Market Basket Insights

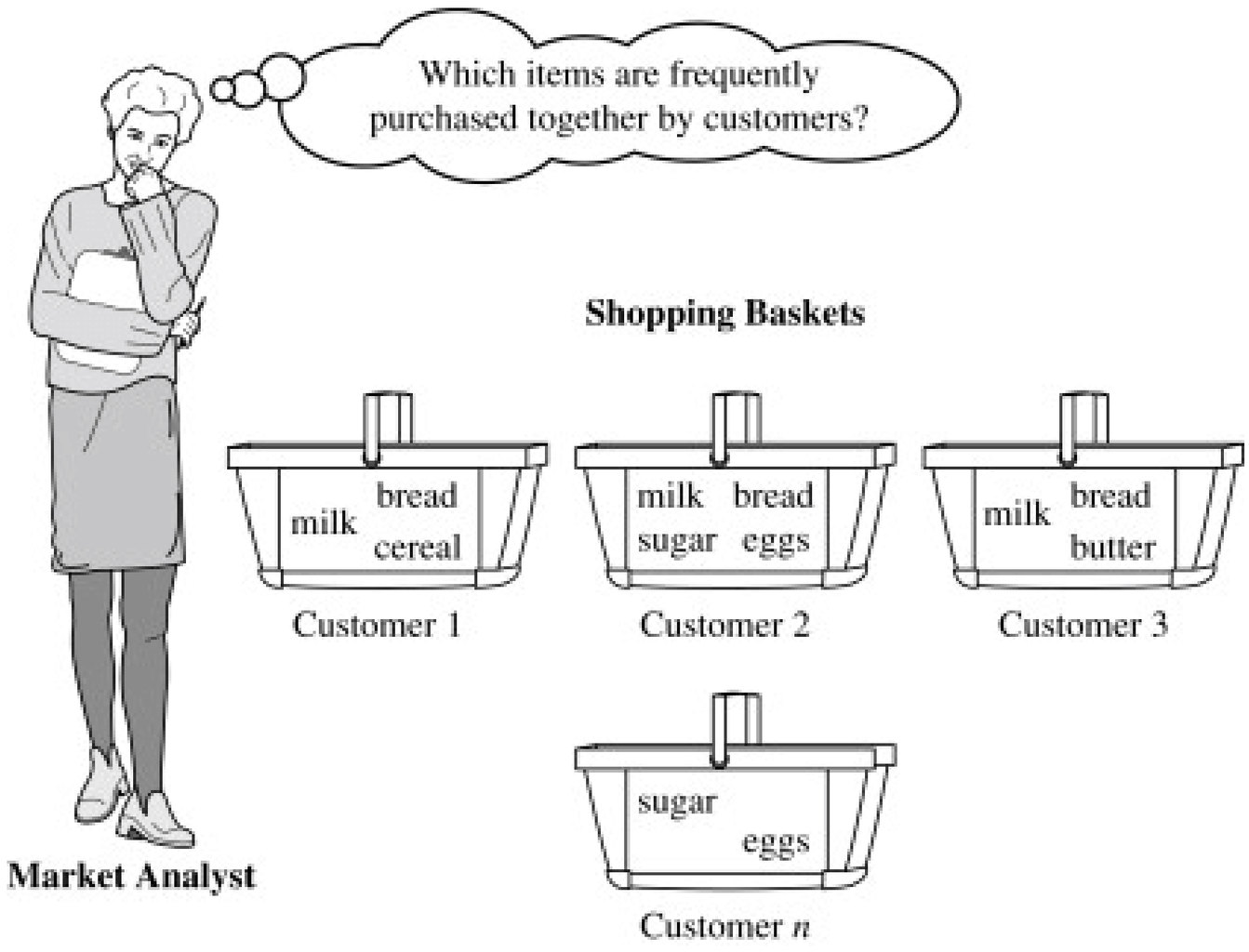
Phase 3 project submission

Project title: market basket insights

Phase 3: Development part 1

Table of contents

What Is Association Rule for Market Basket Analysis?



Let I = {I1, I2,… , I m} be an item set. These item sets are called antecedents. Let D, the data, be a set of database transactions where each transaction T is a nonempty item set such that **T ⊆I**. Each transaction is associated with an identifier called a TID(or T id). Let A be a set of items (item set). T is

the Transaction that is said to contain A if $A \subseteq T$. An **Association Rule** is an implication of form $A \Rightarrow B$, where $A \subset I$, $B \subset I$, and $A \cap B = \varphi$.

