Second Home Assignment - Data Mining - Group 2

April 23, 2023

Ana Silva nº 39074, 15 hours; **Cláudia Afonso** nº 36273, 15 hours; **Martim Silva** nº 51304, 15 hours; **Rita Rodrigues** nº 54859 15 hours

---

[2]:
```
import pickle
import pandas as pd
import numpy as np
import time
import matplotlib.pyplot as plt
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, fpgrowth, fpmax, association_rules
from pyfim import pyeclat
from PD_freqitems import freqitemsets
```

0.1 Pre-Processing of the Data

[3]:
```
lines=open("products.txt", "rt", encoding="utf8").readlines()
products=[0]*len(lines)
for lin in lines[1:]:
    pid, pname, aid, did=lin.strip().split("\t")
    products[int(pid)]=pname
```

[4]:
```
orders=pickle.load(open("order_products.pickle", "rb"))
```

[5]:
```
orders_list = list(orders.values()) # transform dataset into a list of lists
tr_enc = TransactionEncoder() # transform dataset into a transaction array
orders_array = tr_enc.fit(orders_list).transform(orders_list, sparse=True)
orders_df = pd.DataFrame.sparse.from_spmatrix(orders_array) # transform into binary dataset dataframe
```

[6]:
```
M, N = orders_df.shape
print(f"Orders dataset has {M} orders (rows) and {N} products (columns).")
```

```
Orders dataset has 3214874 orders (rows) and 49677 products (columns).
```

0.2 Objective 1 - Analyze the itemset/rules generation procedure

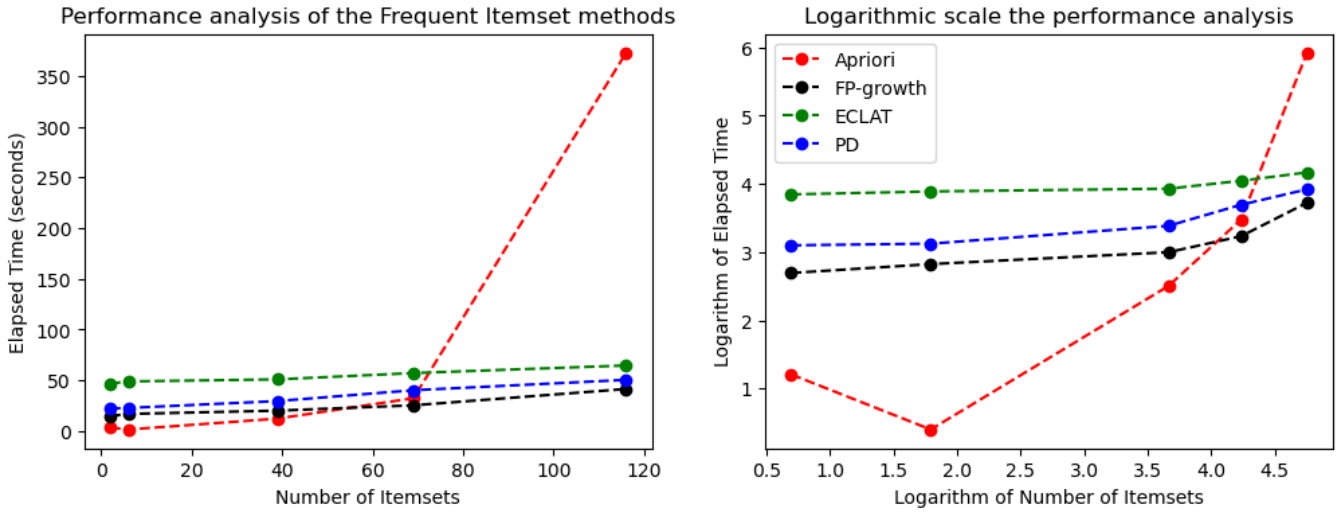
We are going to start the performance analysis of a few Frequent Itemset Mining Methods, namely Apriori, Frequent Pattern growth (FP-growth), Equivalence Class Transformation (ECLAT) and the recursive PD approach developed in the lab classes.

