Assignment No.6

Assignment on Association Rule Learning

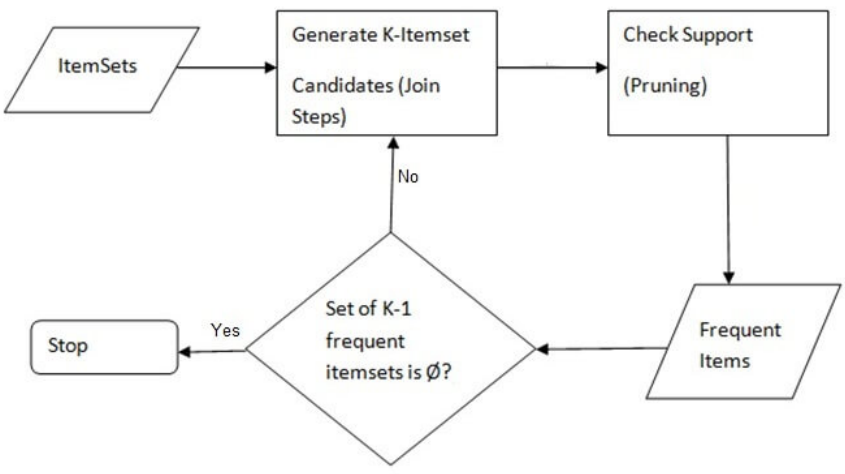
Download Market Basket Optimization dataset from below link.
Data Set: https://www.kaggle.com/hemanthkumar05/market-basket-optimization
This dataset comprises the list of transactions of a retail company over the period of one week. It contains a total of 7501 transaction records where each record consists of the list of items sold in one transaction. Using this record of transactions and items in each transaction, find the association rules between items.
There is no header in the dataset and the first row contains the first transaction, so mentioned header = None here while loading dataset.

a. Follow following steps :
b. Data Preprocessing
c. Generate the list of transactions from the dataset
d. Train Apriori algorithm on the dataset
e. Visualize the list of rules

Theory:

Apriori is a popular algorithm for extracting frequent itemsets with applications in association rule learning. The apriori algorithm has been designed to operate on databases containing transactions, such as purchases by customers of a store. An itemset is considered as "frequent" if it meets a user-specified support threshold. For instance, if the support threshold is set to 0.5 (50%), a frequent itemset is defined as a set of items that occur together in at least 50% of all transactions in the database.

Applying apriori



How does Apriori Algorithm Work ?¶

A key concept in Apriori algorithm is the anti-monotonicity of the support measure. It assumes that

- All subsets of a frequent itemset must be frequent
- Similarly, for any infrequent itemset, all its supersets must be infrequent too

Step 1: Create a frequency table of all the items that occur in all the transactions.

Step 2: We know that only those elements are significant for which the support is greater than or equal to the threshold support.

Step 3: The next step is to make all the possible pairs of the significant items keeping in mind that the order doesn't matter, i.e., AB is same as BA.

Step 4: We will now count the occurrences of each pair in all the transactions

Step 5: Again only those itemsets are significant which cross the support threshold

Step 6: Now let's say we would like to look for a set of three items that are purchased together. We will use the itemsets found in step 5 and create a set of 3 items.

Code:

```
import pandas as pd
import numpy as np
!pip install wordcloud
```

```
Requirement already satisfied: wordcloud in
/usr/local/lib/python3.7/dist-packages (1.5.0)
Requirement already satisfied: numpy>=1.6.1 in
/usr/local/lib/python3.7/dist-packages (from wordcloud) (1.19.5)
Requirement already satisfied: pillow in /usr/local/lib/python3.7/dist-
packages (from wordcloud) (7.1.2)
```

```
import matplotlib.pyplot as plt

import seaborn as sns
plt.style.use('fivethirtyeight')
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
```

Importing the dataset

```
data = pd.read_csv('/content/Market_Basket_Optimisation (1).csv')
data.shape
(7501, 20)
data.head()
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
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	spinach	olive oil
1	burgers	meatballs	eggs	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	chutney	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	turkey	avocado	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	mineral water	milk	energy bar	whole wheat rice	green tea	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

```
data.tail()
```

	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

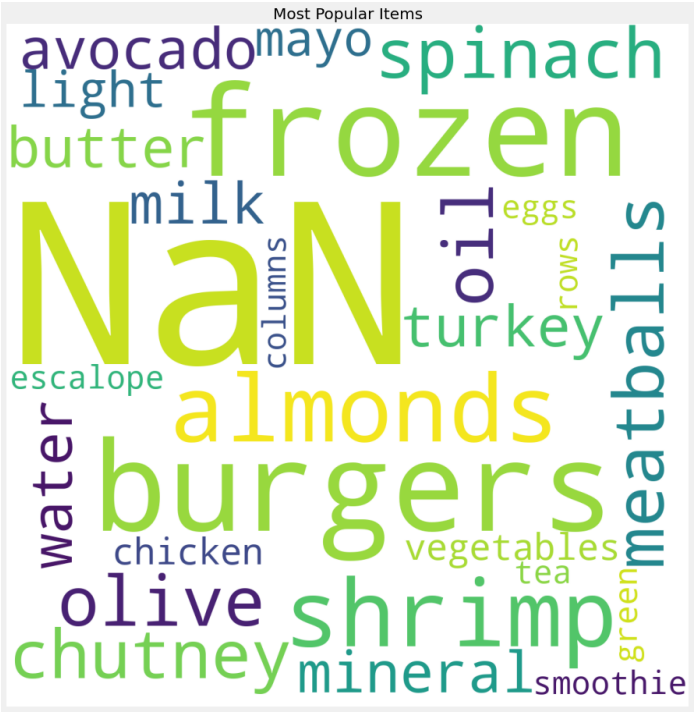
```
data.sample(10)
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
5097	pancakes	cake	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2162	spaghetti	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5967	red wine	tomatoes	tea	escalope	champagne	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7069	burgers	ground beef	energy drink	melons	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5092	spaghetti	honey	french fries	escalope	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2960	cider	french fries	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5903	olive oil	muffins	mint	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4789	frozen vegetables	chocolate	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2704	frozen vegetables	spaghetti	meatballs	chutney	chicken	frozen smoothie	escalope	pancakes	toothpaste	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2746	burgers	pancakes	cake	chocolate	low fat yogurt	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Data Visualizations

```
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud

plt.rcParams['figure.figsize'] = (15, 15)
wc = WordCloud(background_color = 'white', width = 1200,  height = 1200
, max_words = 121).generate(str(data))
plt.imshow(wc)
plt.axis('off')
plt.title('Most Popular Items',fontsize = 20)
plt.show()
```



Data Preprocessing

```
# making each customers shopping items an identical list
trans = []
for i in range(0, 7501):
    trans.append([str(data.values[i,j]) for j in range(0, 20)])

# conveting it into an numpy array
trans = np.array(trans)

# checking the shape of the array
print(trans.shape)
```

```
(7501, 20)
```

```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder

te = TransactionEncoder()
data = te.fit_transform(trans)
data = pd.DataFrame(data, columns = te.columns_)

data.shape
```

```
(7501, 121)
```

```
import warnings
warnings.filterwarnings('ignore')

# getting correlations for 121 items would be messy
# so let's reduce the items from 121 to 50
```

```
data = data.loc[:, ['mineral water', 'burgers', 'turkey', 'chocolate',
'frozen vegetables', 'spaghetti',
                    'shrimp', 'grated cheese', 'eggs', 'cookies', 'fren
ch fries', 'herb & pepper', 'ground beef',
                    'tomatoes', 'milk', 'escalope', 'fresh tuna', 'red
wine', 'ham', 'cake', 'green tea',
                    'whole wheat pasta', 'pancakes', 'soup', 'muffins',
 'energy bar', 'olive oil', 'champagne',
                    'avocado', 'pepper', 'butter', 'parmesan cheese', '
whole wheat rice', 'low fat yogurt', 'chicken', 'vegetables mix', 'pick
les', 'meatballs', 'frozen smoothie', 'yogurt cake']]

# checking the shape
data.shape
(7501, 40)

data.columns
Index(['mineral water', 'burgers', 'turkey', 'chocolate', 'frozen
vegetables',
       'spaghetti', 'shrimp', 'grated cheese', 'eggs', 'cookies',
       'french fries', 'herb & pepper', 'ground beef', 'tomatoes',
'milk',
       'escalope', 'fresh tuna', 'red wine', 'ham', 'cake', 'green
tea',
       'whole wheat pasta', 'pancakes', 'soup', 'muffins', 'energy
bar',
       'olive oil', 'champagne', 'avocado', 'pepper', 'butter',
       'parmesan cheese', 'whole wheat rice', 'low fat yogurt',
'chicken',
       'vegetables mix', 'pickles', 'meatballs', 'frozen smoothie',
       'yogurt cake'],
      dtype='object')

data.head()
```

	mineral water	burgers	turkey	chocolate	frozen vegetables	spaghetti	shrimp	grated cheese	eggs	cookies	french fries	herb & pepper	ground beef	tomatoes	milk	escalope	fresh tuna	red wine	ham	cake	green tea	wh wh pa
0	True	False	False	False	False	False	True	False	False	False	False	False	False	False	False	False	False	False	False	False	True	F
1	False	True	False	False	False	False	False	False	True	False	False	False	False	False	False	False	False	False	False	False	False	F
2	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	F
3	False	False	True	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	F
4	True	False	False	False	False	False	False	False	False	False	False	False	False	False	True	False	False	False	False	False	True	F

```
from mlxtend.frequent_patterns import apriori

#Now, let us return the items and itemsets with at least 5% support:
apriori(data, min_support = 0.01, use_colnames = True)
```

	support	itemsets
0	0.238368	(mineral water)
1	0.087188	(burgers)
2	0.062525	(turkey)
3	0.163845	(chocolate)
4	0.095321	(frozen vegetables)
...	...	...
204	0.010132	(ground beef, mineral water, eggs)
205	0.013065	(milk, mineral water, eggs)
206	0.011065	(ground beef, mineral water, milk)
207	0.010532	(chocolate, spaghetti, eggs)
208	0.010932	(chocolate, spaghetti, milk)

209 rows × 2 columns

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

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

```
frequent_itemsets[ (frequent_itemsets['length'] == 2) &
                   (frequent_itemsets['support'] >= 0.01) ]
```

	support	itemsets	length
24	0.052660	(chocolate, mineral water)	2
25	0.059725	(spaghetti, mineral water)	2
26	0.050927	(mineral water, eggs)	2

```
# getting th item sets with length = 2 and support more han 10%

frequent_itemsets[ (frequent_itemsets['length'] == 1) &
                   (frequent_itemsets['support'] >= 0.01) ]
```

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

```
frequent_itemsets[ frequent_itemsets['itemsets'] == {'eggs', 'mineral water'} ]
```

	support	itemsets	length
26	0.050927	(mineral water, eggs)	2

```
frequent_itemsets[ frequent_itemsets['itemsets'] == {'mineral water'} ]
```

	support	itemsets	length
0	0.238368	(mineral water)	1

```
frequent_itemsets[ frequent_itemsets['itemsets'] == {'milk'} ]
```

	support	itemsets	length
13	0.129583	(milk)	1

```
frequent_itemsets[ frequent_itemsets['itemsets'] == {'chicken'} ]
```

	support	itemsets	length
22	0.059992	(chicken)	1

```
frequent_itemsets[ frequent_itemsets['itemsets'] == {'frozen vegetables'} ]
```

	support	itemsets	length
4	0.095321	(frozen vegetables)	1