Prediction and recommendation of Grocery Products

J-Component Document

NILADRI MITRA 20BIT0381

Submitted to

Prof. Chiranji Lal Chowdhary, SITE

School of Information Technology and Engineering

VIT®

Vellore Institute of Technology

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

TABLE OF CONTENTS

1. ABSTRACT

Various departmental stores have seen their suffering because of the complexity and work overload they have to face. In order to decrease their suffering and increase their profit we are designing this smart grocery management system. We have added the recommender system here which will help the manager to know beforehand what actually a customer may buy. If for example there is a user who buys milk frequently, hence the store manager can keep a track and arrange the items in his grocery accordingly. We will be providing the store manager with the information about the recommendations of the grocery items for a particular customer depending on his previous purchase history.

Understanding that a grocery store has a highly complex working system, like managing items in a group with similar items and also placing those groups of items, etc. So, managing all these things requires a high workload which is not suitable for some grocers as they have to hire some people and hence, they earn less profit. This project can be used heavily in daily markets also because as this is a complete software-based product cost will also be less and even though will help in decreasing the complexity of grocery management system and increasing their profits. This can be easily applicable in almost all the groceries provided they have any software capabilities.

The dataset created for normal transactions purposes includes all grocery details like member id and their date wise transaction details including the items bought over the span of time period mentioned in the dataset. The Dataset used for the recommender system included all the patterns in which buyers buy in grocery.

We will use this dataset to perform the prediction of these frequent item sets and integrate it in a software application to make it easier for the store manager to access. Via the website application, the store manager can easily see a displayed list of frequent items a customer has bought with the help of their customer id. This will help him to be prepared in advance and thus help in keeping a check on the sales.

2. INTRODUCTION

Recommender systems are one of the most successful and widespread applications of artificial intelligence technologies in business. Recommendations are used for making the work of the customer easier and fast. This reduces their valuable time and also their efforts. For this the recommendations given to the customer should be exact and should be fast. And most importantly they should not irritate the customer. These recommendations are mostly given based on their necessity and their interest. Therefore the customer's necessity can be predicted from their purchase history and the customer's interest can be predicted from the people who have interest the same as that customer. In our project of grocery recommendation system, we are going to develop a recommendation system which will recommend the customer products of his interest and necessity.

First, we need to analyze the data from the customer grocery purchase dataset. We then find the products or categories that are most purchased by the customers. After processing the raw file, we get the required results. The next step is to compare the results with the most bought items per customer. This is an important phase as all further processing will depend on the results of this analysis.

Association rule is applied in this system to identify the next section that is in the basket. The main reason to select this algorithm instead of other machine learning algorithms is to give priority to the customer choice that can be identified using past transactions and baskets. If this algorithm fails to identify the next section using the two combinations of sections then 5 similar customers are selected and the next section is identified using their baskets and transactions. It is a technique to identify underlying relations between different items. Take the example of a Supermarket where customers can buy a variety of items. Usually, there is a pattern in what the customers buy. For instance, mothers with babies buy baby products such as milk and diapers.
More profit can be generated if the relationship between the items purchased in different transactions can be identified.

