MARKET BASKET ANALYSIS USING THE APRIORI ALGORITHM

Market basket analysis is a data mining technique that helps businesses discover the patterns and associations among the products that customers buy together. By using the Apriori algorithm, businesses can efficiently generate and evaluate the rules that describe how often and how strongly the products are related. For example, if customers who buy bread also tend to buy butter, then the rule "Bread -> Butter" can be derived and used for marketing purposes.

In this report, I will explain the principles of market basket analysis and how to use the Apriori algorithm in Python. I will also demonstrate the implementation and the results of the algorithm on a sample dataset of supermarket transactions.

Principles of market basket analysis

Market basket analysis is based on the concept of association rules, which are statements that describe the relationship between two or more items in a transactional dataset. For example, the rule "Bread -> Butter" means that customers who buy bread are likely to buy butter as well. Association rules can be useful for understanding customer behavior, optimizing product placement, offering discounts or coupons, or suggesting complementary products.

To measure the quality and the strength of association rules, there are three common metrics: support, confidence and lift.

The apriori algorithm

The Apriori algorithm is a method for finding frequent itemsets and association rules from a transactional dataset. The algorithm follows the Apriori property, which states that if a set of items is frequent, then all of its subsets must also be frequent. This property allows the algorithm to prune the search space and avoid checking the sets that are unlikely to be frequent.

The Apriori algorithm consists of two main steps; frequent itemset generation and association rule generation.

Implementation of apriori in python

Python, with its rich ecosystem of libraries, provides a user-friendly environment for implementing the Apriori algorithm. Here, we will see how to use the mlxtend library to perform market basket analysis on a sample dataset of grocery store transactions. The dataset is taken from Kaggle and it contains 38,765 rows and 3 columns.

```
df.shape
```

```
(38765, 3)
```

```
df.columns
```

```
Index(['Member_number', 'Date', 'itemDescription'], dtype='object')
```

The steps I took to implement the Apriori algorithm in Python are as follows:

1. First, I imported the necessary libraries

```
import pandas as pd
import numpy as np
import time, warnings
import datetime as dt
warnings.filterwarnings('ignore')

#algorithms
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
```

2. Then I loaded the dataset into a pandas DataFrame. It contains information on purchases made at a grocery store, including the transaction date, item description and a unique customer ID (Member_number). I used this data frame to perform market basket analysis and identify item combinations that are frequently bought together.

```
df = pd.read_csv("Groceries_dataset.csv")
df.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

3. I checked for null values but there were none.

```
df.isnull().sum()
```

```
Member_number      0
Date               0
itemDescription    0
dtype: int64
```

4. I converted the 'Date' column to datetime objects so that I won't encounter any issues with the format of my data.

```
# Convert the 'Date' column to datetime objects
df['Date'] = pd.to_datetime(df['Date'])
```

5. To use the Apriori algorithm, I transformed the data into a binary matrix, where each row is a transaction, each column is an item and each cell is a one or a zero, indicating whether the item is present or absent in the transaction. To do this:
   a. I first grouped the data by Member_number and date, so that each group represents a single

transaction.

```
df['single_transaction'] = df['Member_number'].astype(str)+'_'+df['Date'].astype(str)
df.head(5)

#The "single_transaction" variable combines the Member_number and date and tells us the item purchased in one receipt.
```

	Member_number	Date	itemDescription	single_transaction
0	1808	2015-07-21	tropical fruit	1808_2015-07-21
1	2552	2015-05-01	whole milk	2552_2015-05-01
2	2300	2015-09-19	pip fruit	2300_2015-09-19
3	1187	2015-12-12	other vegetables	1187_2015-12-12
4	3037	2015-01-02	whole milk	3037_2015-01-02

b. To transform the table into a binary matrix, we need to pivot it so that each row represents a transaction and each column represents an item. This way, we can use ones and zeros to indicate the presence or absence of each item in each transaction.

