'] = data['Date'].dt.year
    data.head()
```

	BillNo	ItemName	Quantity	Date	Price	CustomerID	Country	Year
0	536365	WHITE HANGING HEART T-LIGHT HOLDER	6.0	2010-12-01 08:26:00	2.55	17850.0	United Kingdom	2010
1	536365	WHITE METAL LANTERN	6.0	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	2010
2	536365	CREAM CUPID HEARTS COAT HANGER	8.0	2010-12-01 08:26:00	2.75	17850.0	United Kingdom	2010
3	536365	KNITTED UNION FLAG HOT WATER BOTTLE	6.0	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	2010
4	536365	RED WOOLLY HOTTIE WHITE HEART.	6.0	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	2010

```
[ ] #kolonları azaltmak
    cols = [3,4,5,6]
    data=data.drop(data.columns[cols],axis=1)
    data.head()
```

Figure 13: Extracting the year

Reducing columns:
We remove specific columns (such as columns 3, 4, 5, 6, etc.) from the dataset. This action cleans out unnecessary or unwanted information for the analysis.

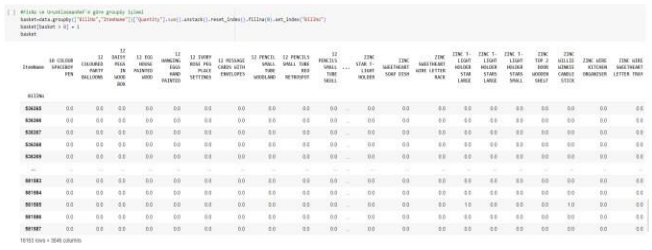


```
[ ] #kolonları azaltmak
    cols = [3,4,5,6]
    data=data.drop(data.columns[cols],axis=1)
    data.head()
```

	BillNo	ItemName	Quantity	Year
0	536365	WHITE HANGING HEART T-LIGHT HOLDER	6.0	2010
1	536365	WHITE METAL LANTERN	6.0	2010
2	536365	CREAM CUPID HEARTS COAT HANGER	8.0	2010
3	536365	KNITTED UNION FLAG HOT WATER BOTTLE	6.0	2010
4	536365	RED WOOLLY HOTTIE WHITE HEART.	6.0	2010

Figure 14: Reducing the columns

Performing a groupby operation on BillNo and ItemName:
In this step, we group the data based on the 'BillNo' and 'ItemName' columns. Then, we sum the 'Quantity' information within these groups. This process consolidates the quantities of products on each bill. Using the 'unstack()' method, we arrange the data into a structured table format and fill in missing values (NaN) with 0. Finally, any cell with a quantity value greater than 0 is marked as 1. The Transaction table is a data structure representing customer purchases. Each row corresponds to a single transaction (BillNo), and each column forms a binary matrix where each entry represents a product. For the Apriori and FP-Growth algorithm, the values in the matrix should be either 1 or 0.



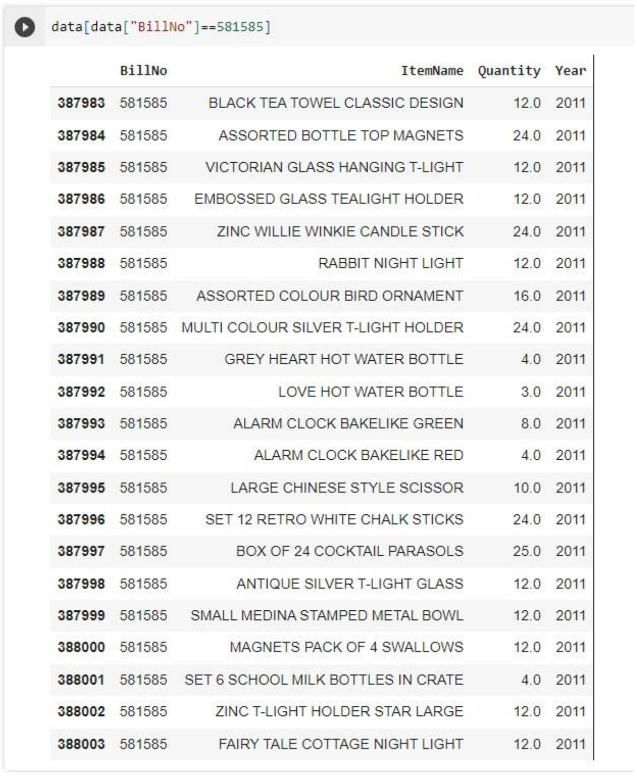
```
[ ] #FisNo ve UrunAdi'na göre groupby işlemi
    basket=data.groupby(["BillNo","ItemName"])["Quantity"].sum().unstack().reset_index().fillna(0).set_index("BillNo")
    basket[basket > 0] = 1
    basket
```

ItemName	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	12 PENCILS SMALL TUBE RED RETROSPOT	12 PENCILS SMALL TUBE SKULL	...	ZINC STAR T-LIGHT HOLDER	ZINC SWEETHEART SOAP DISH	ZINC SWEETHEART WIRE LETTER RACK	ZINC T-LIGHT HOLDER STAR LARGE	ZINC T-LIGHT HOLDER STARS LARGE	ZINC T-LIGHT HOLDER STARS SMALL	ZINC TOP 2 DOOR WOODEN SHELF	ZINC WILLIE WINKIE CANDLE STICK	ZINC WIRE KITCHEN ORGANISER	ZINC WIRE SWEETHEART LETTER TRAY
BillNo																					
536365	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
536366	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
536367	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
536368	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
536369	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
581583	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
581584	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
581585	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	1.0	0.0	0.0	0.0	1.0	0.0	0.0
581586	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
581587	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

18193 rows × 3646 columns

Figure 15: Transaction Table

Displaying information for a specific BillNo:
This code block is used to display details of a specific bill (FisNo). For instance, if there is a bill number 581585, this code retrieves all the details related to that particular bill.



```
data[data["BillNo"]==581585]
```

	BillNo	ItemName	Quantity	Year
387983	581585	BLACK TEA TOWEL CLASSIC DESIGN	12.0	2011
387984	581585	ASSORTED BOTTLE TOP MAGNETS	24.0	2011
387985	581585	VICTORIAN GLASS HANGING T-LIGHT	12.0	2011
387986	581585	EMBOSSED GLASS TEALIGHT HOLDER	12.0	2011
387987	581585	ZINC WILLIE WINKIE CANDLE STICK	24.0	2011
387988	581585	RABBIT NIGHT LIGHT	12.0	2011
387989	581585	ASSORTED COLOUR BIRD ORNAMENT	16.0	2011
387990	581585	MULTI COLOUR SILVER T-LIGHT HOLDER	24.0	2011
387991	581585	GREY HEART HOT WATER BOTTLE	4.0	2011
387992	581585	LOVE HOT WATER BOTTLE	3.0	2011
387993	581585	ALARM CLOCK BAKELIKE GREEN	8.0	2011
387994	581585	ALARM CLOCK BAKELIKE RED	4.0	2011
387995	581585	LARGE CHINESE STYLE SCISSOR	10.0	2011
387996	581585	SET 12 RETRO WHITE CHALK STICKS	24.0	2011
387997	581585	BOX OF 24 COCKTAIL PARASOLS	25.0	2011
387998	581585	ANTIQUE SILVER T-LIGHT GLASS	12.0	2011
387999	581585	SMALL MEDINA STAMPED METAL BOWL	12.0	2011
388000	581585	MAGNETS PACK OF 4 SWALLOWS	12.0	2011
388001	581585	SET 6 SCHOOL MILK BOTTLES IN CRATE	4.0	2011
388002	581585	ZINC T-LIGHT HOLDER STAR LARGE	12.0	2011
388003	581585	FAIRY TALE COTTAGE NIGHT LIGHT	12.0	2011

Figure 16: Checking Condition

IX. ALGORITHMS

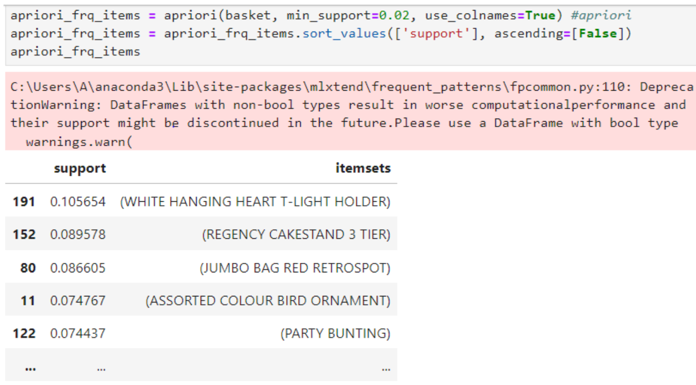
In this market basket analysis, three algorithms have been utilized: Apriori, FP-Growth, and ECLAT algorithms.[1]

Apriori:

Apriori algorithm, commonly employed in market basket analysis, is a rule-based method that extracts association rules by identifying frequent itemsets in transactional data, revealing patterns in customer purchasing behavior.

