Part 1: Research Question

A1) Research Question

The question I will be asking in this analysis is what some of the most frequently co-purchased items in the telecom dataset are.

A2) Define Goal

Our goal will be to use Market Basket Analysis to determine what the top three most frequently co-purchased items are to better guide business decisions.

Part 2: Market Basket Justification

B1) Analysis Explanation and Expected Outcomes

Market basket analysis is based on analyzing the data to find patterns, wanting to know what items tend to be purchased together. It counts how often two or more items appear together, and calculates how likely it is for one item to be purchased with another. From these calculations, it makes “association rules” which help businesses make decisions.

My expected outcome is to find what the most frequently co-purchased items are.

B2) Example of Transaction

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

*import pandas as pd*

*import numpy as np*

*import matplotlib.pyplot as plt*

*import seaborn as sns*

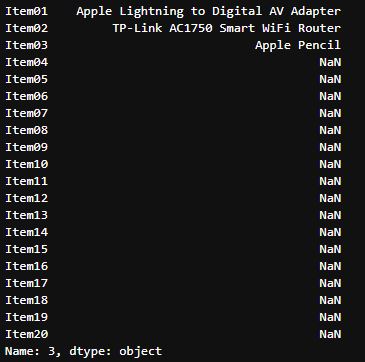
*from mlxtend.frequent\_patterns import association\_rules, apriori*

*from mlxtend.preprocessing import TransactionEncoder*

*df = pd.read\_csv('teleco\_market\_basket.csv')*

*# Example of a single transaction in the dataset*

*df.iloc[3]*



In the transaction example above, we can see that this transaction contains three items, an ‘apple lightning to digital av adapter’, a ‘tp-link ac1750 smart wifi router’, and an apple pencil.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

B3) Summary of Assumption

One assumption of market basket is that joint occurrence of two or more products imply that these products are complements in purchase. Meaning that the purchase of one will lead to the purchase of others. (1)

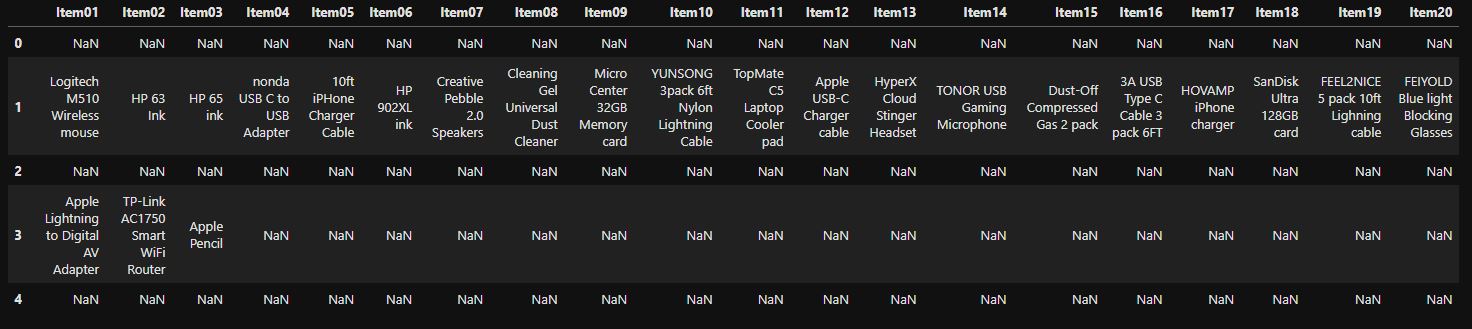
Part 3: Data Preparation and Analysis

C1) Transform Data Set

In preparation of the data, I ran the head of the data to gain some insight. Upon doing so, I discovered that every other row in the data set was empty. I dropped the empty rows, then used nested for loops to change the data to a list of lists. Then I ran the TransactionEncoder() and saved it as a new DataFrame. I checked for any empty columns, found one and removed it, then saved the now prepared data to a new csv file.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