```
df2 = pd.crosstab(df['single_transaction'], df['itemDescription'])
df2.head()
```

itemDescription	Instant food products	UHT-milk	abrasive cleaner	artif. sweetener	baby cosmetics	bags	baking powder	bathroom cleaner	beef	berries	...	turkey	vinegar	waffles	whipped/sour cream	whisky	whit brea
single_transaction																	
1000_2014-06-24	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
1000_2015-03-15	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
1000_2015-05-27	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
1000_2015-07-24	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
1000_2015-11-25	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	

5 rows × 167 columns

6. The next step involves encoding all values in the above data frame to 0 and 1. This means that even if there are multiples of the same items in the same transaction, the value will be encoded to 1s and 0s since market basket analysis does not take purchase frequency into consideration.

```
def encode(item_freq):
    res = 0
    if item_freq > 0:
        res = 1
    return res

basket_input = df2.applymap(encode)
basket_input
```

itemDescription	Instant food products	UHT-milk	abrasive cleaner	artif. sweetener	baby cosmetics	bags	baking powder	bathroom cleaner	beef	berries	...	turkey	vinegar	waffles	whipped/sour cream	whisky	whi brea
single_transaction																	
1000_2014-06-24	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
1000_2015-03-15	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
1000_2015-05-27	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
1000_2015-07-24	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
1000_2015-11-25	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
4999_2015-05-16	0	0	0	0	0	0	0	0	0	0	...	0	0	0	1	0	
4999_2015-12-26	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
5000_2014-09-03	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
5000_2014-11-16	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
5000_2015-10-02	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	

14963 rows × 167 columns

7. Filtering the data. If the customer bought only 1 item during his purchase, we can't use that data because we cannot find any relation between items as there is only one product. So, we need to filter the transactions that bought more than one item.

```
#filtering the data:
data_filter=basket_input[(basket_input>0).sum(axis=1)>=2]
data_filter
```

itemDescription	Instant food products	UHT-milk	abrasive cleaner	artif. sweetener	baby cosmetics	bags	baking powder	bathroom cleaner	beef	berries	...	turkey	vinegar	waffles	whipped/sour cream	whisky	whi brea
single_transaction																	
1000_2014-06-24	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
1000_2015-03-15	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
1000_2015-05-27	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
1000_2015-07-24	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
1000_2015-11-25	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
4999_2015-05-16	0	0	0	0	0	0	0	0	0	0	...	0	0	0	1	0	
4999_2015-12-26	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
5000_2014-09-03	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
5000_2014-11-16	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
5000_2015-10-02	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	

14758 rows × 167 columns

8. After making all the required changes, we can now apply the algorithm. The main aim of the algorithm is to find the frequently bought items in the dataset. The library required for apriori algorithm is 'mlextend', which we imported earlier. In this algorithm, we can define the frequent data by given a support value, here in this case I gave a minimum support of 0.001. Here, the "antecedents" and "consequents" columns show items that are frequently purchased together.

```
frequent_itemsets = apriori(data_filter, min_support=0.001, use_colnames=True)
rules = association_rules(frequent_itemsets, metric="lift")
rules
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	zhangs_metric
0	(bottled water)	(UHT-milk)	0.060984	0.021615	0.001084	0.017778	0.822459	-0.000234	0.996093	-0.186916
1	(UHT-milk)	(bottled water)	0.021615	0.060984	0.001084	0.050157	0.822459	-0.000234	0.988601	-0.180754
2	(other vegetables)	(UHT-milk)	0.122510	0.021615	0.002168	0.017699	0.818820	-0.000480	0.996013	-0.201381
3	(UHT-milk)	(other vegetables)	0.021615	0.122510	0.002168	0.100313	0.818820	-0.000480	0.975329	-0.184445
4	(sausage)	(UHT-milk)	0.061052	0.021615	0.001152	0.018868	0.872893	-0.000168	0.997200	-0.134262
...	...	...	...	...	...	...	...	...	...	...
715	(sausage, whole milk)	(yogurt)	0.009080	0.086055	0.001491	0.164179	1.907839	0.000709	1.093470	0.480207
716	(yogurt, whole milk)	(sausage)	0.011316	0.061052	0.001491	0.131737	2.157789	0.000800	1.081409	0.542704
717	(sausage)	(yogurt, whole milk)	0.061052	0.011316	0.001491	0.024417	2.157789	0.000800	1.013429	0.571451
718	(yogurt)	(sausage, whole milk)	0.086055	0.009080	0.001491	0.017323	1.907839	0.000709	1.008388	0.520651
719	(whole milk)	(sausage, yogurt)	0.157406	0.005827	0.001491	0.009471	1.625184	0.000573	1.003678	0.456549

