Understanding Market Basket Analysis using Python

April 13, 2024

1 Understanding Market Basket Analysis using Python

2 Introduction to Market Basket Analysis

Imagine strolling through a supermarket, your cart gradually filling with groceries. As you reach for the pasta, you notice a variety of pasta sauces conveniently placed nearby. This strategic arrangement isn't just about convenience; it's a calculated decision informed by **Market Basket Analysis (MBA)**. This technique helps businesses understand which products customers often purchase together, optimizing the shopping experience and improving store layout.

3 Understanding Market Basket Analysis

Market Basket Analysis is a technique used to identify the relationships between items frequently bought together. By analyzing these patterns, businesses can tailor their offerings to better meet customer needs, enhancing both the efficiency of shopping and the pleasure derived from it.

4 The Necessity of Market Basket Analysis

Market Basket Analysis is essential as it enables businesses to make data-driven decisions that cater to customer preferences. This strategy not only makes shopping more convenient but also encourages the purchase of complementary products, increasing sales and customer satisfaction.

5 Potential Users of Market Basket Analysis

This analysis tool is invaluable across various sectors:

- **Retail Stores**: Improve product placement and store layout.
- **E-commerce Platforms**: Customize recommendations to enhance user experience.
- **Content Providers**: Offer relevant media suggestions to viewers.
- **Service Industries**: Bundle related services for better deals.

6 The Value Added by Market Basket Analysis

Market Basket Analysis brings significant advantages, including optimized product placement, personalized marketing, and an overall enhanced customer experience. These benefits help in crafting a shopping environment that is both intuitive and user-friendly.

7 Use Cases for Market Basket Analysis

The application of Market Basket Analysis is vast, encompassing:

- **Product Bundling**: Offering discounts on commonly bought items together.
- **Layout Optimization**: Strategically placing related products to boost sales.
- **Enhanced Recommendations**: Providing personalized suggestions on online platforms.

8 Business Benefits of Market Basket Analysis

The insights from Market Basket Analysis have proven extremely valuable, leading to increased sales, higher customer satisfaction, and more efficient marketing strategies. These outcomes stem from a deeper understanding of customer behavior and preferences.

9 Importance of Market Basket Analysis

The strategic importance of Market Basket Analysis lies in its ability to convert raw sales data into actionable insights. This conversion allows businesses to proactively cater to customer needs and stay ahead in a competitive marketplace.

10 Drawbacks of Overlooking Market Basket Analysis

Neglecting Market Basket Analysis can result in missed sales opportunities, inefficient store operations, and ineffective marketing campaigns. Without the insights it provides, businesses may struggle to meet customer expectations and maximize their market potential.

11 A Step-by-Step Guide to Conducting Market Basket Analysis

Market Basket Analysis (MBA) is a powerful analytical tool used by businesses to understand the purchasing habits of their customers by examining the items they frequently buy together. This insight helps companies to make strategic decisions that enhance customer satisfaction and drive sales. Here's a step-by-step guide on how to conduct a Market Basket Analysis, which can be adapted to suit various types of businesses, from retail to e-commerce.

Step 1: Data Collection

The first step in Market Basket Analysis is gathering the necessary data. This typically includes detailed transaction records that list the items purchased together by each customer. Data can be collected through point-of-sale (POS) systems, online sales platforms, or customer loyalty programs. Ensure that the data is clean and comprehensive, including timestamps, item descriptions, quantities, and prices.

Step 2: Data Preparation

Once you have collected the data, the next step is to prepare it for analysis. This involves processing the data to a suitable format, usually a transactional dataset where each row represents a single transaction and the columns represent purchased items. This might require transforming the data into binary form (1s and 0s), indicating whether an item was purchased or not in a particular transaction.

Step 3: Identifying Itemsets

The core of Market Basket Analysis involves identifying frequent itemsets, or combinations of items that appear together in transactions more often than a specified threshold (support threshold). This is typically done using algorithms like **Apriori, FP-Growth, or Eclat**. These algorithms help to efficiently sift through large datasets and find items that meet the minimum support criterion.

