**Assignment 6**

Aim:

The Apriori Algorithm is a popular algorithm used in data mining and association rule learning, specifically for discovering frequent itemsets from a given dataset. The primary aim of the Apriori Algorithm is to identify sets of items that frequently occur together in a transactional dataset, and then generate association rules based on these frequent itemsets.

Theory:

The Apriori Algorithm is based on the concept of the "apriori" property, which states that if an itemset is frequent (i.e., occurs with a minimum threshold support) in a dataset, then all of its subsets must also be frequent. The algorithm starts by identifying frequent itemsets of size 1, which are individual items that occur with a minimum support threshold in the dataset. Then, it generates candidate itemsets of larger sizes by combining frequent itemsets of smaller sizes. The algorithm prunes the candidate itemsets that do not meet the minimum support threshold, and iteratively repeats the process of generating candidates, pruning, and counting the support until no further frequent itemsets can be found.

Case Study:

Market Basket Analysis is one of the key techniques used by large retailers to uncover associations between items. It works by looking for combinations of items that occur together frequently in transactions. To put it another way, it allows retailers to identify relationships between the items that people buy. Association Rules are widely used to analyze retail basket or transaction data and are intended to identify strong rules discovered in transaction data using measures of interestingness, based on the concept of strong rules. The dataset has 38765 rows of the purchase orders of people from the grocery stores. These orders can be analysed and association rules can be generated using Market Basket Analysis by algorithms like Apriori Algorithm.

Some important terms:-

* Support: This says how popular an itemset is, as measured by the proportion of transactions in which an itemset appears.
* Confidence: This says how likely item Y is purchased when item X is purchased, expressed as {X -> Y}. This is measured by the proportion of transactions with item X, in which item Y also appears.
* Lift: This says how likely item Y is purchased when item X is purchased while controlling for how popular item Y is.

[18]:

**import pandas as pd import numpy as np**

**import matplotlib.pyplot as plt import seaborn as sns**

**import re**

**from mlxtend.frequent\_patterns import** apriori

**from mlxtend.frequent\_patterns import** association\_rules **from mlxtend.preprocessing import** TransactionEncoder **from mpl\_toolkits.mplot3d import** Axes3D

**import networkx as nx**

basket = pd.read\_csv("Groceries\_dataset.csv") display(basket.head())

[19]:

|  |  |  |
| --- | --- | --- |
| 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 |

Member\_number Date itemDescription 0 1808 21-07-2015 tropical fruit

Instant food products UHT-milk abrasive cleaner artif. sweetener \

basket.itemDescription = basket.itemDescription.transform(**lambda** x: [x]) basket = basket.groupby(['Member\_number','Date']).sum()['itemDescription'].

↪reset\_index(drop=**True**)

encoder = TransactionEncoder()

transactions = pd.DataFrame(encoder.fit(basket).transform(basket),␣

↪columns=encoder.columns\_)

display(transactions.head())

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | False | False | False | False |
| 1 | False | False | False | False |
| 2 | False | False | False | False |
| 3 | False | False | False | False |
| 4 | False | False | False | False |

baby cosmetics bags baking powder bathroom cleaner beef berries \

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | False False | | | False | False | | False | False |
| 1 | False False | | | False | False | | False | False |
| 2 | False False | | | False | False | | False | False |
| 3 | False False | | | False | False | | False | False |
| 4 | False False | | | False | False | | False | False |
| … | turkey | vinegar | waffles | whipped/sour | cream | whisky | white | bread \ |
| 0 … | False | False | False |  | False | False |  | False |
| 1 … | False | False | False |  | False | False |  | False |
| 2 … | False | False | False |  | False | False |  | False |
| 3 … | False | False | False |  | False | False |  | False |
| 4 … | False | False | False |  | False | False |  | False |

white wine whole milk yogurt zwieback

0 False True True False

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | False | True | False | False |
| 2 | False | False | False | False |
| 3 | False | False | False | False |
| 4 | False | False | False | False |

[5 rows x 167 columns]

[20]:

frequent\_itemsets = apriori(transactions, min\_support= 6/len(basket),␣

↪use\_colnames=**True**, max\_len = 2)

rules = association\_rules(frequent\_itemsets, metric="lift", min\_threshold = 1.