720 rows × 10 columns

9. To get the most frequent item combinations in the entire dataset, let's sort the dataset by support, confidence and lift:

```
# To get the most frequent item combinations in the entire dataset, sort the dataset by support, confidence and lift
rules.sort_values(["support", "confidence", "lift"], axis = 0, ascending = False).head(10)
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	zhangs_metric
622	(rolls/buns)	(whole milk)	0.110005	0.157923	0.013968	0.126974	0.804028	-0.003404	0.964550	-0.214986
623	(whole milk)	(rolls/buns)	0.157923	0.110005	0.013968	0.088447	0.804028	-0.003404	0.976350	-0.224474
695	(yogurt)	(whole milk)	0.085879	0.157923	0.011161	0.129961	0.822940	-0.002401	0.967861	-0.190525
694	(whole milk)	(yogurt)	0.157923	0.085879	0.011161	0.070673	0.822940	-0.002401	0.983638	-0.203508
551	(soda)	(other vegetables)	0.097106	0.122101	0.009691	0.099794	0.817302	-0.002166	0.975219	-0.198448
550	(other vegetables)	(soda)	0.122101	0.097106	0.009691	0.079365	0.817302	-0.002166	0.980729	-0.202951
649	(sausage)	(whole milk)	0.060349	0.157923	0.008955	0.148394	0.939663	-0.000575	0.988811	-0.063965
648	(whole milk)	(sausage)	0.157923	0.060349	0.008955	0.056708	0.939663	-0.000575	0.996140	-0.070851
625	(yogurt)	(rolls/buns)	0.085879	0.110005	0.007819	0.091051	0.827697	-0.001628	0.979147	-0.185487
624	(rolls/buns)	(yogurt)	0.110005	0.085879	0.007819	0.071081	0.827697	-0.001628	0.984071	-0.189562

CONCLUSION

Market Basket Analysis using Apriori Algorithm is performed using an online retail dataset (this dataset was downloaded from kaggle). There are about 734 transactions which are considered as frequently bought item sets. We can see that the most frequently bought item and their support are;

I. Rolls/buns and whole milk (0.013968)
II. Yogurt and whole milk (0.011161)
III. Soda and vegetables (0.009691)
IV. Sausages and whole milk (0.008955)

V. Yogurt and rolls/buns (0.007819)

The result of this analysis can be used for decision-making and marketing strategies. Insights gained from the above experiment are:

1. We can place them side by side in the store or run a promo/discount sale to increase sales.
2. Whenever a customer purchases one of these item pairs, we could recommend the other item pairs to him/her or suggest them as complementary products on the website.
3. We can offer a lower price by bundling both product pair together on the shelf. Thus, helping to generate more income and accelerate sales.

In this report, I have explained the principles of market basket analysis and how to use the Apriori algorithm in Python. I have also demonstrated the implementation and the results of the algorithm on a sample dataset of supermarket transactions. The results show that the algorithm can find frequent itemsets and association rules that reveal the patterns and associations among the products that customers buy together. The results can be useful for businesses to optimize their product offerings, marketing strategies, and customer satisfaction.