Step 4: Rule Generation

After identifying frequent itemsets, the next step is to generate **association rules** from these itemsets. These rules suggest that if a customer buys a certain set of items **(antecedent)**, they are likely to buy another item or set of items **(consequent)**. The strength of these rules is measured using metrics such as **confidence** (the likelihood of purchasing the consequent items, given the antecedent items) and **lift** (the improvement in the ratio of sale of the consequent when the antecedent is sold).

Step 5: Analyzing the Results

Once the rules have been generated, they need to be analyzed to draw meaningful conclusions. This involves sorting the rules by their confidence, lift, and other relevant metrics to identify the most valuable insights. For example, a rule indicating that customers who buy flour and sugar are highly likely to buy baking powder as well might lead a grocery store to place these items near each other.

Step 6: Implementing Insights

The actionable insights gathered from the analysis must then be implemented to optimize business practices. This could include changing the store layout, tailoring marketing campaigns, adjusting inventory levels, or offering bundled products at a discounted rate. The implementation should align with the business strategy and be monitored for effectiveness.

Step 7: Monitoring and Refining

Market Basket Analysis is not a one-time process but requires ongoing monitoring and refinement to adapt to changing customer behaviors and market conditions. Regularly updating the data and repeating the analysis will help maintain the relevance of the findings and ensure that the business continues to meet customer needs effectively.

12 Market Basket Analysis using Python

Download the dataset below to solve this Data Science case study on Market Basket Analysis.

Link to download www.iamhere.com

The dataset provided contains transactional data, with columns representing:

BillNo: A unique identifier for each customer's bill. Itemname: The name of the purchased item. Quantity: The quantity of the item purchased. Price: The price of each item. CustomerID: A unique identifier for each customer.

Perform a Market Basket Analysis on the given dataset to discover associations between purchased items. Specifically, the goal is to identify:

- Which items are frequently purchased together by customers.
- The strength of these associations (e.g., support, confidence, lift).

12.0.1 Guide to Solving a Data Science Case Study on Market Basket Analysis

To get started with this Market Basket Analysis case study, follow these steps:

Step 1: Download the Dataset First, you'll need the data. Download the dataset for this case study from the following link here. This dataset contains transactional data, which is crucial for conducting your analysis.

Understanding the Dataset* Once you have downloaded the dataset, it's important to understand what each column represents:

- **BillNo**: This is a unique identifier for each customer's bill. It helps you track each transaction separately.
- **Itemname**: The name of the item that was purchased. This tells you what was bought in each transaction.
- **Quantity**: The number of units of the item purchased in each transaction.
- **Price**: The cost of each item at the time of purchase.
- **CustomerID**: A unique identifier for each customer who made the purchase.

Step 2: Perform Market Basket Analysis Now, you're set to start the Market Basket Analysis. Your main objectives will be to:

Identify Frequently Purchased Items: Determine which items are commonly bought together by customers. This helps in understanding shopping patterns and can guide product placement and promotions.

Analyze the Strength of Associations: Calculate metrics such as support, confidence, and lift for item associations. These metrics will help you understand not just which items are frequently bought together, but also the strength of these relationships:

- **Support** measures how frequently the itemset appears in the dataset.
- **Confidence** indicates the likelihood that an item B is purchased when item A is purchased.
- **Lift** compares the observed frequency of A and B appearing together with the frequency expected if they were statistically independent (a lift greater than 1 indicates a strong association).

Let's begin the Market Basket Analysis by importing the necessary Python libraries and loading the dataset.

[25]:
```
# Import necessary libraries
import pandas as pd  # For data manipulation and analysis
import plotly.express as px  # For interactive data visualization
import plotly.io as pio  # For Plotly configuration
import plotly.graph_objects as go  # For more advanced Plotly visualizations

# Set Plotly template to a clean white background
pio.templates.default = "plotly_white"

# Read data from a CSV file into a DataFrame
data = pd.read_csv("market_basket_dataset.csv")
```

[26]:
```
# Print the first few rows of the dataset to inspect its structure and contents
print(data.head())
```

	BillNo	Itemname	Quantity	Price	CustomerID
0	1000	Apples	5	8.30	52299
1	1000	Butter	4	6.06	11752
2	1000	Eggs	4	2.66	16415
3	1000	Potatoes	4	8.10	22889
4	1004	Oranges	2	7.26	52255

Let's check for any null values in the data before we proceed further.