The rule $A \Rightarrow B$ holds in the data set(transactions) D with supports, where ' s' is the percentage of transactions in D that contain $A \cup B$ (i.e., the union of set A and set B, or both A and B). This is taken as the probability, $P(A \cup B)$. Rule $A \Rightarrow B$ has confidence **c** in the transaction set D, where c is the percentage of transactions in D containing **A** that also contains **B**. This is taken to be the conditional probability, like P (B|A). That is,

- ***support*** $(A \Rightarrow B) = P(A \cup B)$
- ***confidence*** $(A \Rightarrow B) = P(B|A)$

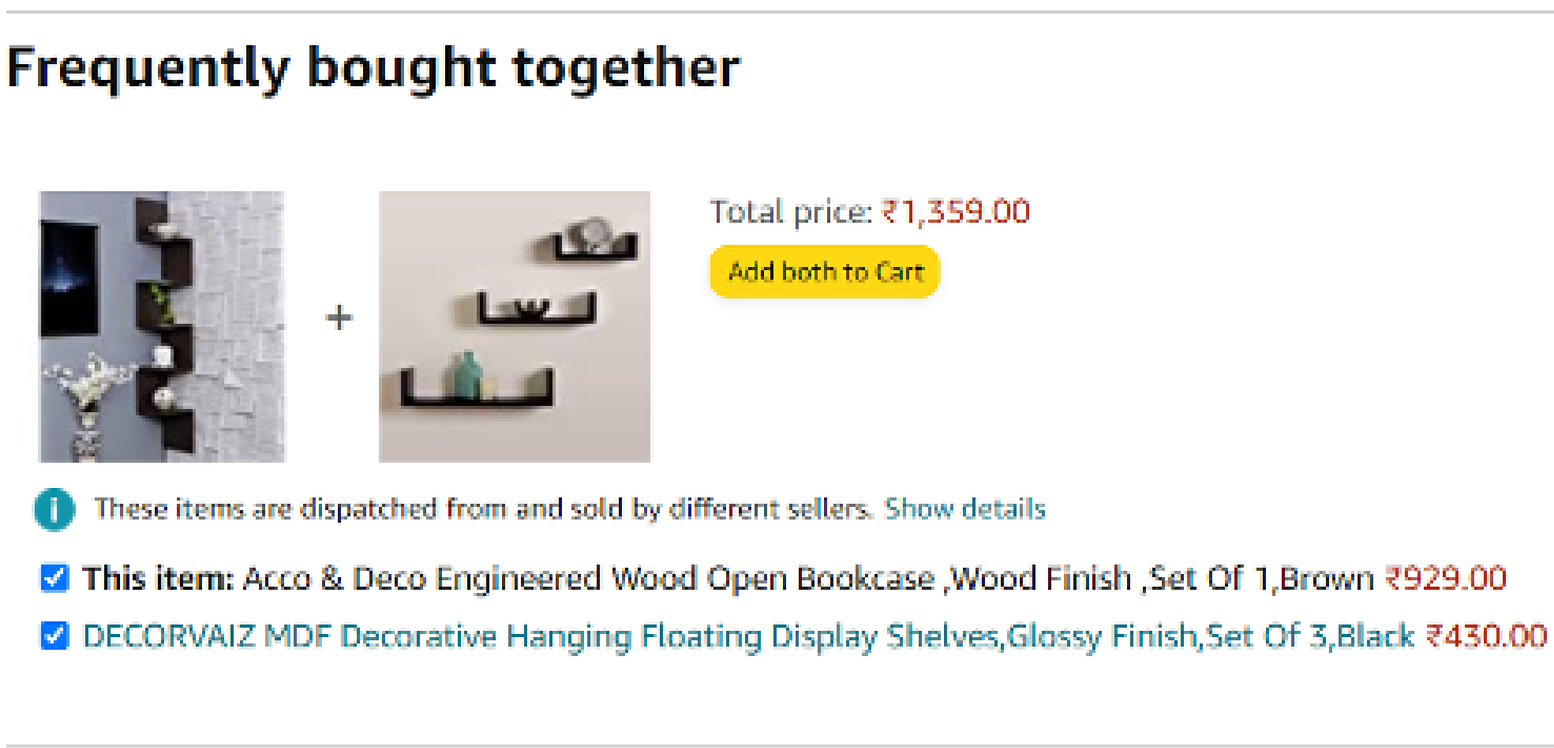
Rules that satisfy both a minimum support threshold (called min sup) and a minimum confidence threshold (called min conf) are called " ***Strong***" .