In this project, the apriori function was created using the mlxtend library.

The Apriori function was invoked with the specified transaction table and minimum support threshold. The support values for the itemsets have been returned as a dataframe and sorted in descending order.



```
apriori_frq_items = apriori(basket, min_support=0.02, use_colnames=True) #apriori
apriori_frq_items = apriori_frq_items.sort_values(['support'], ascending=[False])
apriori_frq_items
```

```
C:\Users\A\anaconda3\Lib\site-packages\mlxtend\frequent_patterns\fpcommon.py:110: Depreca
tionWarning: DataFrames with non-bool types result in worse computationalperformance and
their support might be discontinued in the future.Please use a DataFrame with bool type
  warnings.warn(
```

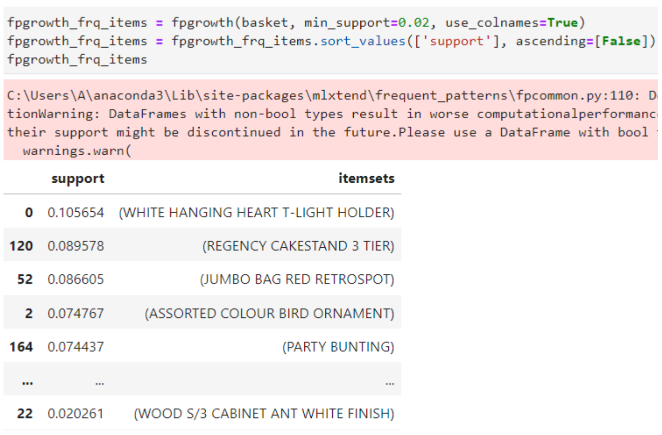
	support	itemsets
191	0.105654	(WHITE HANGING HEART T-LIGHT HOLDER)
152	0.089578	(REGENCY CAKESTAND 3 TIER)
80	0.086605	(JUMBO BAG RED RETROSPOT)
11	0.074767	(ASSORTED COLOUR BIRD ORNAMENT)
122	0.074437	(PARTY BUNTING)
...	...	...

Figure 17: Apriori

FP-Growth:

FP-Growth is a frequent itemset mining algorithm utilized in market basket analysis and association rule discovery, distinguishing itself from Apriori by offering enhanced efficiency and requiring fewer passes through the dataset.

The FP-Growth function, like Apriori, is created using the MLxtend library. MLxtend is a Python library that facilitates the use of frequent itemset mining algorithms, and the FP-Growth function, implemented through MLxtend, efficiently discovers frequent itemsets in a given dataset, providing a performance advantage, particularly with large datasets.



```
fpgrowth_frq_items = fpgrowth(basket, min_support=0.02, use_colnames=True)
fpgrowth_frq_items = fpgrowth_frq_items.sort_values(['support'], ascending=[False])
fpgrowth_frq_items
```

```
C:\Users\A\anaconda3\Lib\site-packages\mlxtend\frequent_patterns\fpcommon.py:110: D
tionWarning: DataFrames with non-bool types result in worse computationalperformanc
their support might be discontinued in the future.Please use a DataFrame with bool
  warnings.warn(
```

	support	itemsets
0	0.105654	(WHITE HANGING HEART T-LIGHT HOLDER)
120	0.089578	(REGENCY CAKESTAND 3 TIER)
52	0.086605	(JUMBO BAG RED RETROSPOT)
2	0.074767	(ASSORTED COLOUR BIRD ORNAMENT)
164	0.074437	(PARTY BUNTING)
...	...	...
