VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

ONLINE SHOPPING CART ANALYSIS TO UNDERSTAND THE CUSTOMER'S ONLINE EXPENDITURE PATTERN

A PROJECT REPORT
For

DATA MINING TECHNIQUES (ITE2006)
in
B.Tech – Information Technology and Engineering

by

Name	***Registration number***
AMAN SOMANI	**19BIT0166**
RITWIK SUKHIJA	**19BIT0411**
VISHNU PRABHAKAR	**19BIT0043**

Under the guidance of

Prof. Valarmathi B

Google Drive Link :
https://drive.google.com/drive/folders/1rY7Ri_eaGTs8TWxXq5Dbyfej0o-2q-Ka?usp=sharing

Abstract :

In day-to-day activities huge amounts of data are generated, as a result the volume of data is increasing dramatically. Mining information from this explosive growth of data has become one of the major challenges for data management and mining communities.Moreover, the majority of the recognized organizations collect and store massive amounts of customer transaction data. However, having these massive data do not meanthe organizations had rich commercial information. The business industries need to discover valuable information and knowledge from this vast quantity of data. This leadsus to online shopping cart analysis. Shopping cart analysis aims at finding out purchasing patterns by discovering important associations among the products which they place in their online shopping carts. It not only assists in decision making process but also increases sales in many e-commerce websites like amazon, flipkart etc...Apriori, FP Growth and Eclat are the most common algorithms for mining frequent itemsets. For all of these algorithms predefined minimum support is needed to satisfy for identifying the frequent itemsets. But when the minimum support is low, a huge numberof candidate sets will be generated which requires large computation. In this project, weplan to follow an approach that has been proposed to avoid this large computation by reducing the items of dataset with top selling products. The top selling products will be marketed more with the help of suggestions to the customers. Various percentages of top selling products like 30%, 40%, 50%, 55% have been taken and for all algorithmsfrequent itemsets and association rules are generated. The results show that if top selling items are used, it is possible to get almost same frequent itemsets and association rules within a short time comparing with those outputs which are derived by computing all the items. From time comparison it is also found that FP Growth algorithm takes smaller time than Apriori algorithm

Keywords:

Market Basket Analysis (MBA), Data Mining, Association Rule Mining (ARM), Product Recommendation system.

Introduction :

Market basket analysis is a data analysis methodology based on Association data mining that merchants employ to boost sales by better understanding customer purchase patterns. It entails evaluating huge data sets, such as purchase histories, to identify product groups and products that are likely to be bought together. Customers' purchase patterns are discovered through market basket analysis, which identifies significant links between the things they place in their shopping baskets. It not only aids in the decision-making process, but it also boosts sales in many businesses. The most common methods for mining frequent itemsets are Apriori, FP Growth and Eclat. Market Basket Analysis is an important part of the analytical system in the retail organisation to determine the placement of goods, designing sales promotion for different segments of customers to improve customer satisfaction and hence the profit of the supermarkets.

Proposed Method:

We will use the concept of Association Rule Mining in our project.
Association rule algorithms includes :

- Apriori Algorithm
- F-P Growth Algorithm (Frequent Pattern Algorithm)
- Eclat Algorithm

So, we will be using Apriori Algorithm and F-P Growth Algorithm and Eclat Algorithm for the Market Basket Analysis (MBA).

Literature Survey :

S.No.	Title of the Paper and Year	Algorithms used	Data set being used	Performance measures	Scope for future work
1.	An Implementation of the FP-growth Algorithm (2010)	FP-growth Algorithm , Apriori and Eclat algorithm	BMS-Webview-1,T10I4D100K 1, census, chess, and mushroom	Accuracy - 72%	The performance of the two projection methods for projecting an FP-tree, especially, why the second is sometimes much slower than the first, needs further investigation
2.	Performance Analysis of Apriori and FP-Growth Algorithms (Association Rule Mining) (2016)	Weka workbench	Supermarket and Voter datasets	Accuracy-87%	The performance of Apriori and FP-Growth algorithms can be analysed by taking less execution time and lesser number of scans for different instances.
3.	Study on Market Basket Analysis with	Apriori Algorithm	(Name not mentioned)	Accuracy - 78%	For future using this project a

	Apriori Algorithm Approach (2021)				shop owner can place some items close together in future for their customers to pick more than one item whether they were previously only going to buy a single item.
4.	Market Basket Analysis with Enhanced Support Vector Machine (ESVM) Classifier for Key Security in Organisation (2019)	Association Rule Mining Algorithm Based on Probabilistic Graphical Model, FPM algorithm, ESVM algorithm	Bank Marketing Dataset	Accuracy - 86%	The works intended for the future will provide the implementation of specific features that the work was not capable of exploring by using a more reliable algorithm in the system, which in turn would facilitate the system in operating rapidly and with more efficiency. Efforts on improving the search techniques can also be useful in boosting the market and profitability.
5.	Market Basket Analysis: Identify the changing trends of market data using association rule mining (2016)	Market Basket algorithm	Extended bakery datasets	Accuracy - 70%	Authors suggested that some areas are still there which need to be focused on. Firstly, results have been influenced

					greatly by the manual threshold values for score, so it is needed to automate the threshold values for better recognition of outliers. Secondly, this approach is specifically targeted at Market Basket Data, it may perhaps be extended to other areas.
6.	Comparative Analysis of Market Basket Analysis through Data Mining Techniques (2021)	Collaborative Filtering Algorithm, ARM	Transactional Dataset	Accuracy - 74%	There are different data mining algorithms available to find out the frequent trends in consumer's buying pattern and also to give the product recommendation on the basis of past purchases made by the consumer.
7.	Analysing Online Transaction Data using Association Rule Mining: Misumi Philippines Market Basket Analysis (2019)	Apriori Algorithm	Transactional Dataset	Accuracy-70%	Producing a list of package items for consumers based on strong rules generated by association rule mining at a lesser runtime rate.
8.	Data Mining Applications for Sales Information System Using Market Basket	Generalised Rule Induction Based Hash Algorithm,	(Name not mentioned)	Accuracy-71%	According to the questionnaire, 85% of consumers rate

	Analysis on Stationery Company (2017)	Apriori Algorithm			the appearance of good, 15% of consumers rate the appearance is very good, 100% of users assess the accuracy of the data generated very good, 25% of users rate the application, simply, 75% of users rated ease of application excellent, 100% of users evaluate reports produced good, 10% of users to assess the suitability of the needs of both, 90% of consumers rate the suitability to the needs of very good.
9.	MARKET BASKET ANALYSIS USING FP GROWTH AND APRIORI ALGORITHM: A CASE STUDY OF MUMBAI RETAIL STORE (2016)	FP GROWTH AND APRIORI ALGORITHM	Synthetic Transactional Dataset	Accuracy-74%	Increasing the sample size of the dataset and accuracy of the analysis.
10.	MARKET BASKET ANALYSIS: UNDERSTANDING INDIAN CONSUMER BUYING BEHAVIOUR OF SPAIN MARKET (2016)	APRIORI ALGORITHM	Market basket dataset	Accuracy-73%	The market basket problem can be seen as the best example of mining association rules. Discovering association rules has been a well-studied area for the past decade.

					Building up on previous researches by using established methods for mining association rules allowed for discovering useful information for the retailer

S.No.	Title of the Paper and Year	Algorithms used	Data set being used	Performance measures	Scope for future work
11.	Consumer purchase patterns based on market basket analysis using apriori algorithms(2020)	Apriori algorithm	Transactional dataset	Accuracy-74%	It is expected that the results of consumer purchasing patterns can help minimarket managers in making decisions to get even better profits.
12.	FP-Tree Based Algorithms Analysis: FP-Growth, COFI-Tree and CT-PRO(2011)	FPGrowth, COFI-Tree, CT-PRO	Transactional dataset	Accuracy-80%	FP-Growth is the first successful tree base algorithm for mining the frequent itemsets. As for large databases its structure does

					not fit into main memory therefore new techniques come into pictures which are the variations of the classic FP-Tree.
13.	Online Shopping: Do Men Behave Differently than Women?(2021)	EFA, CFA and SEM tools	(Name not mentioned)	Accuracy-77%	The study proposed and validated gender specific behavioural determinants of the young Indian online shoppers. Taking this study as a base, further research investigations into gender-wise specific product category behaviour can be studied.
14.	Uncovering Modern Clinical Applications of Fuzi and Fuzi-Based Formulas: A Nationwide Descriptive Study With Market Basket Analysis(2021)	Apriori algorithm	Taiwan National Health Insurance dataset	Accuracy-83%	The results light up the road to the development of new Fuzi-based botanic drugs
15.	Chiller system performance	FP-growth algorithm, Association	Operational Dataset	Accuracy-72%	Tailor-made optimisation strategies and the

	management with market basket analysis(2018)	optimisation algorithm, Apriori algorithm, Self-joining algorithm			associated electricity savings can be further evaluated when developing a COP model with significant variables and predicting its maximum values under different operating conditions.
16.	Improving Efficiency of Apriori Algorithm Using Transaction Reduction (2013)	Apriori Algorithm	(Name not mentioned)	Accuracy-75%	Although this improved algorithm is optimised and efficient, but it has overhead to manage the new database after every generation of Lk. So, there should be some approach which has a very small number of scans of the database. Another solution might be the division of large databases among processors.
17.	Sales Prediction System using Machine Learning(2019)	Decision tree algorithm, XGBoost regressor	Big Mart Companies Real-world data set	Accuracy-81%	In the future, we will use the output of this project as part of the price optimization problem which we are planning to work on.
18.	Research on a Prediction Model of Online Shopping Behaviour Based on Deep Forest Algorithm(2020)	Deep forest algorithm	Customers online shopping behaviour dataset	Accuracy-79%	The combination of algorithm and business is still at the initial stage. Although machine learning

					algorithms are data-driven, there is no uniform standard for the construction and selection of features. Further work needs to be done on them.
19.	AN IMPROVED APRIORI ALGORITHM FOR ASSOCIATION RULES(2014)	Apriori Algorithm	Transactional database	Accuracy-78%	The time consumed to generate candidate support counts in the improved Apriori is less than the time consumed in the original Apriori; the improved Apriori reduces the time consuming by 67.38%. As this is proved and validated by the experiments, it would prove very useful in the future.
20.	A NOVELTY OF DATA MINING FOR PROMOT ING EDUCATION BASED ON FP-GROWTH ALGORITHM(2018)	FP-growth algorithm	Transactional database	Accuracy-76%	From research done on some attributes not used in the resulting rule, so the s election of attributes in the dataset is very important.