*df.head()*

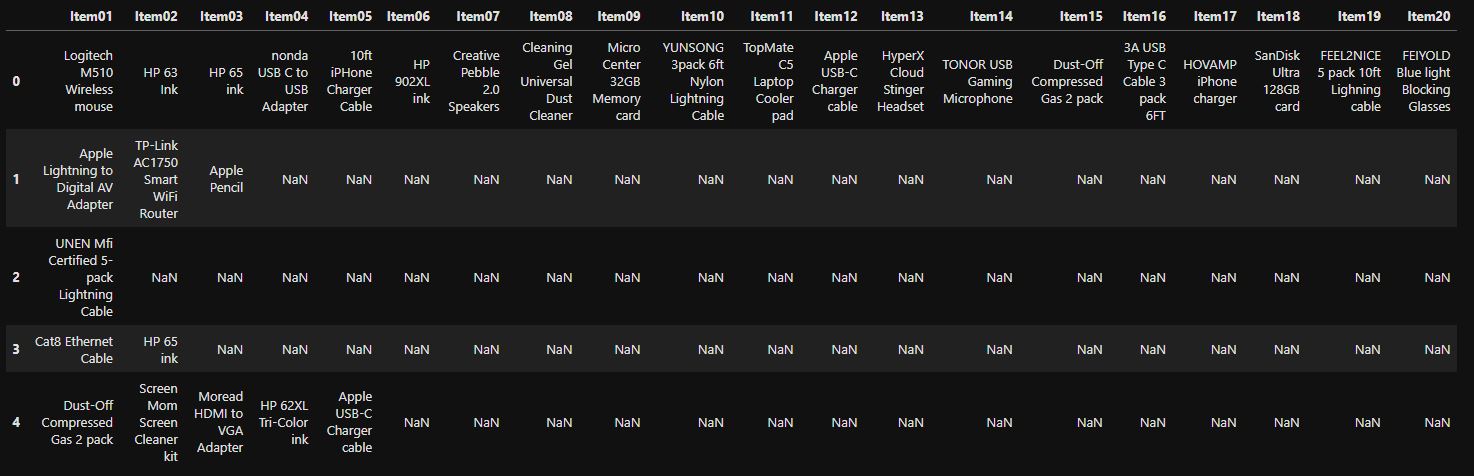


Seeing the empty rows, go ahead and remove them.

*df = df.dropna(how = 'all')*

*df.reset\_index(drop=True, inplace=True)*

*df.head()*



Turn to list of lists, run transaction encoder and save to new DataFrame.

*# Inspiration came from (2)*

*lists = []*

*for i in range(len(df)):*

*row = []*

*for j in range(len(df.columns)):*

*value = str(df.iat[i, j])*

*row.append(value)*

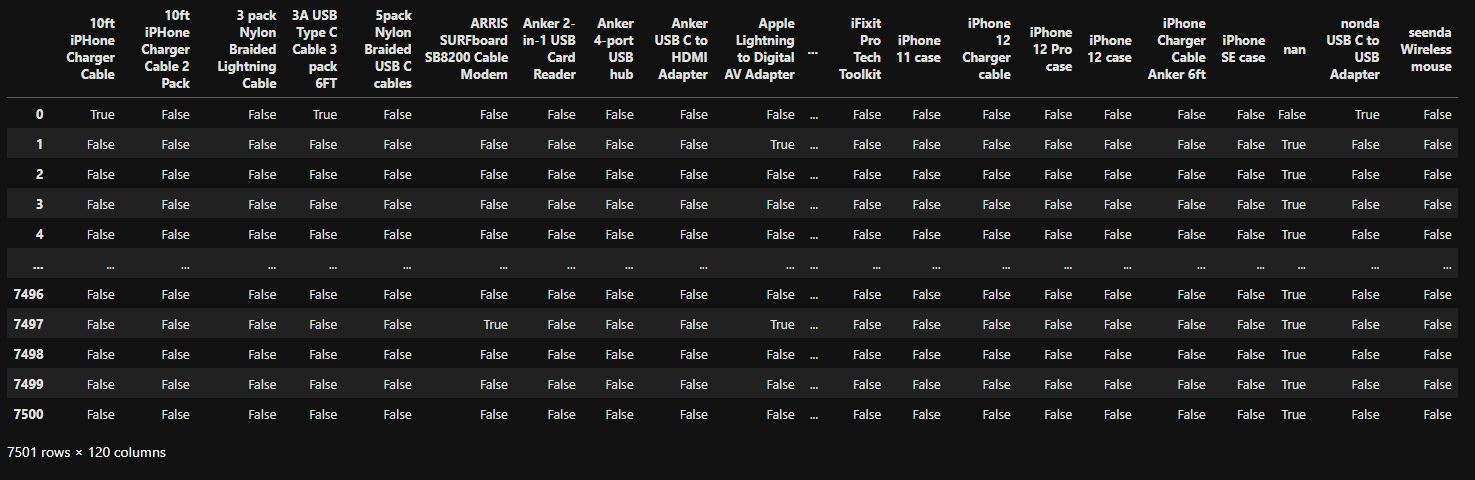
*lists.append(row)*

*encode = TransactionEncoder()*

*array = encode.fit(lists).transform(lists)*

*clean = pd.DataFrame(array, columns = encode.columns\_)*

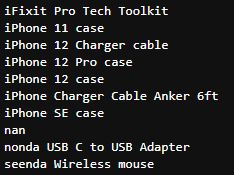
*clean*



Check for empty columns, the screenshot below is the bottom part where the empty column is.