- ***Confidence*** $(A \Rightarrow B) = P(B|A) =$
- ***support*** $(A \cup B)$ ***/support*** $(A) =$
- ***support count*** $(A \cup B)$ ***/ support count*** (A)

Generally, Association Rule Mining can be viewed in a two-step process:

1. Find all frequent item sets: *By definition, each of these items e its will occur at least as frequently as a pre-established minimum support count, min sup.*
2. Generate Association Rules from the frequent item sets: *By definition, these rules must satisfy minimum support and minimum confidence.*

   Example:



Algorithms Used in Market Basket Analysis

There are multiple data mining techniques and algorithms used in Market Basket Analysis. One of the important objectives is “ *to predict the probability of items that are being bought together by customers.*”

- **A priori Algorithm**
- **AIS**
- **SETM Algorithm**
- **FP Growth**

1. Apriority Algorithm

A priori Algorithm is a widely-used and well-known Association Rule algorithm and is a popular algorithm used in market basket analysis. It is also considered accurate and overtop AIS and SETM algorithms. It helps to find frequent item sets in transactions and identifies association rules between these items. The limitation of the A priori Algorithm is *frequent item set generation*. It needs to scan the database many times, leading to increased time and reduced performance as a computationally costly step because of a large dataset. It uses the concepts of Confidence and Support.

2. AIS Algorithm

The AIS algorithm creates multiple passes on the entire database or transactional data. During every pass, it scans all transactions. As you can see, in the first pass, it counts the support of separate items and determines then which of them are frequent in the database. Huge item sets of every pass are enlarged to generate candidate item sets. After each scanning of a transaction, the common item sets between the item sets of the previous pass and the items of this transaction are determined. This algorithm was the first published algorithm which is developed to generate all large item sets in a transactional database. It focused on the enhancement of databases with the necessary performance to process decision support. This technique is bounded to only one item in the consequent.

- **Advantage**: The AIS algorithm was used to find whether there was an association between items or not.
- **Disadvantage**: The main disadvantage of the AIS algorithm is that it generates too many candidates set that turn out to be small. As well as the data structure is to be maintained.

3. SETM Algorithm

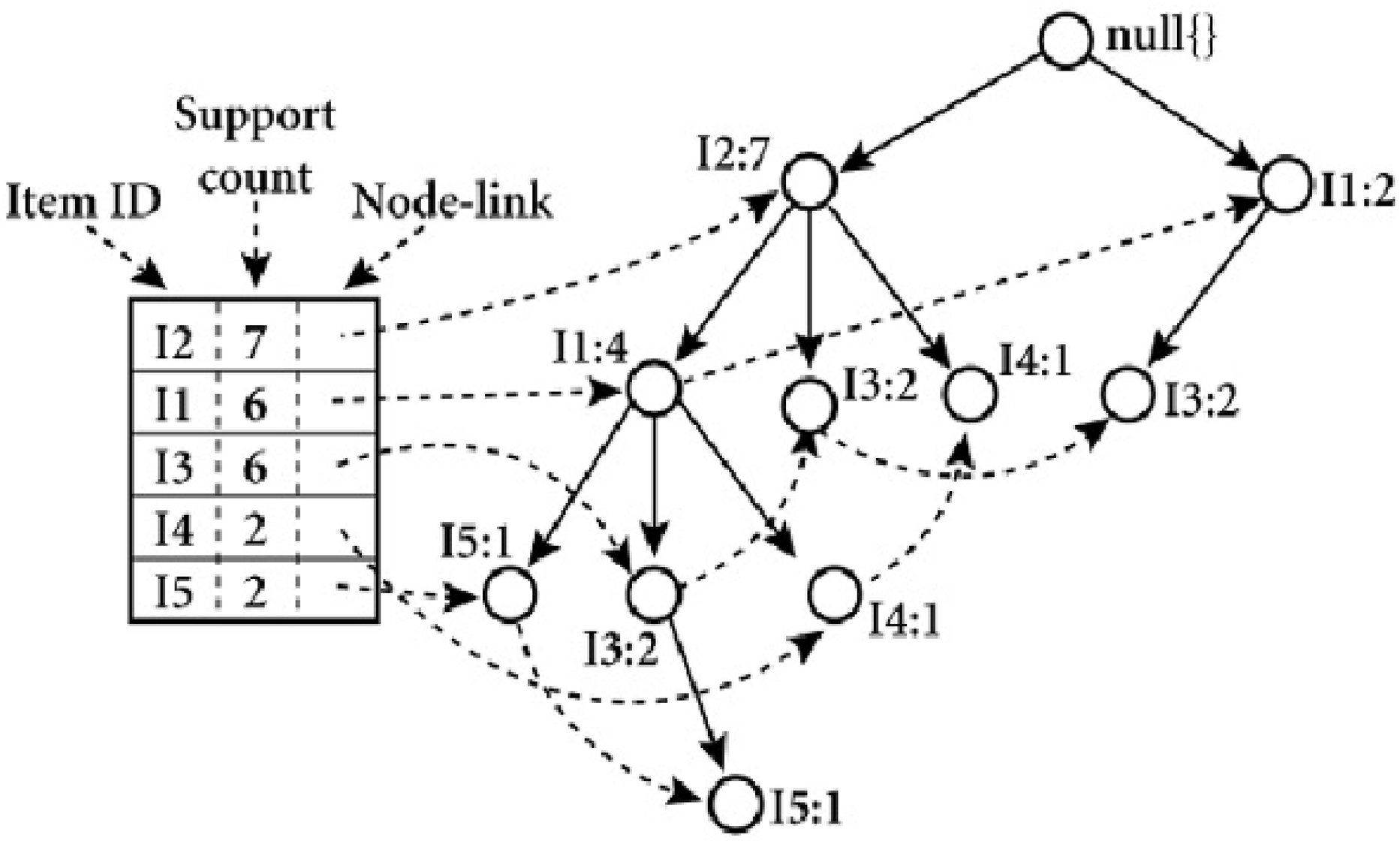
This Algorithm is quite similar to the AIS algorithm. The **SETM** algorithm creates collective passes over the database. As you can see, in the first pass, it counts the support of single items and then determines which of them are frequent in the database. Then, it also generates the candidate item sets by enlarging large item sets of the previous pass. In addition to this, the SETM algorithm recalls the TIDs (transaction ids) of the generating transactions with the candidate item sets.