[27]:
```
# Print the number of missing values in each column of the dataset
print(data.isnull().sum())
```

```
BillNo        0
Itemname      0
Quantity      0
Price         0
CustomerID    0
dtype: int64
```

Next, let's review the summary statistics to better understand the dataset.

[28]:
```
print(data.describe())
```

```
            BillNo    Quantity       Price    CustomerID
count   500.000000  500.000000  500.000000    500.000000
mean   1247.442000    2.978000    5.617660  54229.800000
std     144.483097    1.426038    2.572919  25672.122585
min    1000.000000    1.000000    1.040000  10504.000000
25%    1120.000000    2.000000    3.570000  32823.500000
50%    1246.500000    3.000000    5.430000  53506.500000
75%    1370.000000    4.000000    7.920000  76644.250000
max    1497.000000    5.000000    9.940000  99162.000000
```

[29]:
```
# Print summary statistics of the dataset
print(data.describe())
```

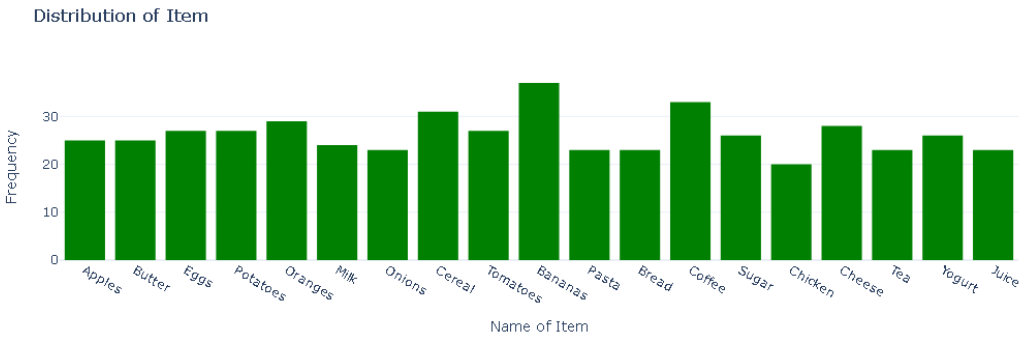
```
            BillNo    Quantity       Price    CustomerID
count   500.000000  500.000000  500.000000    500.000000
mean   1247.442000    2.978000    5.617660  54229.800000
std     144.483097    1.426038    2.572919  25672.122585
min    1000.000000    1.000000    1.040000  10504.000000
25%    1120.000000    2.000000    3.570000  32823.500000
50%    1246.500000    3.000000    5.430000  53506.500000
75%    1370.000000    4.000000    7.920000  76644.250000
max    1497.000000    5.000000    9.940000  99162.000000
```

Let's now examine how the items' sales are distributed across the dataset.

[30]:
```
# Create a histogram with green bars and show labels to visualize the␣
↪distribution of items
```

```
fig = px.histogram(data, x='Itemname', title='Distribution of Item',␣
 ↪color_discrete_sequence=['green'])

# Update layout to add axis labels
fig.update_layout(xaxis_title='Name of Item', yaxis_title='Frequency')

# Show the histogram
fig.show()
```

```
C:\ProgramData\anaconda3\Lib\site-packages\plotly\io\_renderers.py:395:
DeprecationWarning:

distutils Version classes are deprecated. Use packaging.version instead.

C:\ProgramData\anaconda3\Lib\site-packages\plotly\io\_renderers.py:395:
DeprecationWarning:

distutils Version classes are deprecated. Use packaging.version instead.
```