*for col in clean.columns:*

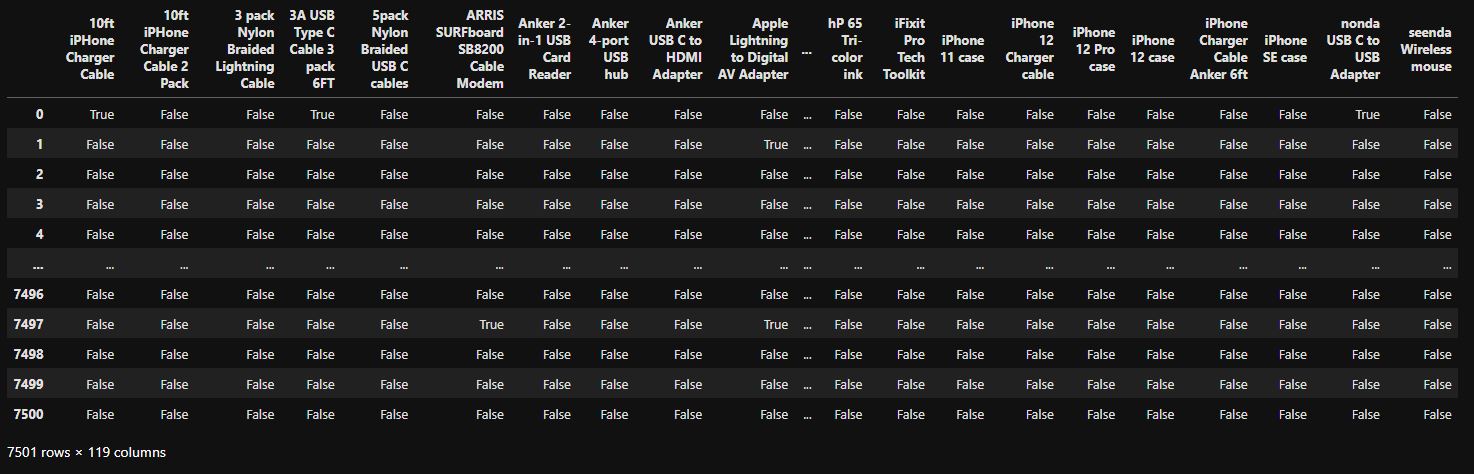
*print(col)*



Drop the empty columns, then check again. Now the data has 119 columns and is ready for analysis. Finally, save to new prepared csv file.

*clean = clean.drop(['nan'], axis = 1)*

*clean*



*clean.to\_csv('teleco\_prepared.csv', index = False)*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

C2) Generate Association Rules

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

*import pandas as pd*

*import numpy as np*

*import matplotlib.pyplot as plt*

*import seaborn as sns*

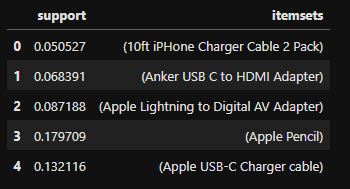
*from mlxtend.frequent\_patterns import association\_rules, apriori*

*from mlxtend.preprocessing import TransactionEncoder*

*df = pd.read\_csv('teleco\_prepared.csv')*

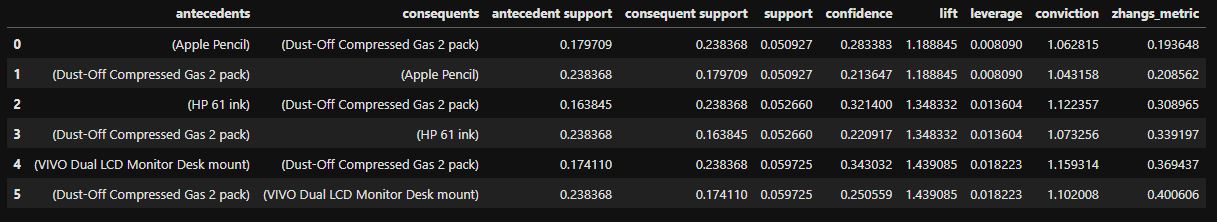
*a\_rules = apriori(df, min\_support = 0.05, use\_colnames = True)*

*a\_rules.head()*



*rules = association\_rules(a\_rules, metric = 'lift', min\_threshold = 1)*

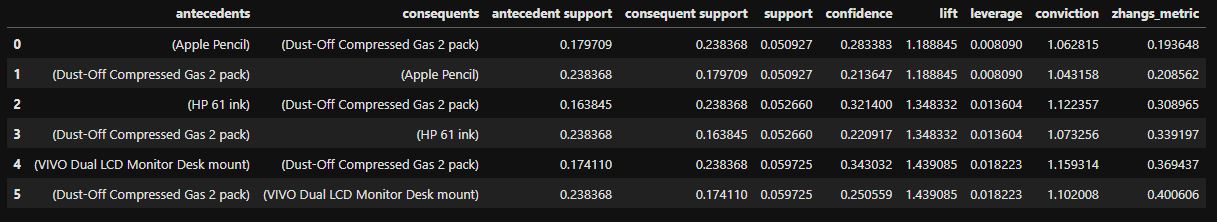
*rules*



We can see here, the code above generated the association rules table properly without errors.

C3) Values for Association Rules Table

The association rules table is above, in C2, but I will put it here again for the rubric.

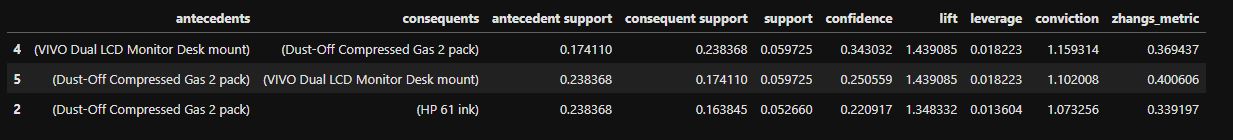


C4) Three Relevant Rules

To find the top three relevant rules, I decided to sort my rules table according to one of the criteria. I chose to sort by lift because I wanted to see the top three rules of items that are more likely to be sold together.

*lift = rules.sort\_values(by = 'lift', ascending = False)*

*lift.head(3)*



As we see in the above screenshot, the top three relevant rules are:

(VIVO Dual LCD Monitor Desk mount) -> (Dust-Off Compressed Gas 2 pack)

(Dust-Off Compressed Gas 2 pack) -> (VIVO Dual LCD Monitor Desk mount)

(Dust-Off Compressed Gas 2 pack) -> (HP 61 ink)

More discussion of what this means for this specific analysis will be shown more in D2.

Part 4: Data Summary and Implications

D1) Significance of Support, Lift, and Confidence

When you break it down, support is how common the groups of items are, confidence is how often one item purchase leads to another, and lift is how much more likely they are bought together. When doing this analysis, I started with a higher support value of 0.05, which cut down the total amount of data I got quite a bit. This does give some different insight when compared with using a lower starting support value.

The significance these three play in the results of this analysis is that we want to see the highest lift and confidence we can get when starting with a high support value. So when we have the combination of items with the highest lift while starting at the highest support, we can determine the likelihood of a customer purchasing the most common combination. We also can look at the highest confidence in addition, usually trying to get the highest number of all three.

D2) Practical Significance

When we look at the support, lift and confidence values of this specific analysis we can draw some conclusions from the data. Since I started with a higher support value of 0.05, I got much less data, but I was interested in the values for the most commonly purchased together items. With this support value, we can see that nearly 6% of purchases included both a dust-off compressed gas 2 pack and a dual monitor desk mount. Then about 5.2% of the purchases include the same dust-off compressed gas packs and HP 61 ink. So, when we look at the purchases of a higher support value, we see that it is more likely that when customers purchase more than one item, it has a higher chance to be the dust-off compressed gas 2 packs.

D3) Recommendation

The question and goal of this analysis was to find the more commonly purchased combination of items. Based on our findings, a good course of action for the company would be to have deals on the dust-off compressed gas 2 pack. In the top 3 relevant rules, we discovered that the dust-off compressed gas 2 pack was in all three of them, two of which have a support value near 0.06, meaning nearly 6% of purchases included a dust-off compressed gas 2 pack. My recommendation would be to lean into this with special sales that include them and making sure stores are always fully stocked with them.

E) Panopto Video

<https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=a5591c58-2878-4b3e-89dd-b070002be23b>

F) Sources

1) *Market Basket analysis*. (2015, September 1). HUA’S Analysis. https://sarahtianhua.wordpress.com/portfolio/market-basket-analysis/

2) Malli. (2023). Create list of lists in Python. *Spark by {Examples}*. https://sparkbyexamples.com/python/create-list-of-lists-in-python/