In [1]:

*#Name:Pawar Ved Balasaheb(T512037)*

**import** numpy **as** np

**import** pandas **as** pd

**import** os

**for** dirname, \_, filenames **in** os**.**walk('/kaggle/input'):

**for** filename **in** filenames:

print(os**.**path**.**join(dirname, filename))

In [3]:

**!**pip install mlxtend

Defaulting to user installation because normal site-packages is not writeable

Requirement already satisfied: mlxtend in c:\users\student\appdata\roaming\python

\python312\site-packages (0.23.1)

Requirement already satisfied: scipy>=1.2.1 in c:\programdata\anaconda3\lib\site- packages (from mlxtend) (1.13.1)

Requirement already satisfied: numpy>=1.16.2 in c:\programdata\anaconda3\lib\site

-packages (from mlxtend) (1.26.4)

Requirement already satisfied: pandas>=0.24.2 in c:\programdata\anaconda3\lib\sit e-packages (from mlxtend) (2.2.2)

Requirement already satisfied: scikit-learn>=1.0.2 in c:\programdata\anaconda3\li b\site-packages (from mlxtend) (1.4.2)

Requirement already satisfied: matplotlib>=3.0.0 in c:\programdata\anaconda3\lib

\site-packages (from mlxtend) (3.8.4)

Requirement already satisfied: joblib>=0.13.2 in c:\programdata\anaconda3\lib\sit e-packages (from mlxtend) (1.4.2)

Requirement already satisfied: contourpy>=1.0.1 in c:\programdata\anaconda3\lib\s ite-packages (from matplotlib>=3.0.0->mlxtend) (1.2.0)

Requirement already satisfied: cycler>=0.10 in c:\programdata\anaconda3\lib\site- packages (from matplotlib>=3.0.0->mlxtend) (0.11.0)

Requirement already satisfied: fonttools>=4.22.0 in c:\programdata\anaconda3\lib

\site-packages (from matplotlib>=3.0.0->mlxtend) (4.51.0)

Requirement already satisfied: kiwisolver>=1.3.1 in c:\programdata\anaconda3\lib

\site-packages (from matplotlib>=3.0.0->mlxtend) (1.4.4)

Requirement already satisfied: packaging>=20.0 in c:\programdata\anaconda3\lib\si te-packages (from matplotlib>=3.0.0->mlxtend) (23.2)

Requirement already satisfied: pillow>=8 in c:\programdata\anaconda3\lib\site-pac kages (from matplotlib>=3.0.0->mlxtend) (10.4.0)

Requirement already satisfied: pyparsing>=2.3.1 in c:\programdata\anaconda3\lib\s ite-packages (from matplotlib>=3.0.0->mlxtend) (3.0.9)

Requirement already satisfied: python-dateutil>=2.7 in c:\programdata\anaconda3\l ib\site-packages (from matplotlib>=3.0.0->mlxtend) (2.9.0.post0)

Requirement already satisfied: pytz>=2020.1 in c:\programdata\anaconda3\lib\site- packages (from pandas>=0.24.2->mlxtend) (2024.1)

Requirement already satisfied: tzdata>=2022.7 in c:\programdata\anaconda3\lib\sit e-packages (from pandas>=0.24.2->mlxtend) (2023.3)

Requirement already satisfied: threadpoolctl>=2.0.0 in c:\programdata\anaconda3\l ib\site-packages (from scikit-learn>=1.0.2->mlxtend) (2.2.0)

Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\lib\site-pack ages (from python-dateutil>=2.7->matplotlib>=3.0.0->mlxtend) (1.16.0)

In [5]:

**import** csv

**from** mlxtend.preprocessing **import** TransactionEncoder

**from** mlxtend.frequent\_patterns **import** apriori, association\_rules

In [7]:

data **=** []

**with** open('Market\_Basket\_Optimisation.csv') **as** file: reader **=** csv**.**reader(file, delimiter**=**',')

**for** row **in** reader: data **+=**[row]

In [9]:

data[1:10] *#list of list*

Out[9]:

In [11]:

Out[11]:

In [13]:

te **=** TransactionEncoder() x **=** te**.**fit\_transform(data)

[['burgers', 'meatballs', 'eggs'], ['chutney'],

['turkey', 'avocado'],

['mineral water', 'milk', 'energy bar', 'whole wheat rice', 'green tea'], ['low fat yogurt'],

['whole wheat pasta', 'french fries'], ['soup', 'light cream', 'shallot'],

['frozen vegetables', 'spaghetti', 'green tea'], ['french fries']]

len(data)

7501

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| In [15]: | x |  | | | | | |
| Out[15]: | array([[False, | True, | True, | ..., | True, | False, | False], |
|  | [False, | False, | False, | ..., | False, | False, | False], |
|  | [False, | False, | False, | ..., | False, | False, | False], |
|  | ..., |  |  |  |  |  |  |
|  | [False, | False, | False, | ..., | False, | False, | False], |
|  | [False, | False, | False, | ..., | False, | False, | False], |
|  | [False, | False, | False, | ..., | False, | True, | False]]) |
| In [17]: | te**.**columns\_ |  |  |  |  |  |  |

Out[17]:

[' asparagus', 'almonds',

'antioxydant juice', 'asparagus',

'avocado',

'babies food', 'bacon',

'barbecue sauce', 'black tea',

'blueberries', 'body spray', 'bramble',

'brownies', 'bug spray',

'burger sauce', 'burgers',

'butter',

'cake',

'candy bars', 'carrots',

'cauliflower', 'cereals',

'champagne', 'chicken',

'chili',

'chocolate',

'chocolate bread', 'chutney',

'cider',

'clothes accessories', 'cookies',

'cooking oil', 'corn',

'cottage cheese', 'cream',

'dessert wine', 'eggplant',

'eggs',

'energy bar',

'energy drink', 'escalope',

'extra dark chocolate', 'flax seed',

'french fries', 'french wine', 'fresh bread', 'fresh tuna',

'fromage blanc',

'frozen smoothie',

'frozen vegetables', 'gluten free bar',

'grated cheese', 'green beans',

'green grapes', 'green tea',

'ground beef', 'gums',

'ham',

'hand protein bar', 'herb & pepper',

'honey',

'hot dogs', 'ketchup',

'light cream', 'light mayo',

'low fat yogurt', 'magazines',

'mashed potato', 'mayonnaise',

'meatballs', 'melons',

'milk',

'mineral water', 'mint',

'mint green tea', 'muffins',

'mushroom cream sauce', 'napkins',

'nonfat milk', 'oatmeal',

'oil',

'olive oil', 'pancakes',

'parmesan cheese', 'pasta',

'pepper',

'pet food', 'pickles',

'protein bar', 'red wine',

'rice',

'salad',

'salmon',

'salt',

'sandwich', 'shallot', 'shampoo', 'shrimp',

'soda',

'soup',

'spaghetti',

'sparkling water', 'spinach',

'strawberries', 'strong cheese', 'tea',

'tomato juice', 'tomato sauce', 'tomatoes',

'toothpaste', 'turkey',

'vegetables mix', 'water spray',

'white wine',

'whole weat flour', 'whole wheat pasta', 'whole wheat rice', 'yams',

'yogurt cake', 'zucchini']

In [19]:

df **=** pd**.**DataFrame(x, columns**=**te**.**columns\_)

In [21]:

df

Out[21]:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **asparagus** | **almonds** | **antioxydant**  **juice** | **asparagus** | **avocado** | **babies food** | **bacon** | **barbecue**  **sauce** |
| **0** | False | True | True | False | True | False | False | False |
| **1** | False | False | False | False | False | False | False | False |
| **2** | False | False | False | False | False | False | False | False |
| **3** | False | False | False | False | True | False | False | False |
| **4** | False | False | False | False | False | False | False | False |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **7496** | False | False | False | False | False | False | False | False |
| **7497** | False | False | False | False | False | False | False | False |
| **7498** | False | False | False | False | False | False | False | False |
| **7499** | False | False | False | False | False | False | False | False |
| **7500** | False | False | False | False | False | False | False | False |

# 7501 rows × 120 columns

freq\_itemset **=** apriori(df, min\_support**=**0.01, use\_colnames**=True**)

In [23]:

In [25]:

freq\_itemset

|  |  |  |  |
| --- | --- | --- | --- |
| Out[25]: |  | **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, mineral water, milk) |
|  | **253** | 0.017064 | (ground beef, mineral water, spaghetti) |
|  | **254** | 0.015731 | (mineral water, spaghetti, milk) |
|  | **255** | 0.010265 | (olive oil, mineral water, spaghetti) |
|  | **256** | 0.011465 | (mineral water, pancakes, spaghetti) |

# 257 rows × 2 columns

In [27]:

rules **=** association\_rules(freq\_itemset, metric**=**'confidence', min\_threshold**=**0.10)

In [29]:

rules **=** rules[['antecedents','consequents','support','confidence']] rules

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Out[29]: | **antecedents** | **consequents** | **support** | **confidence** |
|  | **0** (avocado) | (mineral water) | 0.011598 | 0.348000 |
|  | **1** (burgers) | (cake) | 0.011465 | 0.131498 |
|  | **2** (cake) | (burgers) | 0.011465 | 0.141447 |
|  | **3** (chocolate) | (burgers) | 0.017064 | 0.104150 |
|  | **4** (burgers) | (chocolate) | 0.017064 | 0.195719 |
|  | **...** ... | ... | ... | ... |
|  | **315** (olive oil) | (mineral water, spaghetti) | 0.010265 | 0.155870 |
|  | **316** (mineral water, pancakes) | (spaghetti) | 0.011465 | 0.339921 |
|  | **317** (mineral water, spaghetti) | (pancakes) | 0.011465 | 0.191964 |
|  | **318** (spaghetti, pancakes) | (mineral water) | 0.011465 | 0.455026 |
|  | **319** (pancakes) | (mineral water, spaghetti) | 0.011465 | 0.120617 |

# 320 rows × 4 columns

In [31]:

rules[rules['antecedents'] **==** {'cake'}]['consequents']

Out[31]:

In [ ]:

2 (burgers)

25 (chocolate)

27 (eggs)

29 (french fries)

31 (frozen vegetables)

32 (green tea)

34 (milk)

37 (mineral water)

39 (pancakes)

41 (spaghetti)

Name: consequents, dtype: object