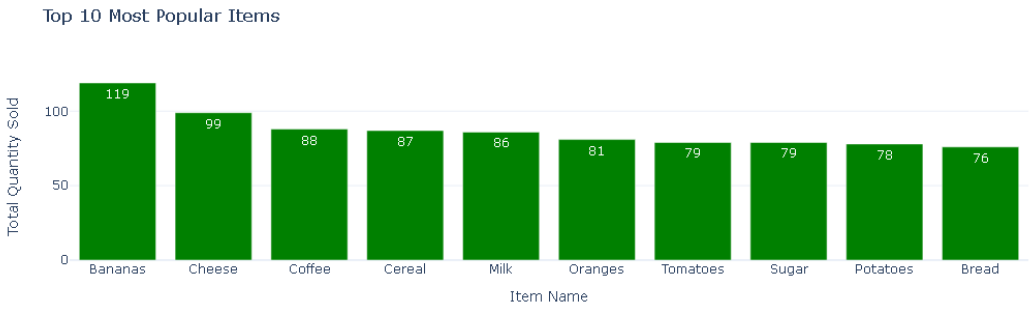


Let's take a look at the top 10 best-selling items in the store."

[31]:
```
# Calculate item popularity by grouping the data by item name and summing the␣
 ↪quantities sold for each item
item_popularity = data.groupby('Itemname')['Quantity'].sum().
 ↪sort_values(ascending=False)

# Specify the number of top items to display
top_n = 10

# Create a new figure
fig = go.Figure()

# Add a bar trace to the figure, representing the top N most popular items
```

```
fig.add_trace(go.Bar(
    x=item_popularity.index[:top_n],  # X-axis: Item names
    y=item_popularity.values[:top_n],  # Y-axis: Total quantities sold
    text=item_popularity.values[:top_n],  # Text displayed on top of each bar
    textposition='auto',  # Automatically position the text on the bars
    marker=dict(color='green')  # Set the color of the bars to green
))

# Update the layout of the figure to set the title and axis labels
fig.update_layout(
    title=f'Top {top_n} Most Popular Items',  # Title of the plot
    xaxis_title='Item Name',  # Label for the x-axis
    yaxis_title='Total Quantity Sold'  # Label for the y-axis
)

# Show the figure
fig.show()
```

```
C:\ProgramData\anaconda3\Lib\site-packages\plotly\io\_renderers.py:395:
DeprecationWarning:

distutils Version classes are deprecated. Use packaging.version instead.

C:\ProgramData\anaconda3\Lib\site-packages\plotly\io\_renderers.py:395:
DeprecationWarning:

distutils Version classes are deprecated. Use packaging.version instead.
```



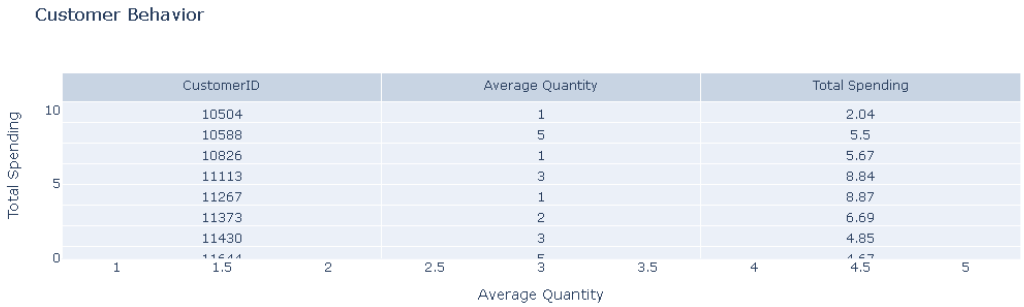
Bananas are the most popular items sold at the store. Next, let's examine customer behavior.

[32]:
```
# Calculate average quantity and spending per customer by grouping the data by␣
↪CustomerID
```

```
customer_behavior = data.groupby('CustomerID').agg({'Quantity': 'mean', 'Price':
 ↪ 'sum'}).reset_index()

# Create a DataFrame to display the values
table_data = pd.DataFrame({
    'CustomerID': customer_behavior['CustomerID'],
    'Average Quantity': customer_behavior['Quantity'],
    'Total Spending': customer_behavior['Price']
})

# Create a subplot with a scatter plot and a table
fig = go.Figure()

# Add a scatter plot
fig.add_trace(go.Scatter(
    x=customer_behavior['Quantity'],  # X-axis: Average quantity per customer
    y=customer_behavior['Price'],  # Y-axis: Total spending per customer
    mode='markers',  # Scatter plot mode
    text=customer_behavior['CustomerID'],  # Text displayed on hover
    marker=dict(size=10, color='coral')  # Marker settings: size and color
))

# Add a table
fig.add_trace(go.Table(
    header=dict(values=['CustomerID', 'Average Quantity', 'Total Spending']), ␣
 ↪# Table header
    cells=dict(values=[table_data['CustomerID'], table_data['Average␣
 ↪Quantity'], table_data['Total Spending']]),  # Table data
))

# Update layout with title and axis labels
fig.update_layout(
    title='Customer Behavior',  # Title of the plot
    xaxis_title='Average Quantity',  # Label for the x-axis
    yaxis_title='Total Spending'  # Label for the y-axis
)

# Show the plot
fig.show()
```

```
C:\ProgramData\anaconda3\Lib\site-packages\plotly\io\_renderers.py:395:
DeprecationWarning:

distutils Version classes are deprecated. Use packaging.version instead.

C:\ProgramData\anaconda3\Lib\site-packages\plotly\io\_renderers.py:395:
DeprecationWarning:
```

```
distutils Version classes are deprecated. Use packaging.version instead.
```



[33]:
```
# Calculate average quantity and spending per customer by grouping the data by␣
 ↪CustomerID
customer_behavior = data.groupby('CustomerID').agg({'Quantity': 'mean', 'Price':
 ↪ 'sum'}).reset_index()

# Create a DataFrame to display the values
table_data = pd.DataFrame({
    'CustomerID': customer_behavior['CustomerID'],
    'Average Quantity': customer_behavior['Quantity'],
    'Total Spending': customer_behavior['Price']
})

# Create a subplot with a scatter plot and a table
fig = go.Figure()


# Add a table
fig.add_trace(go.Table(
    header=dict(values=['CustomerID', 'Average Quantity', 'Total Spending']), ␣
 ↪# Table header
    cells=dict(values=[table_data['CustomerID'], table_data['Average␣
 ↪Quantity'], table_data['Total Spending']],  # Table data
               fill_color=[['lightgreen', 'lightgreen', 'lightgreen']] +␣
 ↪[['lightgreen', 'lightgreen', 'lightgreen']] * len(table_data)),  # Set fill␣
 ↪color for each cell
))


# Show the plot
```

```
fig.show()
```

```
C:\ProgramData\anaconda3\Lib\site-packages\plotly\io\_renderers.py:395:
DeprecationWarning:

distutils Version classes are deprecated. Use packaging.version instead.

C:\ProgramData\anaconda3\Lib\site-packages\plotly\io\_renderers.py:395:
DeprecationWarning:

distutils Version classes are deprecated. Use packaging.version instead.
```

CustomerID	Average Quantity	Total Spending
10504	1	2.04
10588	5	5.5
10826	1	5.67
11113	3	8.84
11267	1	8.87
11373	2	6.69
11430	3	4.85

We're now looking at customer behavior, comparing how much and how often they buy, and checking the exact numbers for each customer in the table.

