1. Data Exploration for customer segmentation and association on typical sales order transaction

The widely practice of performing customer segmentation requires consideration of the business and marketing alignment, hence it cannot be strictly applied using silhouette score or elbow point in K-means clustering technique as prescribed by data science theory in academics.

Organization of the past decade during 20th century would typically use most advance relational databases such as Oracle or PostgreSQL to generate analytical reports for descisioin-making. In recent times, software companies such as SAS, Microsoft Azure or SAP BI & Predictive Analytics are emerging with new product features to marry business intelligence with predictive analytics as a solution that leverage on the existing enterprise relational databases.

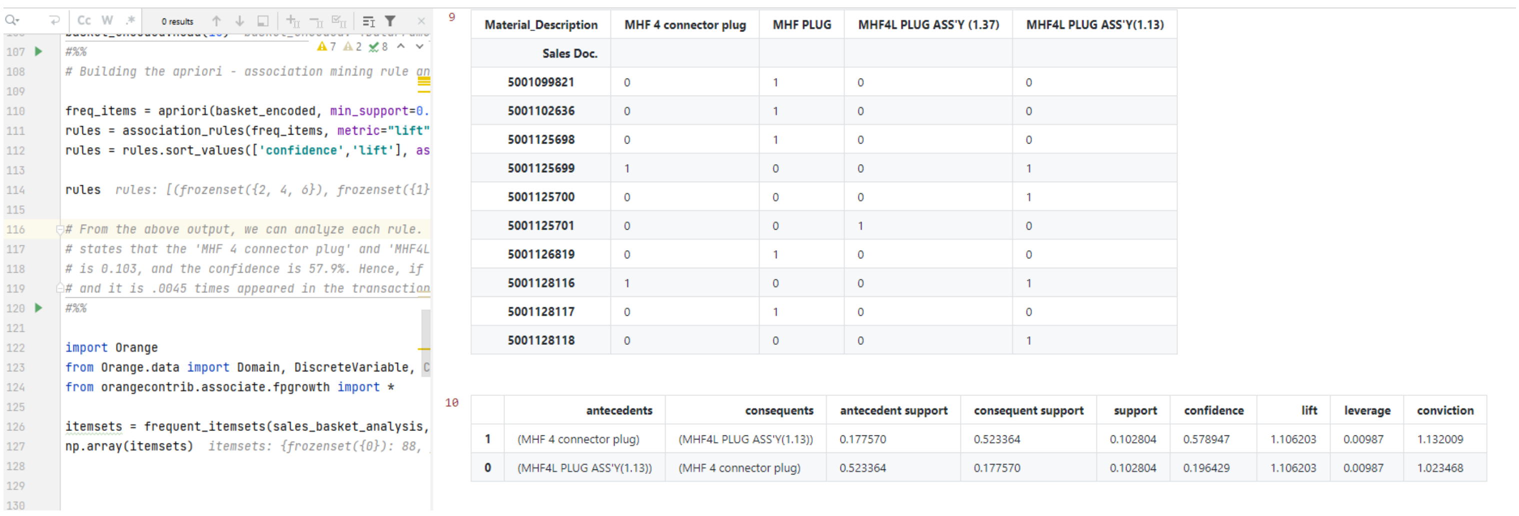
Using sales transaction record whereby sales order comprises of multiple products downloaded from SAP, I have attempted to apply association mining techniques which can be used for identifying the potential for cross-selling. However, this concept of market basket analysis cannnot be applied successfully for those datasets that I have come across with sales orders containing only single sku(s) or material number.

To gain new insights on sales order transaction downloaded from SAP, I applied using python scripts using python package - *mlxtend.frequent_patterns* & *apriori, association_rules* libraries to perform market basket analysis.

From the output, i observed one of the few rules is that the first rule, 'MHF 4 connector plug' → 'MHF4L PLUG ASS Y(1.13)', # states that the 'MHF 4 connector plug' and 'MHF4L PLUG ASS Y(1.13)' are bought frequently by most of the customers. The support for this rule# is 0.103, and the confidence is 57.9%. Hence, if a customer buys 'MHF 4 connector plug', it is 29% chances that he also buys MHF4L PLUG ASS Y(1.13), We can check all these things in other rules also.

Material_Description	MHF 4 connector plug	MHF PLUG	MHF4L PLUG ASS'Y (1.37)	MHF4L PLUG ASS'Y(1.13)
Sales Doc.				
5001099821	0	1	0	0
5001102636	0	1	0	0
5001125698	0	1	0	0
5001125699	1	0	0	1
5001125700	0	0	0	1
5001125701	0	0	1	0
5001126819	0	1	0	0
5001128116	1	0	0	1
5001128117	0	1	0	0
5001128118	0	0	0	1

Fig1: One-hot encoding using typical sales order data extracted from ERP – SAP before processing using association-mining technique



```
#%%
# Building the apriori - association mining rule an

freq_items = apriori(basket_encoded, min_support=0.
rules = association_rules(freq_items, metric="lift"
rules = rules.sort_values(['confidence','lift'], as

rules  rules: [(frozenset({2, 4, 6}), frozenset({1}

# From the above output, we can analyze each rule.
# states that the 'MHF 4 connector plug' and 'MHF4L
# is 0.103, and the confidence is 57.9%. Hence, if
# and it is .0045 times appeared in the transaction
#%%

import Orange
from Orange.data import Domain, DiscreteVariable, C
from orangecontrib.associate.fpgrowth import *

itemsets = frequent_itemsets(sales_basket_analysis,
np.array(itemsets)  itemsets: {frozenset({0}): 88,
```

9

Material_Description	MHF 4 connector plug	MHF PLUG	MHF4L PLUG ASS'Y (1.37)	MHF4L PLUG ASS'Y(1.13)
Sales Doc.				
5001099821	0	1	0	0
5001102636	0	1	0	0
5001125698	0	1	0	0
5001125699	1	0	0	1
5001125700	0	0	0	1
5001125701	0	0	1	0
5001126819	0	1	0	0
5001128116	1	0	0	1
5001128117	0	1	0	0
5001128118	0	0	0	1

10

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
1	(MHF 4 connector plug)	(MHF4L PLUG ASS'Y(1.13))	0.177570	0.523364	0.102804	0.578947	1.106203	0.00987	1.132009
0	(MHF4L PLUG ASS'Y(1.13))	(MHF 4 connector plug)	0.523364	0.177570	0.102804	0.196429	1.106203	0.00987	1.023468

Fig2: Output from association mining technique – Apriori algorithm using Python-scripts

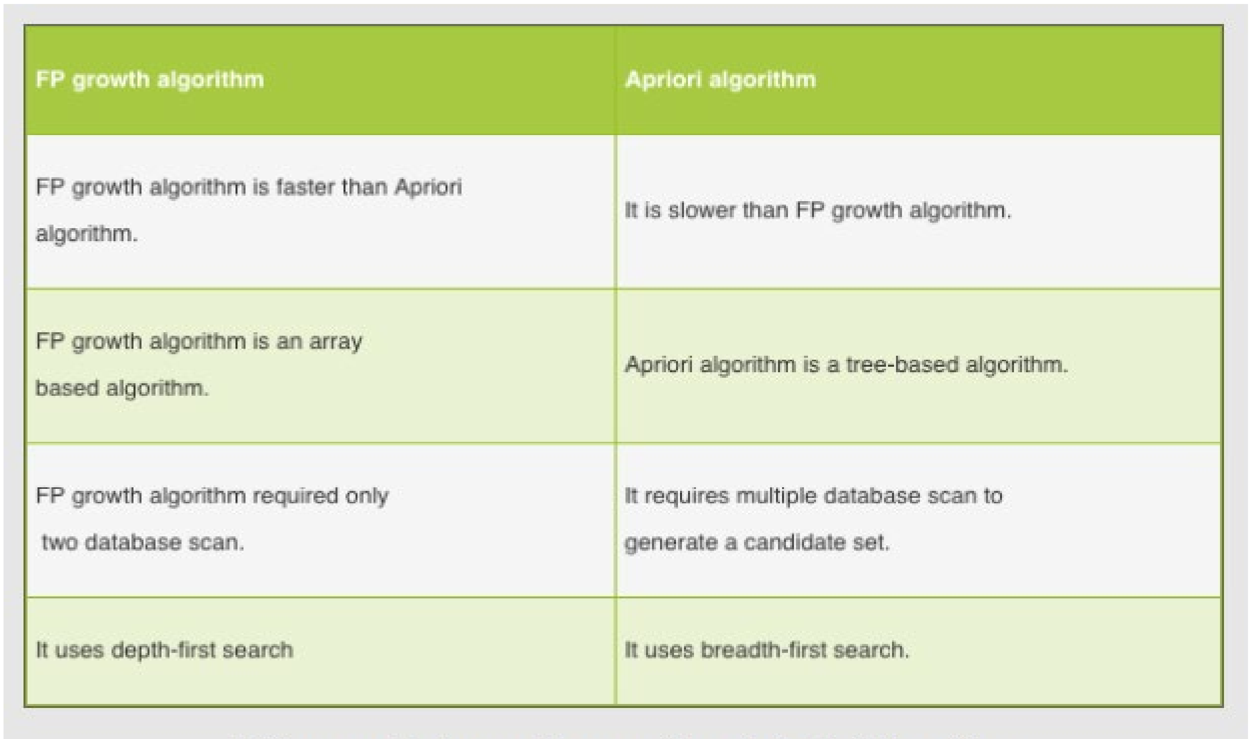
	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
1	(MHF 4 connector plug)	(MHF4L PLUG ASS'Y(1.13))	0.177570	0.523364	0.102804	0.578947	1.106203	0.00987	1.132009
0	(MHF4L PLUG ASS'Y(1.13))	(MHF 4 connector plug)	0.523364	0.177570	0.102804	0.196429	1.106203	0.00987	1.023468

Fig3: Output from association mining technique – Apriori algorithm using typical sales order data extracted from SAP

1.1 Bottleneck or Limitation of Apriori algorithm for association-mining

Apriori is an algorithm for frequent item set 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. This simple association-mining captures the essential idea of cross-selling, here it is not as descriptive as I would described & prescribed by synthesizing the busines problems and competitive threats of industry using quantitative tools in data science.

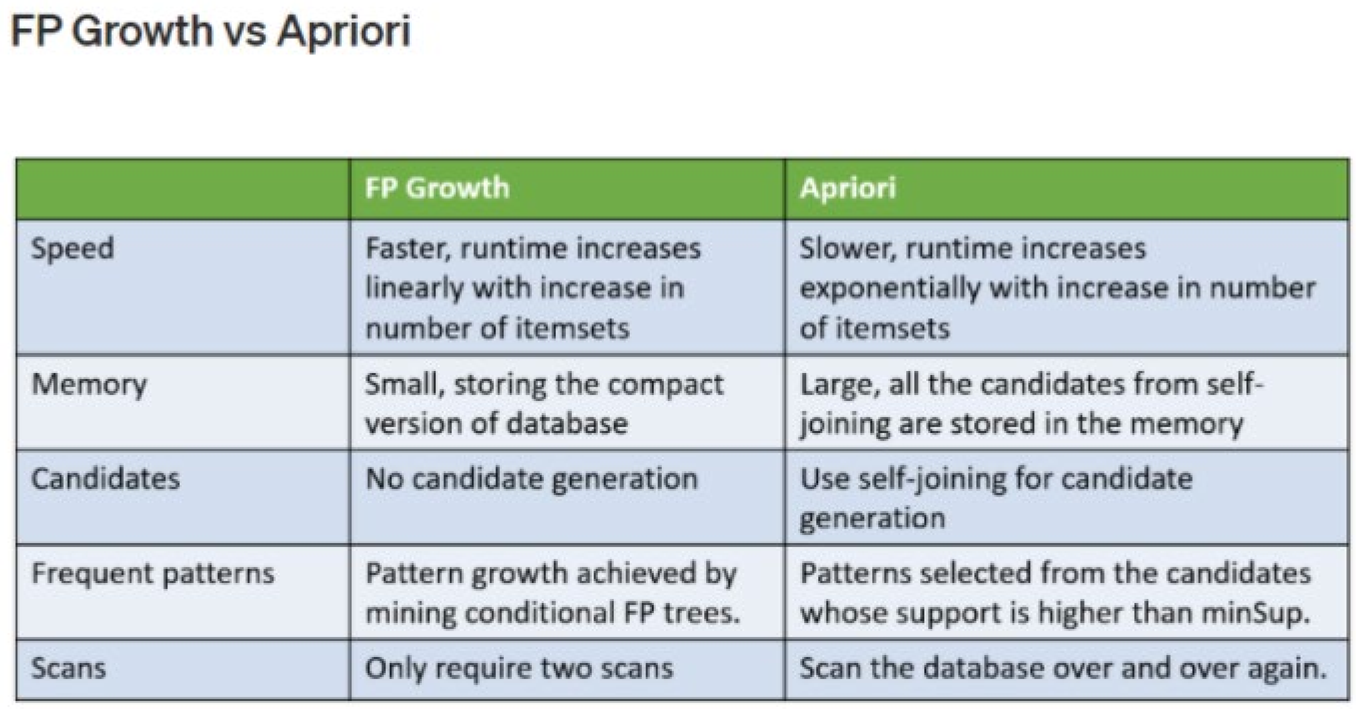
The major limitation in any association rule-mining algorithm is the generation of frequent itemsets. If the transaction dataset is having k unique products, then potentially 2k possible itemsets will be generated. The Apriori algorithm will first generate these itemsets and then proceed to finding the frequent itemsets. This limitation is a huge performance bottleneck as even for around 100 unique products the possible number of itemsets is huge. This limitation makes the Apriori algorithm prohibitively computationally expensive.



FP growth algorithm	Apriori algorithm
FP growth algorithm is faster than Apriori algorithm.	It is slower than FP growth algorithm.
FP growth algorithm is an array based algorithm.	Apriori algorithm is a tree-based algorithm.
FP growth algorithm required only two database scan.	It requires multiple database scan to generate a candidate set.
It uses depth-first search	It uses breadth-first search.

Difference Between Fp growth and Apriori Algorithm

FP growth algorithm and Apriori algorithm they both are used for mining frequent items for boolean Association rule.



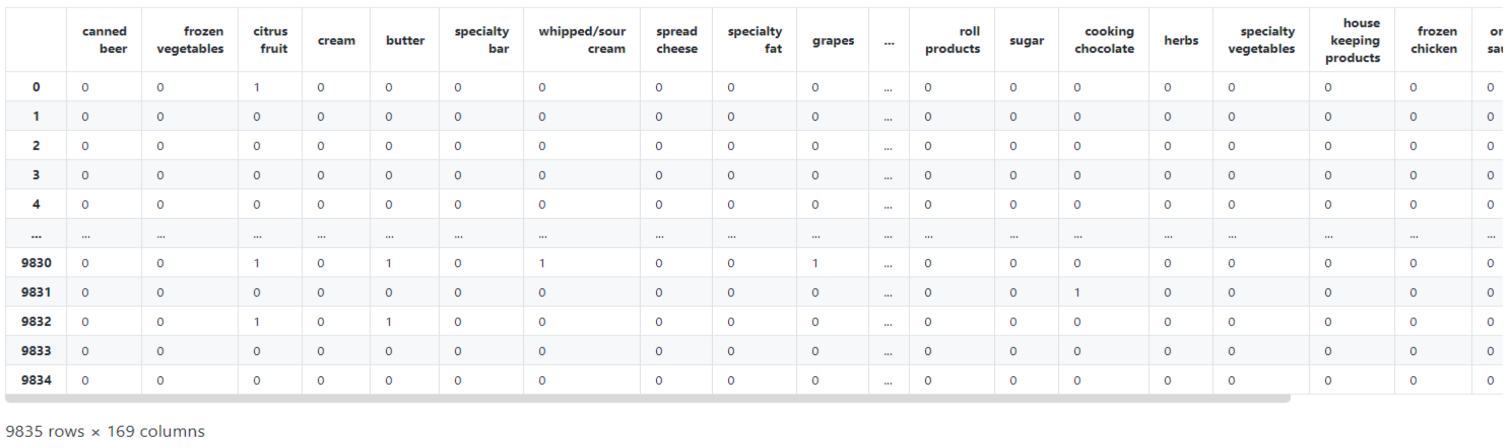
FP Growth vs Apriori

	FP Growth	Apriori
Speed	Faster, runtime increases linearly with increase in number of itemsets	Slower, runtime increases exponentially with increase in number of itemsets
Memory	Small, storing the compact version of database	Large, all the candidates from self-joining are stored in the memory
Candidates	No candidate generation	Use self-joining for candidate generation
Frequent patterns	Pattern growth achieved by mining conditional FP trees.	Patterns selected from the candidates whose support is higher than minSup.
Scans	Only require two scans	Scan the database over and over again.

Fig 4: Difference between FP growth and Apriori Algorithm

As shown earlier instance where I apply using Apriori algorithm to identify the antecedents and consequents, here I'll also use a more efficient algorithm, the FP growth algorithm for finding association rules for typical sales order transaction available literally in any enterprise or retailer.

The FP growth algorithm is superior to Apriori algorithm as it doesn't need to generate all the candidate itemsets. FP growth algorithm uses a special data structure that helps it retains itemset association information.



	canned beer	frozen vegetables	citrus fruit	cream	butter	specialty bar	whipped/sour cream	spread cheese	specialty fat	grapes	...	roll products	sugar	cooking chocolate	herbs	specialty vegetables	house keeping products	frozen chicken	or sa
0	0	0	1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9830	0	0	1	0	1	0	1	0	0	1	...	0	0	0	0	0	0	0	0
9831	0	0	0	0	0	0	0	0	0	0	...	0	0	1	0	0	0	0	0
9832	0	0	1	0	1	0	0	0	0	0	...	0	0	0	0	0	0	0	0
9833	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
9834	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0

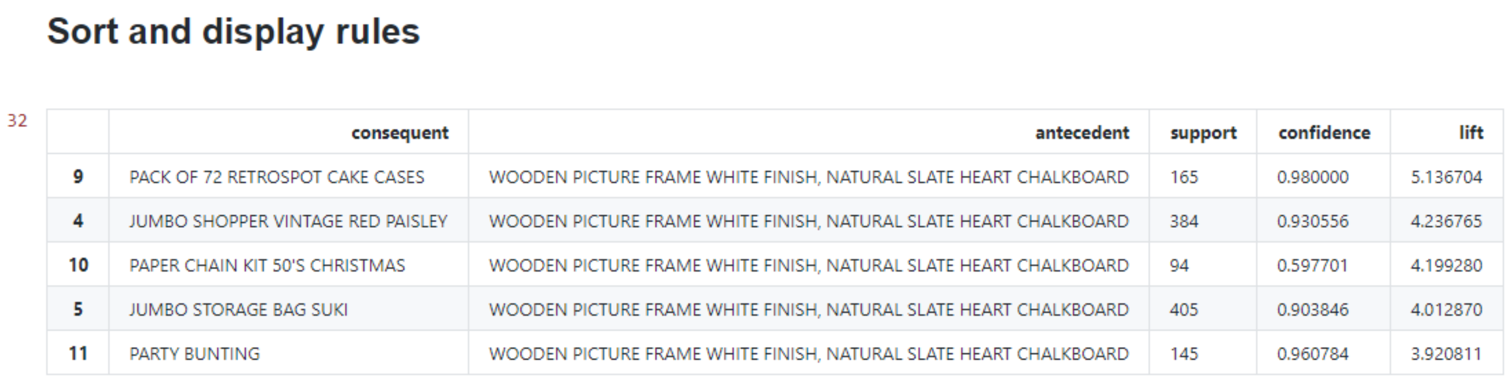
9835 rows × 169 columns

Fig 5: One Hot encoding to translate each transactional (ie: sales order or invoice) across multiple products before performing FP growth algorithm

Due to the nature of the transaction datset, no rules are generated from FP growth algorithm

```
Raw rules data frame of 0 rules generated
Unable to generate any rule
```

If there is frequent itemset pattern in the dataset available, there will be rules generated from FP growth algorithm like the image shown below.



Sort and display rules

32

	consequent	antecedent	support	confidence	lift
9	PACK OF 72 RETROSPOT CAKE CASES	WOODEN PICTURE FRAME WHITE FINISH, NATURAL SLATE HEART CHALKBOARD	165	0.980000	5.136704
4	JUMBO SHOPPER VINTAGE RED PAISLEY	WOODEN PICTURE FRAME WHITE FINISH, NATURAL SLATE HEART CHALKBOARD	384	0.930556	4.236765
10	PAPER CHAIN KIT 50'S CHRISTMAS	WOODEN PICTURE FRAME WHITE FINISH, NATURAL SLATE HEART CHALKBOARD	94	0.597701	4.199280
5	JUMBO STORAGE BAG SUKI	WOODEN PICTURE FRAME WHITE FINISH, NATURAL SLATE HEART CHALKBOARD	405	0.903846	4.012870
11	PARTY BUNTING	WOODEN PICTURE FRAME WHITE FINISH, NATURAL SLATE HEART CHALKBOARD	145	0.960784	3.920811