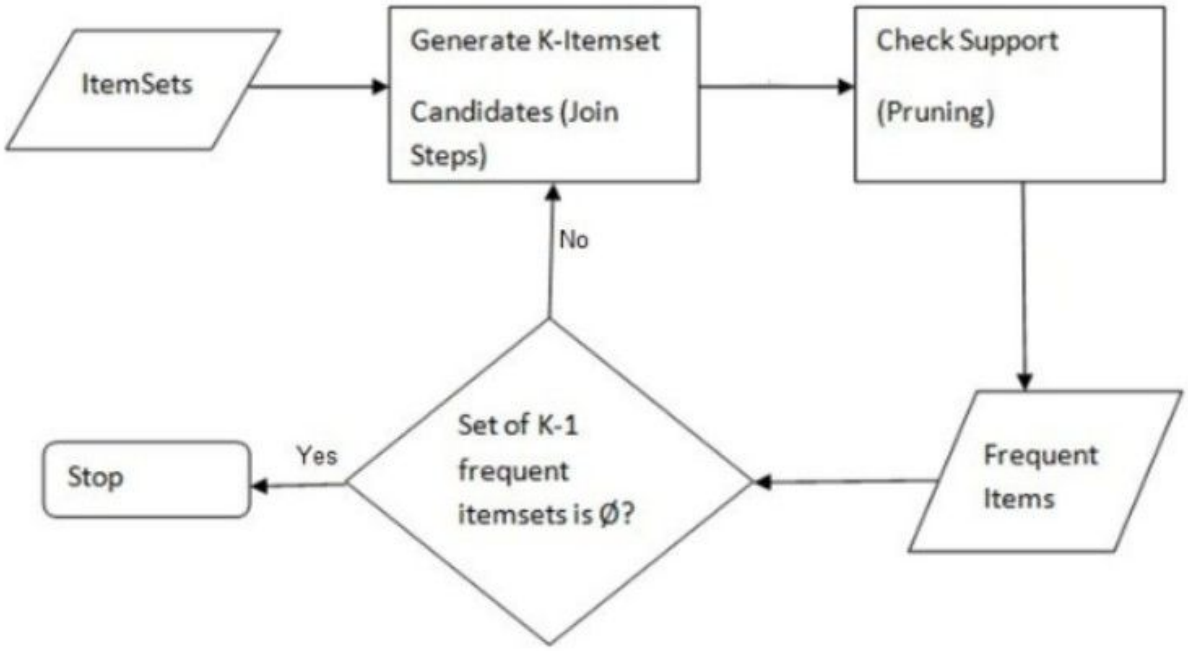
3. LITERATURE REVIEW SURVEY TABLE

Authors and Year	Title (Study)	Concept / Theoretical model/ Framework	Methodology used/ Implementation	Dataset details/ Analysis	Relevant Finding	Limitations/ Future Research/ Gaps identified
Jeff Heaton 2017	Comparing Dataset Characteristics that Favor the Apriori, Eclat or FP-Growth Frequent Itemset Mining Algorithms	This research evaluates the performance of the Apriori, Eclat and FP-Growth frequent itemset mining algorithms implemented by Christian Borgelt Though, association rule mining is a similar algorithm, this research is limited to frequent itemset mining. By limiting the experimentation to a single implementation of frequent itemset mining this research is able to evaluate how the characteristics of the dataset affect the performance of these algorithms.	The research paper evaluates various algorithms. First, it evaluates Frequent Itemset Mining. Next, Naive Algorithm was tested which used a small dataset of its own, to just compare the results from the previous frequent itemset. After this, the Apriori algorithm was implemented and the results were compared. Next its demonstrated how the Eclat algorithm would handle the basket set given earlier in this paper. Lastly the FP growth algorithm is evaluated, and	Generated datasets are used to perform this evaluation. This generated data allows the two independent variables to be adjusted to create a total of 20 different datasets to perform the evaluations. The dataset consists: •Transaction/Basket count: 10 million default • Number of items: 50,000 default • Number of frequent sets: 100 default • Max transaction/basket size: independent variable, 5-100 range	Apriori is an easily understandable frequent itemset mining algorithm. Because of this, Apriori is a popular starting point for frequent itemset study. However, Apriori has serious scalability issues and exhausts available memory a little faster than Eclat and FP-Growth. However, the results of Apriori tend to have more accuracy.	Some of the limitations and future scope of the research paper was to evaluate more algorithms to find frequent itemsets which would thus provide better evaluation, like using clustering, k-means, k nearest neighbors.

			all are compared empirically.	• Frequent set density: independent variable, 0.1 to 0.8 range		
S.Pandya, J.Shah, N.Joshi2,H. Ghayvat , S.C.Mukhopadhyay, M.H.Yap 2016	A Novel Hybrid based Recommendation System based on Clustering and Association Mining	This paper focuses mainly on building an accurate recommendation system that uses opinions about the community of users and determines content of interest using certain rules extractions. Recommendation systems are classified into 3 approaches which are collaborative, content-based or knowledge-based method to have a better recommendation	In this paper, the major challenges such as “data sparsity” and “cold start problem” are addressed. To overcome these challenges, it proposes a new methodology by combining the clustering algorithm with Eclat Algorithm for better rules generation. Firstly cluster the rating matrix based on the user similarity. Then convert the clustered data into Boolean data and applying the Eclat Algorithm on Boolean data efficient rules generation	• The dataset consists of profiles of the users that were created at the beginning. A profile has information about the user and his/her taste which is based on how a user rates the items. In the recommendation process, the engine compares items that were already rated by user with items that he/she did not rate and looks for	Here we have used a k-means clustering to cluster the user profiles and on the cluster data applying eclat algorithm to generate the best rules for recommending the items to the user. The experiments demonstrate that our method achieves the improvement in accuracy and also reduce the sparsity	The future work is to improve the accuracy to the system by enhancing the clustering algorithm.
JinHyun Jooa, SangWon Bangb, GeunDuk Parka 2016	Implementation of a Recommendation System using Association Rules and Collaborative Filtering	In this study, user data are implicitly collected via NFC (Near Field Communication). The advantage of NFC is that most smartphones have a built-in NFC functionality, and only tags are needed to use it. The use of NFC allows to improve inconvenience of	This study collects customer information implicitly and predicts seller with high possibility of customer visit through association rule analysis and collaborative filtering by using NFC. Also, the user’s preferred business that is the closest to their current location is recommended based on GPS data, and a personalized recommendation system that	This study supposed that the implemented mobile coupon recommendation system has been in use by customers for a long time. After using the A restaurant, or the business of the system, the customer collects mileage using the NFC functionality on their smartphone with the business’s NFC reader.	This study implemented the mobile coupon recommendation system, which recommends coupons to the user using association analysis and collaborative filtering based on the consumer usage patterns. In order to minimize user inconvenience, used data was collected with	Implementing extensive data mining techniques can be a future scope. Data mining is a method of finding hidden relationships, patterns and rules in massive amounts of data in order to extract useful information from the data, in the readily-understandable form of new rules, tendencies, and patterns. The information so extracted can be useful for business marketing strategies or

		loosing or keeping paper coupons or RFID card coupons.	maximizes user convenience is implemented, where the user does not have to try to find the closest business manually		an implicit method using NFC, and selected the types of businesses with high possibility of user is likely to use by time slot through association analysis.	when it comes to customers making decisions.

4. PROPOSED WORK



Hardware and software requirements

No hardware requirements. The software requirements are as follows-

- Google Colab used for running
- Python 3.7.2
- Python modules
- pandas and numpy for data manipulation
- seaborn and Matplotlib library for visualization
- mlxtend library for Apriori and Association rule mining techniques

Methodology adapted

To implement the grocery recommendation system, we have used the apriori algorithm. Apriori algorithm, also called frequent pattern mining, refers to the algorithm which is used to calculate the frequent itemsets and association rules between objects. It means how many in what way two or more objects are related to one another. In other words, we can say that the apriori algorithm is an association rule learning that analyzes that people who bought product A also bought product B.

If we establish these association rules, we can say that if Bread is bought, Milk & Eggs are also bought & hence all these can be kept close. It can be helpful in building sections like 'Customers who viewed this also viewed this' like in Amazon.

These frequent itemsets and association rules helps the customers to buy such products more often at any store with ease and increases the sales performance.

Let I={I1, I2, I3, etc} be collection of all possible items for a particular user A. Lets assume I = {Bread, Diaper, Milk, Baby Food, Eggs, Cola} be the dataset.

Item-set: Any subset of I is called an Item-set. Ex: {Babyfood, Diaper},{},{Eggs},etc are Item-sets as all of them are subsets of I. Also, a 'k' element subset will be called K item-sets.

For example:

- {bread, eggs} will be called a 2 item-set.
- {Bread, Diaper, Cola} will be called a 3 item-set
- {Diaper, Milk, Baby Food, Cola} will be called a 4 item-set and so on.