Next, we'll use the Apriori algorithm to create rules that show which items are often bought together. This algorithm helps us see patterns in what people buy, which can guide decisions about where to place products in the store, what promotions to offer, and how to market them. Here's how we set up the Apriori algorithm to find these patterns:

[34]:
```
# Import required functions from mlxtend

#mlxtend module is not installed in your environment. You can install it using␣
 ↪pip. If you're using Jupyter Notebook or a similar environment, you can run␣
 ↪the following command in a new cell:
#!pip install mlxtend

from mlxtend.frequent_patterns import apriori, association_rules

# Group items by BillNo and create a list of items for each bill
basket = data.groupby('BillNo')['Itemname'].apply(list).reset_index()

# Convert the DataFrame containing one-hot encoded items to boolean type
```

```
basket_encoded = basket['Itemname'].str.join('|').str.get_dummies('|').
 ↪astype(bool)

# Find frequent itemsets using Apriori algorithm with lower support
frequent_itemsets = apriori(basket_encoded, min_support=0.01, use_colnames=True)

# Generate association rules with lower lift threshold
rules = association_rules(frequent_itemsets, metric='lift', min_threshold=0.5)

# Display association rules with selected metrics
print(rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']].
 ↪head(10))
```

```
  antecedents consequents   support  confidence      lift
0     (Bread)    (Apples)  0.045752    0.304348  1.862609
1    (Apples)     (Bread)  0.045752    0.280000  1.862609
2    (Butter)    (Apples)  0.026144    0.160000  0.979200
3    (Apples)    (Butter)  0.026144    0.160000  0.979200
4    (Apples)    (Cereal)  0.019608    0.120000  0.592258
5    (Cereal)    (Apples)  0.019608    0.096774  0.592258
6    (Cheese)    (Apples)  0.039216    0.214286  1.311429
7    (Apples)    (Cheese)  0.039216    0.240000  1.311429
8   (Chicken)    (Apples)  0.032680    0.250000  1.530000
9    (Apples)   (Chicken)  0.032680    0.200000  1.530000
```

The output we see shows the association rules between different items—those being purchased and those they are often bought with. Let's break down the output step by step:

- **Antecedents**: These are the items that act as the "if" part of our rule. For instance, in our analysis, items like Bread, Butter, Cereal, Cheese, and Chicken are antecedents.
- **Consequents**: These items are what tend to be purchased alongside the antecedents—the "then" part of our rule.
- **Support**: This measures how often a particular combination of items shows up in the dataset. It tells us the percentage of transactions where the items are bought together. For example, the first rule shows that Bread and Apples appear together in about 4.58% of all transactions.
- **Confidence**: This metric tells us how likely the consequent item is to be bought when the antecedent item is already in the shopping basket. It gives us the probability of purchasing the consequent when the antecedent is bought. For example, according to the first rule, if you buy Bread, there's a 30.43% chance you'll also pick up Apples.
- **Lift**: Lift shows how strong the association is between the antecedent and consequent, taking into account the overall likelihood of the consequent being purchased. A lift value above 1 suggests that the items are more likely to be bought together than separately. A value below 1 would indicate a negative association. For instance, the lift value in the first rule is about 1.86, indicating a positive relationship between buying Bread and Apples.

This is how you can conduct a Market Basket Analysis with Python.

13 Conclusion

In this article, we've explored how to conduct Market Basket Analysis using Python, a powerful tool for uncovering patterns in purchasing data. Starting from downloading and preparing the dataset, to applying the Apriori algorithm for generating association rules, we have walked through each step of the process. This analysis provides insights into the purchasing behaviors of customers, highlighting which items are frequently bought together.

Market Basket Analysis is a valuable tool for businesses seeking to optimize their product offerings, increase cross-selling opportunities, and improve marketing strategies. By understanding these patterns, businesses can make informed decisions about product placement, promotional strategies, and inventory management, all aimed at enhancing the shopping experience and increasing sales. The techniques discussed not only help in visualizing data-driven strategies but also in tailoring business approaches to meet the specific needs of the consumer base.

Implementing Market Basket Analysis is an excellent way for businesses to stay competitive in a market driven by consumer preferences. It can lead to higher revenue, enhanced customer satisfaction, and overall business success. With the foundation laid in this guide, you are now equipped to apply these methods to your own data, enabling you to discover valuable associations that can translate into actionable business strategies.

If you have any questions, please don't hesitate to ask in the comments section below

You can access the notebook via this GitHub Repository

My name is **David Akanji** and you can follow me on linkedin or direct your enquiry to akanjiolubukoladavid@gmail.com

[ ]: