sket-by-using-apriori-algorithm-1

September 19, 2024

[2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

import seaborn as sns
```

[6]:
```
df = pd.read_csv("bread basket.csv")
df.head()
```

[6]:

	Transaction	Item	date_time	period_day	weekday_weekend
0	1	Bread	30-10-2016 09:58	morning	weekend
1	2	Scandinavian	30-10-2016 10:05	morning	weekend
2	2	Scandinavian	30-10-2016 10:05	morning	weekend
3	3	Hot chocolate	30-10-2016 10:07	morning	weekend
4	3	Jam	30-10-2016 10:07	morning	weekend

[7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20507 entries, 0 to 20506
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Transaction      20507 non-null  int64 
 1   Item             20507 non-null  object
 2   date_time        20507 non-null  object
 3   period_day       20507 non-null  object
 4   weekday_weekend  20507 non-null  object
dtypes: int64(1), object(4)
memory usage: 801.2+ KB
```

[8]:
```
df.describe()
```

[8]:

	Transaction
count	20507.000000
mean	4976.202370
std	2796.203001
min	1.000000

```
25%     2552.000000
50%     5137.000000
75%     7357.000000
max     9684.000000
```

[9]:
```
# Converting the 'date_time' column into the right format
df['date_time'] = pd.to_datetime(df['date_time'])
```

[10]:
```
df.head(10)
```

[10]:

	Transaction	Item	date_time	period_day	weekday_weekend
0	1	Bread	2016-10-30 09:58:00	morning	weekend
1	2	Scandinavian	2016-10-30 10:05:00	morning	weekend
2	2	Scandinavian	2016-10-30 10:05:00	morning	weekend
3	3	Hot chocolate	2016-10-30 10:07:00	morning	weekend
4	3	Jam	2016-10-30 10:07:00	morning	weekend
5	3	Cookies	2016-10-30 10:07:00	morning	weekend
6	4	Muffin	2016-10-30 10:08:00	morning	weekend
7	5	Coffee	2016-10-30 10:13:00	morning	weekend
8	5	Pastry	2016-10-30 10:13:00	morning	weekend
9	5	Bread	2016-10-30 10:13:00	morning	weekend

[11]:
```
#Count of unique customers
df['Transaction'].nunique()
```

[11]:
```
9465
```

[12]:
```
# Extracting date
df['date'] = df['date_time'].dt.date

#Extracting time
df['time'] = df['date_time'].dt.time

# Extracting month and replacing it with text
df['month'] = df['date_time'].dt.month
df['month'] = df['month'].replace((1,2,3,4,5,6,7,8,9,10,11,12),
 ↪('January','February','March','April','May','June','July','August',
 ↪'September','October','November','December'))

# Extracting hour
df['hour'] = df['date_time'].dt.hour
# Replacing hours with text
hour_in_num = (1,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23)
hour_in_obj = ('1-2','7-8','8-9','9-10','10-11','11-12','12-13','13-14','14-15',
```

```
                ␣
  ↪'15-16','16-17','17-18','18-19','19-20','20-21','21-22','22-23','23-24')
df['hour'] = df['hour'].replace(hour_in_num, hour_in_obj)

# Extracting weekday and replacing it with text
df['weekday'] = df['date_time'].dt.weekday
df['weekday'] = df['weekday'].replace((0,1,2,3,4,5,6),
                                              ␣
  ↪('Monday','Tuesday','Wednesday','Thursday','Friday','Saturday','Sunday'))

# dropping date_time column
df.drop('date_time', axis = 1, inplace = True)
```

[13]:
```
df.head()
```

[13]:
```
   Transaction           Item period_day weekday_weekend        date  \
0            1          Bread    morning         weekend  2016-10-30
1            2   Scandinavian    morning         weekend  2016-10-30
2            2   Scandinavian    morning         weekend  2016-10-30
3            3  Hot chocolate    morning         weekend  2016-10-30
4            3            Jam    morning         weekend  2016-10-30

       time    month   hour weekday
0  09:58:00  October   9-10  Sunday
1  10:05:00  October  10-11  Sunday
2  10:05:00  October  10-11  Sunday
3  10:07:00  October  10-11  Sunday
4  10:07:00  October  10-11  Sunday
```

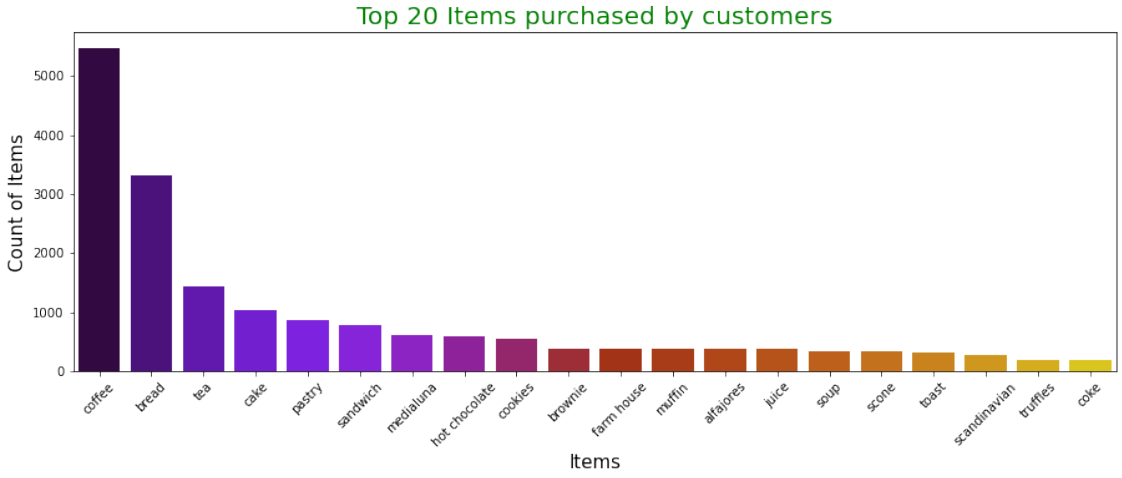
[14]:
```
# cleaning the item column
df['Item'] = df['Item'].str.strip()
df['Item'] = df['Item'].str.lower()
```

[15]:
```
df.head()
```

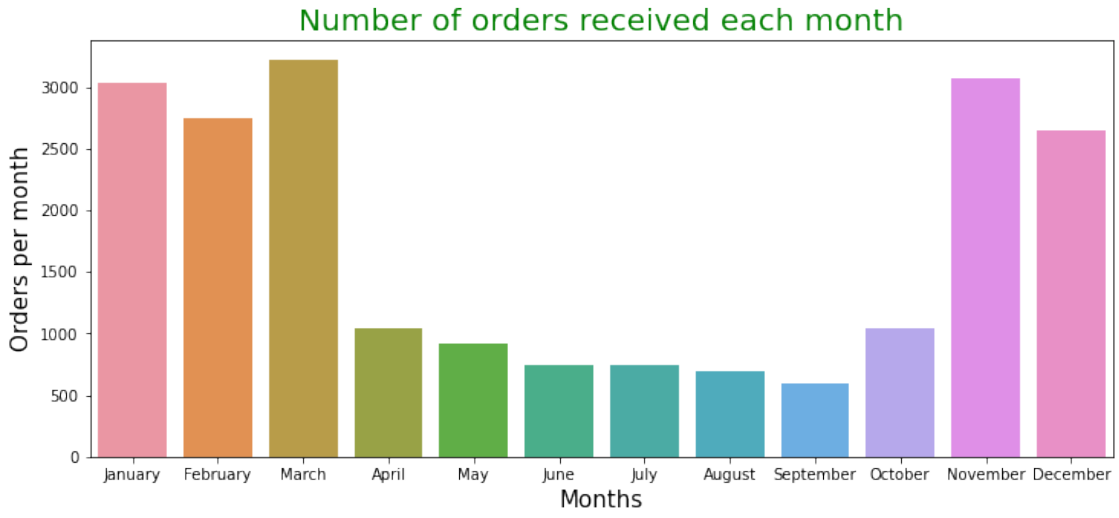
[15]:
```
   Transaction           Item period_day weekday_weekend        date  \
0            1          bread    morning         weekend  2016-10-30
1            2   scandinavian    morning         weekend  2016-10-30
2            2   scandinavian    morning         weekend  2016-10-30
3            3  hot chocolate    morning         weekend  2016-10-30
4            3            jam    morning         weekend  2016-10-30

       time    month   hour weekday
0  09:58:00  October   9-10  Sunday
1  10:05:00  October  10-11  Sunday
2  10:05:00  October  10-11  Sunday
3  10:07:00  October  10-11  Sunday
```

```
4  10:07:00  October  10-11  Sunday
```

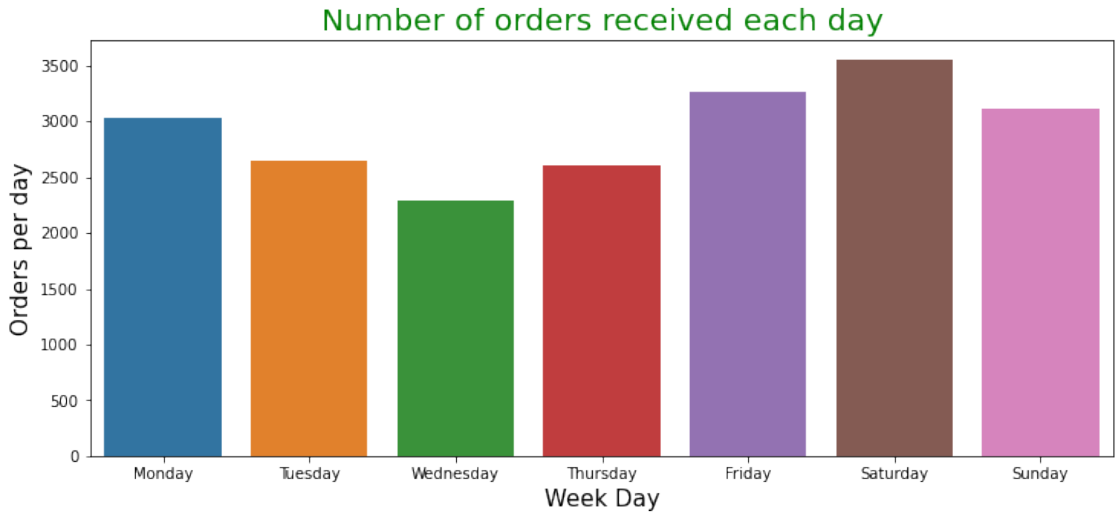
[16]:
```
plt.figure(figsize=(15,5))
sns.barplot(x = df.Item.value_counts().head(20).index, y = df.Item.
 ↪value_counts().head(20).values, palette = 'gnuplot')
plt.xlabel('Items', size = 15)
plt.xticks(rotation=45)
plt.ylabel('Count of Items', size = 15)
plt.title('Top 20 Items purchased by customers', color = 'green', size = 20)
plt.show()
```



