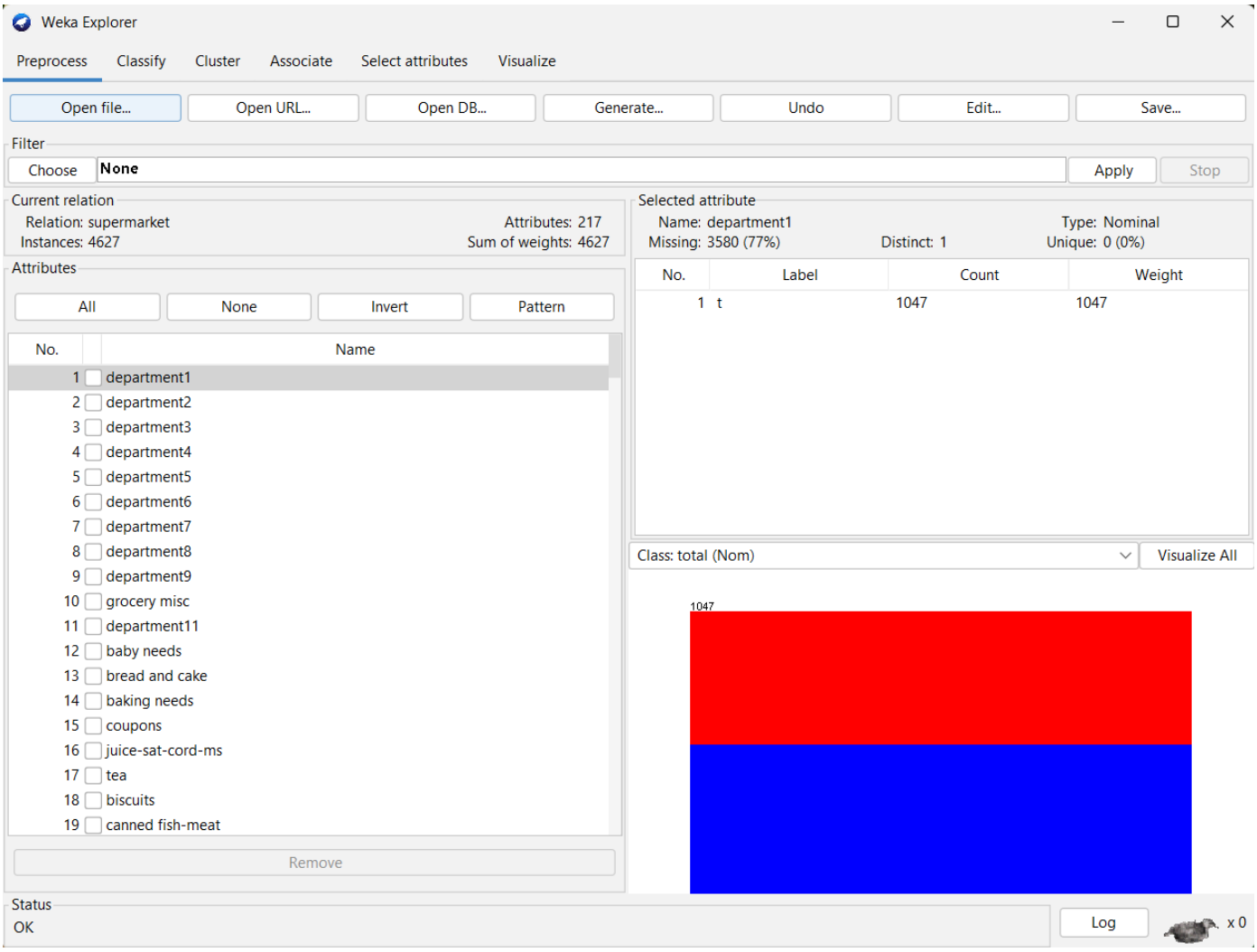
Knowledge Discovery & Data Mining Lab6

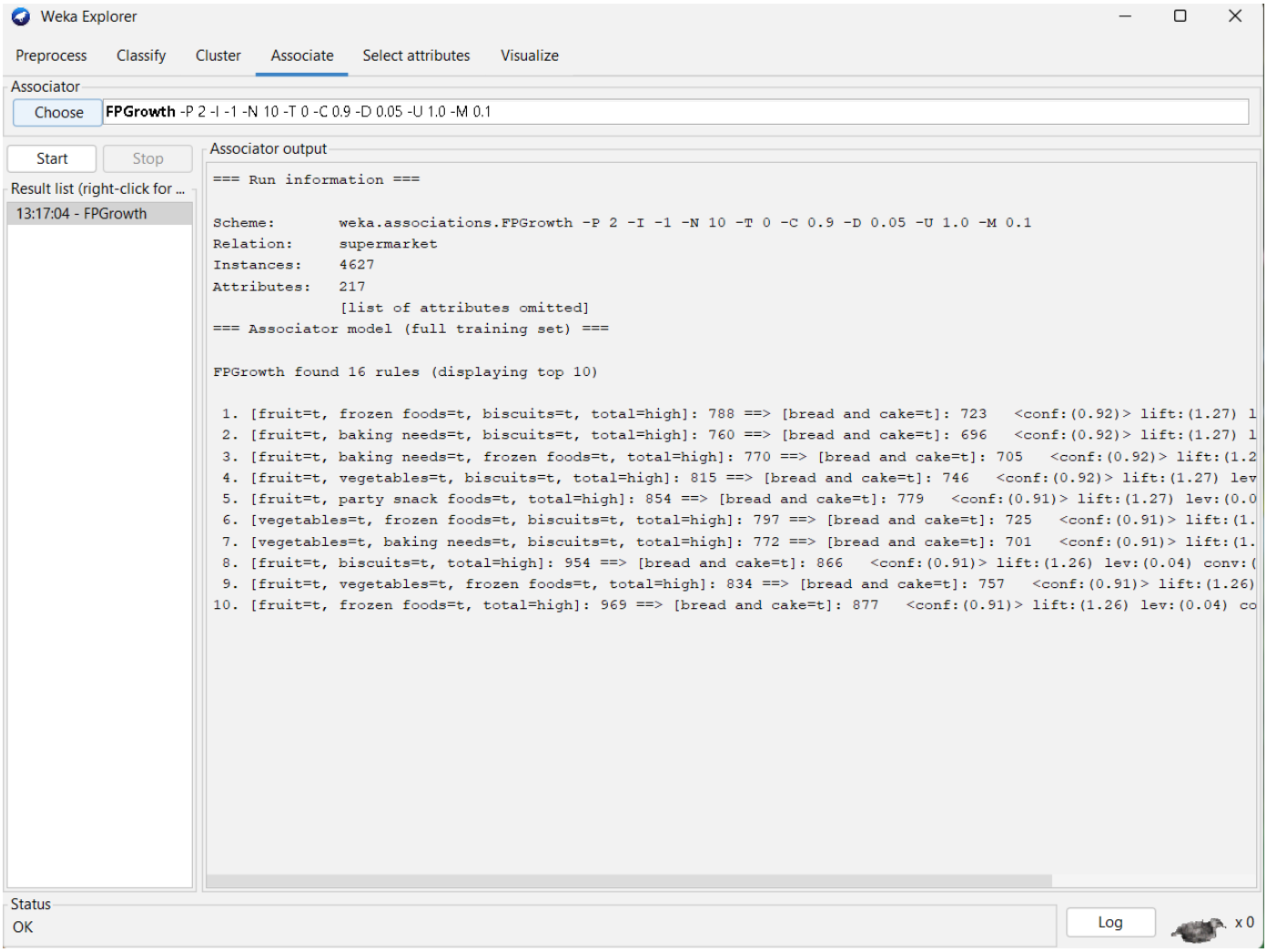
Name : Kunal Sanjay Patil

PRN : 20190802025

AIM :

To implement the FP Growth Algorithm using WEKA and python.




Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Associator
Choose
FPGrowth -P 2 -I -1 -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1
Start
Stop
Result list (right-click for ...
13:17:04 - FPGrowth
Associator output
=== Run information ===
Scheme: weka.associations.FPGrowth -P 2 -I -1 -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1
Relation: supermarket
Instances: 4627
Attributes: 217
[list of attributes omitted]
=== Associator model (full training set) ===
FPGrowth found 16 rules (displaying top 10)
1. [fruit=t, frozen foods=t, biscuits=t, total=high]: 788 ==> [bread and cake=t]: 723 <conf:(0.92)> lift:(1.27) l
2. [fruit=t, baking needs=t, biscuits=t, total=high]: 760 ==> [bread and cake=t]: 696 <conf:(0.92)> lift:(1.27) l
3. [fruit=t, baking needs=t, frozen foods=t, total=high]: 770 ==> [bread and cake=t]: 705 <conf:(0.92)> lift:(1.2
4. [fruit=t, vegetables=t, biscuits=t, total=high]: 815 ==> [bread and cake=t]: 746 <conf:(0.92)> lift:(1.27) lev
5. [fruit=t, party snack foods=t, total=high]: 854 ==> [bread and cake=t]: 779 <conf:(0.91)> lift:(1.27) lev:(0.0
6. [vegetables=t, frozen foods=t, biscuits=t, total=high]: 797 ==> [bread and cake=t]: 725 <conf:(0.91)> lift:(1.
7. [vegetables=t, baking needs=t, biscuits=t, total=high]: 772 ==> [bread and cake=t]: 701 <conf:(0.91)> lift:(1.
8. [fruit=t, biscuits=t, total=high]: 954 ==> [bread and cake=t]: 866 <conf:(0.91)> lift:(1.26) lev:(0.04) conv:(
9. [fruit=t, vegetables=t, frozen foods=t, total=high]: 834 ==> [bread and cake=t]: 757 <conf:(0.91)> lift:(1.26)
10. [fruit=t, frozen foods=t, total=high]: 969 ==> [bread and cake=t]: 877 <conf:(0.91)> lift:(1.26) lev:(0.04) co
Status
OK
Log
x 0

In [1]:

```
pip install mlxtend
```

```
Requirement already satisfied: mlxtend in c:\users\kunal\anaconda3\lib\site-
packages (0.19.0)
Requirement already satisfied: numpy>=1.16.2 in c:\users\kunal\anaconda3\lib
\site-packages (from mlxtend) (1.20.1)
Requirement already satisfied: joblib>=0.13.2 in c:\users\kunal\anaconda3\li
b\site-packages (from mlxtend) (1.0.1)
Requirement already satisfied: pandas>=0.24.2 in c:\users\kunal\anaconda3\li
b\site-packages (from mlxtend) (1.2.4)
Requirement already satisfied: setuptools in c:\users\kunal\anaconda3\lib\si
te-packages (from mlxtend) (52.0.0.post20210125)
Requirement already satisfied: scipy>=1.2.1 in c:\users\kunal\anaconda3\lib
\site-packages (from mlxtend) (1.6.2)
Requirement already satisfied: scikit-learn>=0.20.3 in c:\users\kunal\anacon
da3\lib\site-packages (from mlxtend) (0.24.1)
Requirement already satisfied: matplotlib>=3.0.0 in c:\users\kunal\anaconda3
\lib\site-packages (from mlxtend) (3.3.4)
Requirement already satisfied: cycler>=0.10 in c:\users\kunal\anaconda3\lib
\site-packages (from matplotlib>=3.0.0->mlxtend) (0.10.0)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\kunal\anacon
da3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (2.8.1)
Requirement already satisfied: pillow>=6.2.0 in c:\users\kunal\anaconda3\lib
\site-packages (from matplotlib>=3.0.0->mlxtend) (8.2.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\kunal\anaconda3
\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (1.3.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in
c:\users\kunal\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend)
(2.4.7)
Requirement already satisfied: six in c:\users\kunal\anaconda3\lib\site-pack
ages (from cycler>=0.10->matplotlib>=3.0.0->mlxtend) (1.15.0)
Requirement already satisfied: pytz>=2017.3 in c:\users\kunal\anaconda3\lib
\site-packages (from pandas>=0.24.2->mlxtend) (2021.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\kunal\anacon
da3\lib\site-packages (from scikit-learn>=0.20.3->mlxtend) (2.1.0)
Note: you may need to restart the kernel to use updated packages.
```

In [2]:

```
import pandas as pd
import numpy as np
```

In [3]:

```
data = pd.read_csv('Market_Basket_Optimisation.csv', header=None)
data.head()
```

Out[3]:

	0	1	2	3	4	5	6	7	8	9	10
0	shrimp	almonds	avocado	vegetables mix	green grapes	whole weat flour	yams	cottage cheese	energy drink	tomato juice	low fat yogurt
1	burgers	meatballs	eggs	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	chutney	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	turkey	avocado	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

In [4]:

```
data.shape
```

Out[4]:

```
(7501, 20)
```

In [5]:

```
transaction = []
for i in range(0, data.shape[0]):
    for j in range(0, data.shape[1]):
        transaction.append(data.values[i,j])

# converting to numpy array
transaction = np.array(transaction)
```

In [6]:

```
df = pd.DataFrame(transaction, columns=["items"])
df["incident_count"] = 1

# Delete NaN Items from Dataset
indexNames = df[df['items'] == "nan" ].index
df.drop(indexNames , inplace=True)
```

In [7]:

```
transaction = []
for i in range(data.shape[0]):
    transaction.append([str(data.values[i,j]) for j in range(data.shape[1])])

# creating the numpy array of the transactions
transaction = np.array(transaction)

# importing the required module
from mlxtend.preprocessing import TransactionEncoder

# initializing the transactionEncoder
te = TransactionEncoder()
te_ary = te.fit(transaction).transform(transaction)
dataset = pd.DataFrame(te_ary, columns=te.columns_)

# dataset after encoded
dataset.head()
```

Out[7]:

	asparagus	almonds	antioxydant juice	asparagus	avocado	babies food	bacon	barbecue sauce	black tea	blu
0	False	True	True	False	True	False	False	False	False	
1	False	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	False	
3	False	False	False	False	True	False	False	False	False	
4	False	False	False	False	False	False	False	False	False	

5 rows × 121 columns

In [8]:

```
from mlxtend.frequent_patterns import fpgrowth

#running the fpgrowth algorithm
res=fpgrowth(dataset,min_support=0.05, use_colnames=True)

# printing top 10
res.head(10)
```

Out[8]:

	support	itemsets
0	0.238368	(mineral water)
1	0.132116	(green tea)
2	0.076523	(low fat yogurt)
3	0.071457	(shrimp)
4	0.065858	(olive oil)
5	0.063325	(frozen smoothie)
6	0.999867	(nan)
7	0.179709	(eggs)
8	0.087188	(burgers)
9	0.062525	(turkey)

In [9]:

```
from mlxtend.frequent_patterns import association_rules

# creating asssociation rules
res=association_rules(res, metric="confidence", min_threshold=0.06)

# printing association rules
res
```

Out[9]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	levera
0	(mineral water)	(nan)	0.238368	0.999867	0.238235	0.999441	0.999574	-0.000
1	(nan)	(mineral water)	0.999867	0.238368	0.238235	0.238267	0.999574	-0.000
2	(nan)	(green tea)	0.999867	0.132116	0.131982	0.132000	0.999124	-0.000
3	(green tea)	(nan)	0.132116	0.999867	0.131982	0.998991	0.999124	-0.000
4	(nan)	(low fat yogurt)	0.999867	0.076523	0.076390	0.076400	0.998391	-0.000
...	...	...	...	...	...	...	...	
62	(nan)	(ground beef)	0.999867	0.098254	0.098254	0.098267	1.000133	0.000
63	(nan)	(escalope)	0.999867	0.079323	0.079323	0.079333	1.000133	0.000
64	(escalope)	(nan)	0.079323	0.999867	0.079323	1.000000	1.000133	0.000
65	(cake)	(nan)	0.081056	0.999867	0.081056	1.000000	1.000133	0.000
66	(nan)	(cake)	0.999867	0.081056	0.081056	0.081067	1.000133	0.000

67 rows × 9 columns