Title:

Understanding Market Basket Analysis with the Apriori Algorithm

Introduction:

Market Basket Analysis is a powerful technique in the field of data mining that aims to uncover patterns and associations within transactional datasets. One of the key algorithms used for this purpose is the Apriori algorithm. This algorithm plays a crucial role in identifying relationships between items purchased together, providing valuable insights for businesses in various domains.

Method/Working:

The Apriori algorithm follows a systematic approach to discovering frequent itemsets and generating association rules:

1. **Frequent Itemset Generation:**
    - The algorithm begins by identifying frequent itemsets, which are sets of items that frequently appear together in transactions.
    - It starts with individual items and progressively explores larger itemsets, determining their frequency against a specified threshold (minimum support).
2. **Join Step:**
    - During each iteration, the algorithm joins pairs of known frequent itemsets to create larger candidate itemsets.
    - For instance, if items {A} and {B} are found to be frequent, the algorithm combines them to create the candidate itemset {A, B}.
3. **Prune Step:**
    - After joining, the algorithm prunes candidate itemsets by checking if their subsets are also frequent.
    - If a subset is not frequent, the candidate itemset is eliminated, reducing the search space and enhancing computational efficiency.
4. **Repeat:**
    - Steps 2 and 3 are repeated iteratively until no more frequent itemsets can be generated.
5. **Association Rule Generation:**
    - Once frequent itemsets are identified, association rules are generated based on these itemsets.
    - Association rules have the form A => B, representing relationships between items. These rules are characterized by measures such as support, confidence, and lift.

6. **Rule Evaluation and Selection:**
    - The generated rules are evaluated based on user-specified criteria, such as minimum confidence and minimum lift.
    - Rules meeting the criteria are considered interesting and actionable, providing valns is critical for business success.atns is critical for business success. algorithm, feel free to ask!

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
```

```
!pip install apyori
```

```
Requirement already satisfied: apyori in d:\languages\python\machine_learning\notes\e
```

```
from apyori import apriori
```

Data collection

```
file = pd.read_csv("../datasets/Groceries_dataset.csv")
```

```
file.head(25)
```

	Member_number	Date	itemDescription
0	1808	21-07-2015	tropical fruit
1	2552	05-01-2015	whole milk
2	2300	19-09-2015	pip fruit
3	1187	12-12-2015	other vegetables
4	3037	01-02-2015	whole milk
5	4941	14-02-2015	rolls/buns
6	4501	08-05-2015	other vegetables
7	3803	23-12-2015	pot plants
8	2762	20-03-2015	whole milk
9	4119	12-02-2015	tropical fruit
10	1340	24-02-2015	citrus fruit
11	2193	14-04-2015	beef

```
file.isna().sum()
```

```
Member_number      0
Date               0
itemDescription    0
dtype: int64
```

Splitting the data into features and labels

```
x = file["itemDescription"].value_counts().sort_values(ascending = False)[:10]
```

```
20          2810  08-09-2015          butter milk
```

we are soring in descending order of the `file["itemdescription"]` column and keeping a count of every unique item within the range. the items which occur most often is at the top and then it follows till the item with least frequency comes at last.

- data['itemDescription'].value_counts(): Counts the occurrences of each unique item description.
- .sort_values(ascending=False): Sorts the counts in descending orde. [:10]: Selects the top 10 most frequently occurring items..s.

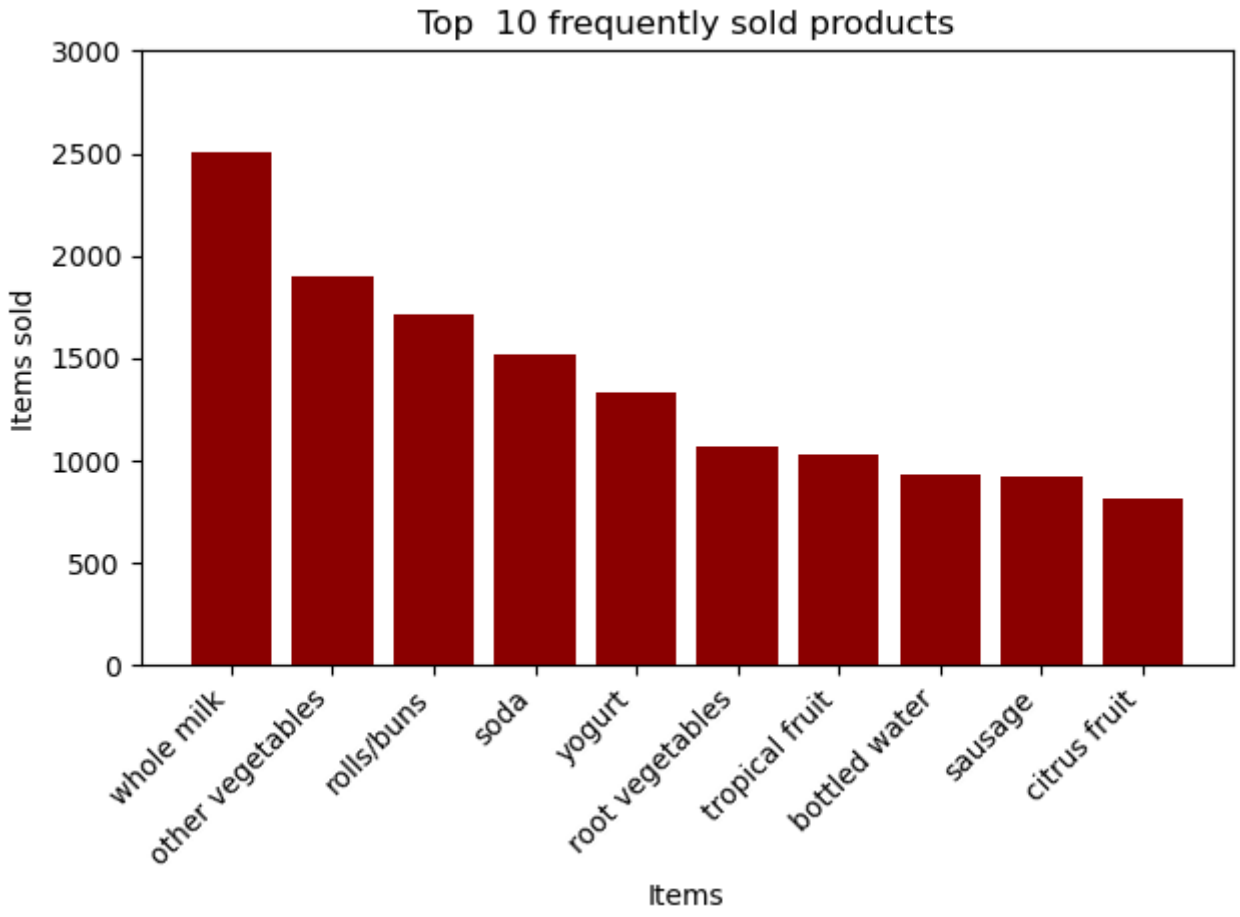
```
x
```

```
itemDescription
whole milk          2502
other vegetables    1898
rolls/buns          1716
soda                1514
```

```
yogurt              1334
root vegetables     1071
tropical fruit      1032
bottled water        933
sausage              924
citrus fruit         812
Name: count, dtype: int64
```

Graphical visualization

```
plt.bar(x.index,x.values,color="darkred")
plt.xlabel("Items")
plt.ylabel("Items sold")
plt.ylim(0,3000)
plt.title("Top  10 frequently sold products")
plt.xticks(rotation=45, ha="right")  # Rotate x-axis labels for better visibility
plt.tight_layout()
plt.show()
```



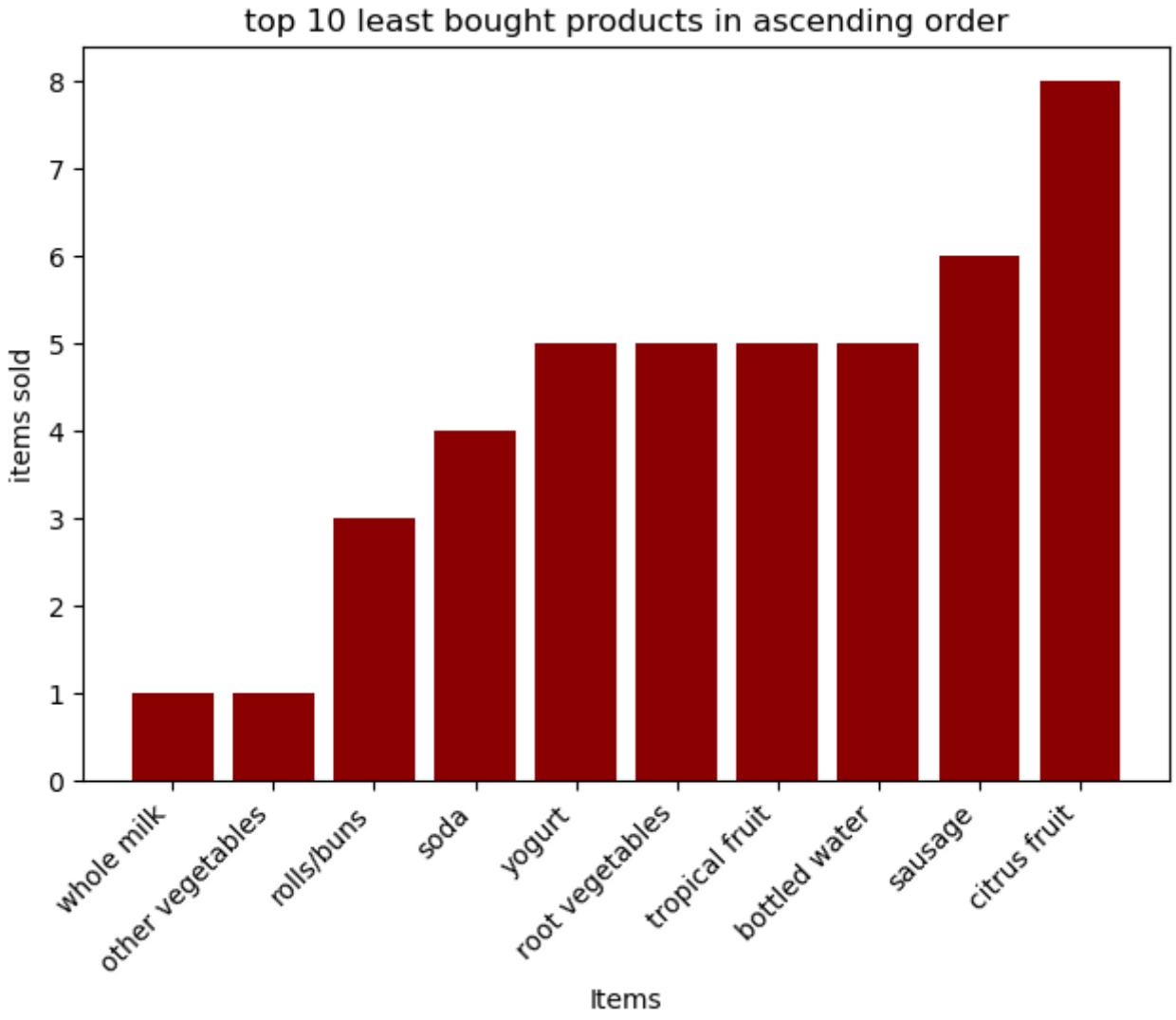

```
y = file["itemDescription"].value_counts().sort_values(ascending=True)[:10]
y
```

```
itemDescription
preservation products    1
kitchen utensil          1
baby cosmetics           3
bags                     4
```

```
frozen chicken           5
make up remover          5
rubbing alcohol          5
toilet cleaner           5
salad dressing           6
whisky                   8
Name: count, dtype: int64
```

In this we sort the least bought products in ascending order. and we keep the count of each item sold. top has the least bought item and 10th position has most bought item in that range.