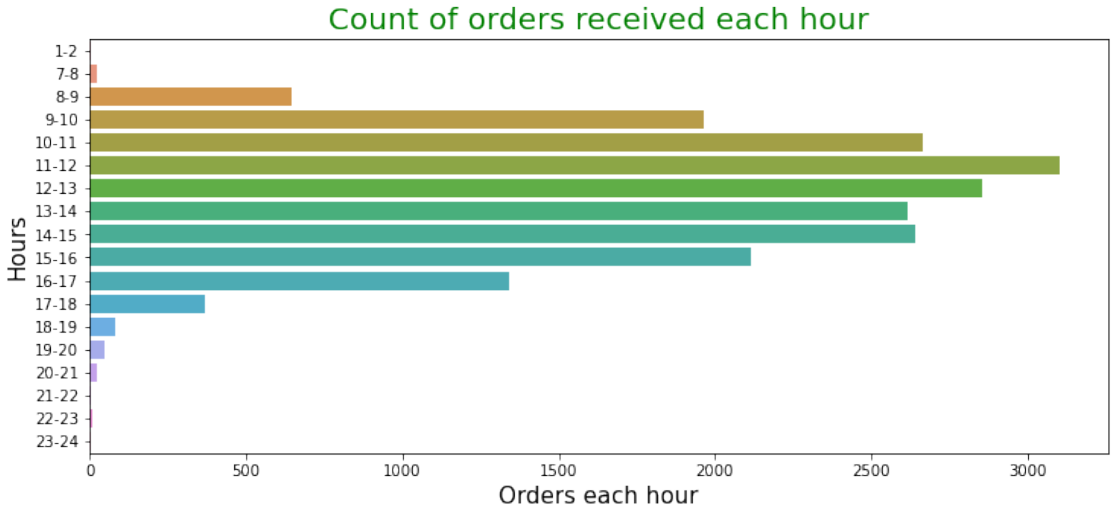
[17]:
```
monthTran = df.groupby('month')['Transaction'].count().reset_index()
monthTran.loc[:,"monthorder"] = [4,8,12,2,1,7,6,3,5,11,10,9]
monthTran.sort_values("monthorder",inplace=True)

plt.figure(figsize=(12,5))
sns.barplot(data = monthTran, x = "month", y = "Transaction")
plt.xlabel('Months', size = 15)
plt.ylabel('Orders per month', size = 15)
plt.title('Number of orders received each month', color = 'green', size = 20)
plt.show()


plt.show()
```



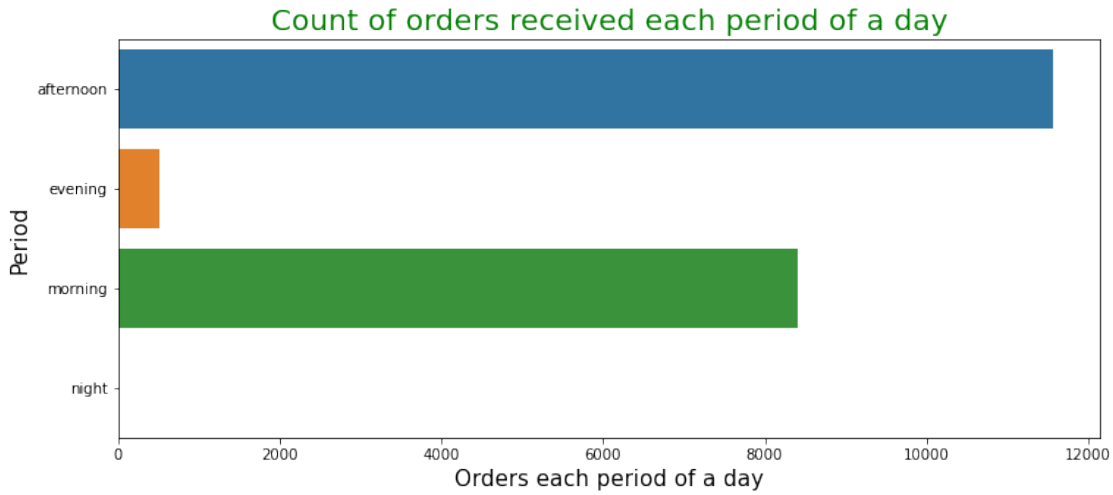
[18]:
```
weekTran = df.groupby('weekday')['Transaction'].count().reset_index()
weekTran.loc[:,"weekorder"] = [4,0,5,6,3,1,2]
weekTran.sort_values("weekorder",inplace=True)

plt.figure(figsize=(12,5))
sns.barplot(data = weekTran, x = "weekday", y = "Transaction")
plt.xlabel('Week Day', size = 15)
plt.ylabel('Orders per day', size = 15)
plt.title('Number of orders received each day', color = 'green', size = 20)
plt.show()


plt.show()
```



[19]:
```
hourTran = df.groupby('hour')['Transaction'].count().reset_index()
hourTran.loc[:,"hourorder"] =␣
 ↪[1,10,11,12,13,14,15,16,17,18,19,20,21,22,23,7,8,9]
hourTran.sort_values("hourorder",inplace=True)

plt.figure(figsize=(12,5))
sns.barplot(data = hourTran, x = "Transaction", y = "hour")
plt.ylabel('Hours', size = 15)
plt.xlabel('Orders each hour', size = 15)
plt.title('Count of orders received each hour', color = 'green', size = 20)
plt.show()
```



[21]:
```
dayTran = df.groupby('period_day')['Transaction'].count().reset_index()
# dayTran.loc[:,"hourorder"] =␣
 ↪[1,10,11,12,13,14,15,16,17,18,19,20,21,22,23,7,8,9]
# dayTran.sort_values("hourorder",inplace=True)

plt.figure(figsize=(12,5))
sns.barplot(data = dayTran, x = "Transaction", y = "period_day")
plt.ylabel('Period', size = 15)
plt.xlabel('Orders each period of a day', size = 15)
plt.title('Count of orders received each period of a day', color = 'green',␣
 ↪size = 20)
plt.show()
```



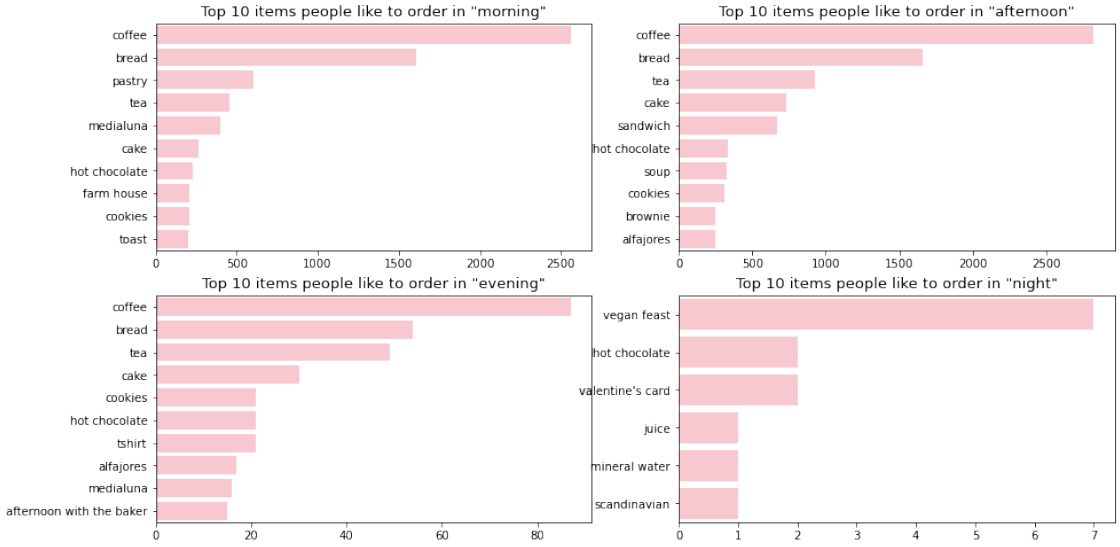
[22]:
```
dates = df.groupby('date')['Transaction'].count().reset_index()
dates = dates[dates['Transaction']>=200].sort_values('date').reset_index(drop =␣
 ↪True)

dates = pd.merge(dates, df[['date','weekday']], on = 'date', how = 'inner')
dates.drop_duplicates(inplace =True)
dates
```

[22]:

	date	Transaction	weekday
0	2016-05-11	275	Wednesday
275	2016-11-19	209	Saturday
484	2016-12-11	221	Sunday
705	2017-01-28	237	Saturday
942	2017-02-18	227	Saturday
1169	2017-03-25	246	Saturday
1415	2017-04-02	292	Sunday
1707	2017-04-03	257	Monday
1964	2017-08-04	205	Friday
2169	2017-11-03	203	Friday

[23]:
```
data = df.groupby(['period_day','Item'])['Transaction'].count().reset_index().
 ↪sort_values(['period_day','Transaction'],ascending=False)
day = ['morning','afternoon','evening','night']

plt.figure(figsize=(15,8))
for i,j in enumerate(day):
    plt.subplot(2,2,i+1)
    df1 = data[data.period_day==j].head(10)
    sns.barplot(data=df1, y=df1.Item, x=df1.Transaction, color='pink')
    plt.xlabel('')
```

```
    plt.ylabel('')
    plt.title('Top 10 items people like to order in "{}"'.format(j), size=13)

plt.show()
```



[25]:
```
!pip install mlxtend
```

```
Collecting mlxtend
  Downloading mlxtend-0.23.1-py3-none-any.whl (1.4 MB)
Requirement already satisfied: pandas>=0.24.2 in c:\users\hp\anaconda3\lib\site-
packages (from mlxtend) (1.2.4)
Requirement already satisfied: numpy>=1.16.2 in c:\users\hp\anaconda3\lib\site-
packages (from mlxtend) (1.22.4)
Requirement already satisfied: scipy>=1.2.1 in c:\users\hp\anaconda3\lib\site-
packages (from mlxtend) (1.6.2)
Requirement already satisfied: scikit-learn>=1.0.2 in
c:\users\hp\anaconda3\lib\site-packages (from mlxtend) (1.3.2)
Requirement already satisfied: matplotlib>=3.0.0 in
c:\users\hp\anaconda3\lib\site-packages (from mlxtend) (3.3.4)
Requirement already satisfied: joblib>=0.13.2 in c:\users\hp\anaconda3\lib\site-
packages (from mlxtend) (1.4.2)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in
c:\users\hp\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend)
(2.4.7)
Requirement already satisfied: pillow>=6.2.0 in c:\users\hp\anaconda3\lib\site-
packages (from matplotlib>=3.0.0->mlxtend) (8.2.0)
Requirement already satisfied: python-dateutil>=2.1 in
c:\users\hp\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend)
(2.8.1)
```

```
Requirement already satisfied: kiwisolver>=1.0.1 in
c:\users\hp\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend)
(1.3.1)
Requirement already satisfied: cycler>=0.10 in c:\users\hp\anaconda3\lib\site-
packages (from matplotlib>=3.0.0->mlxtend) (0.10.0)
Requirement already satisfied: six in c:\users\hp\anaconda3\lib\site-packages
(from cycler>=0.10->matplotlib>=3.0.0->mlxtend) (1.15.0)
Requirement already satisfied: pytz>=2017.3 in c:\users\hp\anaconda3\lib\site-
packages (from pandas>=0.24.2->mlxtend) (2021.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in
c:\users\hp\anaconda3\lib\site-packages (from scikit-learn>=1.0.2->mlxtend)
(3.5.0)
Installing collected packages: mlxtend
Successfully installed mlxtend-0.23.1
```

[26]:
```
from mlxtend.frequent_patterns import association_rules, apriori
```

[34]:
```
import warnings
warnings.filterwarnings('ignore')
```

```
C:\Users\HP\anaconda3\lib\site-packages\ipykernel\ipkernel.py:287:
DeprecationWarning: `should_run_async` will not call `transform_cell`
automatically in the future. Please pass the result to `transformed_cell`
argument and any exception that happen during thetransform in
`preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```

[35]:
```
transactions_str = df.groupby(['Transaction', 'Item'])['Item'].count().
 ↪reset_index(name ='Count')
transactions_str
```

[35]:

	Transaction	Item	Count
0	1	bread	1
1	2	scandinavian	2
2	3	cookies	1
3	3	hot chocolate	1
4	3	jam	1
…	…	…	…
18882	9682	tacos/fajita	1
18883	9682	tea	1
18884	9683	coffee	1
18885	9683	pastry	1
18886	9684	smoothies	1

[18887 rows x 3 columns]

[36]:
```
# making a mxn matrice where m=transaction and n=items and each row represents whether the item was in the transaction or not
my_basket = transactions_str.pivot_table(index='Transaction', columns='Item', values='Count', aggfunc='sum').fillna(0)


my_basket.head()
```

[36]:

Item	adjustment	afternoon with the baker	alfajores	argentina night	\
Transaction					
1	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	
5	0.0	0.0	0.0	0.0	

Item	art tray	bacon	baguette	bakewell	bare popcorn	basket	…	\
Transaction							…	
1	0.0	0.0	0.0	0.0	0.0	0.0	…	
2	0.0	0.0	0.0	0.0	0.0	0.0	…	
3	0.0	0.0	0.0	0.0	0.0	0.0	…	
4	0.0	0.0	0.0	0.0	0.0	0.0	…	
5	0.0	0.0	0.0	0.0	0.0	0.0	…	

Item	the bart	the nomad	tiffin	toast	truffles	tshirt	\
Transaction							
1	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	
5	0.0	0.0	0.0	0.0	0.0	0.0	

Item	valentine's card	vegan feast	vegan mincepie	victorian sponge
Transaction				
1	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0

[5 rows x 94 columns]

[37]:
```
# making a function which returns 0 or 1
# 0 means item was not in that transaction, 1 means item present in that transaction

def encode(x):
```

```
    if x<=0:
        return 0
    if x>=1:
        return 1

# applying the function to the dataset

my_basket_sets = my_basket.applymap(encode)
my_basket_sets.head()
```

[37]:
```
Item         adjustment  afternoon with the baker  alfajores  argentina night  \
Transaction
1                     0                         0          0                0
2                     0                         0          0                0
3                     0                         0          0                0
4                     0                         0          0                0
5                     0                         0          0                0

Item         art tray  bacon  baguette  bakewell  bare popcorn  basket  … \
Transaction                                                             …
1                   0      0         0         0             0       0  …
2                   0      0         0         0             0       0  …
3                   0      0         0         0             0       0  …
4                   0      0         0         0             0       0  …
5                   0      0         0         0             0       0  …

Item         the bart  the nomad  tiffin  toast  truffles  tshirt  \
Transaction
1                   0          0       0      0         0       0
2                   0          0       0      0         0       0
3                   0          0       0      0         0       0
4                   0          0       0      0         0       0
5                   0          0       0      0         0       0

Item         valentine's card  vegan feast  vegan mincepie  victorian sponge
Transaction
1                           0            0               0                 0
2                           0            0               0                 0
3                           0            0               0                 0
4                           0            0               0                 0
5                           0            0               0                 0

[5 rows x 94 columns]
```

[38]:
```
# using the 'apriori algorithm' with min_support=0.01 (1% of 9465)
# It means the item should be present in atleast 94 transaction out of 9465␣
 ↪transactions only when we considered that item in
```

```
# frequent itemset
frequent_items = apriori(my_basket_sets, min_support = 0.01,use_colnames = True)
frequent_items
```

[38]:

	support	itemsets
0	0.036344	(alfajores)
1	0.016059	(baguette)
2	0.327205	(bread)
3	0.040042	(brownie)
4	0.103856	(cake)
..	…	…
56	0.023666	(toast, coffee)
57	0.014369	(sandwich, tea)
58	0.010037	(bread, coffee, cake)
59	0.011199	(pastry, bread, coffee)
60	0.010037	(tea, coffee, cake)

[61 rows x 2 columns]

[39]:
```
# now making the rules from frequent itemset generated above

rules = association_rules(frequent_items, metric = "lift", min_threshold = 1)
rules.sort_values('confidence', ascending = False, inplace = True)
rules
```

[39]:

	antecedents	consequents	antecedent support	\
30	(toast)	(coffee)	0.033597	
29	(spanish brunch)	(coffee)	0.018172	
19	(medialuna)	(coffee)	0.061807	
22	(pastry)	(coffee)	0.086107	
0	(alfajores)	(coffee)	0.036344	
17	(juice)	(coffee)	0.038563	
24	(sandwich)	(coffee)	0.071844	
7	(cake)	(coffee)	0.103856	
26	(scone)	(coffee)	0.034548	
12	(cookies)	(coffee)	0.054411	
14	(hot chocolate)	(coffee)	0.058320	
5	(brownie)	(coffee)	0.040042	
20	(muffin)	(coffee)	0.038457	
2	(pastry)	(bread)	0.086107	
11	(cake)	(tea)	0.103856	
38	(tea, coffee)	(cake)	0.049868	
32	(sandwich)	(tea)	0.071844	
8	(hot chocolate)	(cake)	0.058320	
39	(coffee, cake)	(tea)	0.054728	
10	(tea)	(cake)	0.142631	
37	(pastry)	(bread, coffee)	0.086107	

36	(bread, coffee)	(pastry)	0.090016
6	(coffee)	(cake)	0.478394
34	(bread, coffee)	(cake)	0.090016
9	(cake)	(hot chocolate)	0.103856
33	(tea)	(sandwich)	0.142631
23	(coffee)	(pastry)	0.478394
35	(cake)	(bread, coffee)	0.103856
41	(cake)	(tea, coffee)	0.103856
3	(bread)	(pastry)	0.327205
25	(coffee)	(sandwich)	0.478394
18	(coffee)	(medialuna)	0.478394
40	(tea)	(coffee, cake)	0.142631
15	(coffee)	(hot chocolate)	0.478394
13	(coffee)	(cookies)	0.478394
31	(coffee)	(toast)	0.478394
16	(coffee)	(juice)	0.478394
1	(coffee)	(alfajores)	0.478394
4	(coffee)	(brownie)	0.478394
21	(coffee)	(muffin)	0.478394
27	(coffee)	(scone)	0.478394
28	(coffee)	(spanish brunch)	0.478394

	consequent support	support	confidence	lift	leverage	conviction	\
30	0.478394	0.023666	0.704403	1.472431	0.007593	1.764582	
29	0.478394	0.010882	0.598837	1.251766	0.002189	1.300235	
19	0.478394	0.035182	0.569231	1.189878	0.005614	1.210871	
22	0.478394	0.047544	0.552147	1.154168	0.006351	1.164682	
0	0.478394	0.019651	0.540698	1.130235	0.002264	1.135648	
17	0.478394	0.020602	0.534247	1.116750	0.002154	1.119919	
24	0.478394	0.038246	0.532353	1.112792	0.003877	1.115384	
7	0.478394	0.054728	0.526958	1.101515	0.005044	1.102664	
26	0.478394	0.018067	0.522936	1.093107	0.001539	1.093366	
12	0.478394	0.028209	0.518447	1.083723	0.002179	1.083174	
14	0.478394	0.029583	0.507246	1.060311	0.001683	1.058553	
5	0.478394	0.019651	0.490765	1.025860	0.000495	1.024293	
20	0.478394	0.018806	0.489011	1.022193	0.000408	1.020777	
2	0.327205	0.029160	0.338650	1.034977	0.000985	1.017305	
11	0.142631	0.023772	0.228891	1.604781	0.008959	1.111865	
38	0.103856	0.010037	0.201271	1.937977	0.004858	1.121962	
32	0.142631	0.014369	0.200000	1.402222	0.004122	1.071712	
8	0.103856	0.011410	0.195652	1.883874	0.005354	1.114125	
39	0.142631	0.010037	0.183398	1.285822	0.002231	1.049923	
10	0.103856	0.023772	0.166667	1.604781	0.008959	1.075372	
37	0.090016	0.011199	0.130061	1.444872	0.003448	1.046033	
36	0.086107	0.011199	0.124413	1.444872	0.003448	1.043749	
6	0.103856	0.054728	0.114399	1.101515	0.005044	1.011905	
34	0.103856	0.010037	0.111502	1.073621	0.000688	1.008606	

9	0.058320	0.011410	0.109868	1.883874	0.005354	1.057910
33	0.071844	0.014369	0.100741	1.402222	0.004122	1.032134
23	0.086107	0.047544	0.099382	1.154168	0.006351	1.014740
35	0.090016	0.010037	0.096643	1.073621	0.000688	1.007336
41	0.049868	0.010037	0.096643	1.937977	0.004858	1.051779
3	0.086107	0.029160	0.089119	1.034977	0.000985	1.003306
25	0.071844	0.038246	0.079947	1.112792	0.003877	1.008807
18	0.061807	0.035182	0.073542	1.189878	0.005614	1.012667
40	0.054728	0.010037	0.070370	1.285822	0.002231	1.016827
15	0.058320	0.029583	0.061837	1.060311	0.001683	1.003749
13	0.054411	0.028209	0.058966	1.083723	0.002179	1.004841
31	0.033597	0.023666	0.049470	1.472431	0.007593	1.016699
16	0.038563	0.020602	0.043065	1.116750	0.002154	1.004705
1	0.036344	0.019651	0.041078	1.130235	0.002264	1.004936
4	0.040042	0.019651	0.041078	1.025860	0.000495	1.001080
21	0.038457	0.018806	0.039311	1.022193	0.000408	1.000888
27	0.034548	0.018067	0.037765	1.093107	0.001539	1.003343
28	0.018172	0.010882	0.022747	1.251766	0.002189	1.004682

	zhangs_metric
30	0.332006
29	0.204851
19	0.170091
22	0.146161
0	0.119574
17	0.108738
24	0.109205
7	0.102840
26	0.088224
12	0.081700
14	0.060403
5	0.026259
20	0.022579
2	0.036980
11	0.420538
38	0.509401
32	0.309050
8	0.498236
39	0.235157
10	0.439556
37	0.336907
36	0.338354
6	0.176684
34	0.075356
9	0.523553
33	0.334566
23	0.256084

```
35       0.076520
41       0.540090
3        0.050231
25       0.194321
18       0.305936
40       0.259266
15       0.109048
13       0.148110
31       0.615122
16       0.200428
1        0.220910
4        0.048327
21       0.041623
27       0.163296
28       0.385594
```

[40]:
```
# arranging the data from highest to lowest with respect to 'confidence'

rules.sort_values('confidence', ascending=False)
```

[40]:

	antecedents	consequents	antecedent support	\
30	(toast)	(coffee)	0.033597	
29	(spanish brunch)	(coffee)	0.018172	
19	(medialuna)	(coffee)	0.061807	
22	(pastry)	(coffee)	0.086107	
0	(alfajores)	(coffee)	0.036344	
17	(juice)	(coffee)	0.038563	
24	(sandwich)	(coffee)	0.071844	
7	(cake)	(coffee)	0.103856	
26	(scone)	(coffee)	0.034548	
12	(cookies)	(coffee)	0.054411	
14	(hot chocolate)	(coffee)	0.058320	
5	(brownie)	(coffee)	0.040042	
20	(muffin)	(coffee)	0.038457	
2	(pastry)	(bread)	0.086107	
11	(cake)	(tea)	0.103856	
38	(tea, coffee)	(cake)	0.049868	
32	(sandwich)	(tea)	0.071844	
8	(hot chocolate)	(cake)	0.058320	
39	(coffee, cake)	(tea)	0.054728	
10	(tea)	(cake)	0.142631	
37	(pastry)	(bread, coffee)	0.086107	
36	(bread, coffee)	(pastry)	0.090016	
6	(coffee)	(cake)	0.478394	
34	(bread, coffee)	(cake)	0.090016	
9	(cake)	(hot chocolate)	0.103856	
33	(tea)	(sandwich)	0.142631	

23	(coffee)	(pastry)	0.478394
35	(cake)	(bread, coffee)	0.103856
41	(cake)	(tea, coffee)	0.103856
3	(bread)	(pastry)	0.327205
25	(coffee)	(sandwich)	0.478394
18	(coffee)	(medialuna)	0.478394
40	(tea)	(coffee, cake)	0.142631
15	(coffee)	(hot chocolate)	0.478394
13	(coffee)	(cookies)	0.478394
31	(coffee)	(toast)	0.478394
16	(coffee)	(juice)	0.478394
1	(coffee)	(alfajores)	0.478394
4	(coffee)	(brownie)	0.478394
21	(coffee)	(muffin)	0.478394
27	(coffee)	(scone)	0.478394
28	(coffee)	(spanish brunch)	0.478394

	consequent support	support	confidence	lift	leverage	conviction	\
30	0.478394	0.023666	0.704403	1.472431	0.007593	1.764582	
29	0.478394	0.010882	0.598837	1.251766	0.002189	1.300235	
19	0.478394	0.035182	0.569231	1.189878	0.005614	1.210871	
22	0.478394	0.047544	0.552147	1.154168	0.006351	1.164682	
0	0.478394	0.019651	0.540698	1.130235	0.002264	1.135648	
17	0.478394	0.020602	0.534247	1.116750	0.002154	1.119919	
24	0.478394	0.038246	0.532353	1.112792	0.003877	1.115384	
7	0.478394	0.054728	0.526958	1.101515	0.005044	1.102664	
26	0.478394	0.018067	0.522936	1.093107	0.001539	1.093366	
12	0.478394	0.028209	0.518447	1.083723	0.002179	1.083174	
14	0.478394	0.029583	0.507246	1.060311	0.001683	1.058553	
5	0.478394	0.019651	0.490765	1.025860	0.000495	1.024293	
20	0.478394	0.018806	0.489011	1.022193	0.000408	1.020777	
2	0.327205	0.029160	0.338650	1.034977	0.000985	1.017305	
11	0.142631	0.023772	0.228891	1.604781	0.008959	1.111865	
38	0.103856	0.010037	0.201271	1.937977	0.004858	1.121962	
32	0.142631	0.014369	0.200000	1.402222	0.004122	1.071712	
8	0.103856	0.011410	0.195652	1.883874	0.005354	1.114125	
39	0.142631	0.010037	0.183398	1.285822	0.002231	1.049923	
10	0.103856	0.023772	0.166667	1.604781	0.008959	1.075372	
37	0.090016	0.011199	0.130061	1.444872	0.003448	1.046033	
36	0.086107	0.011199	0.124413	1.444872	0.003448	1.043749	
6	0.103856	0.054728	0.114399	1.101515	0.005044	1.011905	
34	0.103856	0.010037	0.111502	1.073621	0.000688	1.008606	
9	0.058320	0.011410	0.109868	1.883874	0.005354	1.057910	
33	0.071844	0.014369	0.100741	1.402222	0.004122	1.032134	
23	0.086107	0.047544	0.099382	1.154168	0.006351	1.014740	
35	0.090016	0.010037	0.096643	1.073621	0.000688	1.007336	
41	0.049868	0.010037	0.096643	1.937977	0.004858	1.051779	

3	0.086107	0.029160	0.089119	1.034977	0.000985	1.003306
25	0.071844	0.038246	0.079947	1.112792	0.003877	1.008807
18	0.061807	0.035182	0.073542	1.189878	0.005614	1.012667
40	0.054728	0.010037	0.070370	1.285822	0.002231	1.016827
15	0.058320	0.029583	0.061837	1.060311	0.001683	1.003749
13	0.054411	0.028209	0.058966	1.083723	0.002179	1.004841
31	0.033597	0.023666	0.049470	1.472431	0.007593	1.016699
16	0.038563	0.020602	0.043065	1.116750	0.002154	1.004705
1	0.036344	0.019651	0.041078	1.130235	0.002264	1.004936
4	0.040042	0.019651	0.041078	1.025860	0.000495	1.001080
21	0.038457	0.018806	0.039311	1.022193	0.000408	1.000888
27	0.034548	0.018067	0.037765	1.093107	0.001539	1.003343
28	0.018172	0.010882	0.022747	1.251766	0.002189	1.004682

	zhangs_metric
30	0.332006
29	0.204851
19	0.170091
22	0.146161
0	0.119574
17	0.108738
24	0.109205
7	0.102840
26	0.088224
12	0.081700
14	0.060403
5	0.026259
20	0.022579
2	0.036980
11	0.420538
38	0.509401
32	0.309050
8	0.498236
39	0.235157
10	0.439556
37	0.336907
36	0.338354
6	0.176684
34	0.075356
9	0.523553
33	0.334566
23	0.256084
35	0.076520
41	0.540090
3	0.050231
25	0.194321
18	0.305936

```
40      0.259266
15      0.109048
13      0.148110
31      0.615122
16      0.200428
1       0.220910
4       0.048327
21      0.041623
27      0.163296
28      0.385594
```

[ ]: