KDD LAB:6

MAITRI SURVE : 20190802063

In [5]:
```
! pip install plotly
%pip install mlxtend
```

```
Requirement already satisfied: plotly in d:\anaconda\lib\site-packages (5.6.0)
Requirement already satisfied: six in d:\anaconda\lib\site-packages (from plotly) (1.16.0)
Requirement already satisfied: tenacity>=6.2.0 in d:\anaconda\lib\site-packages (from plotly) (8.0.1)
Collecting mlxtend
  Downloading mlxtend-0.19.0-py2.py3-none-any.whl (1.3 MB)
Requirement already satisfied: pandas>=0.24.2 in d:\anaconda\lib\site-packages (from mlxtend) (1.3.4)
Requirement already satisfied: setuptools in d:\anaconda\lib\site-packages (from mlxtend) (58.0.4)
Requirement already satisfied: matplotlib>=3.0.0 in d:\anaconda\lib\site-packages (from mlxtend) (3.4.3)
Requirement already satisfied: joblib>=0.13.2 in d:\anaconda\lib\site-packages (from mlxtend) (1.1.0)
Requirement already satisfied: scikit-learn>=0.20.3 in d:\anaconda\lib\site-packages (from mlxtend) (0.24.2)
Requirement already satisfied: numpy>=1.16.2 in d:\anaconda\lib\site-packages (from mlxtend) (1.20.3)
Requirement already satisfied: scipy>=1.2.1 in d:\anaconda\lib\site-packages (from mlxtend) (1.7.1)
Requirement already satisfied: python-dateutil>=2.7 in d:\anaconda\lib\site-packages (from matplotlib>=3.0.0->
mlxtend) (2.8.2)
Requirement already satisfied: pillow>=6.2.0 in d:\anaconda\lib\site-packages (from matplotlib>=3.0.0->mlxten
d) (8.4.0)
Requirement already satisfied: pyparsing>=2.2.1 in d:\anaconda\lib\site-packages (from matplotlib>=3.0.0->mlxt
end) (3.0.4)
Requirement already satisfied: cycler>=0.10 in d:\anaconda\lib\site-packages (from matplotlib>=3.0.0->mlxtend)
(0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in d:\anaconda\lib\site-packages (from matplotlib>=3.0.0->mlx
tend) (1.3.1)
Requirement already satisfied: six in d:\anaconda\lib\site-packages (from cycler>=0.10->matplotlib>=3.0.0->mlx
tend) (1.16.0)
Requirement already satisfied: pytz>=2017.3 in d:\anaconda\lib\site-packages (from pandas>=0.24.2->mlxtend) (2
021.3)
Requirement already satisfied: threadpoolctl>=2.0.0 in d:\anaconda\lib\site-packages (from scikit-learn>=0.20.
3->mlxtend) (2.2.0)
Installing collected packages: mlxtend
Successfully installed mlxtend-0.19.0
Note: you may need to restart the kernel to use updated packages.
```

In [2]:
```
import pandas as pd
dt= pd.read_csv("Grocery Products Purchase.csv")
dt.shape
```

Out[2]:
```
(9835, 32)
```

In [3]:
```
dt.head()
```

Out[3]:

	Product 1	Product 2	Product 3	Product 4	Product 5	Product 6	Product 7	Product 8	Product 9	Product 10	...	Product 23	Product 24	Product 25	Produ
0	citrus fruit	semi-finished bread	margarine	ready soups	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	Na
1	tropical fruit	yogurt	coffee	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	Na
2	whole milk	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	Na
3	pip fruit	yogurt	cream cheese	meat spreads	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	Na
4	other vegetables	whole milk	condensed milk	long life bakery product	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	Na

5 rows × 32 columns

In [4]:
```
import numpy as np

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

transaction = np.array(transaction)
print(transaction)
```

```
['citrus fruit' 'semi-finished bread' 'margarine' ... 'nan' 'nan' 'nan']
```

In [5]:
```
df = pd.DataFrame(transaction, columns=["items"])
df["incident_count"] = 1

indexNames = df[df['items'] == "nan" ].index
df.drop(indexNames , inplace=True)

# Making a New Appropriate Pandas DataFrame for Visualizations
df_table = df.groupby("items").sum().sort_values("incident_count", ascending=False).reset_index()

#  Initial Visualizations
df_table.head(5).style.background_gradient(cmap='Blues')
```

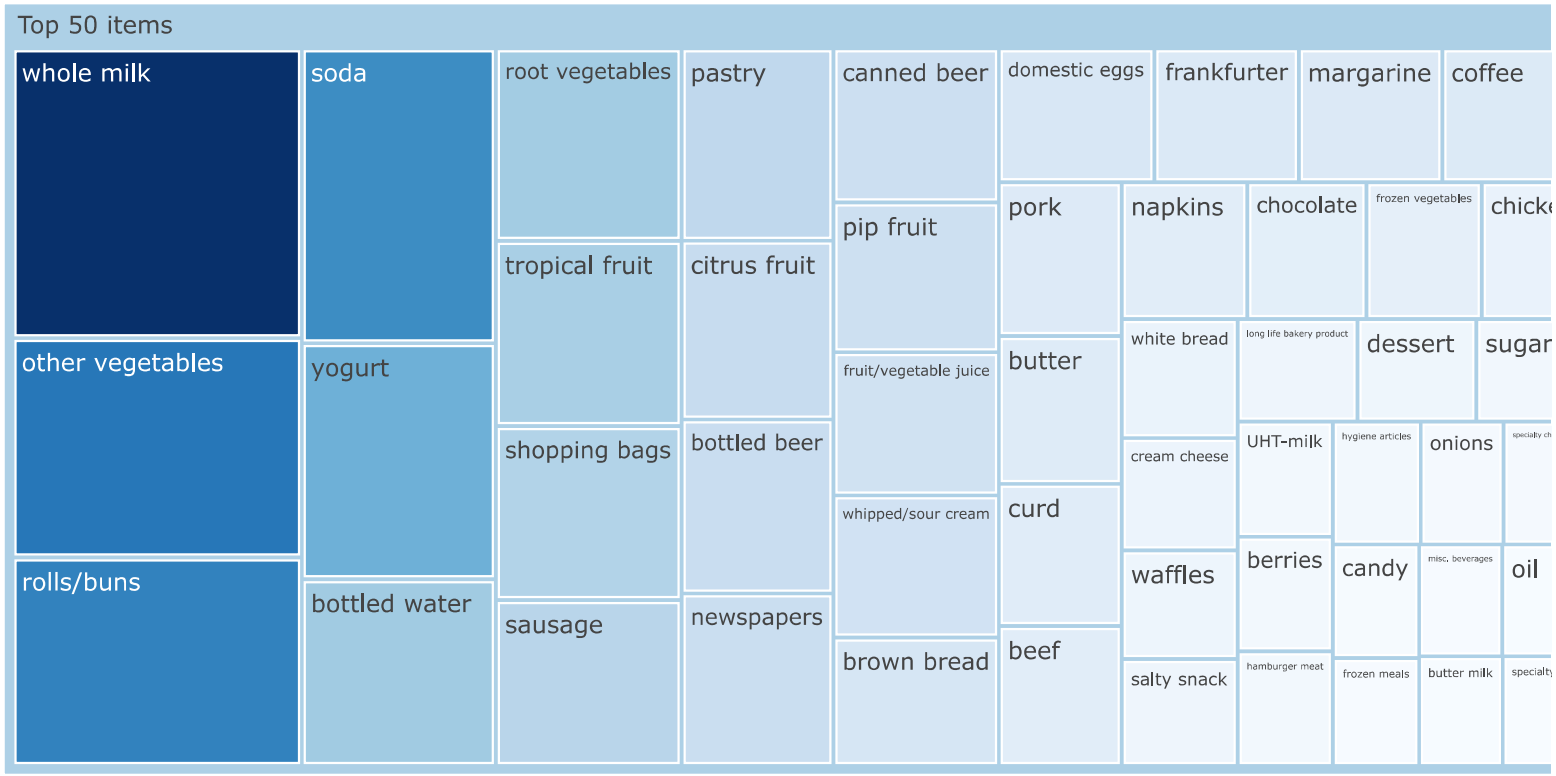
Out[5]:

	items	incident_count
0	whole milk	2513
1	other vegetables	1903
2	rolls/buns	1809
3	soda	1715
4	yogurt	1372

In [6]:
```
# importing required module
import plotly.express as px

# to have a same origin
df_table["all"] = "Top 50 items"

# creating tree map using plotly
fig = px.treemap(df_table.head(50), path=['all', "items"], values='incident_count',
                 color=df_table["incident_count"].head(50), hover_data=['items'],
                 color_continuous_scale='Blues',
                )
# ploting the treemap
fig.show()
```



In [7]:
```
transaction = []
for i in range(dt.shape[0]):
    transaction.append([str(dt.values[i,j]) for j in range(dt.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)
dt = pd.DataFrame(te_ary, columns=te.columns_)

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

Out[7]:

	Instant food products	UHT-milk	abrasive cleaner	artif. sweetener	baby cosmetics	baby food	bags	baking powder	bathroom cleaner	beef	...	turkey	vinegar	waffles	whipped/sour cream	w
0	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	
1	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	
2	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	
3	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	
4	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	

5 rows × 170 columns

In [8]:
```
# select top 30 items
first30 = df_table["items"].head(30).values

# Extract Top 30
dt = dt.loc[:,first30]

# shape of the dt
dt.shape
```

Out[8]:
```
(9835, 30)
```

In [9]:
```
from mlxtend.frequent_patterns import fpgrowth

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

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

Out[9]:

	support	itemsets
0	0.082766	(citrus fruit)
1	0.058566	(margarine)
2	0.139502	(yogurt)
3	0.104931	(tropical fruit)
4	0.058058	(coffee)
5	0.255516	(whole milk)
6	0.075648	(pip fruit)
7	0.193493	(other vegetables)
8	0.055414	(butter)
9	0.183935	(rolls/buns)

In [10]:
```
from mlxtend.frequent_patterns import association_rules

# creating asssociation rules
res=association_rules(res, metric="lift", min_threshold=1)

# printing association rules
res
```

Out[10]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(yogurt)	(whole milk)	0.139502	0.255516	0.056024	0.401603	1.571735	0.020379	1.244132
1	(whole milk)	(yogurt)	0.255516	0.139502	0.056024	0.219260	1.571735	0.020379	1.102157
2	(whole milk)	(other vegetables)	0.255516	0.193493	0.074835	0.292877	1.513634	0.025394	1.140548
3	(other vegetables)	(whole milk)	0.193493	0.255516	0.074835	0.386758	1.513634	0.025394	1.214013
4	(whole milk)	(rolls/buns)	0.255516	0.183935	0.056634	0.221647	1.205032	0.009636	1.048452
5	(rolls/buns)	(whole milk)	0.183935	0.255516	0.056634	0.307905	1.205032	0.009636	1.075696

In [11]:
```
res.sort_values("confidence",ascending=False)
```

Out[11]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(yogurt)	(whole milk)	0.139502	0.255516	0.056024	0.401603	1.571735	0.020379	1.244132
3	(other vegetables)	(whole milk)	0.193493	0.255516	0.074835	0.386758	1.513634	0.025394	1.214013
5	(rolls/buns)	(whole milk)	0.183935	0.255516	0.056634	0.307905	1.205032	0.009636	1.075696
2	(whole milk)	(other vegetables)	0.255516	0.193493	0.074835	0.292877	1.513634	0.025394	1.140548
4	(whole milk)	(rolls/buns)	0.255516	0.183935	0.056634	0.221647	1.205032	0.009636	1.048452
1	(whole milk)	(yogurt)	0.255516	0.139502	0.056024	0.219260	1.571735	0.020379	1.102157