Performance analysis up to a threshold level of support

[7]:
```
D={"threshold": [0.10, 0.05, 0.02, 0.015, 0.010],
   "num_itemsets":[],
   "Apriori":[],
   "FP-growth":[],
   "ECLAT":[],
   "PD":[]}
for min_supp in D["threshold"]:
    t0 = time.time()
    FI_apriori=apriori(orders_df, min_supp)
    t1 = time.time()
    D["num_itemsets"].append(FI_apriori.shape[0])
```

```
    D["Apriori"].append(t1-t0)
    FI_fpg= fpgrowth(orders_df, min_supp)
    t2 = time.time()
    D["FP-growth"].append(t2-t1)
    FI_eclat= pyeclat(orders_list, min_supp)
    t3 = time.time()
    D["ECLAT"].append(t3-t2)
    FI_pdfis= freqitemsets(orders_list, min_supp)
    t4 = time.time()
    D["PD"].append(t4-t3)
df_performance=pd.DataFrame(D)
```

[8]:
```
fig, axes = plt.subplots(1, 2, figsize=(12, 4))
axes[0].plot(df_performance["num_itemsets"], df_performance["Apriori"], color="r", label="Apriori",␣
 ↪marker= 'o', linestyle = '--')
axes[0].plot(df_performance["num_itemsets"], df_performance["FP-growth"], color="k", label="FP-growth",␣
 ↪marker= 'o', linestyle = '--')
axes[0].plot(df_performance["num_itemsets"], df_performance["ECLAT"], color="g", label="ECLAT", marker=␣
 ↪'o', linestyle = '--')
axes[0].plot(df_performance["num_itemsets"], df_performance["PD"], color="b", label="PD", marker= 'o',␣
 ↪linestyle = '--')
axes[0].set_title('Performance analysis of the Frequent Itemset methods')
axes[0].set_xlabel('Number of Itemsets')
axes[0].set_ylabel('Elapsed Time (seconds)')
axes[1].plot(np.log(df_performance["num_itemsets"]), np.log(df_performance["Apriori"]), c="r",␣
 ↪label="Apriori", marker= 'o', linestyle = '--')
axes[1].plot(np.log(df_performance["num_itemsets"]), np.log(df_performance["FP-growth"]), c="k",␣
 ↪label="FP-growth", marker= 'o', linestyle = '--')
axes[1].plot(np.log(df_performance["num_itemsets"]), np.log(df_performance["ECLAT"]), c="g",␣
 ↪label="ECLAT", marker= 'o', linestyle = '--')
axes[1].plot(np.log(df_performance["num_itemsets"]), np.log(df_performance["PD"]), c="b", label="PD",␣
 ↪marker= 'o', linestyle = '--')
axes[1].set_xlabel('Logarithm of Number of Itemsets')
axes[1].set_ylabel('Logarithm of Elapsed Time')
axes[1].set_title('Logarithmic scale the performance analysis')
plt.legend()
plt.show()
```



Based on the performance analysis of the various Frequent Itemset Mining Methods, the chosen method is FP-growth. This was the method that displayed the lowest elapsed time at the lowest threshold level of 0.01. Thus, FP-growth was the faster and more memory-efficient method for this dataset. Based on this method, the support threshold to compute frequent itemsets will be defined below. It is not surprising that the Apriori method performed the worst in this analysis. Indeed, this method can be

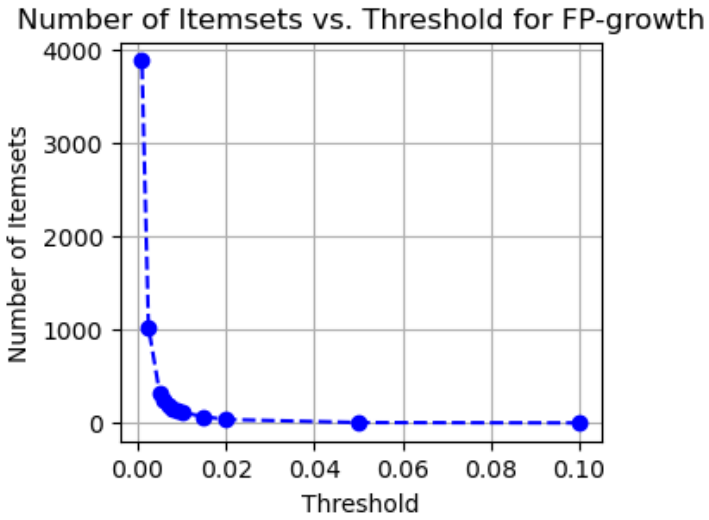
computationally quite costly, particularly for large datasets such as the one considered for this project.