For our implementation, we have chosen items - sets which contain the highest number of items, to provide all the possibilities of recommended items to the particular user.

We decide these frequent itemsets and rules based on following two components -

- Support
- Confidence

Support - Support refers to the default popularity of any product. We can find the support as a quotient of the division of the number of transactions comprising that product by the total number of transactions.
Confidence - Confidence refers to the possibility that the customers bought both I1 and I2 together. So, you need to divide the number of transactions that comprise both biscuits and chocolates by the total number of transactions to get the confidence.

$$\text{Support (A)} = \frac{\text{Number of transaction in which A appears}}{\text{Total number of transactions}}$$

$$\text{Confidence (A}\longrightarrow\text{B)} = \frac{\text{Support(AUB)}}{\text{Support(A)}}$$

Here, an item-set is called frequent if it has a Support>Support_Threshold set. If an Item-set is frequent (Support above Support_Threshold), all its subsets are also frequent. Hence, we don't need to calculate Support for each rule.

Ex: If Support{Bread,Milk,Eggs} is above the Support_Threshold, then, {Bread,Milk}, {Milk,Eggs}, {Bread,Eggs},etc also have their Support above threshold. Hence, a lot of comparisons are saved.

Reasons for choosing Apriori algorithm-

- It is used to calculate large itemsets.
- Simple to understand and apply.
- The join and prune steps of the algorithm can be easily implemented on large datasets

sxasd

Innovation that was carried out

There are many already familiar with clustering algorithms such as K-Means, HAC, or DBSCAN. However, clustering is not the only unsupervised way to find similarities between data points.
Association rule learning techniques are better to determine if certain data points (actions) are more likely to occur together.

Apriori is part of the association rule learning algorithms, which falls under unsupervised Artificial Intelligence. The Apriori model doesn't require a target variable for the model. Instead, the algorithm identifies relationships between data points subject to the specified constraints.
Hence, instead of going the traditional way using classification or clustering, we decided to use Apriori Algorithm under Association rule mining.

DATASET USED/ TOOLS USED

We took an open Groceries Dataset from Kaggle containing descriptions and transaction details of about 38000+ customers.

Tools used: Google Colab

Member_number	Date	itemDescription
1808	21-07-2015	tropical fruit
2552	05-01-2015	whole milk
2300	19-09-2015	pip fruit
1187	12-12-2015	other vegetables
3037	01-02-2015	whole milk
4941	14-02-2015	rolls/buns
4501	08-05-2015	other vegetables
3803	23-12-2015	pot plants
2762	20-03-2015	whole milk
4119	12-02-2015	tropical fruit
1340	24-02-2015	citrus fruit
2193	14-04-2015	beef
1997	21-07-2015	frankfurter
4546	03-09-2015	chicken
4736	21-07-2015	butter
1959	30-03-2015	fruit/vegetable juice
1974	03-05-2015	packaged fruit/vegetables
2421	02-09-2015	chocolate
1513	03-08-2015	specialty bar
1905	07-07-2015	other vegetables

IMPLEMENTATION AND RESULTS

First we import our dataset, which consists of Member Number, Date and Item Description. In the item description, we have the item the user purchased on the respective date.

```
data.head()
```

	Member_number	Date	itemDescription
0	1808	21-07-2015	tropical fruit
1	2552	05-01-2015	whole milk
2	2300	19-09-2015	pip fruit
3	1187	12-12-2015	other vegetables
4	3037	01-02-2015	whole milk

```
data.sort_values(by=['Date'])
```

	Member_number	Date	itemDescription
13581	4942	01-01-2014	butter
9557	1922	01-01-2014	tropical fruit
30568	3797	01-01-2014	whole milk
13449	2237	01-01-2014	bottled water
9172	2226	01-01-2014	sausage
...	...	...	...

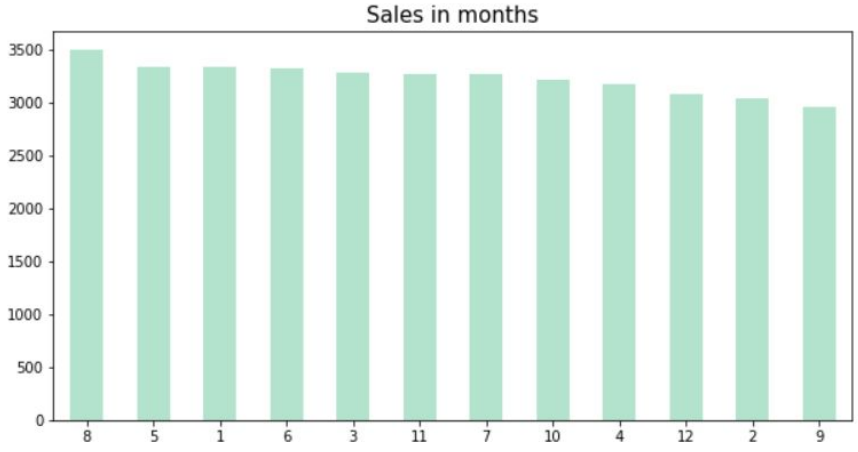
Now, we modify our dataset a little to help visualize better so as to understand the relationships between different attributes of our data in more depth and to be able to build our recommendation model accurately.

```
[8]  #Creating new columns based on the date column
     data_eda['year'] = data_eda.index.year
     data_eda['month'] = data_eda.index.month
     data_eda['day'] = data_eda.index.day
     data_eda['weekday'] = data_eda.index.strftime('%A')
     data_eda['monthName'] = data_eda.index.strftime('%B')
     data_eda.head()
```

Date	Member_number	itemDescription	year	month	day	weekday	monthName
2015-07-21	1808	tropical fruit	2015	7	21	Tuesday	July
2015-05-01	2552	whole milk	2015	5	1	Friday	May
2015-09-19	2300	pip fruit	2015	9	19	Saturday	September
2015-12-12	1187	other vegetables	2015	12	12	Saturday	December
2015-01-02	3037	whole milk	2015	1	2	Friday	January

```
plt.figure(figsize=(10, 5))
data_eda['month'].value_counts(sort=True).plot(kind='bar',rot=0,colormap='Pastel2')
plt.title('Sales in months', size=15)
```

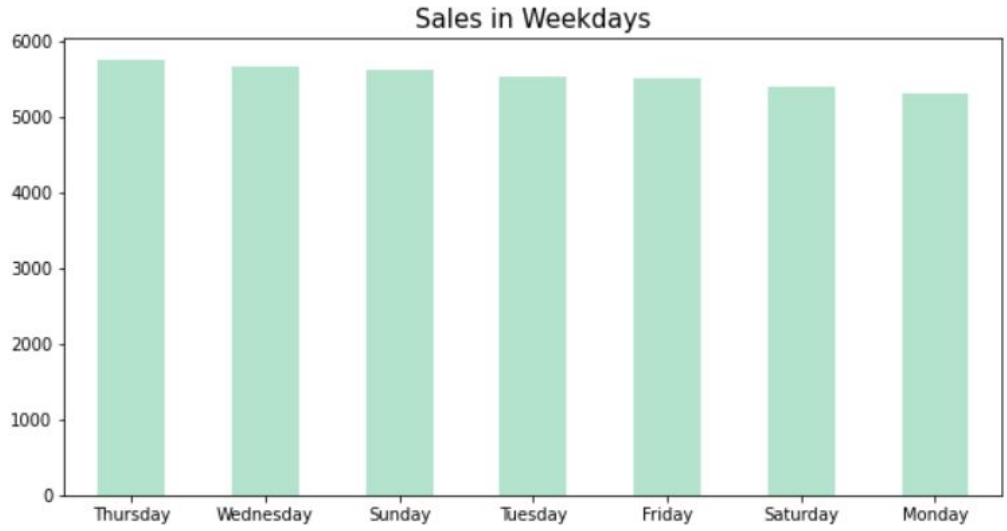
```
Text(0.5, 1.0, 'Sales in months')
```



This graph depicts the sales in every month (1- “Jan”, 2- “Feb” and so on) in descending order so as to give us a detailed idea about what months usually the stock should be kept high.

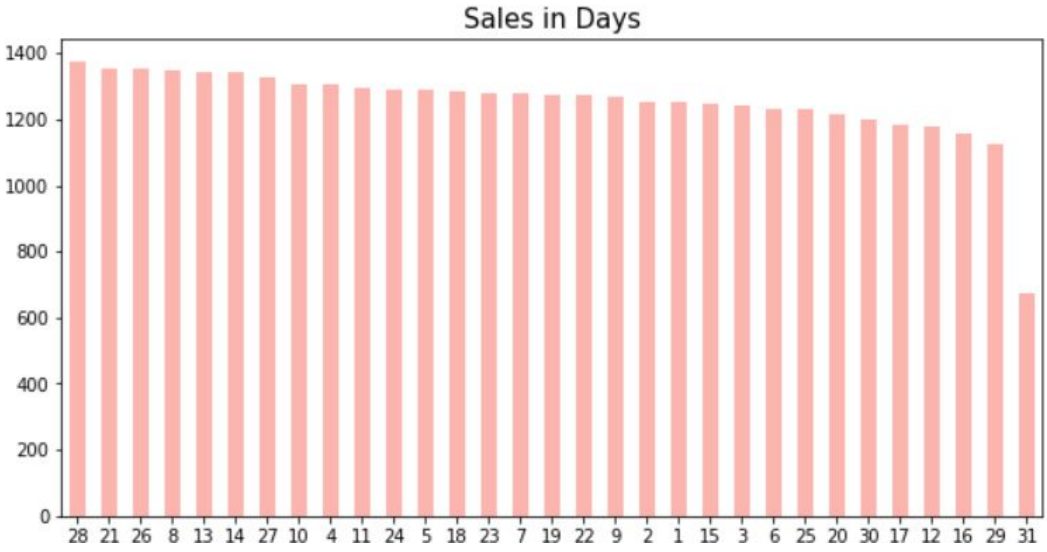
```
plt.figure(figsize=(10, 5))
data_eda['weekday'].value_counts(sort=True).plot(kind='bar',rot=0,colormap='Pastel2')
plt.title('Sales in Weekdays', size=15)
```

```
Text(0.5, 1.0, 'Sales in Weekdays')
```

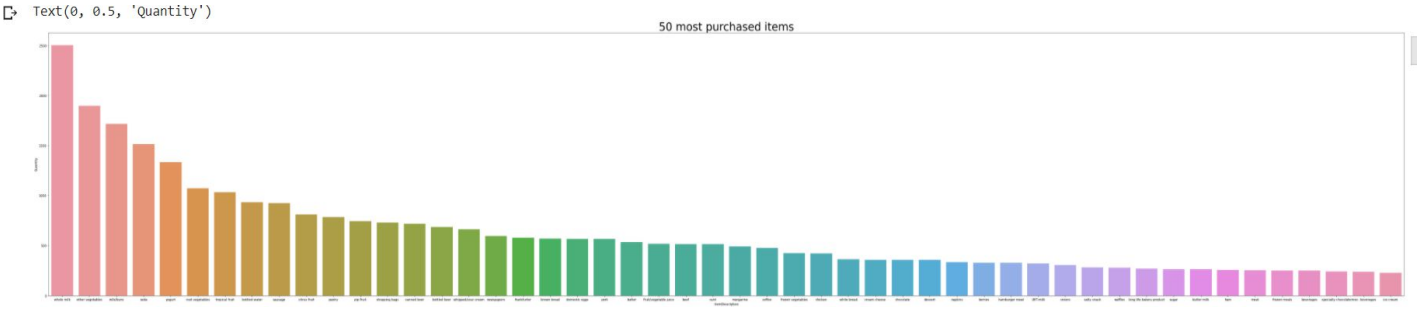



```
plt.figure(figsize=(10, 5))
data_eda['day'].value_counts(sort=True).plot(kind='bar',rot=0,colormap='Pastel1')
plt.title('Sales in Days', size=15)
```

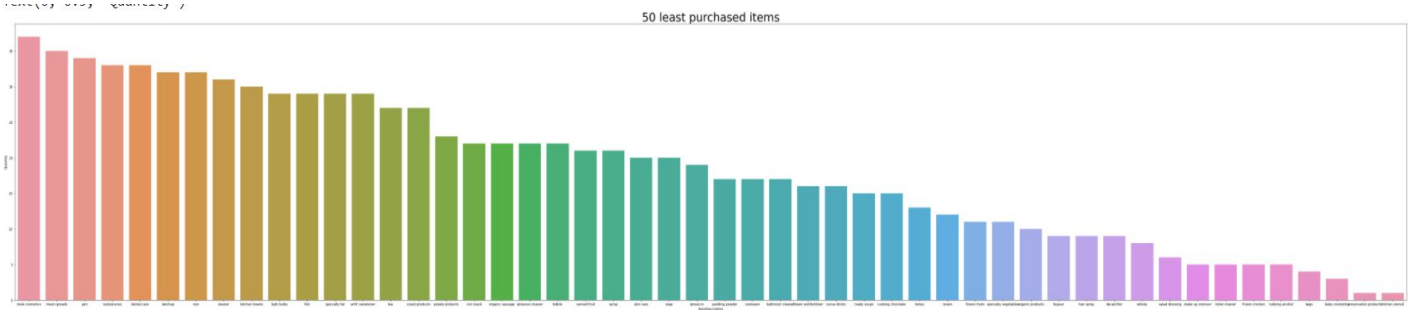
```
Text(0.5, 1.0, 'Sales in Days')
```



We similarly found out the sales in weekdays and days which hence provides us with a summary of when the sales are maximum and when they are minimum.



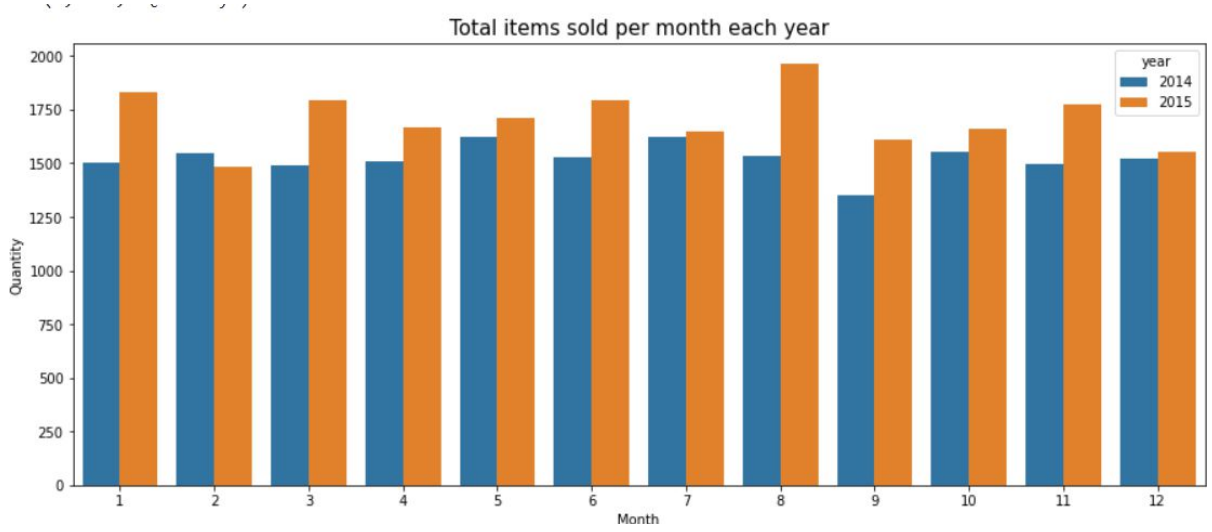
This graph shows us the 50 most purchased items, which helps us to get an idea of the most frequent items purchased overall, as well as the shop-owner to know.

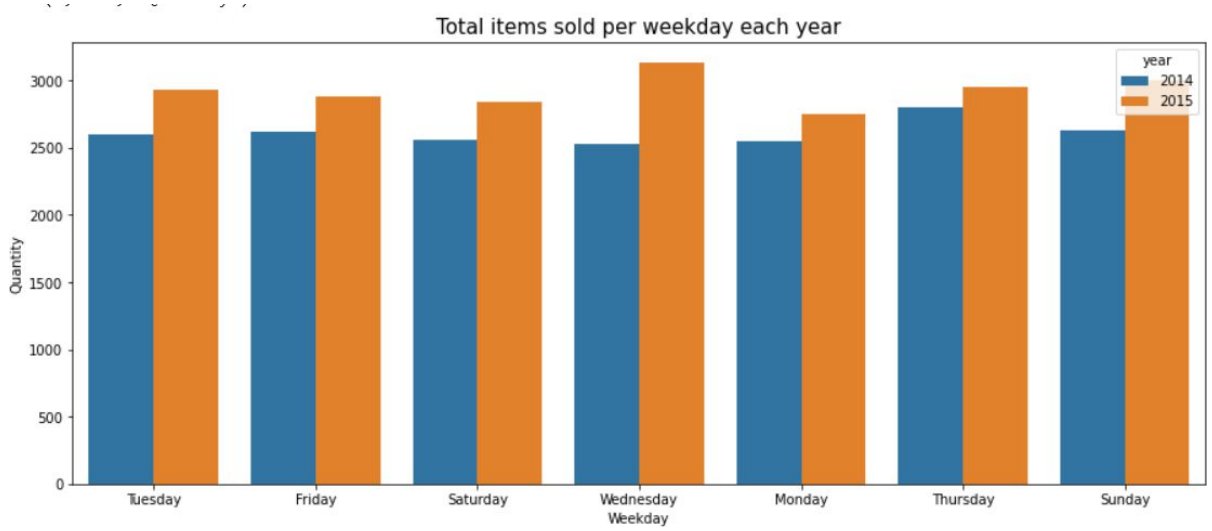


Similarly we also found out the least purchased items, hence to help us generate the user-based frequent itemsets much better.

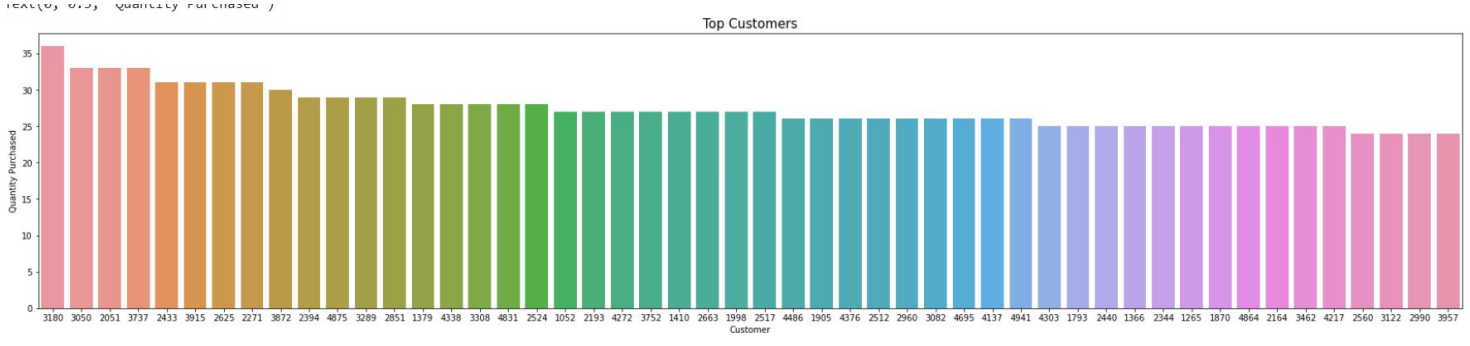
	itemDescription	Count
0	whole milk	2502
1	other vegetables	1898
2	rolls/buns	1716
3	soda	1514
4	yogurt	1334

These are some of the top 50 frequent items purchased.



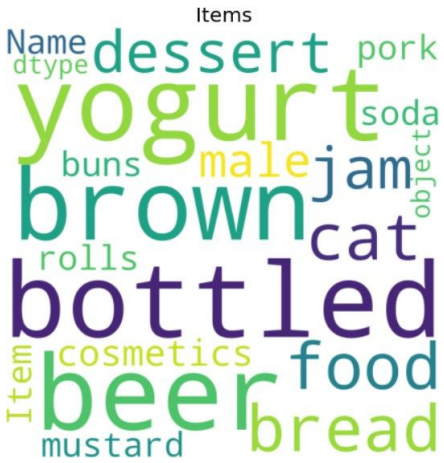


Here we have a side by side comparison of the sales in both the years: 2014 and 2015. We have done a weekly and monthly comparison, and we found out that the sales increased significantly in the year 2015 as compared to the previous year.



	Member_number	Count
0	3180	36
1	3050	33
2	2051	33
3	3737	33
4	2433	31

Here is a glimpse of our 50 top customers, who have purchased the most items in the duration of two years. This really helps us to identify relationships between frequent itemsets, as well as find similar users. This will also help the shop-owner to know his/her key customers and their frequently bought items, so as to be prepared first-hand.



This is a word cloud of some of the most frequently bought items.

APRIORI MODEL

Apriori Model

```
[21] user_id = input()
```

```
1004
```

```
[22] df3 = data.loc[data['Member_number'] == int(user_id)]
```

```
[23] df3
```

	Member_number	Date	itemDescription
4936	1004	02-12-2015	other vegetables
10792	1004	04-02-2014	pip fruit
11398	1004	19-08-2014	root vegetables
12114	1004	01-07-2014	canned beer
12293	1004	10-06-2014	rolls/buns
13767	1004	13-10-2014	whole milk
14500	1004	10-07-2014	other vegetables
21085	1004	02-12-2015	hygiene articles
26941	1004	04-02-2014	whole milk

This is where we input the member number of the member whose frequent itemsets need to be generated. We will now limit our findings to this dataset.

```
freq_items = apriori(transactions, min_support=0.001, use_colnames=True)
#support = particular itemset/ total no of transactions
```

```
transactions
```

```
[['dish cleaner', 'cling film/bags'],
 ['canned beer', 'frozen fish'],
 ['other vegetables', 'hygiene articles'],
 ['pip fruit', 'whole milk', 'tropical fruit'],
 ['rolls/buns', 'red/blush wine', 'chocolate'],
 ['other vegetables', 'shopping bags'],
 ['whole milk', 'chocolate', 'packaged fruit/vegetables', 'rolls/buns'],
 ['root vegetables', 'whole milk', 'pastry']]
```

This gives us our user's transaction history. It is a list in a list. Every list contains the itemset of one transaction made, And here, in this example, similarly, we have 8 transactions made by the user. Now that we have got our main list, we define a minimum support, and find itemsets that have maximum items and are above our minimum support.

	support	itemsets	length
32	0.125	(whole milk, root vegetables)	2
33	0.125	(tropical fruit, whole milk)	2
34	0.125	(chocolate, rolls/buns, packaged fruit/vegetab...	3
35	0.125	(whole milk, chocolate, packaged fruit/vegetab...	3
36	0.125	(red/blush wine, chocolate, rolls/buns)	3
37	0.125	(whole milk, chocolate, rolls/buns)	3
38	0.125	(whole milk, rolls/buns, packaged fruit/vegeta...	3
39	0.125	(whole milk, root vegetables, pastry)	3
40	0.125	(tropical fruit, pip fruit, whole milk)	3
41	0.125	(whole milk, chocolate, rolls/buns, packaged f...	4

We now run the data through our model, which gives us a dataframe with length of itemsets arranged in ascending order, all above the min support (0.001).

From this, we now choose the itemsets with the maximum length, and add all the items to a list, making sure no item is repeated twice, as can be seen below. The list contains of that user's most purchased items:

```
['whole milk', 'chocolate', 'rolls/buns', 'packaged fruit/vegetables']
```

A more detailed view of our apriori model, can be seen below. We make a rules table based on the association rule mining.

The table of some of the rules is as follows:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
82	(whole milk, chocolate)	(rolls/buns, packaged fruit/vegetables)	0.125	0.125	0.125	1.000000	8.000000	0.109375	inf
83	(whole milk, rolls/buns)	(chocolate, packaged fruit/vegetables)	0.125	0.125	0.125	1.000000	8.000000	0.109375	inf
84	(whole milk, packaged fruit/vegetables)	(chocolate, rolls/buns)	0.125	0.250	0.125	1.000000	4.000000	0.093750	inf
85	(chocolate, rolls/buns)	(whole milk, packaged fruit/vegetables)	0.250	0.125	0.125	0.500000	4.000000	0.093750	1.7500
86	(chocolate, packaged fruit/vegetables)	(whole milk, rolls/buns)	0.125	0.125	0.125	1.000000	8.000000	0.109375	inf
87	(rolls/buns, packaged fruit/vegetables)	(whole milk, chocolate)	0.125	0.125	0.125	1.000000	8.000000	0.109375	inf
88	(whole milk)	(chocolate, rolls/buns, packaged fruit/vegetab...	0.375	0.125	0.125	0.333333	2.666667	0.078125	1.3125
89	(chocolate)	(whole milk, rolls/buns, packaged fruit/vegeta...	0.250	0.125	0.125	0.500000	4.000000	0.093750	1.7500
90	(rolls/buns)	(whole milk, chocolate, packaged fruit/vegetab...	0.250	0.125	0.125	0.500000	4.000000	0.093750	1.7500
91	(packaged fruit/vegetables)	(whole milk, chocolate, rolls/buns)	0.125	0.125	0.125	1.000000	8.000000	0.109375	inf

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(frozen fish)	(canned beer)	0.125	0.125	0.125	1.000000	8.000000	0.109375	inf
1	(canned beer)	(frozen fish)	0.125	0.125	0.125	1.000000	8.000000	0.109375	inf
2	(chocolate)	(packaged fruit/vegetables)	0.250	0.125	0.125	0.500000	4.000000	0.093750	1.750
3	(packaged fruit/vegetables)	(chocolate)	0.125	0.250	0.125	1.000000	4.000000	0.093750	inf
4	(red/blush wine)	(chocolate)	0.125	0.250	0.125	1.000000	4.000000	0.093750	inf
5	(chocolate)	(red/blush wine)	0.250	0.125	0.125	0.500000	4.000000	0.093750	1.750
6	(chocolate)	(rolls/buns)	0.250	0.250	0.250	1.000000	4.000000	0.187500	inf
7	(rolls/buns)	(chocolate)	0.250	0.250	0.250	1.000000	4.000000	0.187500	inf
8	(whole milk)	(chocolate)	0.375	0.250	0.125	0.333333	1.333333	0.031250	1.125
9	(chocolate)	(whole milk)	0.250	0.375	0.125	0.500000	1.333333	0.031250	1.250

Code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
% matplotlib inline
from google.colab import drive
drive.mount('/content/gdrive')
data=pd.read_csv("/content/gdrive/MyDrive/Groceries_dataset.csv")
#/content/gdrive/MyDrive/Groceries_dataset.csv
data.shape
data.head()
data.sort_values(by=['Date'])
#data_eda = data.copy()
data_eda = pd.read_csv("/content/gdrive/MyDrive/Groceries_dataset.csv",index_col='Dat
e', parse_dates=True)
data_eda.head()
#Number of unique Costumers and Items
print(data_eda.Member_number.nunique())
print(data_eda.itemDescription.nunique())
#Creating new columns based on the date column
data_eda['year'] = data_eda.index.year
data_eda['month'] = data_eda.index.month
data_eda['day'] = data_eda.index.day
data_eda['weekday'] = data_eda.index.strftime('%A')
data_eda['monthName'] = data_eda.index.strftime('%B')
data_eda.head()
data_eda['year'].value_counts()
plt.figure(figsize=(10, 5))
data_eda['month'].value_counts(sort=True).plot(kind='bar',rot=0,colormap='Pastel2')
plt.title('Sales in months', size=15)
plt.figure(figsize=(10, 5))
data_eda['weekday'].value_counts(sort=True).plot(kind='bar',rot=0,colormap='Pastel2')
plt.title('Sales in Weekdays', size=15)
plt.figure(figsize=(10, 5))
data_eda['day'].value_counts(sort=True).plot(kind='bar',rot=0,colormap='Pastel1')
plt.title('Sales in Days', size=15)
df1 = pd.DataFrame(data_eda['itemDescription'].value_counts().head(50)).reset_index()
df1.columns =['itemDescription','Count']
df1.head()
plt.figure(figsize=(80, 15))
sns.barplot(x='itemDescription',y='Count', data=df1)
plt.title('50 most purchased items', size=35)
plt.ylabel('Quantity')
df2 = pd.DataFrame(data_eda['itemDescription'].value_counts().tail(50)).reset_index()
df2.columns =['itemDescription','Count']
plt.figure(figsize=(80, 15))
sns.barplot(x='itemDescription',y='Count', data=df2)
plt.title('50 least purchased items', size=35)
```

```
plt.ylabel('Quantity')
#Total items sold per month each year
plt.figure(figsize=(15,6))
ax = sns.countplot(x='month', hue='year', data=data_eda)
plt.title('Total items sold per month each year', size=15)
plt.xlabel('Month')
plt.ylabel('Quantity')
#Total items sold per weekday each year
plt.figure(figsize=(15,6))
ax = sns.countplot(x='weekday', hue='year', data=data_eda)
plt.title('Total items sold per weekday each year', size=15)
plt.xlabel('Weekday')
plt.ylabel('Quantity')
#Total items sold per day each year
plt.figure(figsize=(25,8))
ax = sns.countplot(x='day', hue='year', data=data_eda)
plt.title('Total items sold per day each year', size=15)
plt.xlabel('Day')
plt.ylabel('Quantity')
total_items = len(data_eda)
total_days = len(np.unique(data_eda.index.date))
total_months = len(np.unique(data_eda.index.month))
average_items = total_items / total_days
unique_items = data_eda.itemDescription.unique().size

print("Total {} items sold in {} days throughout 2 years".format(total_items, total_d
ays))
print("With an average of {} items sold daily".format(average_items))
#Top Customers
plt.figure(figsize=(30,6))
ax = sns.countplot(x='Member_number', data=data_eda, order=data_eda.Member_number.val
ue_counts().iloc[:50].index)
plt.title('Top Customers', size=15)
plt.xlabel('Customer')
plt.ylabel('Quantity Purchased')



user_id = input()
df3 = data.loc[data['Member_number'] == int(user_id)]
df3
df3.shape
df3.values
transactions = [a[1]['itemDescription'].tolist() for a in list(df3.groupby(['Date']))
]
transactions
from mlxtend.preprocessing import TransactionEncoder
te = TransactionEncoder()
te_ary = te.fit(transactions).transform(transactions)
te.columns_
te_ary
```

```
transactions = pd.DataFrame(te_ary, columns=te.columns_)
transactions
#Count is the total no of transactions in each colums, and freq is the no of times *T
OP variable* appeared in that column
pf = transactions.describe()
pf
#Count - Freq from above table - to find no of TRUE values
pf.iloc[0]-pf.iloc[3]
f = pf.iloc[0]-pf.iloc[3]
a = f.tolist()
b = list(f.index)
item = pd.DataFrame([[a[r],b[r]]for r in range(len(a))], columns=['Count','Item'])
item = item.sort_values(['Count'], ascending=False)
item
import matplotlib.pyplot as plt
import seaborn as sns

from wordcloud import WordCloud

plt.rcParams['figure.figsize'] = (8, 8)
wordcloud = WordCloud(background_color = 'white', width = 1200,  height = 1200, max_w
ords = 121).generate(str(item['Item']))
plt.imshow(wordcloud)
plt.axis('off')
plt.title('Items',fontsize = 20)
plt.show()
from mlxtend.frequent_patterns import apriori, association_rules
import matplotlib.pyplot as plt
freq_items = apriori(transactions, min_support=0.001, use_colnames=True)
#support = particular itemset/ total no of transactions
freq_items.shape
freq_items.head(10)
freq_items['length'] = freq_items['itemsets'].apply(lambda x: len(x))
freq_items['length'].unique()
freq_items.head(100)
freq_items.tail(10)
length = freq_items.iloc[-1]['length']
length
l = freq_items.loc[freq_items['length'] == length]
l
names=[]
for i in l.itemsets:
  for j in i:
    if j not in names:
      names.append(j)
names
rules = association_rules(freq_items, metric="confidence", min_threshold=0.001)
rules.shape
rules.head(10)
```

Colab Link –
https://colab.research.google.com/drive/1NWRlIdaoVMUOC3u5z1W9JZDo0nYlS7ym?usp=sharing

Dataset Link -
https://drive.google.com/file/d/1RaVoQc8KhqQ1kmUaVv5QDiYzaTVpt1K_/view?usp=sharing

CONCLUSION

In this project we have successfully implemented a functional grocery recommendation system with the help of apriori algorithm. We also linked our results into a website for better visibility of a novice user. To get to the results, we also performed several EDAs. It was done in order to understand the data and learn it in a better way. The shortcomings of the Apriori algorithm might have been a setback in the implementation of our project though. The Hash-based itemset counting: A k-itemset whose corresponding hashing bucket count is below the threshold cannot be frequent. Transaction reduction: A transaction that does not contain any frequent k-itemset is useless in subsequent scans. Partitioning: Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB. Sampling: mining on a subset of given data, lower support threshold + a method to determine the completeness. Dynamic itemset counting: add new candidate itemsets only when all of their subsets are estimated to be frequent. Apriori algorithm is also an expensive method to find support since the calculation has to pass through the whole database. Sometimes, you need a huge number of candidate rules, so it becomes computationally more expensive.

REFERENCES

Eirinaki, Gao, Varlamis, Tserpes (2018) Recommender Systems for Large-Scale Social Networks: A review of challenges and solutions, ELESVIER, Future Generation Computer Systems 78 (2018) 413–418

Shaikh, Rathi, Janrao (2017): Recommendation system in E-commerce websites: AGraph Based Approached, 2017 IEEE 7th International Advance Computing Conference

Sidorov, Velasquez, Stamatatos, Gelbukh, Chanona-Hernández (2014): Syntactic N-grams as machine learning features for natural language processing, ELSEVIER Expert Systems with Applications 41 (2014) 853–860.

Comparing Dataset Characteristics that Favor the Apriori, Eclat or FP-Growth Frequent Itemset Mining Algorithms Authors: Jeff Heaton College of Engineering and Computing Published: Nova Southeastern University, 2017

A novel hybrid based recommendation system based on clustering, association mining Authors:

S. Pandya ; J. Shah ; N. Joshi ; H. Ghayvat ; S. C. Mukhopadhyay ; M. H. Yap Published: 2016 10th International Conference on Sensing Technology (ICST)

Implementation of a Recommendation System using Association Rules and Collaborative Filtering Authors: JinHyun Jooa, SangWon Bangb, GeunDuk Parka Published: Information Technology and Quantitative Management (ITQM 2016)

A recommendation engine by using association rules Authors: Ozgur Cakira 1, Murat Efe Aras b Published: WCBEM 2012

Advanced eclat algorithm for frequent itemsets generation Authors: M. Kaur, Urvashi Garg, S. Kaur Published: January 2015 International Journal of Applied Engineering Research

Grocery Stop database management tool Author: Tapan Desai: www.slideshare.net/tapandesai2992/grocery-station

Online Store Product Recommendation System Uses Apriori Method: C S Fatoni , E Utami and F W Wibowo, 2018