↪5)

display(rules.head())

print("Rules identified: ", len(rules))

antecedents consequents antecedent support \

|  |  |  |
| --- | --- | --- |
| 0 (UHT-milk) | (butter milk) | 0.021386 |
| 1 (butter milk) | (UHT-milk) | 0.017577 |
| 2 (UHT-milk) | (cream cheese ) | 0.021386 |
| 3 (cream cheese ) | (UHT-milk) | 0.023658 |
| 4 (soda) | (artif. sweetener) | 0.097106 |

consequent support support confidence lift leverage conviction \ 0 0.017577 0.000601 0.028125 1.600131 0.000226 1.010854

1 0.021386 0.000601 0.034221 1.600131 0.000226 1.013289

2 0.023658 0.000869 0.040625 1.717152 0.000363 1.017685

3 0.021386 0.000869 0.036723 1.717152 0.000363 1.015922

4 0.001938 0.000468 0.004818 2.485725 0.000280 1.002893

zhangs\_metric 0 0.383247

1 0.381761

2 0.426767

3 0.427761

[21]:

4 0.661986

Rules identified: 190

sns.set(style = "whitegrid")

fig = plt.figure(figsize=(12, 12))

ax = fig.add\_subplot(projection = '3d')

x = rules['support']

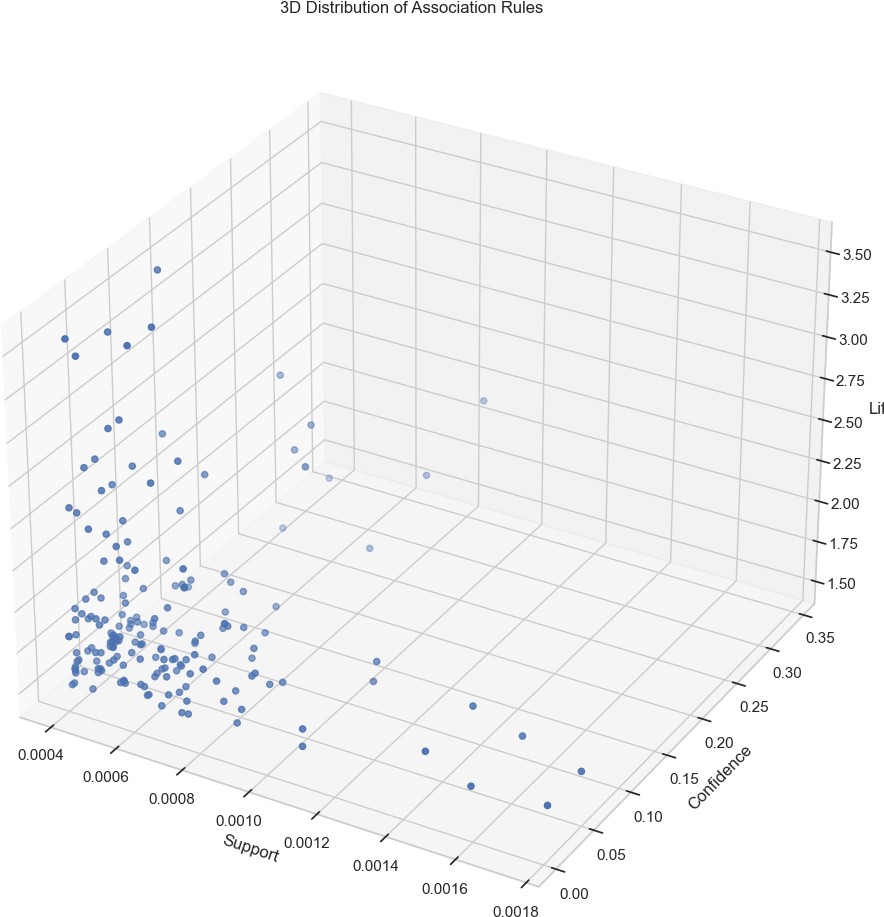
y = rules['confidence'] z = rules['lift']

ax.set\_xlabel("Support") ax.set\_ylabel("Confidence") ax.set\_zlabel("Lift")

ax.scatter(x, y, z)

ax.set\_title("3D Distribution of Association Rules")

plt.show()