22	0.020261	(WOOD S/3 CABINET ANT WHITE FINISH)

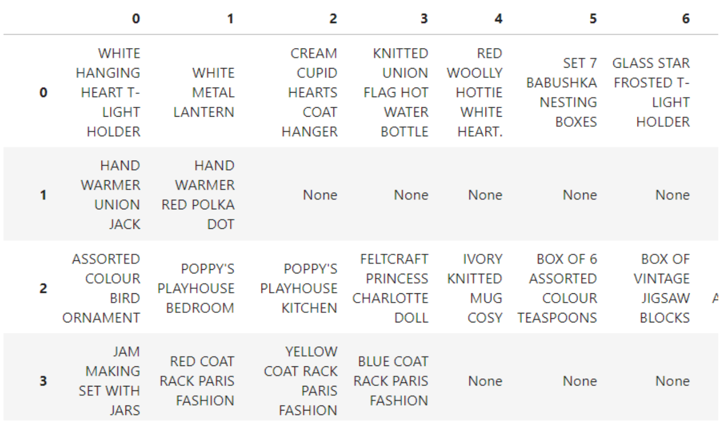
Figure 18: FP-Growth

ECLAT:

The ECLAT (Equivalence Class Transformation) algorithm, like Apriori, is a frequent itemset mining algorithm used for market basket analysis and association rule discovery. Unlike Apriori, it adopts a more efficient approach by updating only the frequency table instead of generating all combinations when forming candidate itemsets.

The "pyECLAT" library is a straightforward Python package used to discover relationships between variables in various data frames based on the ECLAT (Equivalence Class Transformation) method. Unlike the Apriori method, the ECLAT approach doesn't rely on confidence and lift calculations but is centered around determining support combinations of variables, returning two dictionaries containing information about frequency and support.

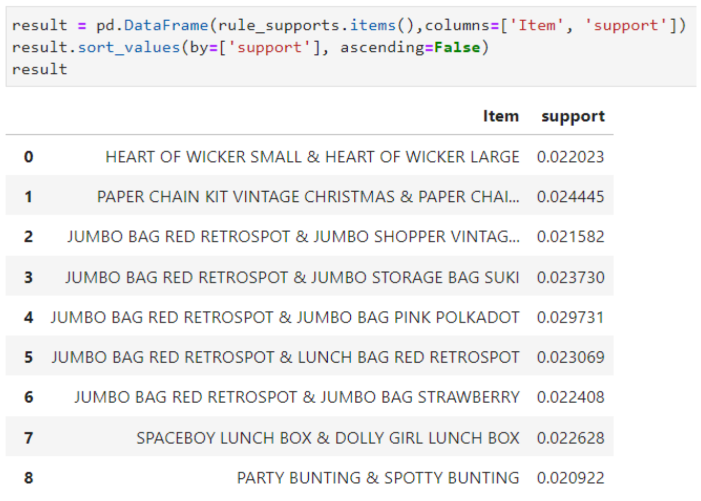
We realized that we needed to change the format of the dataset in order to use the pyECLAT library.



	0	1	2	3	4	5	6
0	WHITE HANGING HEART T-LIGHT HOLDER	WHITE METAL LANTERN	CREAM CUPID HEARTS COAT HANGER	KNITTED UNION FLAG HOT WATER BOTTLE	RED WOOLLY HOTTIE WHITE HEART.	SET 7 BABUSHKA NESTING BOXES	GLASS STAR FROSTED T-LIGHT HOLDER
1	HAND WARMER UNION JACK	HAND WARMER RED POLKA DOT	None	None	None	None	None
2	ASSORTED COLOUR BIRD ORNAMENT	POPPY'S PLAYHOUSE BEDROOM	POPPY'S PLAYHOUSE KITCHEN	FELTCRAFT PRINCESS CHARLOTTE DOLL	IVORY KNITTED MUG COSY	BOX OF 6 ASSORTED COLOUR TEASPOONS	BOX OF VINTAGE JIGSAW BLOCKS
3	JAM MAKING SET WITH JARS	RED COAT RACK PARIS FASHION	YELLOW COAT RACK PARIS FASHION	BLUE COAT RACK PARIS FASHION	None	None	None

Figure 19: New DataFrame

After creating a new dataframe, we utilized the necessary functions of the pyECLAT library to execute the ECLAT algorithm. The library returned a dataframe containing the support values of the itemsets.



```
result = pd.DataFrame(rule_supports.items(),columns=['Item', 'support'])
result.sort_values(by=['support'], ascending=False)
result
```

	Item	support
0	HEART OF WICKER SMALL & HEART OF WICKER LARGE	0.022023
1	PAPER CHAIN KIT VINTAGE CHRISTMAS & PAPER CHAI...	0.024445
2	JUMBO BAG RED RETROSPOT & JUMBO SHOPPER VINTAG...	0.021582
3	JUMBO BAG RED RETROSPOT & JUMBO STORAGE BAG SUKI	0.023730
4	JUMBO BAG RED RETROSPOT & JUMBO BAG PINK POLKADOT	0.029731
5	JUMBO BAG RED RETROSPOT & LUNCH BAG RED RETROSPOT	0.023069
6	JUMBO BAG RED RETROSPOT & JUMBO BAG STRAWBERRY	0.022408
7	SPACEBOY LUNCH BOX & DOLLY GIRL LUNCH BOX	0.022628
8	PARTY BUNTING & SPOTTY BUNTING	0.020922

Figure 20: ECLAT

X. TIME ANALYSIS

When compared in terms of time, it was observed that the most efficient algorithm among the Apriori, FP-Growth, and ECLAT algorithms was the FP-Growth algorithm. The least efficient algorithm was the ECLAT algorithm. We utilized the time library to observe this mathematically and presented the time data we obtained graphically.

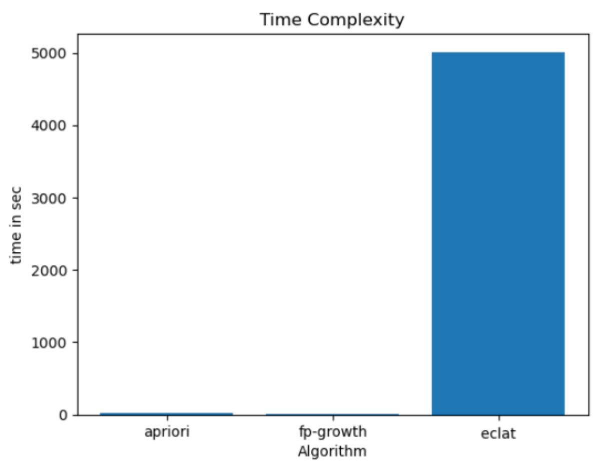
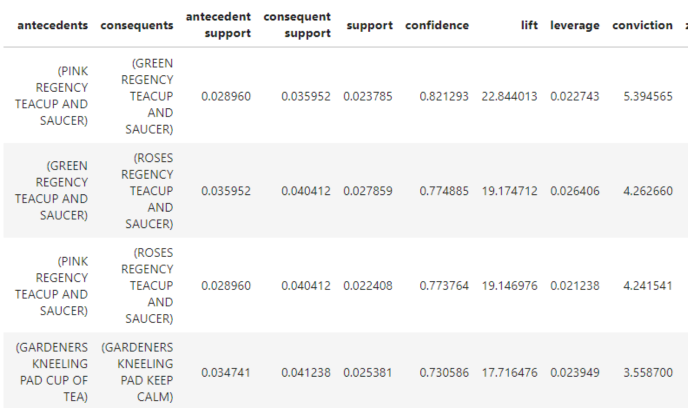


Figure 21: Time Analysis

XI. ASSOCIATION RULES

Association rules are utilized to uncover relationships within large datasets. Association rules analysis examines relationships in large datasets, using metrics like support, confidence, lift, leverage, conviction, and Zhang's metric to uncover patterns and insights.

In our project, we employed them to identify the most commonly co-purchased items. We utilized the data frames generated by the Apriori and FP-Growth algorithms as parameters for the association_rules function.



antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
(PINK REGENCY TEACUP AND SAUCER)	(GREEN REGENCY TEACUP AND SAUCER)	0.028960	0.035952	0.023785	0.821293	22.844013	0.022743	5.394565
(GREEN REGENCY TEACUP AND SAUCER)	(ROSES REGENCY TEACUP AND SAUCER)	0.035952	0.040412	0.027859	0.774885	19.174712	0.026406	4.262660
(PINK REGENCY TEACUP AND SAUCER)	(ROSES REGENCY TEACUP AND SAUCER)	0.028960	0.040412	0.022408	0.773764	19.146976	0.021238	4.241541
(GARDENERS KNEELING PAD CUP OF TEA)	(GARDENERS KNEELING PAD KEEP CALM)	0.034741	0.041238	0.025381	0.730586	17.716476	0.023949	3.558700

Figure 22: Association Rules

The most commonly co-purchased items have been observed. These data can be utilized by companies to identify products eligible for bundled discounts or to strategically group best-selling items for enhanced sales.[2]

Later, a comparison was made between the association analysis results of the Apriori and FP-Growth algorithms.

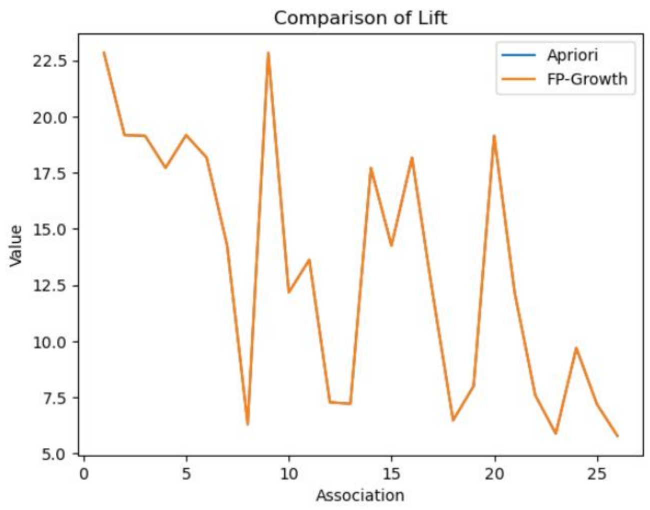


Figure 23: Comparison of Lift

The lift value is a crucial metric used in association analysis, measuring the strength of a relationship within a rule. If the lift value is greater than 1, it indicates that the rule contains a predictable relationship and performs better than random chance.

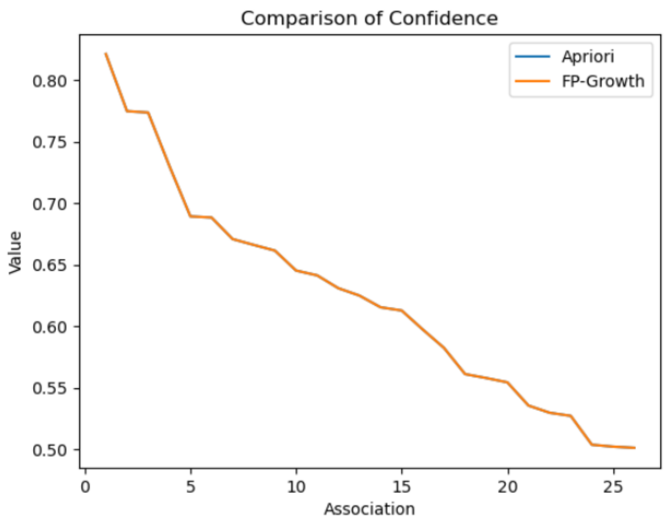


Figure 24: Comparison of Confidence

A high confidence value indicates that when the antecedent occurs, there is a high probability of the consequent also occurring, signifying the robust validity of the rule.

