Market Basket Analysis

July 2024

Disclaimer

To ensure data privacy, all product references in this analysis have been replaced with generic placeholders. The placeholder images used are royalty-free and bear no association with Apple or any other entity. Although they resemble Apple products, this analysis is not affiliated with Apple in any way.

Frequent Itemset Mining

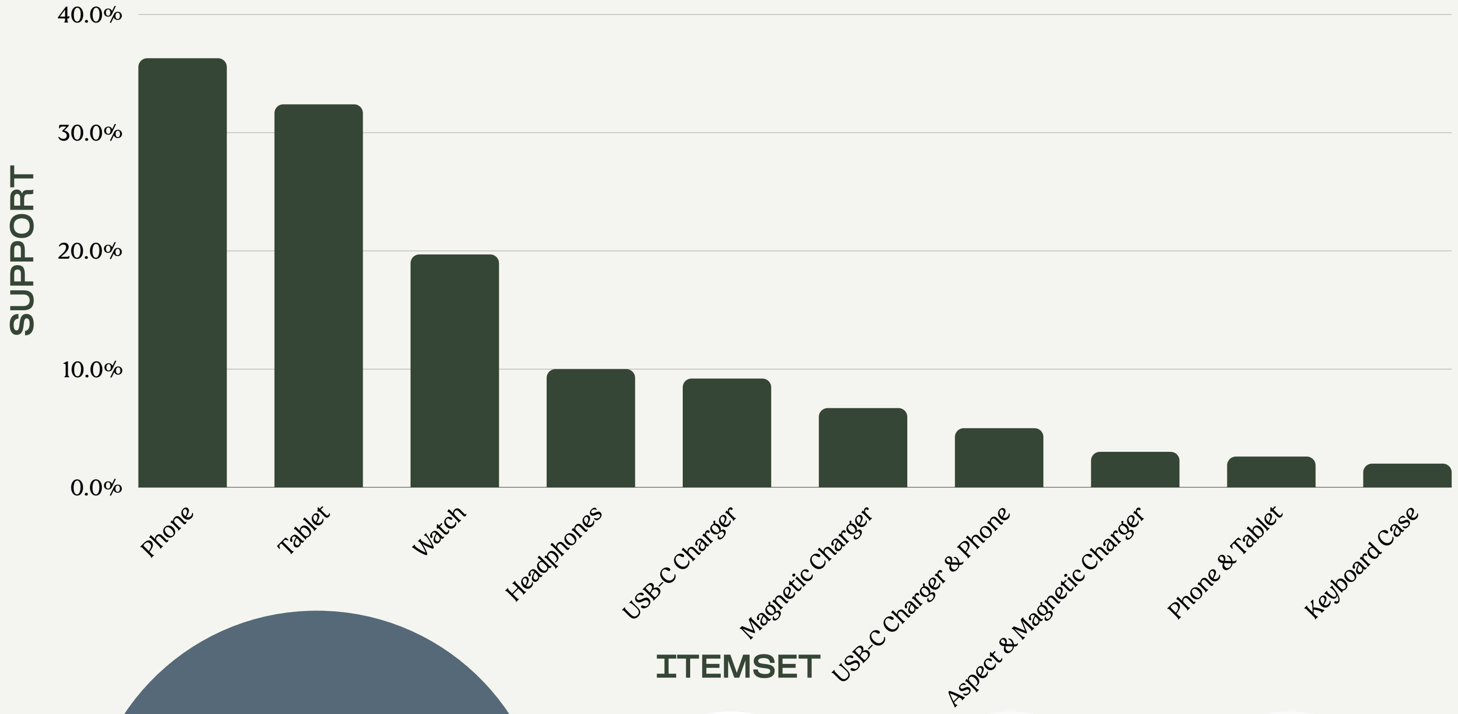
WHAT IS IT?

Frequent itemset mining determines which products are frequently purchased together. It's valuable for e-commerce businesses to enhance bundle offerings and improve cross-selling strategies.

METHODOLOGY

Orders from August 1, 2023 to current were exported for analysis. Using an Apriori Algorithm package in Python, it was determined which products are frequently purchased together (visit FAQ for more technical information).

August 1, 2023 was chosen as it marks the release date of the newest product.



1st
Phone

2nd
Tablet

3rd
Watch

4th
Headphones

Association Rule Mining

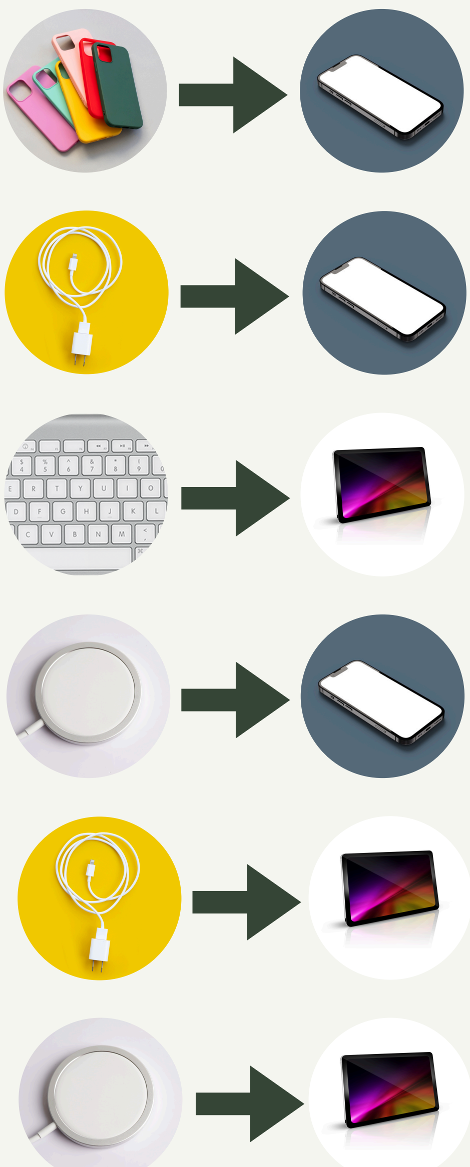
WHAT IS IT?

Association rule mining takes frequent itemset mining a step further by creating conditional “if-then” rules to determine how likely a product is to occur in an order, given the presence of another.

A → B 20%

INTERPRETATION

When product A is in an order, there’s a 20% chance product B is also in the order



72% — LIFT: 24.7

INTERPRETATION

An order with a Case is 24.7x more likely to contain a Phone, compared to the probability of an order containing a Phone regardless of whether it included a Case

> 1: Occurs less than expected
= 1: Occurs as expected
< 1: Occurs more than expected

54% — LIFT: 1.5 — CURRENT BUNDLE

53% — LIFT: 1.6 — CURRENT BUNDLE

45% — LIFT: 1.2 — CURRENT BUNDLE

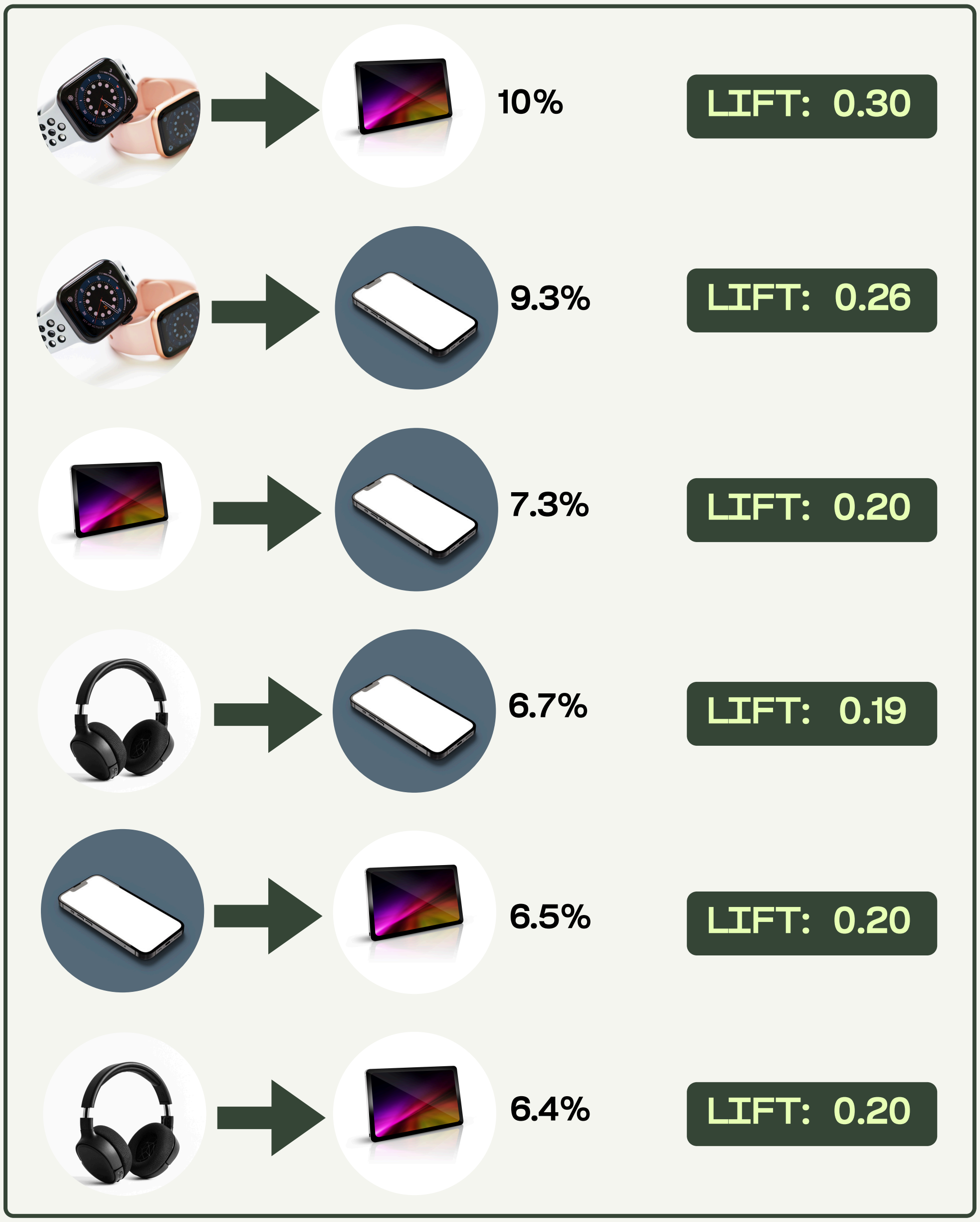
15% — LIFT: 0.46

BUNDLE IDEA

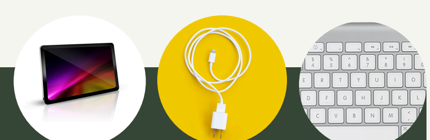
A new bundle with a tablet, keyboard case, and usb-c charger could be effective.

12% — LIFT: 0.37

Association Rules (Cont'd)



CROSS-SELLING OPPORTUNITY



RECOMMENDED ACTIONS

Consider implementation of a new bundle featuring a tablet, usb-c charger and keyboard case

Adjust marketing strategy to improve cross-selling

- Rebuy could be optimized; some suggestions are ***not*** logical.
- Improve email marketing 'You May Also Like' section logic
- Creative cross selling strategy brainstorming

Data-Driven Decision Making

REBUY ENGINE OPTIMIZATIONS

Rebuy Engine is a third-party SaaS platform designed to enhance upselling and cross-selling. A standout feature is the cart upsell, often labeled "You May Also Like" or similar. This mirrors the strategy used in brick-and-mortar stores where complementary items, like tomatoes and basil, are placed nearby to boost cart value.

The software incorporates machine learning technology to apply pattern mining techniques on historical sales data for better recommendations. However, the AI-generated recommendations often lack logic. Some recommended products are incompatible, infrequently purchased together, or *not* the most profitable options.

Below are some examples of poor recommendations from the Rebuy Engine AI:

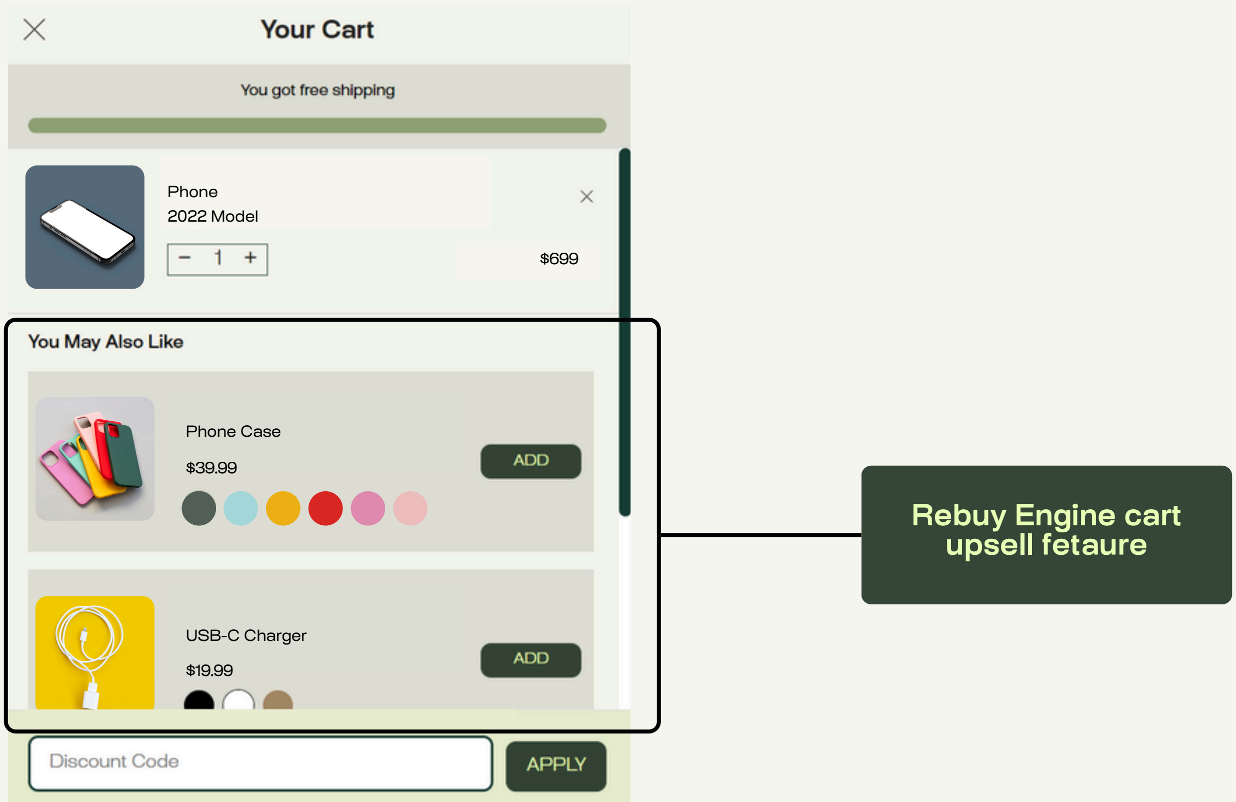
- **Problem:** Product A is often recommended despite having zero profitability and only breaking even.
- **Solution:** Exclude the Product A from product recommendations.

- **Problem:** Product B, an accessory not compatible with the Product C, is frequently recommended with it.
- **Solution:** Manually adjust product recommendations to ensure compatibility.

- **Problem:** Product D & Product E are frequently recommended and purchased but have a minimal impact on total revenue.
- **Solution:** Exclude these items from product recommendations to focus on more profitable products.

To optimize product recommendations, we will no longer rely on the AI. Instead, we will create manual logic based on the results of market basket analysis, product revenue, profitability data (not included for confidentiality purposes), and human knowledge of product offerings.

Data-Driven Decision Making (Cont'd)



EMAIL MARKETING OPTIMIZATIONS

We are extending the same methodology to our "You Might Also Like" sections in email marketing communications. By leveraging market basket analysis, product revenue, profitability data, and our deep understanding of our product offerings, we ensure that each recommendation is relevant, compatible, and profitable.

This approach replaces poor AI-generated suggestions with carefully curated selections, enhancing the value and appeal of our email marketing efforts.

Frequently Asked Questions

WHAT IS THE APRIORI ALGORITHM?

The Apriori algorithm is a data mining technique used to identify frequent itemsets and generate association rules in large datasets. It starts by finding individual items that meet a minimum support threshold, then combines them to form larger itemsets. The algorithm iteratively counts occurrences of these itemsets, retaining only those that are frequent. This process continues until no more frequent itemsets are found. From these, it generates association rules with a specified confidence level. The Apriori algorithm is especially useful in market basket analysis to discover products often bought together, aiding in optimizing product placement and cross-selling strategies.

ARE THESE METRICS RELIABLE?

Yes, these metrics are reliable. Data analysis is only as good as the quality of the input data. Poorly cleaned data can lead to inaccurate data-driven decision-making. For this analysis, thorough data quality measures were implemented throughout the cleaning process to ensure high data integrity.

WHY NOT USE AI RECOMMENDATIONS?

It's important to note that while artificial intelligence is powerful, it's not always the best solution. In this example, the Rebuy Engine only considers limited historical data and lacks comprehensive product knowledge. This can lead to recommendations that are not logical or beneficial. By combining human expertise with AI-assisted data analysis, we can ensure more relevant and profitable product suggestions.

WHAT TOOLS WERE USED FOR THIS ANALYSIS

This analysis was conducted using Python, leveraging libraries such as Pandas for data manipulation, Mlxtend for frequent itemset and association rule mining, and Matplotlib/Seaborn for data visualization. Data was stored and processed using Google Drive and other relevant tools.

Python Code

```
# Load necessary libraries
import pandas as pd
import os
from google.colab import drive
drive.mount('/content/drive')
from mlxtend.frequent_patterns import apriori,
association_rules
from mlxtend.preprocessing import TransactionEncoder
import warnings
warnings.filterwarnings("ignore",
category=DeprecationWarning)
pd.options.display.max_rows = 29

# Select Google Drive
os.chdir('/content/drive/MyDrive/')

# Import order data
import1 = pd.read_csv('orders_export_1.csv')
import2 = pd.read_csv('orders_export_2.csv')

# Import dictionary to standardize product names
productList = pd.read_csv('productList.csv')

# Set dictionary
dictionary = pd.Series(productList.new.values,
index=productList.old).to_dict()

# Concatenate Import1 & Import 2 in Dataframe
df = pd.concat([import1, import2], ignore_index=True)

# Replace Lineitem name with dictionary - standardized names
df['Lineitem name'] = df['Lineitem name'].replace(dictionary)

# Create subset with needed variables & preview dataframe
# NOTE: 'NAME' is order id but I did not replace the column
name
df_sub = df[['Lineitem name', 'Name', 'Email']]
df_sub.head()

# Convert Dataframe to pivot table for Apriori Algorithm &
create subset without 'Name' (order id)
df_pivot = df_sub.pivot_table(index='Name',
columns=df_sub.groupby('Name').cumcount(), values='Lineitem
name', aggfunc='first').reset_index()
df_pivot_sub = df_pivot.drop(columns='Name')
```

Python Code

```
# Preview pivot table
df_pivot_sub

# Create empty list
store_data_list = []


# Populate empty list with transactions
df_pivot_sub.apply(lambda x: store_data_list.append([i for i in
list(x) if pd.isnull(i) == False]), axis=1)

# Transform transaction list to one-hot encoded dataframe
tr = TransactionEncoder()
tr_arr = tr.fit(store_data_list).transform(store_data_list)
transactions = pd.DataFrame(tr_arr, columns=tr.columns_)
transactions.head()

# Import package and calculate frequent itemsets
frequent_itemsets = apriori(transactions, min_support=0.005,
use_colnames=True)
pd.set_option('max_colwidth', 600)

# Save as CSV to Google Drive
frequent_itemsets.to_csv('frequent_itemsets.csv')

# View frequent itemsets
frequent_itemsets.head(10)

# Import package and calculate association rules
association = association_rules(frequent_itemsets,
metric="confidence", min_threshold=0.005)

# Save as CSV to Google Drive
association.to_csv('association.csv')

# View association rules
association.head(15)
```