- **Advantage**: While generating candidate, the SETM algorithm arranges candidate item sets together with the TID (transaction Id) in a sequential manner.
- **Disadvantage**: For every item set, there is an association with Tid; hence it requires more space to store a huge number of TIDs.

4. FP Growth

FP Growth is known as Frequent Pattern Growth Algorithm. FP growth algorithm is a concept of representing the data in the form of an FP tree or Frequent Pattern. Hence FP Growth is a method of *Mining Frequent Item sets*. This algorithm is advancement to the **Apriority Algorithm**. There is no need for candidate generation to generate a frequent pattern. This frequent pattern tree structure maintains the association between the item sets.

A Frequent Pattern Tree is a tree structure that is made with the earlier item sets of the data. The main purpose of the FP tree is to mine the most frequent patterns. Every node of the FP tree represents an item of that item set. The root node represents the null value, whereas the lower nodes represent the item sets of the data. The association of these nodes with the lower nodes, that is, between item sets, is maintained while creating the tree.



Implementing Market Basket Analysis in Python

The Method

Here are the steps involved in using the a priori algorithm to implement MBA:

1. First, define the minimum support and confidence for the association rule.
2. Find out all the subsets in the transactions with higher support(sup) than the minimum support.
3. Find all the rules for these subsets with higher confidence than minimum confidence.
4. Sort these association rules in decreasing order.
5. Analyze the rules along with their confidence and support.

The Dataset

In this implementation, we have to use the Store Data dataset that is publicly available on Kaggle. This dataset contains a total of 7501 transaction records, where every record consists of a list of items sold in just one transaction.

Implementing Market Basket Analysis Using the A priori Method

The A priori algorithm is frequently used by data scientists. We are required to import the necessary libraries. Python provides the ***pyori*** as an API that is required to be imported to run the A priori Algorithm.

```
from IPython.core.display import HTML
HTML("""
<style>
.output_png {
    display: table-cell;
    text-align: center;
    vertical-align: middle;
    horizontal-align: middle;
}
h1,h2 {
    text-align: center;
    background-color: pink;
    padding: 20px;
    margin: 0;
    color: black;
    font-family: ariel;
    border-radius: 80px
}

h3 {
    text-align: center;
    border-style: solid;
    border-width: 3px;
    padding: 12px;
    margin: 0;
    color: black;
    font-family: ariel;
    border-radius: 80px;
    border-color: gold;
}

body, p {
    font-family: ariel;
    font-size: 15px;
    color: charcoal;
}
div {
    font-size: 14px;
    margin: 0;

}

h4 {
    padding: 0px;
    margin: 0;
    font-family: ariel;
    color: purple;
```

```
}
</style>
""")
```

Flow of Execution:

1. Loading Necessary Packages
2. Loading dataset
3. Data Pre-Processing
4. Performing EDA
5. Apriori Implementation
6. Result Customization

Step - 1 : Loading Necessary Packages

```
!pip install apyori  ## Installing apriori library
Collecting apyori
  Downloading apyori-1.1.2.tar.gz (8.6 kB)
  Preparing metadata (setup.py) ... - done
Building wheels for collected packages: apyori
  Building wheel for apyori (setup.py) ... - \ done
  Created wheel for apyori: filename=apyori-1.1.2-py3-none-any.whl size=5974
sha256=c03b4c07b988bef21b16adf3c29d4091a48e0ba3cc7a41e46b557ef451395440
  Stored in directory:
/root/.cache/pip/wheels/cb/f6/e1/57973c631d27efd1a2f375bd6a83b2a616c4021f24aab84080
Successfully built apyori
Installing collected packages: apyori
Successfully installed apyori-1.1.2
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting
behaviour with the system package manager. It is recommended to use a virtual environment
instead: https://pip.pypa.io/warnings/venv
import numpy as np # linear algebra
import pandas as pd # Data pre-processing
import seaborn as sns # Required for plotting
import matplotlib.pyplot as plt # Required for plotting
```

Step - 2 : Loading dataset

```
df = pd.read_csv("../input/groceries-dataset/Groceries_dataset.csv") ## Loading dataset
df.head()
```

	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

```
df.info() # Checking data type information for validation purposes
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38765 entries, 0 to 38764
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
```

```
 0   Member_number    38765 non-null  int64
 1   Date             38765 non-null  object
 2   itemDescription  38765 non-null  object
dtypes: int64(1), object(2)
memory usage: 908.7+ KB
```

Interpretation: - No Null values should be present

```
df.isnull().sum().sort_values(ascending=False) ## Checking availability of NULL values
Member_number      0
Date               0
itemDescription    0
dtype: int64
```

Note - No NULLs present

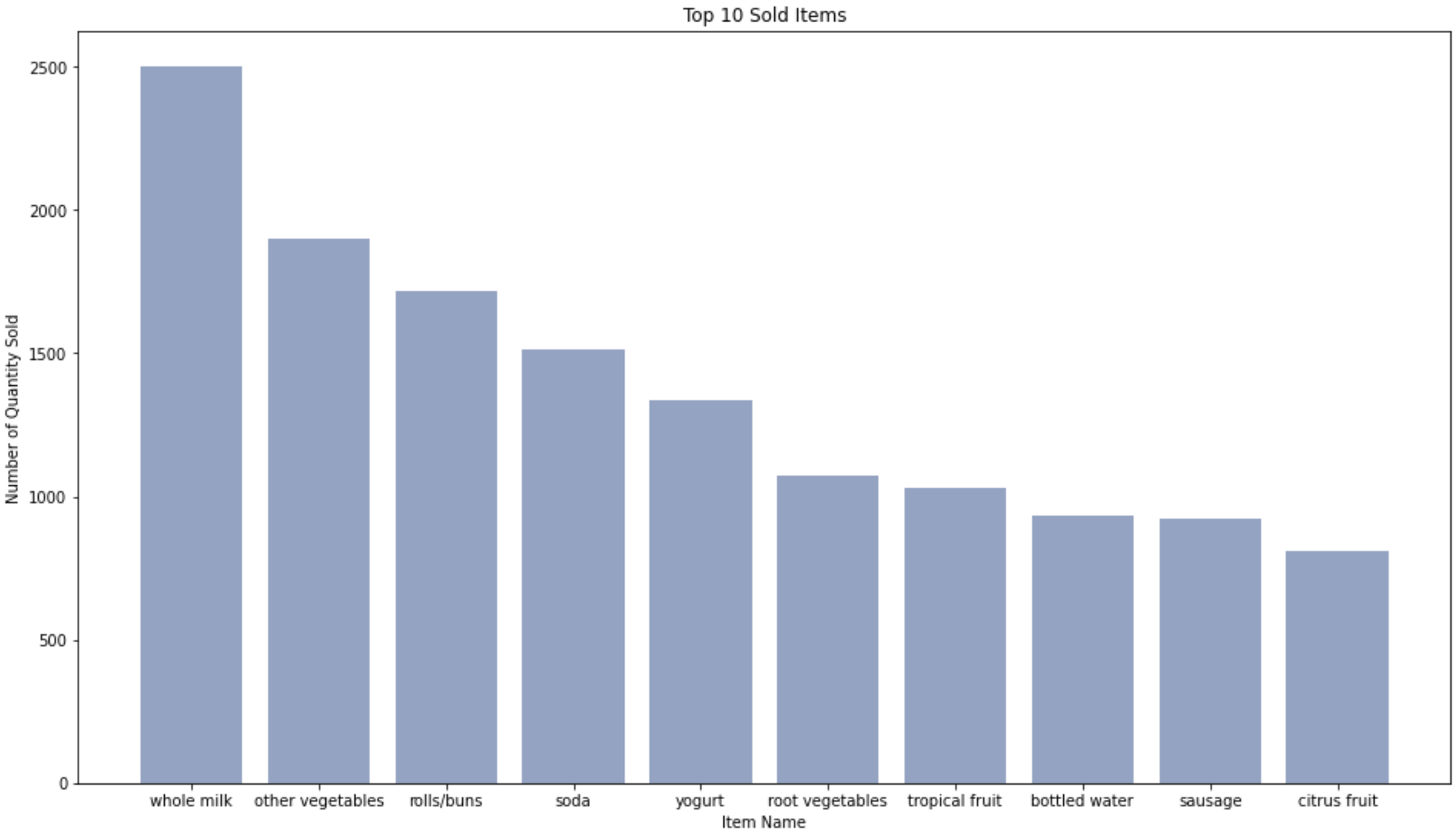
Step - 3 : Data Pre-Processing

```
df['Date'] = pd.to_datetime(df['Date']) ## Type-Conversion from Object to Dateime
df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38765 entries, 0 to 38764
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Member_number    38765 non-null  int64
 1   Date             38765 non-null  datetime64[ns]
 2   itemDescription  38765 non-null  object
dtypes: datetime64[ns](1), int64(1), object(1)
memory usage: 908.7+ KB
df.head() ## Schema check
```

	Member_number	Date	itemDescription
0	1808	2015-07-21	tropical fruit
1	2552	2015-05-01	whole milk
2	2300	2015-09-19	pip fruit
3	1187	2015-12-12	other vegetables
4	3037	2015-01-02	whole milk

Step - 4 : Performing EDA

Step - 4.1 : Top 10 Sold Items




```
## Creating distribution of Item Sold

Item_distr = df.groupby(by =
"itemDescription").size().reset_index(name='Frequency').sort_values(by =
'Frequency',ascending=False).head(10)

## Declaring variables

bars = Item_distr["itemDescription"]
height = Item_distr["Frequency"]
x_pos = np.arange(len(bars))

## Defining Figure Size

plt.figure(figsize=(16,9))

# Create bars
plt.bar(x_pos, height, color=(0.3, 0.4, 0.6, 0.6))

# Add title and axis names
plt.title("Top 10 Sold Items")
plt.xlabel("Item Name")
plt.ylabel("Number of Quantity Sold")

# Create names on the x-axis
plt.xticks(x_pos, bars)

# Show graph
plt.show()
```

Step - 4.2 : Month-Year Sales

```
df_date=df.set_index(['Date']) ## Setting date as index for plotting purpose
df_date
```

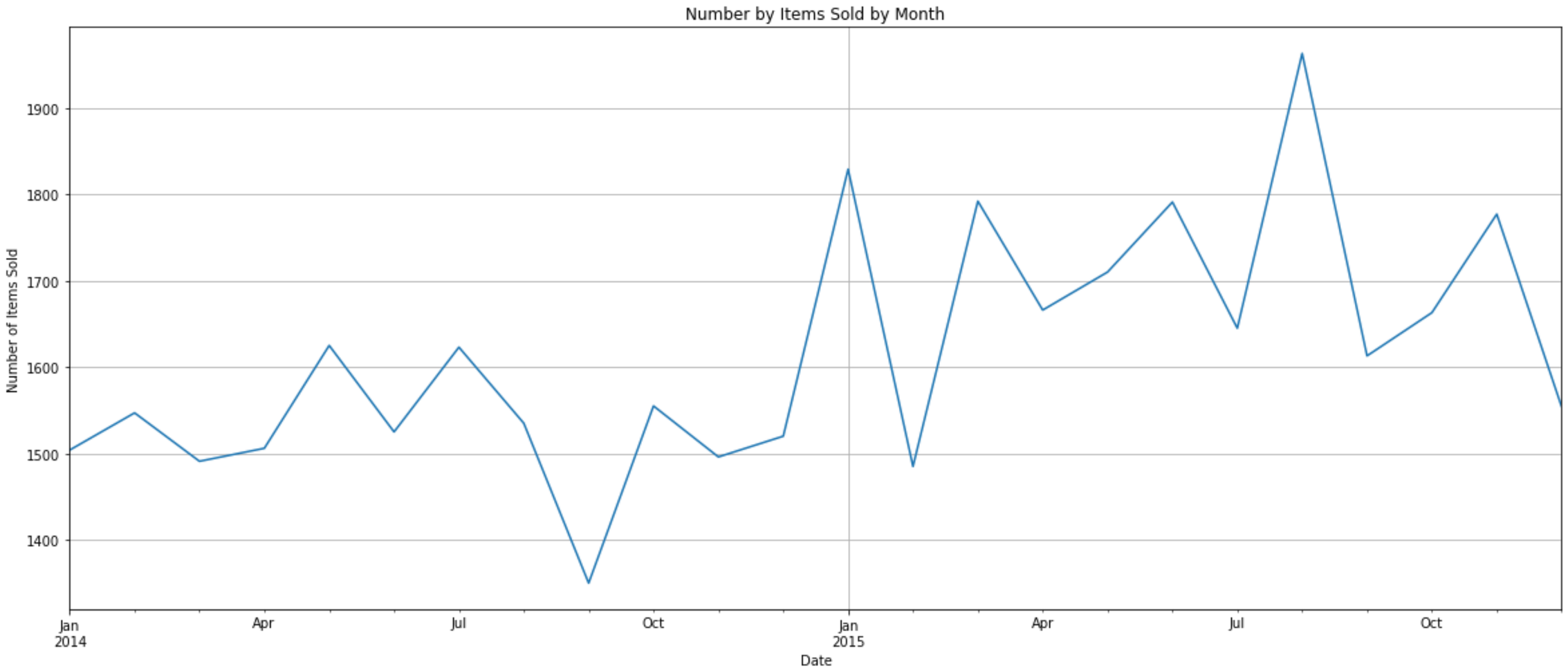
	Member_number	item Description
Date		
2015-07-21	1808	tropical fruit
2015-05-01	2552	whole milk
2015-09-19	2300	pip fruit
2015-12-12	1187	other vegetables
2015-01-02	3037	whole milk
...	...	...
2014-08-10	4471	sliced cheese
2014-02-23	2022	candy
2014-04-16	1097	cake bar
2014-03-12	1510	fruit/vegetable juice
2014-12-26	1521	cat food

38765 rows × 2 columns

```
df_date.resample("M")['itemDescription'].count().plot(figsize = (20,8), grid = True, title
= "Number by Items Sold by Month").set(xlabel = "Date", ylabel = "Number of Items Sold")
```
```
[Text(0.5, 0, 'Date'), Text(0, 0.5, 'Number of Items Sold')]
```

Step - 5 : Apriori Implementation

Apriori is an algorithm for frequent itemset mining and association rule learning over relational databases. It proceeds by identifying the frequent individual items in the database and extending them to larger and larger item sets as long as those item sets appear sufficiently often in the database. The frequent itemsets determined by Apriori can be used to determine association rules which highlight general trends in the database: this has applications in domains such as market basket analysis.



Step - 5.1 : Data Preparation

```
cust_level = df[["Member_number", "itemDescription"]].sort_values(by = "Member_number",
ascending = False) ## Selecting only required variables for modelling
cust_level['itemDescription'] = cust_level['itemDescription'].str.strip() # Removing white
spaces if any
cust_level
```

	Member number	Item Description
3578	5000	soda
34885	5000	semi-finished bread
11728	5000	fruit/vegetable juice
9340	5000	bottled beer
19727	5000	root vegetables
...	...	...
13331	1000	whole milk
17778	1000	pickled vegetables
6388	1000	sausage
20992	1000	semi-finished bread

	Member number	Item Description
8395	1000	whole milk

38765 rows × 2 columns

Step - 5.2 : Create Transaction list

```
transactions = [a[1]['itemDescription'].tolist() for a in
list(cust_level.groupby(['Member_number']))] ## Combing all the items in list format for
each cutomer
```

Step - 5.3 : Train Model

```
from apyori import apriori ## Importing apriori package
rules = apriori(transactions = transactions, min_support = 0.002, min_confidence = 0.05,
min_lift = 3, min_length = 2, max_length = 2) ## Model Creation
results = list(rules) ## Storing results in list format for better visualisation
results
[RelationRecord(items=frozenset({'UHT-milk', 'kitchen towels'}),
support=0.00230876346844536,
ordered_statistics=[OrderedStatistic(items_base=frozenset({'kitchen towels'}),
items_add=frozenset({'UHT-milk'}), confidence=0.3000000000000004,
lift=3.821568627450981)]),
 RelationRecord(items=frozenset({'potato products', 'beef'}), support=0.0025654181631605 95,
 ordered_statistics=[OrderedStatistic(items_base=frozenset({'potato products'}),
items_add=frozenset({'beef'}), confidence=0.454545454545454 6, lift=3.802184939523995 5)]),
 RelationRecord(items=frozenset({'canned fruit', 'coffee'}), support=0.00230876346844536,
ordered_statistics=[OrderedStatistic(items_base=frozenset({'canned fruit'}),
items_add=frozenset({'coffee'}), confidence=0.4285714285714286, lift=3.728954081632653 4)]),
 RelationRecord(items=frozenset({'domestic eggs', 'meat spreads'}),
support=0.0035915854284248334,
ordered_statistics=[OrderedStatistic(items_base=frozenset({'meat spreads'}),
items_add=frozenset({'domestic eggs'}), confidence=0.4, lift=3.004238921001926 7)]),
 RelationRecord(items=frozenset({'flour', 'mayonnaise'}), support=0.00230876346844536,
ordered_statistics=[OrderedStatistic(items_base=frozenset({'flour'}),
items_add=frozenset({'mayonnaise'}), confidence=0.06338028169014086,
lift=3.338599162542825), OrderedStatistic(items_base=frozenset({'mayonnaise'}),
items_add=frozenset({'flour'}), confidence=0.12162162162162163, lift=3.338599162542825)]),
 RelationRecord(items=frozenset({'napkins', 'rice'}), support=0.003078501795792714 3,
ordered_statistics=[OrderedStatistic(items_base=frozenset({'rice'}),
items_add=frozenset({'napkins'}), confidence=0.2448979591836735, lift=3.011395094315329)]),
 RelationRecord(items=frozenset({'sparkling wine', 'waffles'}),
support=0.0025654181631605 95,
ordered_statistics=[OrderedStatistic(items_base=frozenset({'sparkling wine'}),
items_add=frozenset({'waffles'}), confidence=0.2173913043478260 8,
lift=3.150153547761435 3)])]
```

Step - 6 : Result Customization

```
## Creating user-defined function for arranging the results obtained from model into
readable format

def inspect(results):
    lhs         = [tuple(result[2][0][0])[0] for result in results]
    rhs         = [tuple(result[2][0][1])[0] for result in results]
    supports    = [result[1] for result in results]
    confidences = [result[2][0][2] for result in results]
    lifts       = [result[2][0][3] for result in results]
    return list(zip(lhs, rhs, supports, confidences, lifts))
```

```
resultsinDataFrame = pd.DataFrame(inspect(results), columns = ['Left Hand Side', 'Right Hand Side', 'Support', 'Confidence', 'Lift'])
resultsinDataFrame.nlargest(n=10, columns="Lift") ## Showing best possible scenarios
```

	Left Hand Side	Right Hand Side	Support	Confidence	Lift
0	kitchen towels	UHT-milk	0.002309	0.300000	3.821569
1	potato products	beef	0.002565	0.454545	3.802185
2	canned fruit	coffee	0.002309	0.428571	3.728954
4	flour	mayonnaise	0.002309	0.063380	3.338599
6	sparkling wine	waffles	0.002565	0.217391	3.150154
5	rice	napkins	0.003079	0.244898	3.011395
3	meat spreads	domestic eggs	0.003592	0.400000	3.004239

Conclusion

In this tutorial, we discussed Market Basket Analysis and learned the steps to implement it from scratch using Python. We then implemented Market Basket Analysis using Apriori Algorithm. We also looked into the various uses and advantages of this algorithm and learned that we could also use FP Growth and AIS algorithms to implement Market Basket Analysis.

Key Takeaways

- Market Basket Analysis is a business strategy used to design store layouts based on customers' shopping behavior and purchase histories.
- This idea is also applicable to machine learning algorithms to teach machines to help businesses, especially in the e-commerce sector.
- In this article, we have gone through a step-by-step guide to implementing the apriori algorithm in Python and also looked into the math behind the association rules.