[23]:

**def** draw\_network(rules, rules\_to\_show): *# Directional Graph from NetworkX* network = nx.DiGraph()

*# Loop through number of rules to show*

**for** i **in** range(rules\_to\_show):

*# Add a Rule Node*

network.add\_nodes\_from(["R"+str(i)])

**for** antecedents **in** rules.iloc[i]['antecedents']: *# Add antecedent node and link to rule* network.add\_nodes\_from([antecedents])

network.add\_edge(antecedents, "R"+str(i), weight = 2)

**for** consequents **in** rules.iloc[i]['consequents']: *# Add consequent node and link to rule* network.add\_nodes\_from([consequents])

network.add\_edge("R"+str(i), consequents, weight = 2)

color\_map=[]

*# For every node, if it's a rule, colour as Black, otherwise Orange*

**for** node **in** network:

**if** re.compile("^[R]\d+$").fullmatch(node) != **None**: color\_map.append('black')

# else:

color\_map.append('orange')

*# Position nodes using spring layout*

pos = nx.spring\_layout(network, k=16, scale=1)

*# Draw the network graph*

nx.draw(network, pos, node\_color = color\_map, font\_size=8)

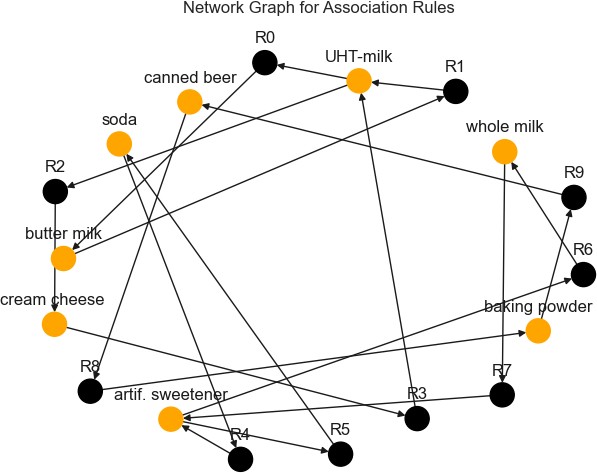
*# Shift the text position upwards*

**for** p **in** pos:

pos[p][1] += 0.12

nx.draw\_networkx\_labels(network, pos) plt.title("Network Graph for Association Rules") plt.show()

draw\_network(rules, 10)



[24]:

milk\_rules = rules[rules['consequents'].astype(str).str.contains('whole milk')] milk\_rules = milk\_rules.sort\_values(by=['lift'],ascending = [**False**]).

↪reset\_index(drop = **True**)

display(milk\_rules.head())

|  |  |  |  |
| --- | --- | --- | --- |
| antecedents | consequents | antecedent support | consequent support \ |
| 0 (brandy) | (whole milk) | 0.002540 | 0.157923 |
| 1 (softener) | (whole milk) | 0.002740 | 0.157923 |
| 2 (canned fruit) | (whole milk) | 0.001403 | 0.157923 |
| 3 (syrup) | (whole milk) | 0.001403 | 0.157923 |
| 4 (artif. sweetener) | (whole milk) | 0.001938 | 0.157923 |
| support confidence | lift leverage conviction | | zhangs\_metric |
| 0 0.000869 0.342105 | 2.166281 0.000468 1.279957 | | 0.539750 |
| 1 0.000802 0.292683 | 1.853328 0.000369 1.190523 | | 0.461695 |
| 2 0.000401 0.285714 | 1.809201 0.000179 1.178908 | | 0.447899 |
| 3 0.000401 0.285714 | 1.809201 0.000179 1.178908 | | 0.447899 |
| 4 0.000535 0.275862 | 1.746815 0.000229 1.162868 | | 0.428360 |