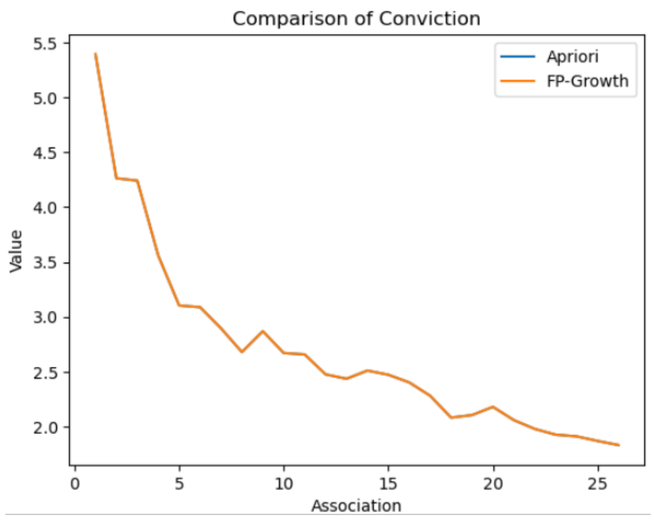


Figure 25: Comparison of Conviction

The conviction value is a metric used to assess the reliability of a rule in association analysis. A high conviction value indicates that a rule has a strong relationship between the antecedent and consequent, but it is not consistent with randomness.

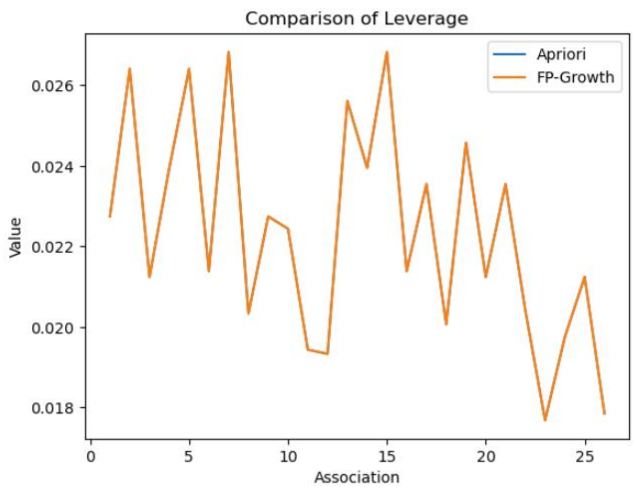


Figure 26: Comparison of Leverage

Leverage is a metric in association analysis that measures the difference between the observed and expected frequency of a rule; a positive value indicates that the frequency is higher than expected, while a negative value suggests it is lower than expected.

Upon examining the association rules values of the Apriori and FP-Growth algorithms, it was observed that these values were entirely overlapping. Both algorithms generated similar rule sets on the same dataset, indicating that they identified the same relationships. However, a notable difference was observed in the faster execution of the FP-Growth algorithm compared to Apriori.

CONCLUSION

Market Basket Analysis, Apriori, FP-Growth, and Eclat algorithms were explored to uncover patterns and associations within transactional data. The comparative evaluation indicated that FP-Growth excelled in terms of speed, making it a preferable choice for large-scale datasets. Both FP-Growth and Apriori consistently delivered similar average values for lift, support, and confidence, suggesting that they are interchangeable in terms of association rule mining performance. However, caution is advised when considering the Eclat algorithm, as it exhibited significant computational slowness. In conclusion, Market Basket Analysis, when implemented with efficient algorithms like FP-Growth or Apriori, proves to be a valuable tool for identifying product associations and improving strategic decision-making in areas such as product placement, promotions, and inventory management. The choice between FP-Growth and Apriori should be guided by the specific requirements of the analysis, considering the dataset size and the need for faster processing.

ACKNOWLEDGMENT

We extend our appreciation to the creators of the dataset utilized in our market basket analysis project, with special thanks to the Kaggle for providing this valuable data. Our gratitude also goes to the numerous researchers and organizations that have shared their findings on market basket analysis. We would like to express our special thanks to the individuals who generously dedicated their time and expertise, guiding us in understanding and implementing machine learning techniques tailored to the specifics of the market basket dataset.

References

[1] M. Dhanabhakyam and M. Punithavalli, "A Survey on Data Mining Algorithms for Market Basket Analysis," Global Journal of Computer Science and Technology, vol. 11, no. 11, pp. 1-6, July 2011.

[2] M. Kaura and S. Kanga, "Market Basket Analysis: Identify the changing trends of market data using association rule mining," in International Conference on Computational Modeling and Security (CMS), 2016.