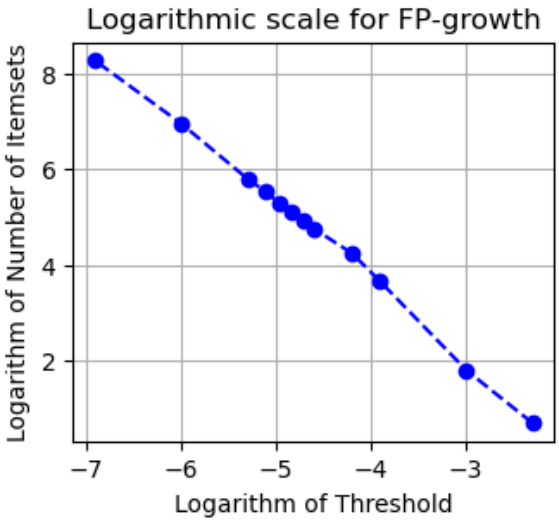
Defining a good support threshold for analysis

Since the best performing method was FP-growth, we decided to lower the support threshold further using only this method due to computational and time constraints imposed by the other methods (particularly Apriori). This was done to evaluate the number of itemsets for each corresponding level of support threshold and thus select a good support threshold for analysis according to our computational capabilities.

[9]:
```
F={"threshold": [0.10, 0.05, 0.02, 0.015, 0.010, 0.009, 0.008, 0.007, 0.006, 0.005, 0.0025, 0.001],
   "num_itemsets":[],
   "FP-growth":[]}
for min_supp in F["threshold"]:
    t0 = time.time()
    FI_fpg= fpgrowth(orders_df, min_supp)
    t1 = time.time()
    F["num_itemsets"].append(FI_fpg.shape[0])
    F["FP-growth"].append(t1-t0)
df_fpgrowth=pd.DataFrame(F)
```

[10]:
```
fig, axes = plt.subplots(1, 2, figsize=(9, 3))
axes[0].plot(df_fpgrowth['threshold'], df_fpgrowth['num_itemsets'], '--bo')
axes[0].set_xlabel('Threshold')
axes[0].set_ylabel('Number of Itemsets')
axes[0].grid(True)
axes[0].set_title('Number of Itemsets vs. Threshold for FP-growth')
axes[1].plot(np.log(df_fpgrowth['threshold']), np.log(df_fpgrowth['num_itemsets']), '--bo')
axes[1].set_xlabel('Logarithm of Threshold')
axes[1].set_ylabel('Logarithm of Number of Itemsets')
axes[1].grid(True)
axes[1].set_title('Logarithmic scale for FP-growth')
fig.subplots_adjust(wspace=0.5)
plt.show()
```





As observed from the graphical representations above, as the support threshold is decreased, the number of frequent itemsets increases. As a good support threshold, we decided to choose the value of 0.001 since it generates a reasonable amount of itemsets and is computationally feasible in our own personal computers.

0.3 Objective 2 - Identify the most relevant rules

In this second stage of the project, the goal is to identify buying patterns that reflect items that are frequently associated or purchased together. These patterns can be represented in the form of association rules.

Generating all available itemsets and rules from a predefined support level and identifying a set of relevant rules

[11]:
```python
# Define a function to convert tuple of indexes to tuple of product names
def indexes_to_names(index_tuple, product_list):
    return tuple(product_list[int(i)] for i in index_tuple)
```

[12]:
```python
frequent_itemsets = fpgrowth(orders_df, min_support=0.001, use_colnames=True)
```

[13]:
```python
frequent_itemsets['itemsets'] = frequent_itemsets['itemsets'].apply(lambda x: indexes_to_names(x,␣
↪products))
frequent_itemsets['length'] = frequent_itemsets['itemsets'].apply(lambda x: len(x))
print('The number of generated frequent itemsets was:', frequent_itemsets.shape[0])
```

```
The number of generated frequent itemsets was: 3883
```

[14]:
```python
all_rules = association_rules(frequent_itemsets, metric='confidence', min_threshold=0.25)
```

In general, to generate strong association rules from the frequent itemsets, these rules must satisfy minimum support and minimum confidence values. However, strong association rules are not necessarily interesting or relevant. Thus, additional interestingness measures such as lift, leverage, conviction and zhang's metric can be used to augment the support-confidence framework for association rules, thereby discovering correlation relationships between associated items.

To generate relevant rules, the support was set to 0.0020, the confidence was set to above 0.25, the lift was set to above 25, the leverage was set to above 0.0020, the conviction was set to above 1.0 and zhang's metric was set to above 0.9. With these values, the rules below were generated.