```
plt.bar(x.index,y.values,color="darkred")
plt.xlabel("Items")
plt.ylabel("items sold")
plt.title("top 10 least bought products in ascending order")
plt.tight_layout()
plt.xticks(rotation = 45, ha="right")
plt.show()
```



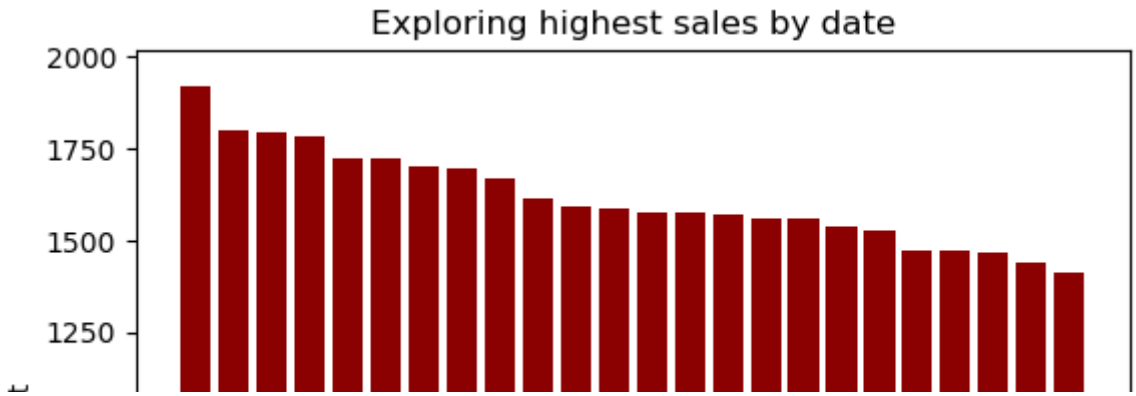

```
pd.DataFrame(file["Member_number"]).value_counts().sort_values(ascending=False)[:10]
```

```
Member_number
3180             36
```

```
3050             33
2051             33
3737             33
3915             31
2433             31
2625             31
2271             31
3872             30
3289             29
Name: count, dtype: int64
```

```
file["Year"]=file["Date"].str.split("-").str[-1]

#Creating a new column in Month-Year format by splitting the date by - and filtering out
file["Month-Year"] = file['Date'].str.split("-").str[1] + "-" + file['Date'].str.split("-
```

```
import matplotlib.pyplot as plt

# Count transactions for each Month-Year
month_year_counts = file["Month-Year"].value_counts(ascending=False)

# Create a bar chart
plt.bar(month_year_counts.index, month_year_counts.values, color='darkred')

# Set labels and title
plt.xlabel('Date')
plt.ylabel('Count')
plt.title('Exploring highest sales by date')

# Rotate x-axis labels for better visibility
plt.xticks(rotation=45, ha='right')

# Display the chart
plt.show()
```




```
products = file['itemDescription'].unique()
products[:10]
```

```
array(['tropical fruit', 'whole milk', 'pip fruit', 'other vegetables',
       'rolls/buns', 'pot plants', 'citrus fruit', 'beef', 'frankfurter',
       'chicken'], dtype=object)
```

```
#For modelling and finding the relationship between products we need to be working with n
file1=file.copy()
one_hot = pd.get_dummies(file1['itemDescription'])
file1.drop(['itemDescription'], inplace =True, axis=1)

file1 = file1.join(one_hot)

file1.head()
```

	Member_number	Date	Year	Month-Year	Instant food products	UHT-milk	abrasive cleaner	artif. sweetener	baby cosmetics
0	1808	21-07-2015	2015	07-2015	False	False	False	False	False
1	2552	05-01-2015	2015	01-2015	False	False	False	False	False
2	2300	19-09-2015	2015	09-2015	False	False	False	False	False
3	1187	12-12-2015	2015	12-2015	False	False	False	False	False
4	3037	01-02-2015	2015	02-2015	False	False	False	False	False

5 rows × 171 columns

```
file2 = file1.groupby(['Member_number', 'Date'])[products[:]].sum()

file2.head(3)
```

		tropical fruit	whole milk	pip fruit	other vegetables	rolls/buns	pot plants	citrus fruit
Member_number	Date							
1000	15-03-2015	0	1	0	0	0	0	0
	24-06-2014	0	1	0	0	0	0	0
	24-07-2015	0	0	0	0	0	0	0

3 rows × 167 columns

```
file2 = file2.reset_index()[products]

file2.head()
```

	tropical fruit	whole milk	pip fruit	other vegetables	rolls/buns	pot plants	citrus fruit	beef	frankfurter
0	0	1	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0

5 rows × 167 columns

```
def product_names(x):
    for product in products:
        if x[product] >0:
            x[product] = product
    return x
#Apply the created function on data2 dataset.
file2 = file2.apply(product_names, axis=1)
file2.head()
```

	tropical fruit	whole milk	pip fruit	other vegetables	rolls/buns	pot plants	citrus fruit	beef	frankfurter
0	0	whole milk	0	0	0	0	0	0	0
1	0	whole milk	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0

```
x = file2.values
#Convert into list values in each row if value is not zero
x = [sub[~(sub==0)].tolist() for sub in x if sub [sub != 0].tolist()]
transactions = x
transactions[0:10]
```

```
[['whole milk', 'yogurt', 'sausage', 'semi-finished bread'],
 ['whole milk', 'pastry', 'salty snack'],
 ['canned beer', 'misc. beverages'],
 ['sausage', 'hygiene articles'],
 ['soda', 'pickled vegetables'],
 ['frankfurter', 'curd'],
 ['whole milk', 'rolls/buns', 'sausage'],
 ['whole milk', 'soda'],
 ['beef', 'white bread'],
 ['frankfurter', 'soda', 'whipped/sour cream']]
```

```
associations = apriori(transactions, min_support = 0.00030, min_confidence = 0.05, min_li
association_results = list(associations)
print(association_results[0])
```

```
RelationRecord(items=frozenset({'liver loaf', 'fruit/vegetable juice'}), support=0.00
```

```
for item in association_results:

    #for each item filter out the item pair and create item list containing individual it
    itemset = item[0]
    items = [x for x in itemset]

    #Print the relationship( First value in items to second value in items)
    print("Rule : ", items[0], " -> " + items[1])

    #Print support,confidence and lift value of each itemset
    print("Support : ", str(item[1]))
    print("Confidence : ",str(item[2][0][2]))
    print("Lift : ", str(item[2][0][3]))

    print("=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>")
```

```
Rule :  liver loaf  -> fruit/vegetable juice
Support :  0.00040098910646260775
```

```
Confidence :  0.12
Lift :  3.5276227897838903
=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>
Rule :  ham  -> pickled vegetables
Support :  0.0005346521419501437
Confidence :  0.05970149253731344
Lift :  3.4895055970149254
=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>
Rule :  roll products   -> meat
Support :  0.0003341575887188398
Confidence :  0.06097560975609757
Lift :  3.620547812620984
=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>
Rule :  misc. beverages  -> salt
Support :  0.0003341575887188398
Confidence :  0.05617977528089888
Lift :  3.5619405827461437
=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>
Rule :  spread cheese  -> misc. beverages
Support :  0.0003341575887188398
Confidence :  0.05
Lift :  3.170127118644068
=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>
Rule :  soups  -> seasonal products
Support :  0.0003341575887188398
Confidence :  0.10416666666666667
Lift :  14.704205974842768
=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>
Rule :  spread cheese  -> sugar
Support :  0.00040098910646260775
Confidence :  0.06
Lift :  3.3878490566037733
=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>
```