S.No.	**Title of the Paper and Year**	**Algorithms used**	**Data set being used**	**Performance measures**	**Scope for future work**
21.	Market Basket Analysis on Sales Transactions for Micro,	Apriori algorithm	The sales transaction dataset	Accuracy- 73%	Further research into creating an automatic data

	Small and Medium Enterprises Using Apriori Algorithm to Support Business Promotion Strategy in RDA Hijab				processing system (data preprocessing) so that the process is more efficient and the data processing process is shorter. In addition, creating a priori process with python language so that the execution process is faster.
22	Personalised Market Basket Prediction with Temporal Annotated Recurring Sequences	Apriori algorithm,FP-Growth algorithm	Coop dataset	Accuracy- 72%	Furthermore, we would like to exploit TARS for developing analytical services in other domains, such as mobility data, musical listening sessions and health data. Finally, in line with, it would be interesting to study if there is an improvement in the quality of the prediction if the user-centric models are exploited for developing a collective or hybrid predictive approach
23.	Data Mining Applications for Sales Information System Using Market Basket Analysis on Stationery Company	D. Apriori Algorithm	transaction dataset	Accuracy- 85%	Further Applications can perform data mining process based on existing sales data. Then program can help decide when to make the process of bundling.

24.	A Market Basket Analysis of Beef Calf Management Practice Adoption	Apriori algorithm	Oklahoma Beef Management dataset	Accuracy-70%	we can use market basket analysis to identify which combinations of research-based recommended calf health management practices are bundled frequently on the ranch, which are less frequently bundled, and to gain insight as to how to help producers both recognize their importance and assist them in practice implementation to increase profitability through joint adoption of practices.
25.	MARKET BASKET ANALYSIS USING FP GROWTH AND APRIORI ALGORITHM: A CASE STUDY OF MUMBAI RETAIL STORE	GROWTH AND APRIORI ALGORITHM	Transaction dataset	Accuracy-70%	A synthetic data set has been used with 77 items each for analysis. A set of association rules are obtained by applying Apriori algorithm and FP growth.
26.	Comparing unsupervised probabilistic machine learning methods for market basket analysis	(MH-RM) algorithm	real-world point-of-sale transactions	Accuracy-81%	To infer managerial implications we determine both probability increases of other categories and expected relative basket size increases due to promoting a

					product category. Product categories with high expected relative increases constitute candidates for a promotion whose the objective is to increase basket size.
27.	Market basket analysis by solving the inverse Ising problem: Discovering pairwise interaction strengths among products	Metropolis–Hasting (MH) algorithm	transactional sample dataset	Accuracy-75%	Furthermore, the use of the couplings to obtain a network representation of the purchasing system and the hierarchical structure that emerges from the topology of that network, is extremely interesting because it allows to understand the link that exists between the energy of a state and its revealed hierarchy
28.	Application of Market Basket Analysis for the Visualization of Transaction Data Based on Human Lifestyle and Spectroscopic Measurements	Apriori algorithm	transactional sample datase		The method has been conventionally utilized in social sciences such as marketing and has not previously been implemented for use in metabolomics or metabonomics.
29.	Mobile Agent Based Market Basket Analysis on Cloud	apriori algorithm	transactional sample datase	Accuracy-75%	In future works we consider using more automation in the application by providing

					information without registration process and whole transaction will happen on the mobile number provided. We can integrate the routing of GPS to provide direction and distance measurement between shop and customer.
30.	Market basket analysis of crash data from large jurisdictions and its potential as a decision support tool	Apriori algorithm	non-intersection crash data	Accuracy-75%	These investigations are necessary for developing decision support tools based on association rule mining. As with the market basket analysis in the retail sector where it is up to the data owners to re-shelve their items based on the results, it would be up to the agencies to act on these broad patterns discovered from the data to develop policy initiatives and/or specific solutions for reduction in injuries and fatalities on roadways.

Gap Identified:

For all three of these algorithms, predefined minimum support is needed to satisfy for identifying the frequent itemsets. But when the minimum support is low, a huge number of candidate sets will be generated which requires large computation. In this project, we plan to follow an approach that has been proposed to avoid this large computation by reducing the items in the dataset with top selling products.

Existing Systems:

1. Online Store Product Recommendation System
2. Discovering Region-Based Association Rule in the IoT Environment

Datasets Description & Sample data

Data Set Information:

This data is taken from : https://www.kaggle.com/datasets

The data involved in any sale transaction in e-commerce websites such as amazon, such as data of items purchased, time of purchase, total sales volume, item price. E-commerce companies require additional data for managers to make strategic decisions that can increase company profits, such as the most sold product information, slightly sold products, and rarely sold products. To maintain inventory, it is essential to know the pattern of consumer spending that often occurs at these websites by analyzing the data of sales transactions. The placement of the product layout is still less accurate and optimal because it is only based on management's perception by categorizing the existing products and has not been reviewed from the consumer's point of view. So that, the researcher's initiative to try to provide solutions in the placement of the product layout.

Sample Data:

Dataset Link :

https://docs.google.com/spreadsheets/d/1OJsJu58wObv9py9m6_COXT9_4gWlAG44/edit?usp=sharing&ouid=107248981638858011160&rtpof=true&sd=true

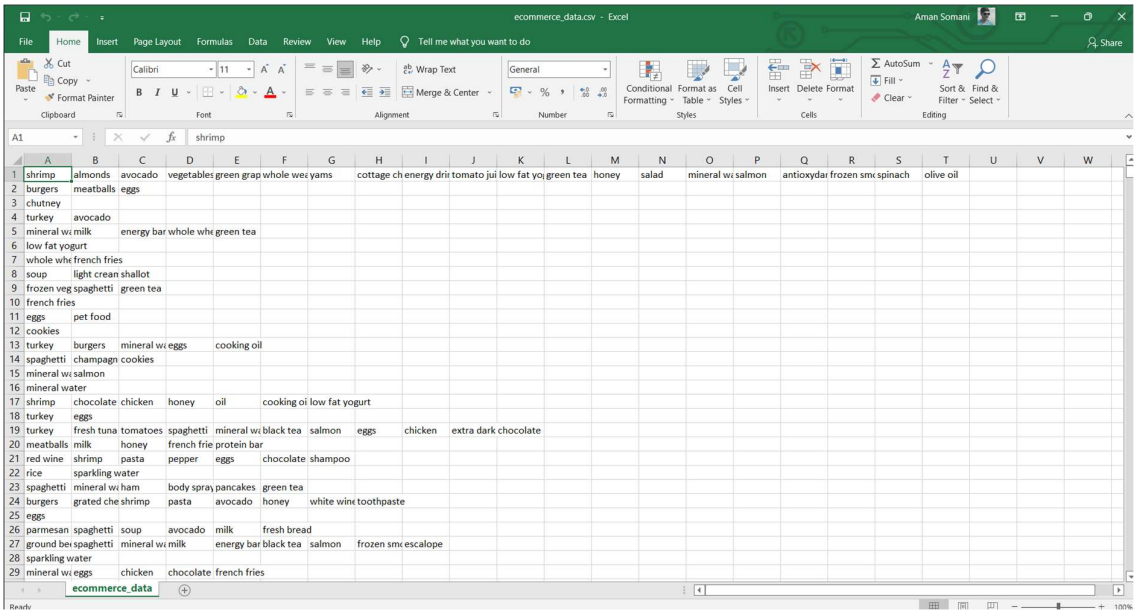


Fig1.

Proposed Algorithms With Flowchart:

Initially the dataset is taken as input and then the pre-processing of the data in the dataset takes place. Then, two different steps occur parallelly:-
1. Apirori, FP growth and Eclat algorithms are applied to the interested section of the dataset.
2. Before applying the mining algorithms the number of entries in the existing dataset is reduced by checking which of them are top selling products. After this is done the algorithms are applied on the resulting reduced dataset.
Once both the above results are obtained the results are compared and then analyzed.

Apriori algorithm:

Association rule mining finds interesting associations and/or correlation relationships among large set of data items. Association rules shows attribute value conditions that occur frequently together in a given dataset. A typical and widely used example of association rule mining is Market Basket Analysis. For example, data are collected from the supermarkets. Such market basket databases consist of a large number of transaction records. Each record lists all items bought by a customer on a single purchase transaction. Association rules provide information of this type in the form of “IF-THEN” statements. The rules are computed from the data, an association rule has two numbers that express the degree of uncertainty about the rule.

FP-GROWTH (Frequent Pattern Growth):

This algorithm is an improvement to the Apriori method. A frequent pattern is generated without the need for candidate generation. FP growth algorithm represents the database in the form of a tree called a frequent pattern tree or FPtree. This tree structure will maintain the association between the itemset. The database is fragmented using one frequent item. This fragmented part is called "pattern fragment". The itemset of these fragmented patterns are analyzed. Thus, with this method, the search for frequent itemset is reduced comparatively.

ECLAT Algorithm:

The ECLAT algorithm stands for Equivalence Class Clustering and bottom-up Lattice Traversal. It is one of the popular methods of Association Rule mining. It is a more efficient and scalable version of the Apriori algorithm. While the Apriori algorithm works in a horizontal sense imitating the Breadth-First Search of a graph, the ECLAT algorithm works in a vertical manner just like the Depth-First Search of a graph. This vertical approach of the ECLAT algorithm makes it a faster algorithm than the Apriori algorithm.

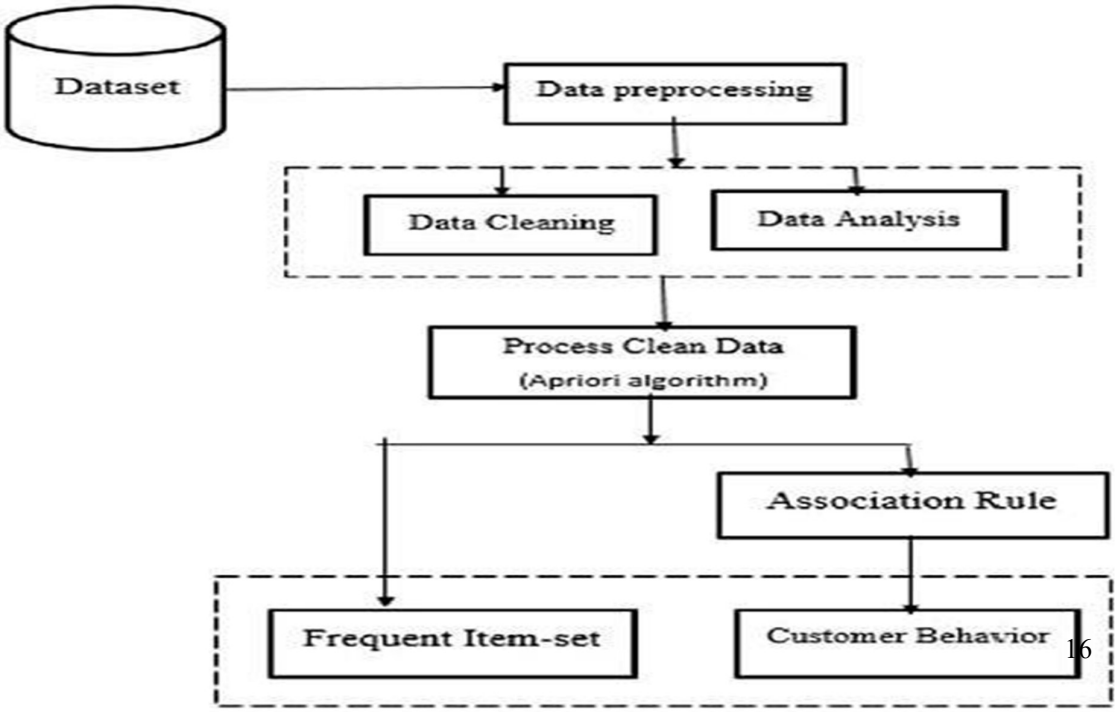


Fig2.

EXPERIMENTS RESULTS:

The above process generates Association Rules with various fixed metrics such as Support, Confidence, lift, etc. which are used to analyze retail basket or

transaction data. These metrics help us understand the strength of association between antecedent and consequent.

In our experiment we have made 2 Association Rule with the help of Apriori and FpGrowth and Eclat algorithm.

- In the first Association rule we have taken minimum support value of frequent itemset is 0.05 and confidence of 0.3. For an example when antecedent is chocolate and consequent is mineral water then that means that 30% of the transactions containing chocolate and also contain mineral water.
- In the Second Association rule we have taken minimum support value of frequent item set is 0.05 and minimum lift value of 1.3. If the lift value is greater than 1that means the probability of occurrence of the antecedent and that of the consequent are greater.

Implementation :

We have implemented the code of Apriori ,FP-growth and ECLAT algorithms in Jupyter Notebook.

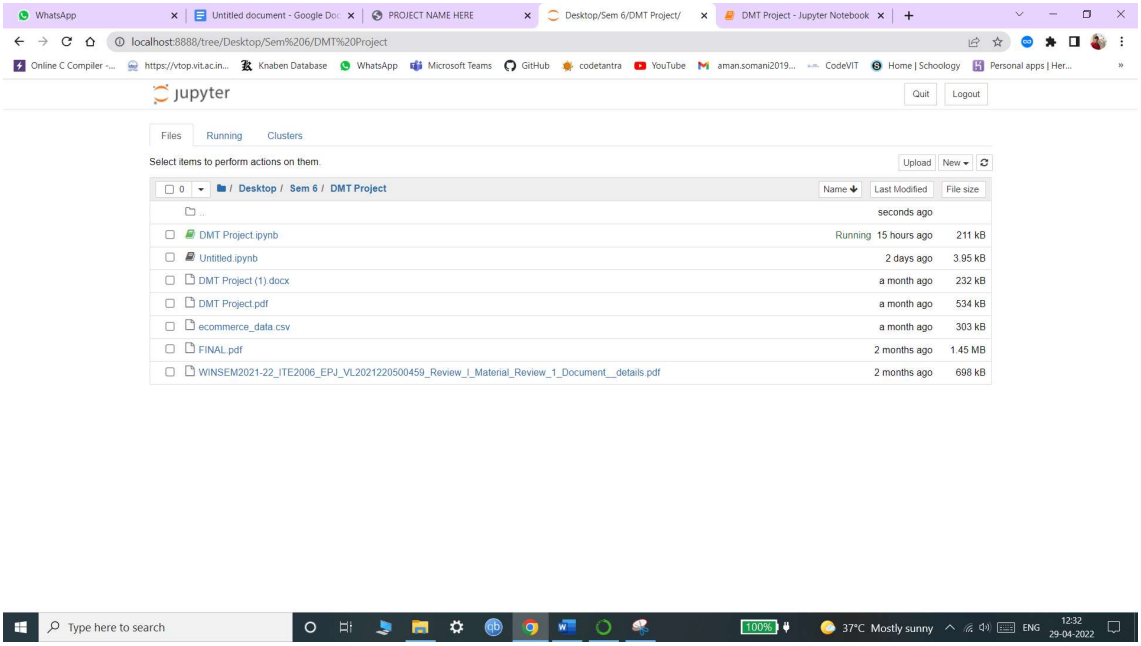


Fig3.

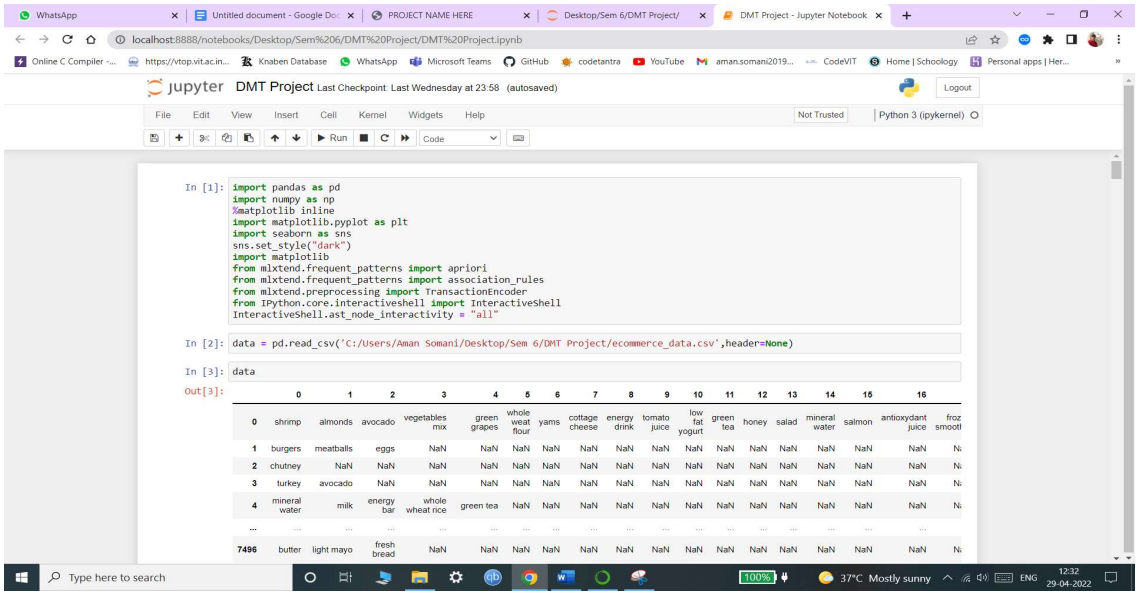


Fig4.

Code Output Screenshots:

Importing Libraries:

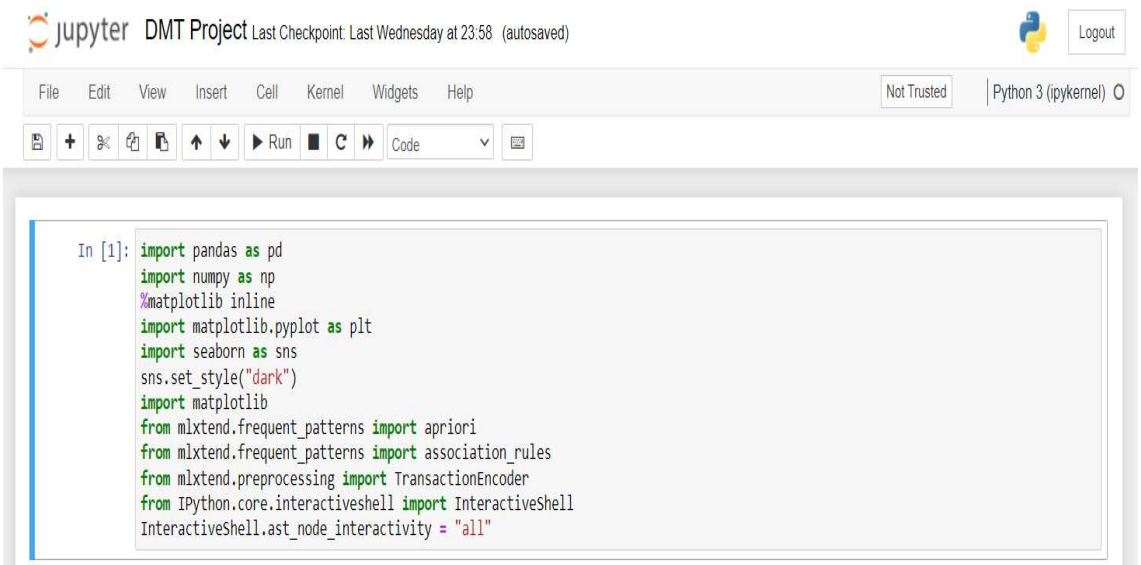


Fig5.

Reading Datasets:

```
In [2]: data = pd.read_csv('C:/Users/Aman Somani/Desktop/Sem 6/DMT Project/ecommerce_data.csv',header=None)

In [3]: data
```

Out[3]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
0	shrimp	almonds	avocado	vegetables mix	green grapes	whole weat flour	yams	cottage cheese	energy drink	tomato juice	low fat yogurt	green tea	honey	salad	mineral water	salmon	antioxydant juice	froz smoothi
1	burgers	meatballs	eggs	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
2	chutney	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
3	turkey	avocado	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
4	mineral water	milk	energy bar	whole wheat rice	green tea	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
7496	butter	light mayo	fresh bread	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
7497	burgers	frozen vegetables	eggs	french fries	magazines	green tea	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
7498	chicken	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
7499	escalope	green tea	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
7500	eggs	frozen smoothie	yogurt cake	low fat yogurt	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na

7501 rows × 20 columns

Fig6.

```
In [4]: data.shape

Out[4]: (7501, 20)

In [5]: data.head()
        data.tail()
```

Out[5]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	
0	shrimp	almonds	avocado	vegetables mix	green grapes	whole weat flour	yams	cottage cheese	energy drink	tomato juice	low fat yogurt	green tea	honey	salad	mineral water	salmon	antioxydant juice	frozen smoothie	spina
1	burgers	meatballs	eggs	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
2	chutney	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
3	turkey	avocado	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
4	mineral water	milk	energy bar	whole wheat rice	green tea	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na

Out[5]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
7496	butter	light mayo	fresh bread	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7497	burgers	frozen vegetables	eggs	french fries	magazines	green tea	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7498	chicken	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7499	escalope	green tea	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7500	eggs	frozen smoothie	yogurt cake	low fat yogurt	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Fig7.

Append the dataset values into an array:

```
In [6]: trans=[]
        for i in range(len(data)):
          trans.append([str(data.values[i,j]) for j in range(0,20)])
        trans=np.array(trans)
        print(trans.shape)

        (7501, 20)

In [7]: trans

Out[7]: array([['shrimp', 'almonds', 'avocado', ..., 'frozen smoothie',
                'spinach', 'olive oil'],
               ['burgers', 'meatballs', 'eggs', ..., 'nan', 'nan', 'nan'],
               ['chutney', 'nan', 'nan', ..., 'nan', 'nan', 'nan'],
               ...,
               ['chicken', 'nan', 'nan', ..., 'nan', 'nan', 'nan'],
               ['escalope', 'green tea', 'nan', ..., 'nan', 'nan', 'nan'],
               ['eggs', 'frozen smoothie', 'yogurt cake', ..., 'nan', 'nan',
                'nan']], dtype='<U20')

In [8]: t=TransactionEncoder()
        data=t.fit_transform(trans)
        data=pd.DataFrame(data,columns=t.columns_,dtype=int)
        data.shape

Out[8]: (7501, 121)
```

Fig8.

Drop the ‘nan’ values:

```
In [9]: data.drop('nan',axis=1,inplace=True)

In [10]: data.shape
         'nan' in data.columns

Out[10]: (7501, 120)

Out[10]: False

In [11]: data.head()
```

Out[11]:

	asparagus	almonds	antioxydant juice	asparagus	avocado	babies food	bacon	barbecue sauce	black tea	blueberries	...	turkey	vegetables mix	water spray	white wine	whole weat flour	whole wheat pasta	wh wh
0	0	1	1	0	1	0	0	0	0	0	...	0	1	0	0	1	0	
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	
3	0	0	0	0	1	0	0	0	0	0	...	1	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	

5 rows × 120 columns

Fig9.

Exploratory Data Analysis (EDA) of 20 top selling items:

```
In [12]: r=data.sum(axis=0).sort_values(ascending=False)[:20]
         plt.figure(figsize=(20,10))
         s=sns.barplot(x=r.index,y=r.values)
         s.set_xticklabels(s.get_xticklabels(), rotation=90)

Out[12]: <Figure size 1440x720 with 0 Axes>

Out[12]: [Text(0, 0, 'mineral water'),
          Text(1, 0, 'eggs'),
          Text(2, 0, 'spaghetti'),
          Text(3, 0, 'french fries'),
          Text(4, 0, 'chocolate'),
          Text(5, 0, 'green tea'),
          Text(6, 0, 'milk'),
          Text(7, 0, 'ground beef'),
          Text(8, 0, 'frozen vegetables'),
          Text(9, 0, 'pancakes'),
          Text(10, 0, 'burgers'),
          Text(11, 0, 'cake'),
          Text(12, 0, 'cookies'),
          Text(13, 0, 'escalope'),
          Text(14, 0, 'low fat yogurt'),
          Text(15, 0, 'shrimp'),
          Text(16, 0, 'tomatoes'),
          Text(17, 0, 'olive oil'),
          Text(18, 0, 'frozen smoothie'),
          Text(19, 0, 'turkey')]
```

Fig9.

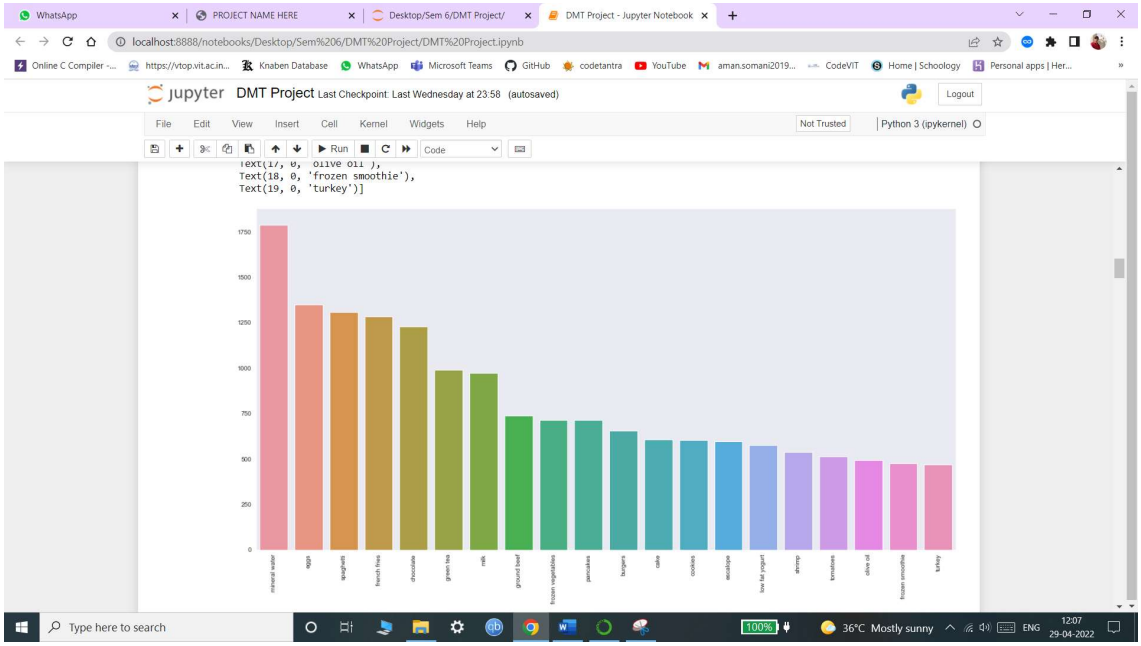


Fig10.

Tree Map of the 20 most selling items:

```
In [13]: import squarify

In [14]: my_values=r.values
         cmap = matplotlib.cm.Blues
         mini=min(my_values)
         maxi=max(my_values)
         norm = matplotlib.colors.Normalize(vmin=mini, vmax=maxi)
         colors = [cmap(norm(value)) for value in my_values]
         plt.figure(figsize=(10,10))
         squarify.plot(sizes=r.values, label=r.index, alpha=.7,color=colors)
         plt.title("Tree map of top 20 items")
         plt.axis('off')

Out[14]: <Figure size 720x720 with 0 Axes>

Out[14]: <AxesSubplot:>

Out[14]: Text(0.5, 1.0, 'Tree map of top 20 items')

Out[14]: (0.0, 100.0, 0.0, 100.0)
```

Fig11.

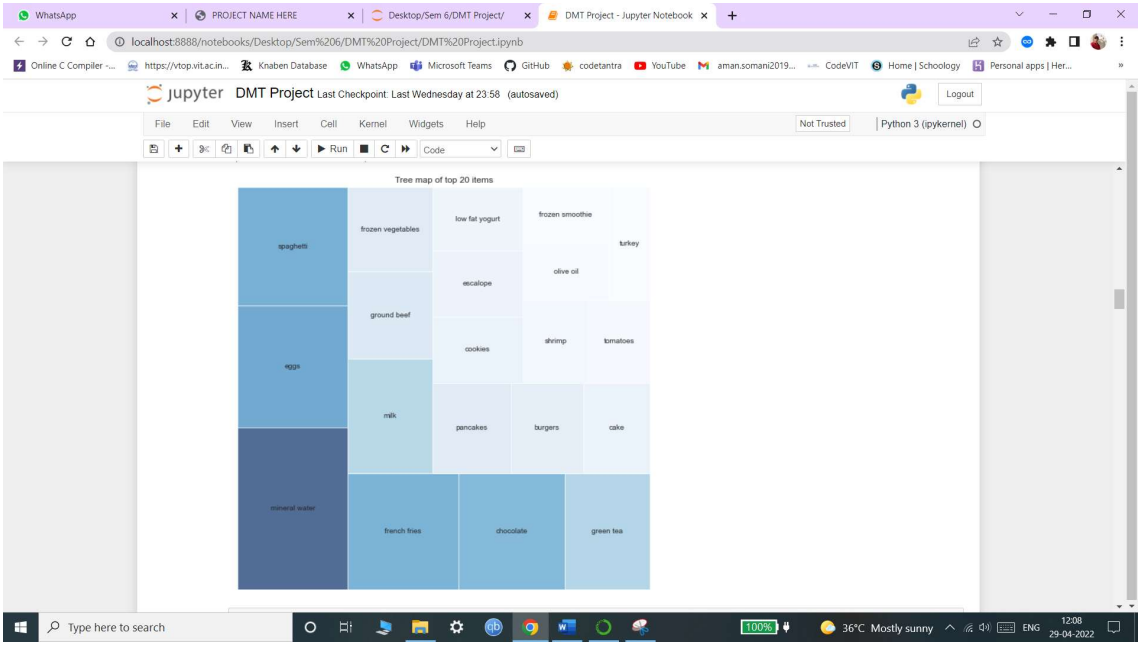


Fig12.

Making frequent items lists using Apriori where minimum support= 0.05:

```
In [15]: freq_items=apriori(data,min_support=0.05,use_colnames=True)
```

```
In [16]: freq_items
```

Out[16]:

	support	itemsets
0	0.087188	(burgers)
1	0.081056	(cake)
2	0.059992	(chicken)
3	0.163845	(chocolate)
4	0.080389	(cookies)
5	0.051060	(cooking oil)
6	0.179709	(eggs)
7	0.079323	(escalope)
8	0.170911	(french fries)
9	0.063325	(frozen smoothie)
10	0.095321	(frozen vegetables)
11	0.052393	(grated cheese)
12	0.132116	(green tea)
13	0.098254	(ground beef)
14	0.076523	(low fat yogurt)
15	0.129583	(milk)
16	0.238368	(mineral water)
17	0.065858	(olive oil)
18	0.095054	(pancakes)

Fig13.

Association rule1 of the freq_items where min_threshold for lift =1.3 :

```
In [17]: res=association_rules(freq_items,metric="lift",min_threshold=1.3)
```

```
In [18]: res
```

Out[18]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(chocolate)	(mineral water)	0.163845	0.238368	0.052660	0.321400	1.348332	0.013604	1.122357
1	(mineral water)	(chocolate)	0.238368	0.163845	0.052660	0.220917	1.348332	0.013604	1.073256
2	(spaghetti)	(mineral water)	0.174110	0.238368	0.059725	0.343032	1.439085	0.018223	1.159314
3	(mineral water)	(spaghetti)	0.238368	0.174110	0.059725	0.250559	1.439085	0.018223	1.102008

Fig14.

Association rule of the freq_items where min_support= 0.05 and min_confidence =0.3:

```
In [19]: res1=association_rules(freq_items,metric="confidence",min_threshold=0.3)
         res1
```

Out[19]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(chocolate)	(mineral water)	0.163845	0.238368	0.052660	0.321400	1.348332	0.013604	1.122357
1	(spaghetti)	(mineral water)	0.174110	0.238368	0.059725	0.343032	1.439085	0.018223	1.159314

Fig15.

Association rule3 of the freq_items where some selected attributes are displayed:

```
In [20]: res2 = res[['antecedents','consequents','support','confidence','lift']]
         res2
```

Out[20]:

	antecedents	consequents	support	confidence	lift
0	(chocolate)	(mineral water)	0.052660	0.321400	1.348332
1	(mineral water)	(chocolate)	0.052660	0.220917	1.348332
2	(spaghetti)	(mineral water)	0.059725	0.343032	1.439085
3	(mineral water)	(spaghetti)	0.059725	0.250559	1.439085

Fig16.

Listing frequent_itemsets where minimum support = 0.05:

```
In [22]: frequent_itemsets = apriori(data, min_support = 0.05, use_colnames=True)
         frequent_itemsets['length'] = frequent_itemsets['itemsets'].apply(lambda x: len(x))
         frequent_itemsets
```

Out[22]:

	support	itemsets	length
0	0.087188	(burgers)	1
1	0.081056	(cake)	1
2	0.059992	(chicken)	1
3	0.163845	(chocolate)	1
4	0.080389	(cookies)	1
5	0.051060	(cooking oil)	1
6	0.179709	(eggs)	1
7	0.079323	(escalope)	1
8	0.170911	(french fries)	1
9	0.063325	(frozen smoothie)	1
10	0.095321	(frozen vegetables)	1
11	0.052393	(grated cheese)	1
12	0.132116	(green tea)	1
13	0.098254	(ground beef)	1
14	0.076523	(low fat yogurt)	1
15	0.129583	(milk)	1
16	0.238368	(mineral water)	1
17	0.065858	(olive oil)	1
18	0.095054	(pancakes)	1

Fig17.

Listing items where length of item sets are 2 and support is greater than 0.01 :

```
In [23]: frequent_itemsets[ (frequent_itemsets['length'] == 2) &
         (frequent_itemsets['support'] >= 0.05) ]
```

Out[23]:

	support	itemsets	length
25	0.052660	(chocolate, mineral water)	2
26	0.050927	(mineral water, eggs)	2
27	0.059725	(spaghetti, mineral water)	2

Fig18.

Listing items where length of item set is 1 and support is greater than 0.01 :

```
In [24]: frequent_itemsets[ (frequent_itemsets['length'] == 1) &
         (frequent_itemsets['support'] >= 0.05) ]
```

Out[24]:

	support	itemsets	length
0	0.087188	(burgers)	1
1	0.081056	(cake)	1
2	0.059992	(chicken)	1
3	0.163845	(chocolate)	1
4	0.080389	(cookies)	1
5	0.051060	(cooking oil)	1
6	0.179709	(eggs)	1
7	0.079323	(escalope)	1
8	0.170911	(french fries)	1
9	0.063325	(frozen smoothie)	1
10	0.095321	(frozen vegetables)	1
11	0.052393	(grated cheese)	1
12	0.132116	(green tea)	1
13	0.098254	(ground beef)	1
14	0.076523	(low fat yogurt)	1
15	0.129583	(milk)	1
16	0.238368	(mineral water)	1
17	0.065858	(olive oil)	1
18	0.095054	(pancakes)	1
19	0.071457	(shrimp)	1
20	0.050527	(soup)	1
21	0.174110	(spaghetti)	1
22	0.068391	(tomatoes)	1
23	0.062525	(turkey)	1
24	0.058526	(whole wheat rice)	1

Fig19.

Importing fpgrowth from mlxtend.frequent_patterns:

```
In [25]: from mlxtend.frequent_patterns import fpgrowth
```

Fig20.

Making frequent item sets using fpgrowth where minimum support= 0.05:

```
In [26]: freq_items=fpgrowth(data,min_support=0.05,use_colnames=True)
```

```
In [27]: freq_items
```

Out[27]:

	support	itemsets
0	0.238368	(mineral water)
1	0.132116	(green tea)
2	0.076523	(low fat yogurt)
3	0.071457	(shrimp)
4	0.065858	(olive oil)
5	0.063325	(frozen smoothie)
6	0.179709	(eggs)
7	0.087188	(burgers)
8	0.062525	(turkey)
9	0.129583	(milk)
10	0.058526	(whole wheat rice)
11	0.170911	(french fries)
12	0.050527	(soup)
13	0.174110	(spaghetti)
14	0.095321	(frozen vegetables)
15	0.080389	(cookies)
16	0.051060	(cooking oil)
17	0.163845	(chocolate)

Fig21.

Association rules of the freq_items where min_threshold for lift is 1 and confidence =0.3 :

```
In [28]: res=association_rules(freq_items,metric="lift",min_threshold=1)
```

```
In [29]: res
```

Out[29]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(mineral water)	(eggs)	0.238368	0.179709	0.050927	0.213647	1.188845	0.008090	1.043158
1	(eggs)	(mineral water)	0.179709	0.238368	0.050927	0.283383	1.188845	0.008090	1.062815
2	(spaghetti)	(mineral water)	0.174110	0.238368	0.059725	0.343032	1.439085	0.018223	1.159314
3	(mineral water)	(spaghetti)	0.238368	0.174110	0.059725	0.250559	1.439085	0.018223	1.102008
4	(chocolate)	(mineral water)	0.163845	0.238368	0.052660	0.321400	1.348332	0.013604	1.122357
5	(mineral water)	(chocolate)	0.238368	0.163845	0.052660	0.220917	1.348332	0.013604	1.073256

```
In [30]: res1 = association_rules(freq_items,metric="confidence",min_threshold=0.3)
         res1
```

Out[30]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(spaghetti)	(mineral water)	0.174110	0.238368	0.059725	0.343032	1.439085	0.018223	1.159314
1	(chocolate)	(mineral water)	0.163845	0.238368	0.052660	0.321400	1.348332	0.013604	1.122357

Fig22.

Comparison of RunTime between Apriori and FPGrowth using Barplot and Lineplot:

In [31]:
```
import time
l=[0.01,0.02]
t=[]
for i in l:
    t1=time.time()
    apriori(data,min_support=i,use_colnames=True)
    t2=time.time()
    t.append((t2-t1)*1000)
```

Out[31]:

	support	itemsets
0	0.020397	(almonds)
1	0.033329	(avocado)
2	0.010799	(barbecue sauce)
3	0.014265	(black tea)
4	0.011465	(body spray)
...	...	...
252	0.011065	(ground beef, milk, mineral water)
253	0.017064	(ground beef, spaghetti, mineral water)
254	0.015731	(spaghetti, milk, mineral water)
255	0.010265	(spaghetti, olive oil, mineral water)
256	0.011465	(pancakes, spaghetti, mineral water)

257 rows × 2 columns

Fig23.

Out[31]:

	support	itemsets
0	0.020397	(almonds)
1	0.033329	(avocado)
2	0.033729	(brownies)
3	0.087188	(burgers)
4	0.030129	(butter)
...	...	...
98	0.020131	(whole wheat rice, mineral water)
99	0.022930	(olive oil, spaghetti)
100	0.025197	(pancakes, spaghetti)
101	0.021197	(spaghetti, shrimp)
102	0.020931	(tomatoes, spaghetti)

103 rows × 2 columns

Fig24.

In [32]:
```
l=[0.01,0.02]
f=[]
for i in l:
    t1=time.time()
    fpgrowth(data,min_support=i,use_colnames=True)
    t2=time.time()
    f.append((t2-t1)*1000)
```

Out[32]:

	support	itemsets
0	0.238368	(mineral water)
1	0.132116	(green tea)
2	0.076523	(low fat yogurt)
3	0.071457	(shrimp)
4	0.065858	(olive oil)
...	...	...
252	0.011465	(cake, burgers)
253	0.014131	(cake, green tea)
254	0.010265	(cake, frozen vegetables)
255	0.011865	(cake, pancakes)
256	0.010265	(cereals, mineral water)

257 rows × 2 columns

Out[32]:

	support	itemsets
0	0.238368	(mineral water)
1	0.132116	(green tea)

Fig25.

Barplot :

In [33]:
```
sns.barplot(x=l,y=f,label="fpgrowth")
sns.barplot(x=l,y=t,label="apriori")
plt.xlabel("Min_support Threshold")
plt.ylabel("Run Time in ms")
```

Out[33]: <AxesSubplot:>

Out[33]: <AxesSubplot:>

Out[33]: Text(0.5, 0, 'Min_support Threshold')

Out[33]: Text(0, 0.5, 'Run Time in ms')

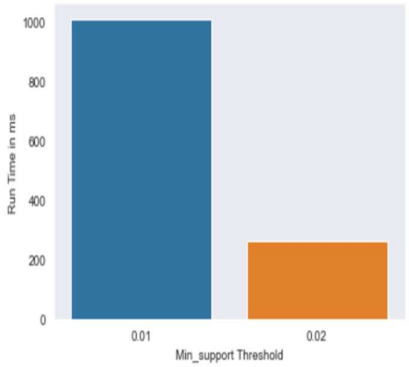


Fig26.

Lineplot :

```
In [34]: sns.lineplot(x=l,y=f,label="fpgrowth")
         sns.lineplot(x=l,y=t,label="apriori")
         plt.xlabel("Min_support Threshold")
         plt.ylabel("Run Time in ms")
```

```
Out[34]: <AxesSubplot:>

Out[34]: <AxesSubplot:>

Out[34]: Text(0.5, 0, 'Min_support Threshold')

Out[34]: Text(0, 0.5, 'Run Time in ms')
```

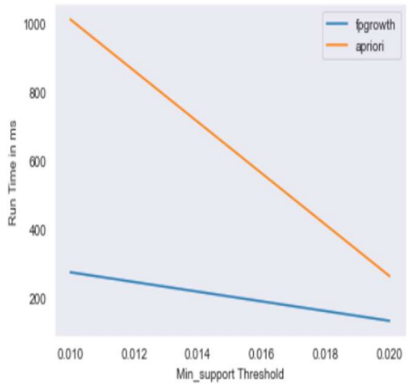


Fig27.

Eclat Algorithm :

Importing the required modules and dataset . After this training the dataset:

```
In [35]: #install ayori

In [36]: pip install apyori

         Requirement already satisfied: apyori in c:\users\aman somani\documents\anaconda\lib\site-packages (1.1.2)
         Note: you may need to restart the kernel to use updated packages.

In [37]: #import modules
         import numpy as np
         import matplotlib.pyplot as plt
         import pandas as pd

In [38]: #import dataset
         dataset = pd.read_csv('C:/Users/Aman Somani/Desktop/Sem 6/DMT Project/ecommerce_data.csv', header = None)
         transactions = []
         for i in range(0, 7501):
           transactions.append([str(dataset.values[i,j]) for j in range(0, 20)])

In [39]: #train eclat model
         from apyori import apriori
         rules = apriori(transactions = transactions, min_support = 0.003, min_confidence = 0.2, min_lift = 3, min_length = 2, max_length
```

Fig28.

Visualising and organising Data :

```
In [40]: #visualising data
         results = list(rules)
         print(results)

         [RelationRecord(items=frozenset({'chicken', 'light cream'}), support=0.004532728969470737, ordered_statistics=[OrderedStatistic
         (items_base=frozenset({'light cream'}), items_add=frozenset({'chicken'}), confidence=0.29059829059829057, lift=4.8439506172839
         5)]), RelationRecord(items=frozenset({'escalope', 'mushroom cream sauce'}), support=0.005732568990801226, ordered_statistics=[O
         rderedStatistic(items_base=frozenset({'mushroom cream sauce'}), items_add=frozenset({'escalope'}), confidence=0.300699300699300
         7, lift=3.790832696715049)]), RelationRecord(items=frozenset({'escalope', 'pasta'}), support=0.005865884548726837, ordered_stat
         istics=[OrderedStatistic(items_base=frozenset({'pasta'}), items_add=frozenset({'escalope'}), confidence=0.3728813559322034, lif
         t=4.700811850163794)]), RelationRecord(items=frozenset({'fromage blanc', 'honey'}), support=0.003332888948140248, ordered_stati
         stics=[OrderedStatistic(items_base=frozenset({'fromage blanc'}), items_add=frozenset({'honey'}), confidence=0.2450980392156863,
         lift=5.164270764485569)]), RelationRecord(items=frozenset({'ground beef', 'herb & pepper'}), support=0.015997866951073192, orde
         red_statistics=[OrderedStatistic(items_base=frozenset({'herb & pepper'}), items_add=frozenset({'ground beef'}), confidence=0.32
         34501347708895, lift=3.291993841349285)]), RelationRecord(items=frozenset({'tomato sauce', 'ground beef'}), support=0.00533262
         2317024397, ordered_statistics=[OrderedStatistic(items_base=frozenset({'tomato sauce'}), items_add=frozenset({'ground beef'}),
         confidence=0.3773584905660377, lift=3.840659481324083)]), RelationRecord(items=frozenset({'olive oil', 'light cream'}), support
         =0.003199573390214638, ordered_statistics=[OrderedStatistic(items_base=frozenset({'light cream'}), items_add=frozenset({'olive
         oil'}), confidence=0.20512820512820515, lift=3.1147098515519573)]), RelationRecord(items=frozenset({'whole wheat pasta', 'olive
         oil'}), support=0.007998933475536596, ordered_statistics=[OrderedStatistic(items_base=frozenset({'whole wheat pasta'}), items_a
         dd=frozenset({'olive oil'}), confidence=0.2714932126696833, lift=4.122410097642296)]), RelationRecord(items=frozenset({'pasta',
         'shrimp'}), support=0.005065991201173177, ordered_statistics=[OrderedStatistic(items_base=frozenset({'pasta'}), items_add=froze
         nset({'shrimp'}), confidence=0.3220338983050847, lift=4.506672147735896)])]

In [41]: #organising data
         def inspect(results):
             lhs         = [tuple(result[2][0][0])[0] for result in results]
             rhs         = [tuple(result[2][0][1])[0] for result in results]
             supports    = [result[1] for result in results]
             return list(zip(lhs, rhs, supports))
         resultsinDataFrame = pd.DataFrame(inspect(results), columns = ['Product 1', 'Product 2', 'Support'])
```

Fig29.

Observing the results obtained on the dataset:

```
In [42]: print(resultsinDataFrame)
```

```
              Product 1    Product 2   Support
0          light cream      chicken  0.004533
1  mushroom cream sauce     escalope  0.005733
2                pasta     escalope  0.005866
3        fromage blanc        honey  0.003333
4        herb & pepper  ground beef  0.015998
5         tomato sauce  ground beef  0.005333
6          light cream    olive oil  0.003200
7    whole wheat pasta    olive oil  0.007999
8                pasta       shrimp  0.005066
```

```
In [43]: #sorting data on the basis of support
         resultsinDataFrame.nlargest(n = 10, columns = 'Support')
         print(resultsinDataFrame)
```

Out[43]:

	Product 1	Product 2	Support
4	herb & pepper	ground beef	0.015998
7	whole wheat pasta	olive oil	0.007999
2	pasta	escalope	0.005866
1	mushroom cream sauce	escalope	0.005733
5	tomato sauce	ground beef	0.005333
8	pasta	shrimp	0.005066
0	light cream	chicken	0.004533
3	fromage blanc	honey	0.003333
6	light cream	olive oil	0.003200

```
              Product 1    Product 2   Support
0          light cream      chicken  0.004533
```

Fig30.

Code :

In [1]:
import pandas as pd
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style("dark")
import matplotlib
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
from mlxtend.preprocessing import TransactionEncoder
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
In [2]:
data = pd.read_csv(C:/Users/Aman Somani/Desktop/Sem 6/DMT Project/ecommerce_data.csv',header=None)
In [3]:
data
In [4]:
data.shape
In [5]:
data.head()
data.tail()
In [6]:
trans=[]

```
for i in range(len(data)):
 trans.append([str(data.values[i,j]) for j in range(0,20)])
trans=np.array(trans)
print(trans.shape)
In [7]:
trans
In [8]:
t=TransactionEncoder()
data=t.fit_transform(trans)
data=pd.DataFrame(data,columns=t.columns_,dtype=int)
data.shape
In [9]:
data.drop('nan',axis=1,inplace=True)
In [10]:
data.shape
'nan' in data.columns
In [11]:
data.head()
In [12]:
r=data.sum(axis=0).sort_values(ascending=False)[:20]
plt.figure(figsize=(20,10))
s=sns.barplot(x=r.index,y=r.values)
s.set_xticklabels(s.get_xticklabels(), rotation=90)
In [13]:
import squarify
In [14]:
my_values=r.values
cmap = matplotlib.cm.Blues
mini=min(my_values)
maxi=max(my_values)
norm = matplotlib.colors.Normalize(vmin=mini, vmax=maxi)
colors = [cmap(norm(value)) for value in my_values]
plt.figure(figsize=(10,10))
squarify.plot(sizes=r.values, label=r.index, alpha=.7,color=colors)
plt.title("Tree map of top 20 items")
plt.axis('off')
In [15]:
freq_items=apriori(data,min_support=0.05,use_colnames=True)
In [16]:
freq_items
In [17]:
res=association_rules(freq_items,metric="lift",min_threshold=1.3)
In[18]:
res
In [19]:
res1=association_rules(freq_items,metric="confidence",min_threshold=0.3)
res1
In [20]:
res2 = res[['antecedents','consequents','support','confidence','lift']]
res2
```

In [21]:
```
res3= res[res['confidence']>=0.1]
res3
```
In [22]:
```
frequent_itemsets = apriori(data, min_support = 0.05, use_colnames=True)
frequent_itemsets['length'] = frequent_itemsets['itemsets'].apply(lambda x: len(x))
frequent_itemsets
```
In [23]:
```
frequent_itemsets[ (frequent_itemsets['length'] == 2) &
(frequent_itemsets['support'] >= 0.05) ]
```
In [24]:
```
frequent_itemsets[ (frequent_itemsets['length'] == 1) &
(frequent_itemsets['support'] >= 0.05) ]
```
In [25]:
```
from mlxtend.frequent_patterns import fpgrowth
```
In [26]:
```
freq_items=fpgrowth(data,min_support=0.05,use_colnames=True)
```
In [27]:
```
freq_items
```
In [28]:
```
res=association_rules(freq_items,metric="lift",min_threshold=1)
res
```
In [29]:
```
res1 = association_rules(freq_items,metric="confidence",min_threshold=0.3)
res1
```
In[30]:
```
import time
l=[0.01,0.02]
t=[]
for i in l:
 t1=time.time()
 apriori(data,min_support=i,use_colnames=True)
 t2=time.time()
 t.append((t2-t1)*1000)
```
In[31]:
```
l=[0.01,0.02]
f=[]
for i in l:
 t1=time.time()
 fpgrowth(data,min_support=i,use_colnames=True)
 t2=time.time()
 f.append((t2-t1)*1000)
```
In[32]:
```
sns.lineplot(x=l,y=f,label="fpgrowth")
sns.lineplot(x=l,y=t,label="apriori")
plt.xlabel("Min_support Threshold")
plt.ylabel("Run Time in ms")
```
In[33]:
```
sns.barplot(x=l,y=f,label="fpgrowth")
sns.barplot(x=l,y=t,label="apriori")
```

```
plt.xlabel("Min_support Threshold")
plt.ylabel("Run Time in ms")


#install ayori
#pip install apyori

#import modules
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

#import dataset
dataset = pd.read_csv(C:/Users/Aman Somani/Desktop/Sem 6/DMT
Project/ecommerce_data.csv', header = None)
transactions = []
for i in range(0, 7501):
  transactions.append([str(dataset.values[i,j]) for j in range(0, 20)])

#train eclat model
from apyori import apriori
rules = apriori(transactions = transactions, min_support = 0.003, min_confidence = 0.2,
min_lift = 3, min_length = 2, max_length = 2)

#visualising data
results = list(rules)
print(results)

#organising data
def inspect(results):
    lhs         = [tuple(result[2][0][0])[0] for result in results]
    rhs         = [tuple(result[2][0][1])[0] for result in results]
    supports    = [result[1] for result in results]
    return list(zip(lhs, rhs, supports))
resultsinDataFrame = pd.DataFrame(inspect(results), columns = ['Product 1', 'Product 2',
'Support'])

print(resultsinDataFrame)

#sorting data on the basis of support
resultsinDataFrame.nlargest(n = 10, columns = 'Support')
print(resultsinDataFrame)
```

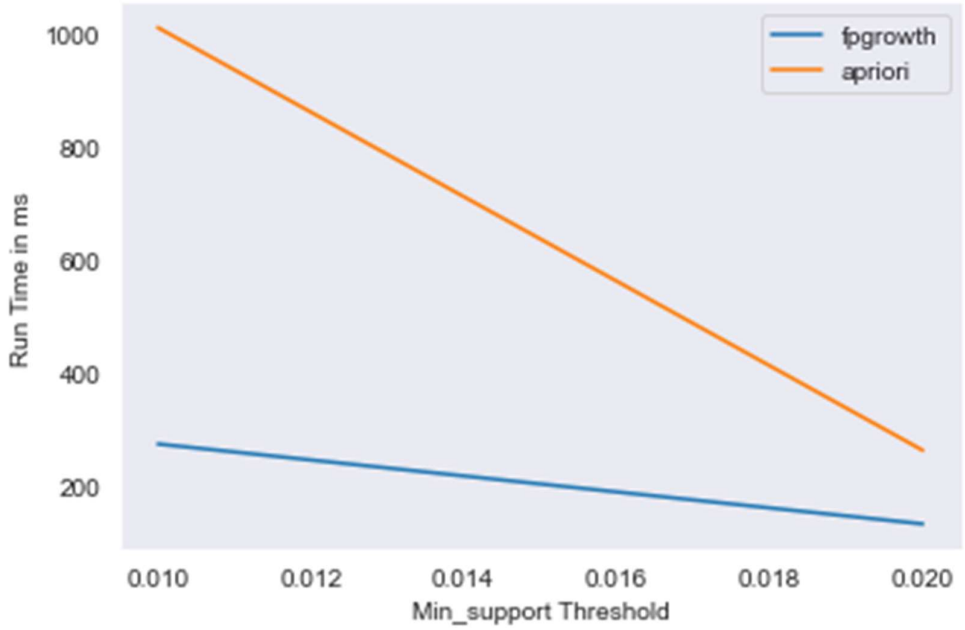
COMPARITIVE STUDY/RESULTS AND DISCUSSIONS:

Comparative Study: Apriori Vs FP-Growth Vs Eclat:

Apriori scans the dataset in each of its steps, so it becomes time-consuming for data where the number of items is larger.

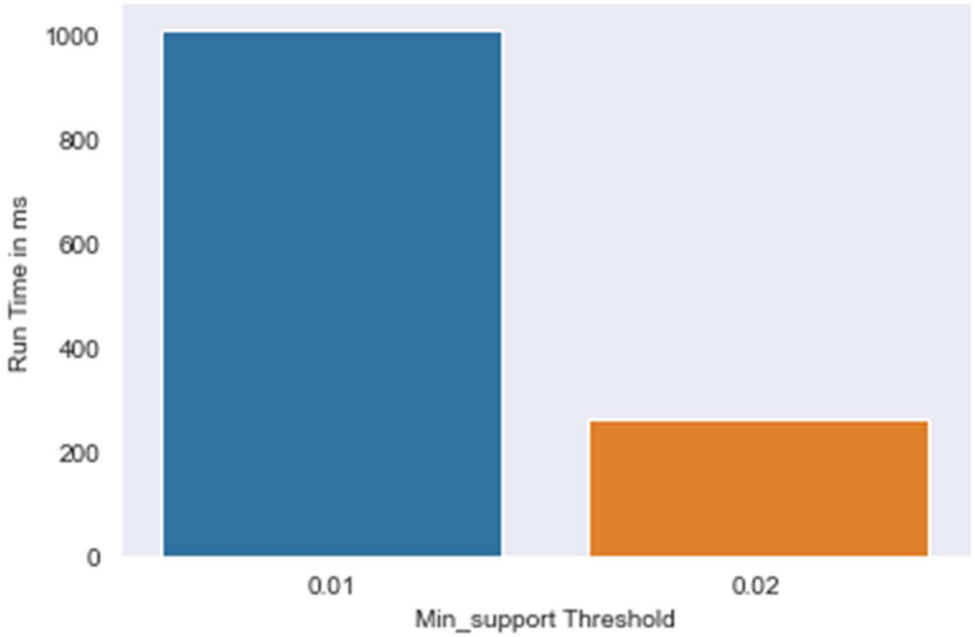
FP-Growth requires only one scan of the dataset in its beginning steps so it consumes lesstime.

The ECLAT algorithm does not involve the repeated scanning of the data to compute the individual support values.



Line plot for Apriori Vs FP-Growth

Fig31.



Bar plot for Apriori Vs FP-Growth

Fig32.

As we can see clearly in the diagram that the FP growth algorithm takes much lessertime than Apriori.

Results: Association Rules are generated for Apriori and FPgrowth algorithm forantecedent and consequent items in our dataset.

Association rule of the freq_items where min_support= 0.05 and min_confidence =0.3:

```
In [19]: res1=association_rules(freq_items,metric="confidence",min_threshold=0.3)
         res1
```

Out[19]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(chocolate)	(mineral water)	0.163845	0.238368	0.052660	0.321400	1.348332	0.013604	1.122357
1	(spaghetti)	(mineral water)	0.174110	0.238368	0.059725	0.343032	1.439085	0.018223	1.159314

Fig 33.

Association rule of the freq_items where min_support= 0.05 and min_threshold for lift =1.3:

```
In [17]: res=association_rules(freq_items,metric="lift",min_threshold=1.3)

In [18]: res
```

Out[18]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(chocolate)	(mineral water)	0.163845	0.238368	0.052660	0.321400	1.348332	0.013604	1.122357
1	(mineral water)	(chocolate)	0.238368	0.163845	0.052660	0.220917	1.348332	0.013604	1.073256
2	(spaghetti)	(mineral water)	0.174110	0.238368	0.059725	0.343032	1.439085	0.018223	1.159314
3	(mineral water)	(spaghetti)	0.238368	0.174110	0.059725	0.250559	1.439085	0.018223	1.102008

Fig34.

Tree Map of the 20 most selling items:

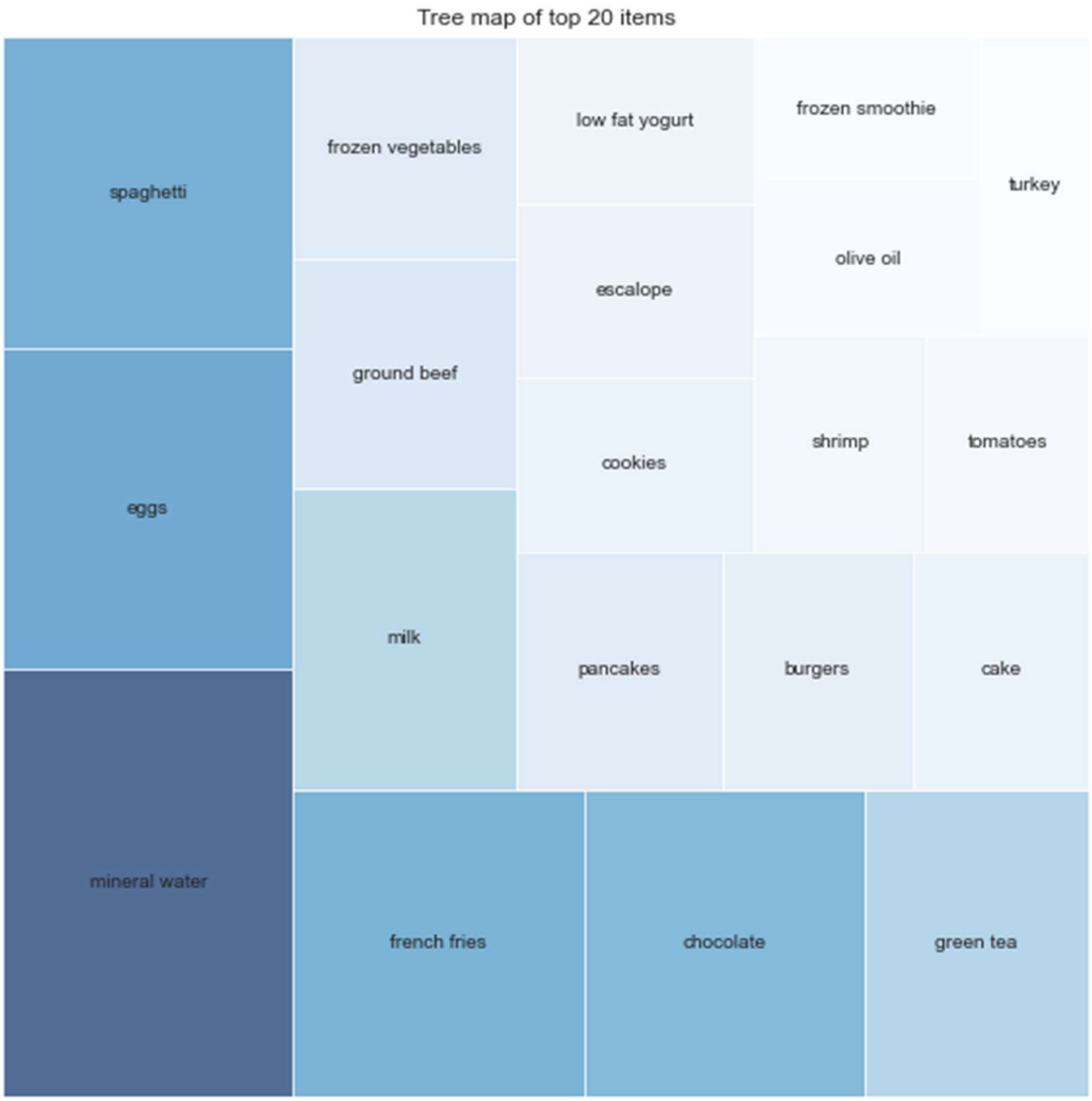


Fig 35.

Bar Plot of 20 top selling items in our dataset:

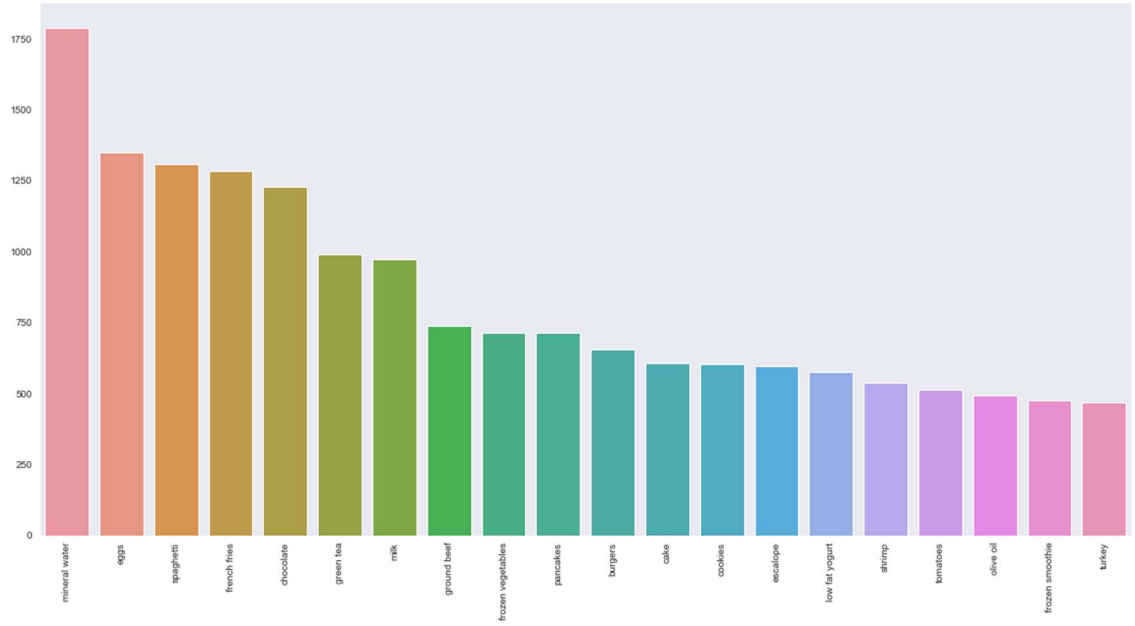


Fig36.

CONCLUSION AND FUTURE WORK:

Conclusion: From the output above, we see that the top associations are not surprising,with one item for a specific purpose gets purchased along with another flavour required for the same purpose: E.g.: - Bread and Butter are strongly associated as they are consumed together. Similarly, Toothbrush and Toothpaste, Chocolate and Chips, Soap and Shampoo also have strong associations as they are frequently purchased together and used for similar purposes.

Future Work: In the future, once common application of association rules mining is in the domain of recommender systems and item pairs have been identified as having positive relationship, recommendations can be made to customers in order to increase sales. Also, there is a possibility of introducing customers to items they never would have tried before.

References:

https://www.sciencedirect.com/science/article/pii/S1877050916305208

https://www.researchgate.net/publication/228913454_An_Implementation_of_the_FP-growth_Algorithm

https://www.researchgate.net/publication/329515220_Performance_Analysis_of_Apriori_and_FP-Growth_Algorithms_Association_Rule_Mining

https://www.irjet.net/archives/V8/i5/IRJET-V8I5831.pdf
https://www.ijeat.org/wp-content/uploads/papers/v9i2/B3186129219.pdf

https://ieeexplore.ieee.org.egateway.vit.ac.in/document/9410737?arnumber=9410737

https://dl.acm.org.egateway.vit.ac.in/doi/10.1145/3377170.3377226

https://www.proquest.com/docview/1789980619?accountid=38207

https://www.researchgate.net/publication/342382244_Consumer_purchase_patterns_based_on_market_basket_analysis_using_apriori_algorithms

https://www.researchgate.net/publication/251422282_FPTree_Based_Algorithms_Analysis_FP_Growth_COFI-Tree_and_CT-PRO

https://www.researchgate.net/publication/352384736_Online_Shopping_Do_Men_Behave_Differently_than_Women

https://www.researchgate.net/publication/351112666_Uncovering_Modern_Clinical_Applications_of_Fuzi_and_Fuzi-Based_Formulas_A_Nationwide_Descriptive_Study_With_Market_Basket_Analysis

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.299.6609&rep=rep1&type=pdf

http://www.ijsred.com/volume2/issue2/IJSRED-V2I2P83.pdf

https://sci-hub.hkvisa.net/10.1109/ICAIBD49809.2020.9137436

https://arxiv.org/ftp/arxiv/papers/1403/1403.3948.pdf

https://osf.io/preprints/inarxiv/jpsfa/

https://ieeexplore.ieee.org.egateway.vit.ac.in/document/9689770

https://ieeexplore.ieee.org.egateway.vit.ac.in/document/8262592

https://www.proquest.com/docview/2571494191/fulltextPDF/76BE6BE78A804ADFPQ/1?accountid=38207

https://www.proquest.com.egateway.vit.ac.in/docview/1789980619/fulltextPDF/DB0BE13D8FDC4517PQ/1?accountid=38207

https://eds.p.ebscohost.com.egateway.vit.ac.in/eds/viewarticle/render?data=dGJyMPPp44rp2%2fdV0%2bnjisfk5Ie46bNPtquyS7Ok63nn5Kx94um%2bS62orUquqLA4trCyS7imtTi%2fw6SM8Nfsi9%2fZ8oHt5Od8u6ixUbasr1Gwr6SL59q7S%2bGtskiy3OFFt9q2fKuq5E6xo997r9erfeSrsH3h2OF9s6fgWPDb4oHxnOp57N27feyc4nq72%2bKL4%2bLhPvLX5VW%2fxKR57LPjfPHb6nusqKtNsKSuRbarrk%2bvqLdIs6qwPuTl8IXf6rt%2b8%2bLqjOPu8gAA&vid=2&sid=3c7404fc-9d8d-4f63-ac1a-ef52ecbce51b@redis

https://www.sciencedirect.com.egateway.vit.ac.in/science/article/pii/S0378437119302171#sec5

https://pubs.acs.org.egateway.vit.ac.in/doi/10.1021/acs.analchem.5b04182

https://ieeexplore.ieee.org.egateway.vit.ac.in/document/7033341

https://www.sciencedirect.com.egateway.vit.ac.in/science/article/pii/S0925753507001877?via%3Dihub#aep-abstract-id5