[15]:
```python
rules=all_rules[(all_rules["support"] >= 0.0020) & (all_rules["confidence"] > 0.25) &
                (all_rules["lift"] > 25) & (all_rules["leverage"] > 0.0020) & (all_rules["conviction"] >␣
↪1.0) &
                (all_rules["zhangs_metric"] > 0.90)]
rules
```

[15]:
```
                                         antecedents  \
296             (Mayonnaise Dressing with Olive Oil)
297                           (Golden Grahams Cereal)
329            (Multi Kids Complete + Fiber Gummies)
330  ("Organic Superfoods Morning Banana, Blueberry...
341  ("Organic Superfoods Morning Banana, Blueberry...
342                                     (Pasta Sauce)
368                              (Savory Cracker Mix)
369                           (Golden Grahams Cereal)

                                         consequents  antecedent support  \
296                           (Golden Grahams Cereal)            0.005643
297             (Mayonnaise Dressing with Olive Oil)            0.005990
329  ("Organic Superfoods Morning Banana, Blueberry...            0.006457
330            (Multi Kids Complete + Fiber Gummies)            0.009298
341                                     (Pasta Sauce)            0.009298
342  ("Organic Superfoods Morning Banana, Blueberry...            0.006192
368                           (Golden Grahams Cereal)            0.005094
369                              (Savory Cracker Mix)            0.005990

     consequent support   support  confidence       lift  leverage  \
296            0.005990  0.002069    0.366718  61.222024  0.002036
297            0.005643  0.002069    0.345485  61.222024  0.002036
329            0.009298  0.002902    0.449513  48.343394  0.002842
330            0.006457  0.002902    0.312147  48.343394  0.002842
341            0.006192  0.002493    0.268090  43.292848  0.002435
342            0.009298  0.002493    0.402552  43.292848  0.002435
368            0.005990  0.002247    0.441106  73.640837  0.002217
369            0.005094  0.002247    0.375136  73.640837  0.002217

     conviction  zhangs_metric
296    1.569617       0.989248
297    1.519226       0.989594
```

```
329    1.799684       0.985679
330    1.444411       0.988506
341    1.357827       0.986070
342    1.658222       0.982989
368    1.778532       0.991471
369    1.592197       0.992365
```

As observed from the results above, 8 relevant rules were generated. Surprisingly, some of these rules appear to be connected since they have the same itemsets, but in reverse order. For instance, rules 296 and 297 have the same itemsets ("Mayonnaise Dressing with Olive Oil" and "Golden Grahams Cereal") but one is the antecedent for rule 296 and is the consequent for rule 297, while the other is the antecedent for rule 297 but is the consequent for rule 296. This same pattern of reverse antecedent-consequent itemsets appears in the following consecutive pairs of rules, suggesting that the itemsets belonging to each pair of connected rules are correlated with each other.

Here, we will analyse the first two rules and then generalize to give an overall picture. For the first two rules (296 and 297), the support is 0.21% which means that 0.21% of the orders contain both "Mayonnaise Dressing with Olive Oil" and "Golden Grahams Cereal". The confidence levels are similar for both rules (36.67% for rule 296 and 34.5% for rule 297), which means that 36.7% of orders than contain "Mayonnaise Dressing with Olive Oil" also contain "Golden Grahams Cereal", while 34.5% of orders that contain "Golden Grahams Cereal" also contain "Mayonnaise Dressing with Olive Oil". The lift is 61.22 which indicates a strong positive association between "Mayonnaise Dressing with Olive Oil" and "Golden Grahams Cereal". The leverage value is 0.002 which is means that the joint occurrence of the antecedent and consequent is 0.2% higher than what would be expected if they were independent. The conviction for both rules is approximately 1.5 which means that "Mayonnaise Dressing with Olive Oil" and "Golden Grahams Cereal" are positively correlated. The value for the Zhang's metric is approximately 0.99 which suggests that this rule is highly interesting and that a strong relationship between the mentioned itemsets exists, to the point where the presence of one strongly predicts the presence of the other in the shopping cart.

Overall, the support values vary between 0.002069 and 0.002902 which means that 0.21% to 0.29% of orders contain the itemsets corresponding to the rules above. These values are not very high, but this is to be expected since our dataset is quite large, containing a huge amount of orders as well as products. The confidence values vary between 0.268090 and 0.449513 which means that 26.8% to 45% of orders containing the antecedent of the corresponding rule contain its respective consequent. Again, these values are a bit on the lower side, but are not unreasonably low. On the contrary, the lift values are quite high, oscillating between 43.3 and 73.6, which suggests a strong correlation between the respective antecedent and consequent of the corresponding rules. The leverage values were quite low, varying between 0.20% and 0.28%. However, the leverage metric is calculated on the basis of the support values for the antecedent and consequent, as thus these low values are to be expected. The conviction values are higher than 1, oscillating between 1.36 and 1.80, which indicates some degree of dependence between the respective antecedents and consequents. The Zhang's metric was quite high, varying between 0.98 and 0.99, which indicates a strong association between the corresponding antecedents and consequents.

To conclude, the generated rules are quite relevant. The presence of antecedent predicts the presence of the consequent in the shopping cart.

Identifying the Closed and Maximal Itemsets for the same level of support

To mine the set of closed itemsets, the pyfim implementation of ECLAT was used while to mine the set of maximal itemsets, the FP-Max implementation at MLxtend was employed, both with the same level of support considered previously.

[16]:
```
orders_closed = pyeclat(orders_list, 0.001, target="c")
```

[17]:
```
orders_closed['itemsets'] = orders_closed['itemsets'].apply(lambda x: indexes_to_names(x, products))
orders_closed['length'] = orders_closed['itemsets'].apply(lambda x: len(x))
print('The number of generated closed itemsets was:', orders_closed.shape[0])
```

```
The number of generated closed itemsets was: 3883
```

[18]:
```
orders_maximal=fpmax(orders_df, min_support=0.001, use_colnames=True)
```

[19]:
```
orders_maximal["itemsets"] = orders_maximal["itemsets"].apply(lambda x: indexes_to_names(x, products))
orders_maximal['length'] = orders_maximal['itemsets'].apply(lambda x: len(x))
print('The number of generated maximal itemsets was:', orders_maximal.shape[0])
```

```
The number of generated maximal itemsets was: 3358
```

Generating the most relevant rules for closed and maximal itemsets with adequate statistics

[20]:
```
all_closed_rules = association_rules(orders_closed, metric="confidence", min_threshold=0.25)
```

To generate relevant rules for closed itemsets, the support was set to 0.0020, the confidence was set to above 0.25, the lift was set to above 10, the leverage was set to above 0.0020, the conviction was set to above 1.0 and zhang's metric was set to above 0.9. With these values, the rules below were generated.

[21]:
```
closed_rules=all_closed_rules[(all_closed_rules["support"] >= 0.0020) & (all_closed_rules["confidence"]␣
↪> 0.25) &
                              (all_closed_rules["lift"] > 10) & (all_closed_rules["leverage"] > 0.0020) &
                              (all_closed_rules["conviction"] > 1.0) &␣
↪(all_closed_rules["zhangs_metric"] > 0.90)]
closed_rules
```

[21]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	zhangs_metric
53	(Non Fat Raspberry Yogurt)	(Icelandic Style Skyr Blueberry Non-fat Yogurt)	0.005094	0.005990	0.002247	0.441106	73.640837	0.002217	1.778532	0.991471
54	(Icelandic Style Skyr Blueberry Non-fat Yogurt)	(Non Fat Raspberry Yogurt)	0.005990	0.005094	0.002247	0.375136	73.640837	0.002217	1.592197	0.992365
68	(Icelandic Style Skyr Blueberry Non-fat Yogurt)	(Vanilla Skyr Nonfat Yogurt)	0.005990	0.005643	0.002069	0.345485	61.222024	0.002036	1.519226	0.989594
69	(Vanilla Skyr Nonfat Yogurt)	(Icelandic Style Skyr Blueberry Non-fat Yogurt)	0.005643	0.005990	0.002069	0.366718	61.222024	0.002036	1.569617	0.989248
85	(Total 2% with Strawberry Lowfat Greek Straine...	(Total 2% Lowfat Greek Strained Yogurt with Pe...	0.006192	0.009298	0.002493	0.268090	43.292848	0.002435	1.357827	0.986070
86	(Total 2% Lowfat Greek Strained Yogurt with Pe...	(Total 2% with Strawberry Lowfat Greek Straine...	0.009298	0.006192	0.002493	0.402552	43.292848	0.002435	1.658222	0.982989
91	(Total 2% with Strawberry Lowfat Greek Straine...	(Total 2% Lowfat Greek Strained Yogurt With Bl...	0.006457	0.009298	0.002902	0.312147	48.343394	0.002842	1.444411	0.988506
92	(Total 2% Lowfat Greek Strained Yogurt With Bl...	(Total 2% with Strawberry Lowfat Greek Straine...	0.009298	0.006457	0.002902	0.449513	48.343394	0.002842	1.799684	0.985679
104	(Yellow Bell Pepper)	(Orange Bell Pepper)	0.007919	0.012190	0.002216	0.279822	22.955853	0.002119		
121	(Sparkling Lemon Water)	(Lime Sparkling Water)	0.010199	0.014478	0.002658	0.260644	18.002365	0.002511		
122	(Sparkling Lemon Water)	(Sparkling Water Grapefruit)	0.010199	0.023605	0.002869	0.281322	11.918090	0.002628		
145	(Lime Sparkling Water)	(Sparkling Water Grapefruit)	0.014478	0.023605	0.004132	0.285395	12.090626	0.003790		

```
104    1.371621       0.964073
121    1.332946       0.954183
122    1.358600       0.925533
145    1.366343       0.930767
```

As observed from the results above, 12 relevant rules were generated. Interestingly, the antecedents and consequents of the generated relevant rules belong to the same category and thereby share a high degree of similarity between them. For instance, the antecedents and consequents of rules 53, 54, 68, 69, 85, 86, 91 and 92 are all different types of yogurt, while those of rule 104 are types of bell peppers and those of rules 121, 122 and 145 are all types of water.

Overall, the support values vary between 0.002216 and 0.004132 which means that 0.2% to 0.4% of orders contain the itemsets corresponding to the rules above. Again, these values are not very high, but this is to be expected since our dataset is quite large, containing a huge amount of orders as well as products. The confidence values vary between 0.260644 and 0.449513 which means that 26.1% to 45.0% of orders containing the antecedent of the corresponding rule contain its respective consequent. Again, these values are a bit on the lower side, but are not unreasonably low. On the contrary, the lift values are quite high, oscillating between 11.92 and 73.64, which suggests a strong correlation between the respective antecedent and consequent of the corresponding rules. The leverage values were quite low, varying between 0.21% and 0.38%. However, as has been said previously, the leverage metric is calculated on the basis of the support values for the antecedent and consequent, as thus these low values are to be expected. The conviction values are higher than 1, oscillating between 1.33 and 1.80, which indicates some degree of dependence between the respective antecedents and consequents. The Zhang's metric was quite high, varying between 0.93 and 0.99, which indicates a strong association between the corresponding antecedents and consequents.

To conclude, the generated rules are quite relevant. The presence of antecedent predicts the presence of the consequent in the shopping cart.

To generate rules for maximal itemsets, we first tried to pass the "orders_maximal" dataframe as an argument of the "association_rules" function. However, this gave an error due to "missing antecedent and/or consequent information" in this dataframe. Most metrics computed by "association_rules" depend on the consequent and antecedent support score of a given rule provided in the "frequent_itemsets" dataframe. Since the maximal itemsets are a subset of the frequent itemsets, to get around this issue we introduced the missing itemsets into the "orders_maximal" dataframe.

[22]:
```
maximal_set = set(map(frozenset, orders_maximal['itemsets']))
frequent_set = set(map(frozenset, frequent_itemsets['itemsets']))
missing_itemsets = frequent_set - maximal_set
missing_itemsets = frequent_itemsets[frequent_itemsets['itemsets'].apply(frozenset).
 ↪isin(missing_itemsets)]
orders_maximal = orders_maximal.append(missing_itemsets)
```

[23]:
```
all_maximal_rules = association_rules(orders_maximal, metric="confidence", min_threshold=0.25)
```

To generate relevant rules for maximal itemsets, the support was set to 0.0020, the confidence was set to above 0.25, the lift was set to above 25, the leverage was set to above 0.0020, the conviction was set to above 1.0 and zhang's metric was set to above 0.9. With these values, the rules below were generated.

[24]:
```
maximal_rules=all_maximal_rules[(all_maximal_rules["support"] >= 0.0020) &␣
 ↪(all_maximal_rules["confidence"] > 0.25) &
                                (all_maximal_rules["lift"] > 25) & (all_maximal_rules["leverage"] > 0.
 ↪0020) &
                                (all_maximal_rules["conviction"] > 1.0) &␣
 ↪(all_maximal_rules["zhangs_metric"] > 0.90)]
maximal_rules
```

[24]:
```
                                      antecedents  \
53                              (Savory Cracker Mix)
54                           (Golden Grahams Cereal)
68           (Mayonnaise Dressing with Olive Oil)
69                           (Golden Grahams Cereal)
367            (Multi Kids Complete + Fiber Gummies)
368  ("Organic Superfoods Morning Banana, Blueberry...
369  ("Organic Superfoods Morning Banana, Blueberry...
370                                  (Pasta Sauce)

                                      consequents  antecedent support  \
```

```
53                            (Golden Grahams Cereal)            0.005094
54                               (Savory Cracker Mix)            0.005990
68                            (Golden Grahams Cereal)            0.005643
69                  (Mayonnaise Dressing with Olive Oil)            0.005990
367  ("Organic Superfoods Morning Banana, Blueberry...            0.006457
368               (Multi Kids Complete + Fiber Gummies)            0.009298
369                                      (Pasta Sauce)            0.009298
370  ("Organic Superfoods Morning Banana, Blueberry...            0.006192

     consequent support   support  confidence       lift  leverage  \
53             0.005990  0.002247    0.441106  73.640837  0.002217
54             0.005094  0.002247    0.375136  73.640837  0.002217
68             0.005990  0.002069    0.366718  61.222024  0.002036
69             0.005643  0.002069    0.345485  61.222024  0.002036
367            0.009298  0.002902    0.449513  48.343394  0.002842
368            0.006457  0.002902    0.312147  48.343394  0.002842
369            0.006192  0.002493    0.268090  43.292848  0.002435
370            0.009298  0.002493    0.402552  43.292848  0.002435

     conviction  zhangs_metric
53     1.778532       0.991471
54     1.592197       0.992365
68     1.569617       0.989248
69     1.519226       0.989594
367    1.799684       0.985679
368    1.444411       0.988506
369    1.357827       0.986070
370    1.658222       0.982989
```

As observed from the results above, 8 relevant rules were generated. Interestingly, the antecedents and consequents of the generated relevant rules for maximal itemsets are quite similar to those obtained previously for the mining of frequent itemsets. In particular, rules 53 and 54 obtained for maximal itemsets correspond to rules 368 and 369 obtained for the frequent itemsets; rules 68 and 69 obtained for maximal itemsets correspond to rules 296 and 297 obtained for the frequent itemsets; rules 367 and 368 obtained for maximal itemsets correspond to rules 329 and 330 obtained for the frequent itemsets; rules 369 and 370 obtained for maximal itemsets correspond to rules 341 and 342 obtained for the frequent itemsets.

0.4 Conclusions

In this project, we explored the mining of frequent itemsets, closed itemsets, and maximal itemsets from the Instacart Market Basket Analysis dataset. We used the FP-Growth and ECLAT methods to mine these itemsets from the dataset.

We initiated our analysis by evaluating the performance of several frequent itemset mining methods, namely Apriori, FP-growth, ECLAT and the recursive PD approach developed in the lab classes. In this context, we observed that the most suitable method for our dataset is the FP-growth, since it generated a good amount of itemsets (3883) in a reasonable time-frame and is thus computationally feasible to run in our own personal computers.

Using the FP-growth method, the frequent itemsets were subsequently generated for a support threshold of 0.001 and a set of relevant rules was identified with adequate statistics (support, confidence, lift, leverage, conviction and Zhang's metric). We then mined closed itemsets using ECLAT and maximal itemsets using FP-Max, which provided additional interesting rules.

In conclusion, mining frequent itemsets, closed itemsets, and maximal itemsets from large datasets like the Instacart Market Basket Analysis can provide valuable insights into customer behavior and purchasing patterns. These insights can be used by businesses to optimize their marketing strategies, improve customer satisfaction, and increase sales.

0.5 Bibliography

[1] Han J, Kamber M, Pei J. Data mining concepts and techniques third edition. University of Illinois at Urbana-Champaign Micheline Kamber Jian Pei Simon Fraser University. 2012.

[2] Azevedo PJ, Jorge AM. Comparing rule measures for predictive association rules. In Machine Learning: ECML 2007: 18th European Conference on Machine Learning, Warsaw, Poland, September 17-21, 2007. Proceedings 18 2007 (pp. 510-517). Springer Berlin Heidelberg.