Association Rules: Identifying asset classes of interest for investors - Karl Merisalu

Background: One of the services UBS provides its clients is offering investment advice. Offering investment advice is a heavily regulated process with many criterias and guidelines to be followed, however, once the investment framework is agreed and one is getting to a more granular level to selecting specific financial instruments, the decision can come down to client's individual preference. At this stage, it is beneficial if the client advisor knows the client well, because he/she can then make more relevant investment recommendations.

Problem: How would I know what exactly are the investment preferences of my client?

I could (and should) do my homework to get to know the client as much as possible, but this is a very time consuming task and all client advisors do that. In addition to the basic client research, I could also take a look at investment patterns of our firm's other clients, who have similar holdings to my current client and see what else are they invested in. Based on this, I could suggest new investments to my current client if relevant-aplicable.

Task: I will work together with my group members (Colleague X, Colleague Y, Colleague Z) to develop a recommendation engine based on association rules algorithm, which will recommend investments to clients based on other clients' investments.

Often times, such an analysis is conducted to analyse people's shopping baskets when making purchases in large grocery stores for example. This will help stores to place items that are bought together close to each other for shoppers' convenience (or not, if they want clients to walk through the whole store to pick up their items). I will try to convert this idea to finance sector.

The project will be conducted in Python ecosystem to ensure the analysis would be fast and replicable on other collections of financial instruments. Python also has good tools/libraries to conduct association rules analysis, which may be less intuitive and customisable in other ecosystems (e.g. in MS Excel).

In this group project, for clarity, my part is overall coordination and division of tasks. I also obtained internal data by communicating with colleagues from various functions, anonymised the data, did an initial analysis and wrote initial conclusions. After these steps my work was reviewed and complemented by my colleagues.

All group members are responsible for reviewing all of the below and making sure they understand and are able to reproduce everything individually.

Given that all group members work in different departments of UBS Asset Management, this project will be a great example of project management, collaborating across teams/departments and making sure that everyone understand all aspects of the work.

Limitations: Making investment recommendations is based on several factors, out of which only one could be client's preference, so the results of such analysis should be taken with caution and be considered in combination with other factors.

Secondly, I have managed to obtain data on some client investment classes, but not on individual holdings at securities level of client portfolios. For the purposes of this project, this high level granularity is sufficient as the code could be easily run on more granular data later on.

Thirdly, the dataset is heavily redacted, anonymised and represents only a fraction of the full picture, due to data confidentiality measures. As such these results should not be interpreted in real life.

Summary of actions: To summarize, this is an association rules group project, where I'm going to divide anonymised client investments into groups based on their investment asset classes. Then, I'm going to analyse the data and try to uncover insights into the patterns of client investments. This project will be useful for my firm, because the insights into client investment patterns will enable UBS client advisors to connect better with their clients and make potentially interesting investment recommendations to clients.

In addition to making investment recommendations based on the findings of this project, the code developed could be used as a research tool, by applying the same method on a much more granular (single stock) level to uncover new insghts.

In the end of the project I will conclude findings and make suggestions for further research.

Following is a step-by-step guide on how I'm going to accomplish it:
1) Use case *(for real life perspective)*
2) Setting up the environment *(enables me to use pre-developed complex functions on data)*
3) Importing the dataset *(import and cleaning of the data)*
4) Getting an overview of frequencies of each asset class
5) Grouping data by client ID
6) Transposing grouped data into a sparse matrix
7) Using mlxtend's apriori method to generate a list of frequent itemsets
8) Using mlxtend's association_rules method to generate rules
9) Conclusion / Analysis *(interpretation and suggestions for further research)*

1) Use case

To start with and put our analysis into perspective:

An investor has a potfolio exposure to Fixed Income asset class only and wishes to create a diversified investment portfolio by introducing another asset class to his/her holdings. I'm applying association rules analysis to the asset class universe to see which groups may come up as likely recommendations based on other client portfolios/holdings. Ideally, the results will give the investor valuable information for the asset class selection process. The association rules analysis is run on a selection of randomised and anonymised data.

2) Setting up the enviroment / importing relevant libraries

First I import required libraries, which will help me to do data manipulation and run association rules algorithms

In [6]:

```
import pandas as pd
import numpy as np
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
```

3) Importing the dataset

The data used is based on internally sourced data. I have taken measures to anonymise, slice and randomise the data so that no client identifying information is revealed (nor could be reverse engineered). The data consists of: 1) anonymised client ID and 2) Asset Class the client is invested in.

These steps were necessary because **UBS has 4 data confidentiality classes:**
- **Strictly confidential** – not to be further distributed
- **Confidential** – only for targeted audience
- **Internal** – can be shared with colleagues internally
- **Public** – can be shared internally and externally

This dataset could now be considered as fictious/public data, so there is no need to hide/transform any of the data further

In [7]:

```
amData = pd.read_csv('instacart\AMdata3.csv')
amData.head()
```

Out[7]:

	client id	AssetClass
0	1	Equities
1	2	Multi Asset
2	3	Real Estate & Private Markets
3	4	Equities
4	5	Equities

4) Getting an overview of fequencies of each asset class

First let's take a look at the distribution of which asset classes do clients invest in

In [8]:

```
amData['AssetClass'].value_counts().rename("freq")
```

Out[8]:

```
Equities                          309
Fixed Income                      215
Multi Asset                       148
Real Estate & Private Markets     122
Hedge Funds                        35
Distribution Partners               6
Money Market                        4
Name: freq, dtype: int64
```

Comment: It looks like Equities, Fixed Income, Multi Asset and RE&PM are by far the most common asset classes clients have invested in

5) Grouping data by client id to have an overview of each client id exposure to asset classes

Because some clients have invested in several asset classes, I will group asset classes by client id next:

In [9]:

```
client_baskets = amData.groupby(['client id']).AssetClass.apply(np.array).reset_index()
client_baskets.head()
```

Out[9]:

	client id	AssetClass
0	1	[Equities]
1	2	[Multi Asset]
2	3	[Real Estate & Private Markets]
3	4	[Equities]
4	5	[Equities]

6) Transposing grouped data into a sparse matrix

Now I transform the grouped dataset into a sparse matrix (as shown below). This is necessary to later perform apriori analytics, such as finding frequent item sets.

In [10]:

```
te = TransactionEncoder()
te_ary = te.fit(client_baskets['AssetClass']).transform(client_baskets['AssetClass'])
dataset = pd.DataFrame(te_ary, columns=te.columns_)
dataset.head()
```

Out[10]:

	Distribution Partners	Equities	Fixed Income	Hedge Funds	Money Market	Multi Asset	Real Estate & Private Markets
0	False	True	False	False	False	False	False
1	False	False	False	False	False	True	False
2	False	False	False	False	False	False	True
3	False	True	False	False	False	False	False
4	False	True	False	False	False	False	False

Comment: 'True' values indicate client's exposure to given asset class. Initial look at the above shows that many clients only seem to have invested in one asset class. This should be taken as a warning for later, because I base on my analysis on making recommendations on similar asset class "baskets". By only having clients who invest in 1 asset class, I wouldn't be able to make recommendations based on purchase patterns (apart from that clients only seem to be interested in having exposure to a single asset class)

7) Using mlxtend's apriori method to generate a list of frequent itemsets

I managed to get a large sparse matrix of client asset class exposures, but it is difficult to read. Let's summarize below which item sets (asset classes) clients are invested in simultaneously most frequently.

In [11]:

```
frequent_itemsets = apriori(dataset, min_support=0.01, use_colnames=True)
frequent_itemsets['length'] = frequent_itemsets['itemsets'].apply(lambda x: len(x))

# if I wanted to limit itemsets to certain number I would use the following sample cod
e:
# frequent_itemsets = frequent_itemsets[ (frequent_itemsets['length'] >= 2)]

frequent_itemsets.sort_values('support', ascending=False)
```

Out[11]:

	support	itemsets	length
0	0.397172	(Equities)	1
1	0.276350	(Fixed Income)	1
3	0.190231	(Multi Asset)	1
4	0.156812	(Real Estate & Private Markets)	1
2	0.044987	(Hedge Funds)	1
5	0.032134	(Fixed Income, Equities)	2
6	0.017995	(Equities, Multi Asset)	2
8	0.014139	(Fixed Income, Multi Asset)	2
7	0.012853	(Equities, Real Estate & Private Markets)	2

Comment: I choose min_support as 0.01 because I only want to focus on most frequently invested baskets. As you can see above, client accounts have mostly invested in a single asset classes. Having min_support as 0.01 will also let us consider most frequent baskets with 2 asset classes, however, the occurrance of this is already much lower than being invested only in a single asset class.

8) Using mlxtend's association_rules method to generate rules

Now I conduct the association rules analysis to be able to tell which asset classes is the client likely to be invested in, given that a client is invested in x asset class.

I choose confidence as my metric of choice, because this will/would help us to make recommendations to other clients with similar existing investments.

I set the confidence threshold to 1% in order to see some number of results. As described earlier, most client accounts are invested only in 1 asset, so the most likely other asset class investment would show as "none".

In [12]:

```
rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.01)
rules["antecedent_len"] = rules["antecedents"].apply(lambda x: len(x))
rules["consequent_len"] = rules["consequents"].apply(lambda x: len(x))
rules.sort_values('confidence', ascending = False)

# If I wanted to see relationships between specific antecedent or consequent lengths
# I sould use the following code sample: rules[ (rules['antecedent_len'] == 1) &
#                                              (rules['consequent_len'] == 2) ].sort_
values('confidence', ascending = False)
```

Out[12]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leve
0	(Fixed Income)	(Equities)	0.276350	0.397172	0.032134	0.116279	0.292767	-0.07
3	(Multi Asset)	(Equities)	0.190231	0.397172	0.017995	0.094595	0.238170	-0.05
5	(Real Estate & Private Markets)	(Equities)	0.156812	0.397172	0.012853	0.081967	0.206377	-0.04
1	(Equities)	(Fixed Income)	0.397172	0.276350	0.032134	0.080906	0.292767	-0.07
7	(Multi Asset)	(Fixed Income)	0.190231	0.276350	0.014139	0.074324	0.268950	-0.03
6	(Fixed Income)	(Multi Asset)	0.276350	0.190231	0.014139	0.051163	0.268950	-0.03
2	(Equities)	(Multi Asset)	0.397172	0.190231	0.017995	0.045307	0.238170	-0.05
4	(Equities)	(Real Estate & Private Markets)	0.397172	0.156812	0.012853	0.032362	0.206377	-0.04

9) Conclusion / Analysis:

Interpretation: based on above results, if the investor definitely wants/needs to add a new asset class to existing portfolio, client advisor should recommend investor to add Equities exposure to his/her Fixed Income only portfolio. This recommendation is based on other clients with Fixed Income holdings, who have also exposure to another asset class. It is arguable if this is a good justification in itself, but given all other conditions have been filled such recommendation may be of interest to the client.

Such a recommendation can actually make a perfect sense, because Fixed Income and Equities asset classes are the most commonly invested assets, they often have good diversification benefits between each other and they are the most straightforward to understand.

Adding Equities to Fixed Income portfolio is a recommendation one could justify in many ways, but it is interesting that I managed to uncover this pattern with an algorithm that doesn't know anything about finance. A shortcut perhaps?

Perhaps not a real shortcut in this case, but if I was to run association rules analysis on specific financial instrument level portfolios and with much more data, I am likely to come across shortcuts. This is because it can take impossibly long time to to go through large portfolios instrument-by-instrument and form a view, so it's much more efficient to identify patterns first and then investigate them directly to understand if and why they make sense or not.

Theoretical insights summary:
1) Individual asset class support: client IDs are most frequently exposed to Equities asset class. 39.7% of all client IDs are exposed to Equities asset class.

The most frequent combination of different asset classes for client IDs is Fixed Income and Equities. 3.2% of all client IDs are exposed to Fixed Income AND Equities asset classes.

2) Antecedent-consequent confidence: 11.6% of client IDs who are exposed to Fixed Income are also exposed to Equities. This is the highest such confidence metric that could be potentially used in fund recommentations.

The 2nd highest confidence metric is "when Multi Asset --> (then) Equities" exposure, which occurs at 9.5% of client IDs

3) Lift: When looking at lift, however, I notice that all shown metrics are below 1. This means that the presence of antecedent has a negative effect on presence of consequent. As such, the first recommendation to investor should be not to invest in any other asset class *(only in the case where the investor wants to add another asset class, we can recommend Equities)*. This is caused by a large marjority of client IDs being exposed to only 1 asset class (meaning client ID having no other exposure is the most likely situation).

To explain the lift phenomenon: it is most likely caused by clients making new investments from under new client IDs (especially when dealing with very large investment accounts).

Further research:
1) Run these algorithms on more granular specific financial instrument level data
2) Run these algorithms on different kind of corporate data like errors made by employees in a firm for example. Introduce additional variables like holiday periods, quarter ends and etc. It would be interesting to see if it is possible to uncover causes of making errors. If yes, company could intervene and avoid such costs.