Author: Antonello Dettori

Data Mining (DM 2020/2021)

RAM vendors dataset report

Task 1: Data Understanding and Preparation

1.1.1: Data semantics

The project is composed of multiple datasets to explore, here we will see a brief rundown of their content, in order to keep this section brief only features whose meaning is not apparent will be expanded upon.

Geography dataset

Contains a list of locations in which RAM sales can take place/are shipped to.

- geo_code: ID of each entry
- continent/country/region/currency: information about the location

RAM dataset

Contains a list of RAM modules, with their characteristics, that are available for sale.

- ram_code: ID of each entry
- brand: name of the brand under which the module is sold
- name: name of the series of the module (ie. Corsair Vengeance: Corsair -> brand, Vengeance -> name)
- memory: size of the memory in GigaBytes
- memory_type: the module's memory technology (ie. DDR3)
- clock

Sales dataset

Contains the sales entries with prices and related IDs from the other datasets.

- ID/ram_code: ID of each entry
    - From a correlation check it can be seen that there is a 1-to-1 relationship between these features, making one of them redundant

- time_code: day in which the sale took place (format Year/Month/Day)
- geo_code/vendor_code: IDs of the other datasets, used to link the place of sale/vendors and RAM features
- sales_uds: sales value in US dollars
    - In order to make sure that the name is just a misspelling of USD we can group the value of sales by country and perform the the difference between sales_uds and sales_currency: all the differences in the US have a value of 0
    - This value is used throughout the notebook in order to simplify the comparison of results across different locations
- sales_currency: sales value in the local currency of the buyer
    - There is a 1-to-many relationship between vendors and currencies, meaning that the feature is tied to the buyer's currency and not the seller's one

Time dataset

Contains a list of dates in which the sales can take place.

- time_code: ID of each entry
- year/month/day/week

Vendor dataset

Contains the list of vendors that sell the modules.

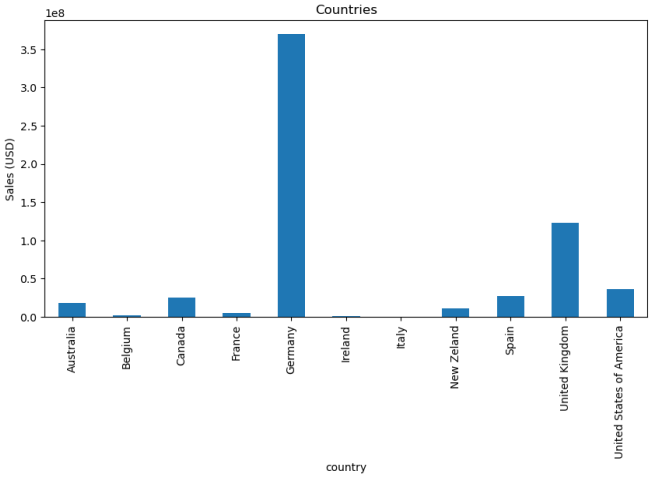
- vendor_code: ID of each entry
- name: name of the vendor

1.1.2: Distribution of the variables and statistics

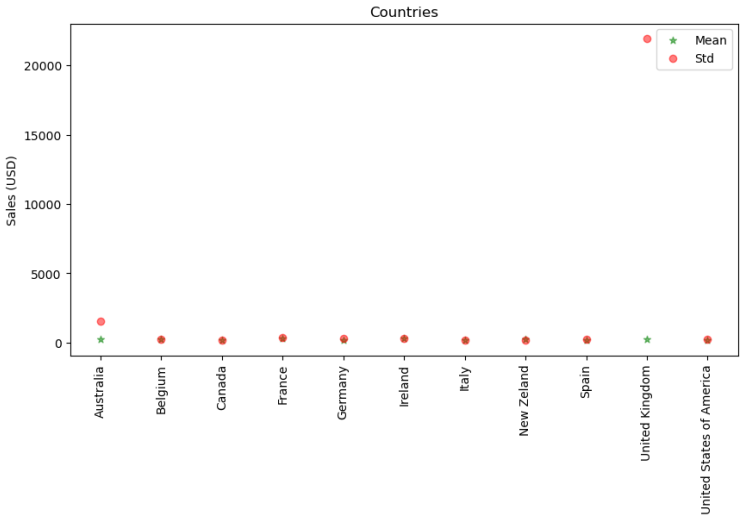
Here we will take a look at some of the most interesting variable distributions across the dataset.
For a more in-depth look check out the Jupyter notebook.

Geography dataset

The first plot shows the distribution of the total value in USD of the sales across countries.
The country with the highest revenue is Germany by a wide margin, followed by the UK and the US.
Most of the sales take place in Europe with the Euro currency.

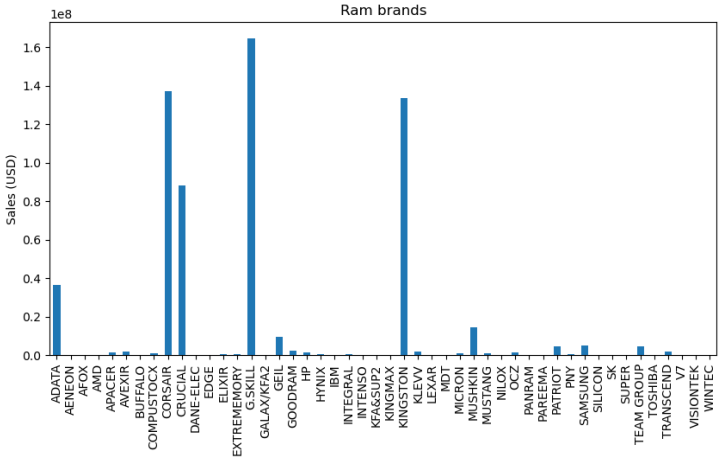


The second plot shows the average value and standard deviation of the sale price of the modules sold grouping by country.
These two values are very similar for most countries, except for the UK and Australia where there are big differences.
This suggests the presence of extreme values in the dataset and therefore of possible outliers.

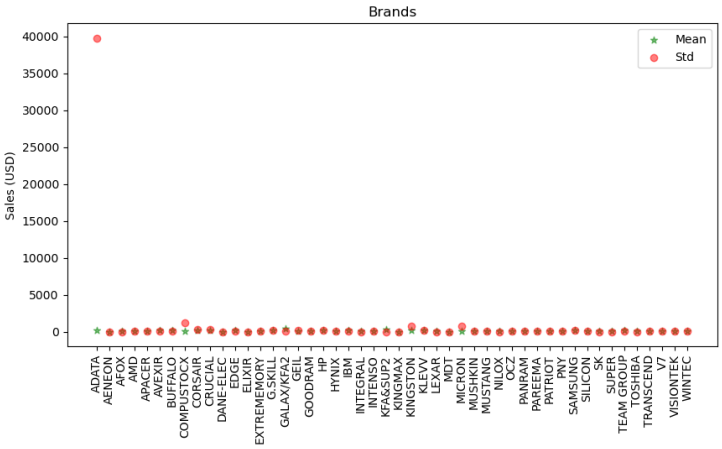


RAM dataset

The brands with the highest revenue are G.Skill followed by Corsair, Kingston and Crucial.



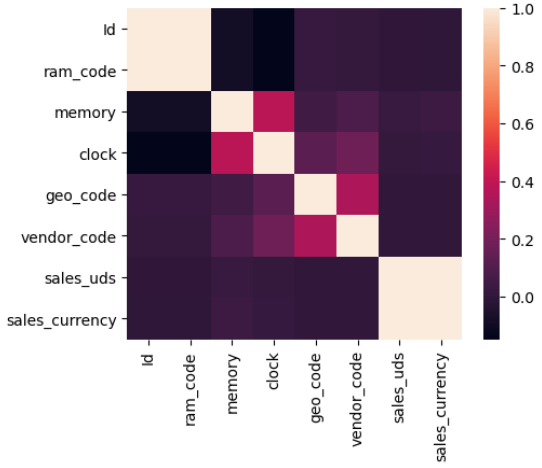
Adata is the brand that shows the highest variance in its selling price (in USD), same as before we took this as an indicator of potential outliers in the data.



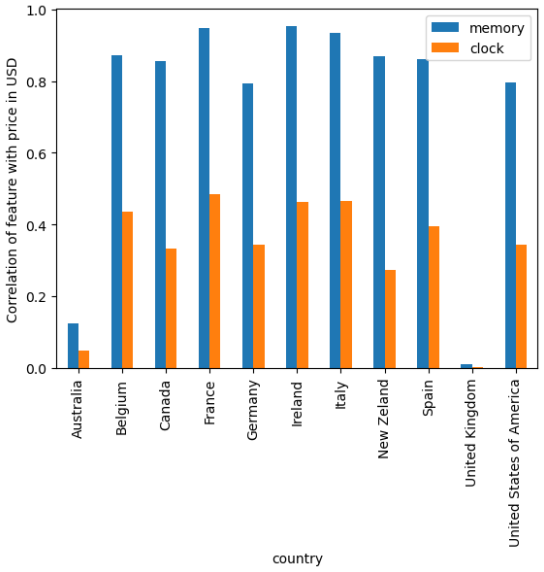
Sales dataset

Since this is the dataset that ties all the other ones together we will take into account features outside the sales dataset itself.

The correlation heatmap shows that there are no strong correlations between the features except for the couples (Id, ram_code) and (sales_uds, sales_currency).
The first couple we already mentioned in 1.1, the second reflects the fact that the sales occur only in countries with a relatively strong and stable currency whose value across time didn't change drastically.

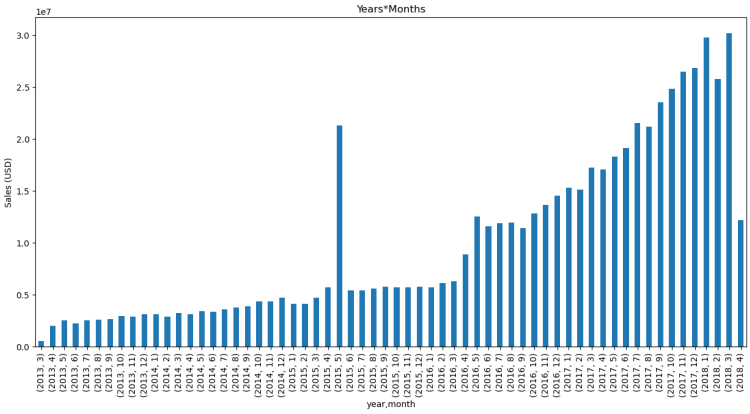


Another interesting thing that can be observed in the heatmap is that the correlation between the ram's features (clock and memory) is very low.
This is suspicious behaviour since the value of a module should, in general, be determined by its characteristics, however this problem can be explained by taking a closer look at the problem.
In the plot on the right we can see that the memory size is strongly correlated with the price in almost every country, with the exception of Australia and the UK.
Further investigation showed that the causes for this phenomenon are limited to a handful of dates in which some modules were sold at extreme prices.



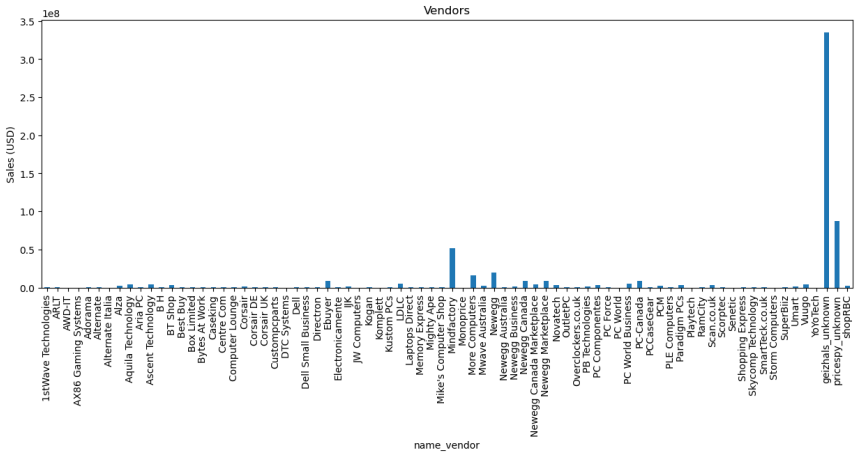
Time dataset

From the distribution of sales (USD) per month we can see that there is a constant growth in the total value of the sales across time.
The notable exception is the spike in May 2015, which also contains one of the

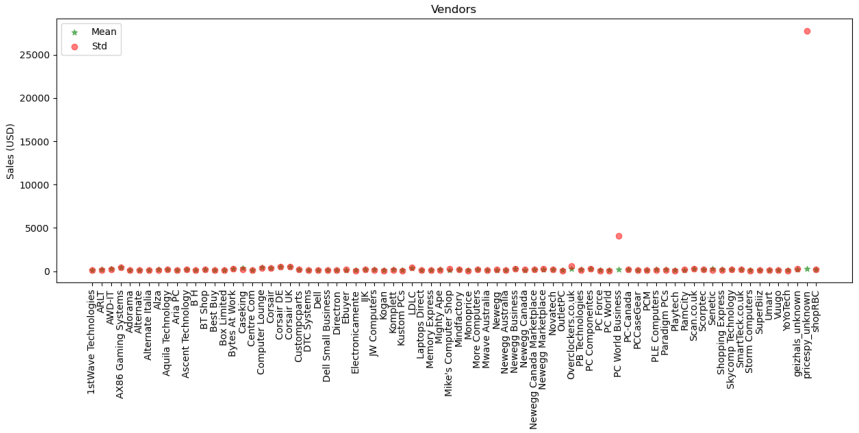


entries of the UK which drastically lowers the correlation between price and features.
The reason for the abrupt decline in sales is due to the end of the dataset.

Vendor dataset



The vendors with the highest revenue are geizhals_unknown by far, followed by pricespy_unknown and Mindfactory.
The first two of these vendors are, respectively, German and UK based price comparison engines which do not sell directly to the end-user.
This helps in understanding the stark difference between vendors.



From the mean and std values we can see that two vendors, PC World Business and pricespy_unknown, contain potential outliers.
To further support this thesis it's easy to find out that both vendors are from the UK.

1.1.3: Assessing data quality

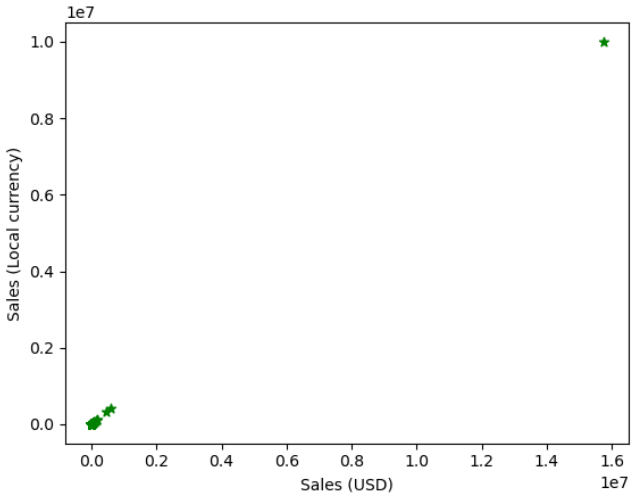
The data was checked on a feature per feature basis for:

- NaN values/empty strings: nothing found
- Extreme values: some have been found through analysis as shown in the plots above, the rest through techniques of outlier detection
- Negative values (ie. memory size <= 0): nothing found
- Duplicate values: nothing found

Task 1.2: Data Preparation:

1.2.1: Handling of extreme values

Since the majority of the numerical values across the datasets are either uninfluential (IDs) or have an easily verifiable range of valid values (dates/memory), the efforts are concentrated on the couple (sales_currency, sales_uds).



In order to deal with the extreme values, after extensive experimentation, the best method seems to be Local Outlier Factor (LOF), which manages to find a good amount of data points both in the upper (> 10^7 US$) and lower (< 10 US$) end of the distribution of the sales prices.

1.2.2: Vendor profile definition

Following are the features used to try to understand the behaviour of the vendors.

Numerical features

- I: total number of modules sold
- Iu: number of unique modules sold
- TotRev: total revenue of vendor
- MaxValuePerOrder: max price of module sold
- AvgValuePerOrder: average price of module sold
- AvgMemoryPerOrder: average amount of memory across sold modules
- IMaxMonthSales: max revenue in any single month
- IAvgMonthSales: average revenue per month
- IMaxMonthItems: max volume of items sold in any single month
- IAvgMonthItems: average volume of items sold per month
- TotMonthBusiness: how many months the seller has been selling for
- AvgItemAvail: average across items of the availability of a product for sale (in days)
- Eram: entropy of ram modules sold by the vendor

- Egeo: entropy of countries in which the sales took place
- Ecur: entropy of currencies used

Categorical features

- TopBrand: top grossing brand for the vendor
- TopCountry: top grossing country for the vendor
- TopCurrency: currency with the highest associated revenue
- TopSalesMonth: month of the year with the highest revenue (average across years)

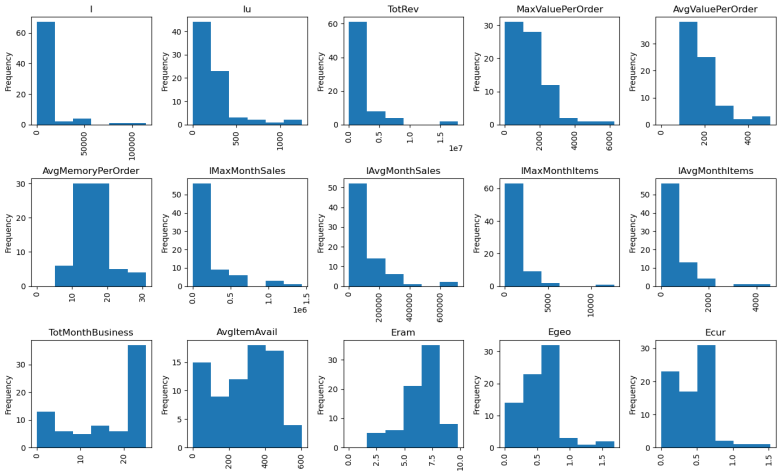
1.2.3: Distribution of the vendor variables

The notebook contains the complete process, including the comparison between these features generated starting from the sales dataset with/without outliers and the outlier detection step for this new dataset.

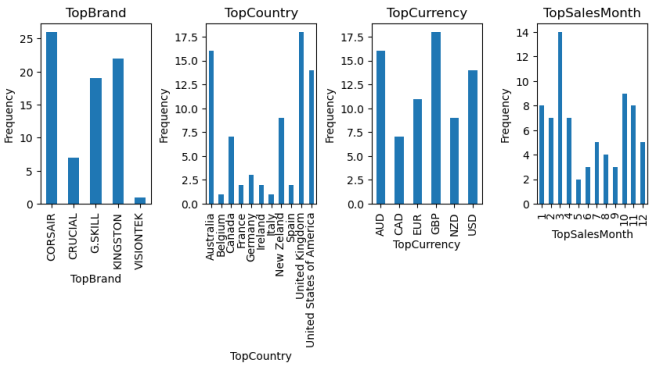
Here are included the final distributions (post outlier removal) of the features described above.

The total number of vendors is 78, 3 of them (including the two price search engines) are classified as outliers.

Numerical features

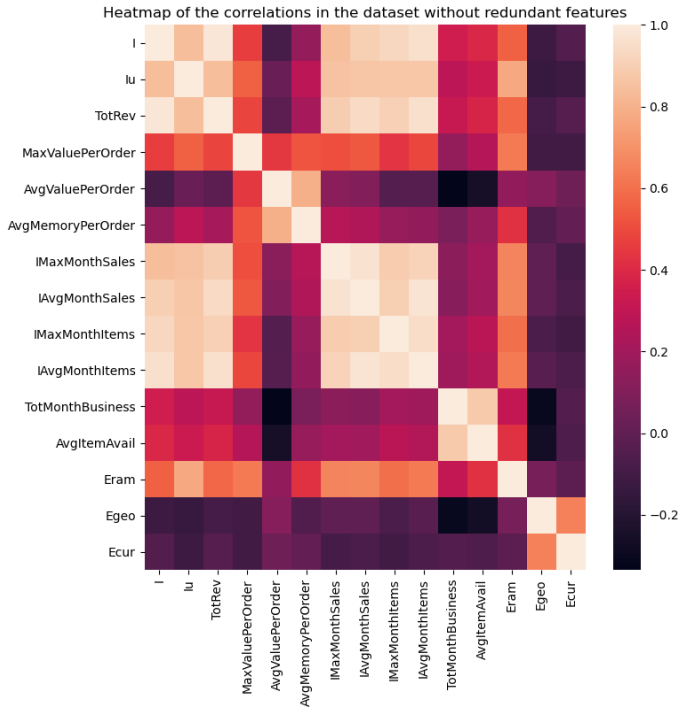


Categorical features



1.2.4: Pairwise correlations of numerical features

From the heatmap we can see that there are many pairwise strong correlations.
As a consequence, the following redundant features are removed (correlation > 0.8):



- TotRev: very strong correlation with I/Iu
- IMaxMonthSales: strong correlation with I
- IMaxMonthItems: same as above
- IAvgMonthItems: strong correlation with IAvgMonthSales
- AvgItemAvail: strong correlation with TotMonthBusiness

Not all strongly correlated features are removed, the following are kept:

- IAvgMonthSales: only feature kept related to price over time
- AvgMemoryPerOrder: potentially interesting for understanding the market niche that the vendor sells to

All the other features not mentioned are also kept for further analysis.

Task 2: Clustering analysis

2.1: Methodology

The approach taken to finding the optimal clustering is mostly the same across all the techniques used.
There is an initial of the metrics applicable to the clustering technique (ie SSE with knee method, Silhouette, KNN distance), follows a plot of the similarity matrix and a parallel coordinates plot to visualise the distribution of the features across labels and at the end of each clustering section there is an interpretation of what the optimal clustering represents. To keep the section brief we will only report the final findings for each technique used.

2.2: Metrics used

- Internal:
    - Silhouette: a measure of both cohesion and separation (higher is better)
    - SSE: a measure of cohesion (lower is better)
    - Davies-Bouldin: a measure of separation (lower is better)
- External:
    - Homogeneity
    - Completeness
    - Mutual information

When evaluating these metrics it's important to keep in mind that they can be biased towards a particular clustering technique due to their specific definition (ie. K-means results in clustering with a small SSE).
Silhouette is generally considered the more comprehensive metric for different clusterings, therefore it is frequently used as a deciding factor.

2.2: Pre-processing

The pre-processing for this task consists of:

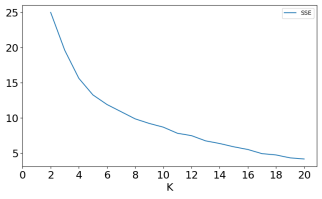
- Removal of outliers: only the vendor dataset without outliers is used due to negative impact on some clusterings during preliminary testing
- Normalisation step: since the outliers are already taken care of, the transformation chosen is Min-max normalisation which also allows for an immediate interpretation of the results

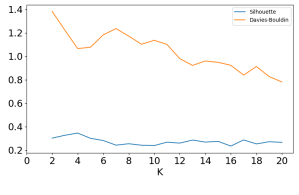
2.3: Analysis of prototype-based clustering results

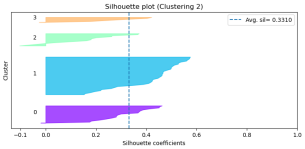
The algorithms tested are K-means from sklearn and Fuzzy c-means from pyclustering.

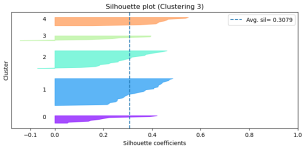
K-means

The optimal value of K was found by first narrowing down the range of Ks to [2,5] using the knee method with the SSE of the clusterings and the metrics.



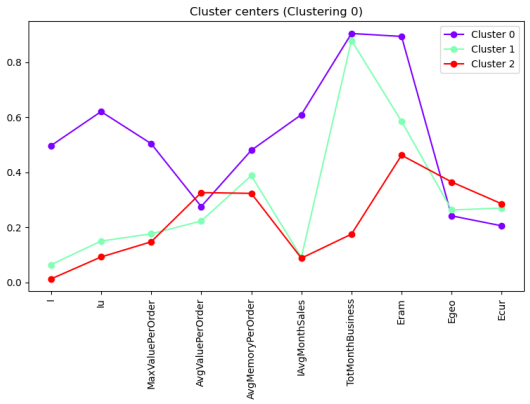






The silhouette score and the individual data point contribution points to the clustering 1 being the optimal one.

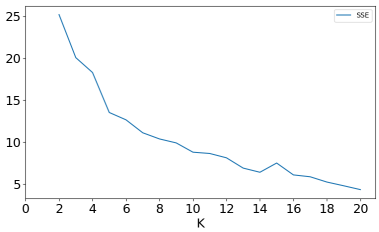
The distribution of the features show, on average, that:

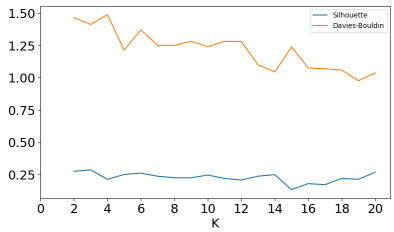


- Cluster 0 vendors: sell a big (I) amount of varied (Eram) ram and have been in business for a long time (TotMonthBusiness)
- Cluster 1 vendors: sell a small amount of ram but have been in business for a long time
- Cluster 2 vendors: sell a small amount of ram and are new to the business

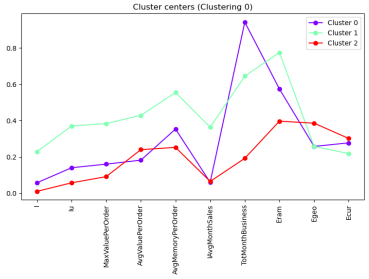
The resulting new labels are, respectively, big-old, **small-old** and **small-new** vendors.

Fuzzy c-means

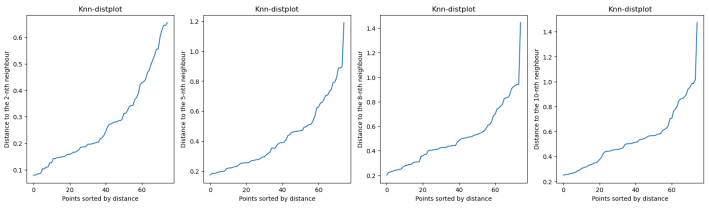


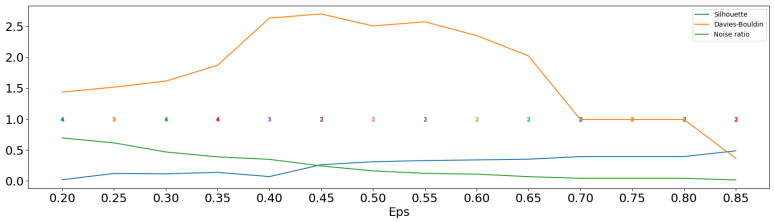


The results were very similar in their behaviour to the K-means clusterings (expected given the similarities between the two algorithms), however the metrics are generally worse (smaller separation/cohesion between clusters).
The final interpretation is identical to the previous one.



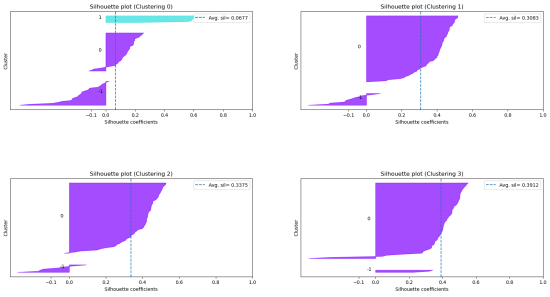
2.4: Analysis of density-based clustering results



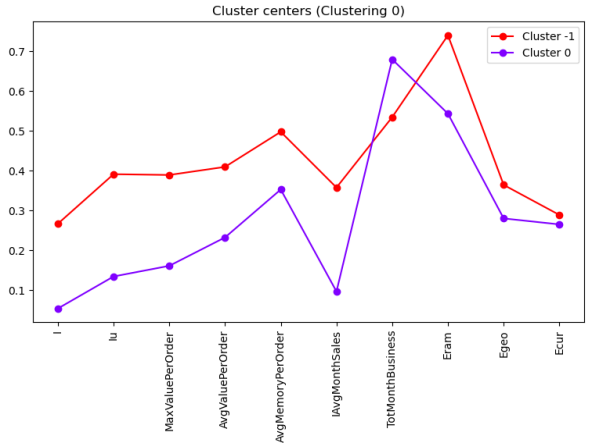


In order to determine the optimal value of minEps the knee method applied on the plot of the KNN distance to different neighbours was used, resulting in a range [0.4, 0.8].
In order to find the optimal value a plot of the metrics was used (the noise ratio and the

number of clusters per minEps are used to get a feel for the concentration of data points among the different clusters).



The individual data point silhouette plot showed that the clustering 1 (range [0.5, 0.6]) has an average silhouette in line with the others but it's cluster 0 contains no entries with a negative silhouette.
The other clusterings were excluded either to a low average silhouette (clust. 0), a low amount of data points in the noise clustering (clust. 3, only 3 data points in label -1) or because their behaviour was identical (clust. 2).

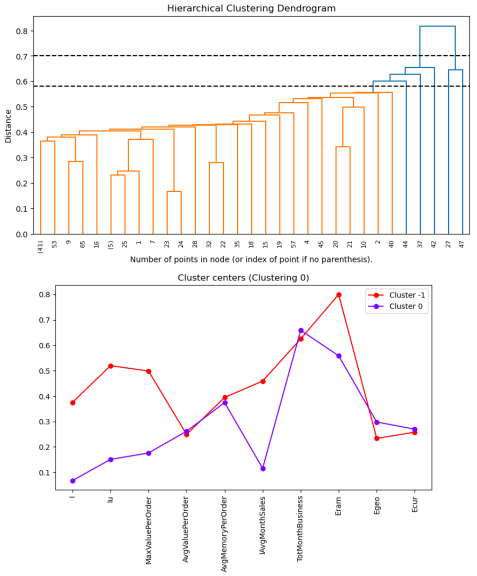


The interpretation for this clustering is simpler compared to the previous cases due to there being less distinguishing features: **big** (label -1), **small** (label 0).

2.5: Analysis of hierarchical clustering results

In order to find the optimal cut for the dendrograms generated by each hierarchical clustering multiple cuts are taken at varying heights that present interesting properties (i.e. drastic change in distance between levels).

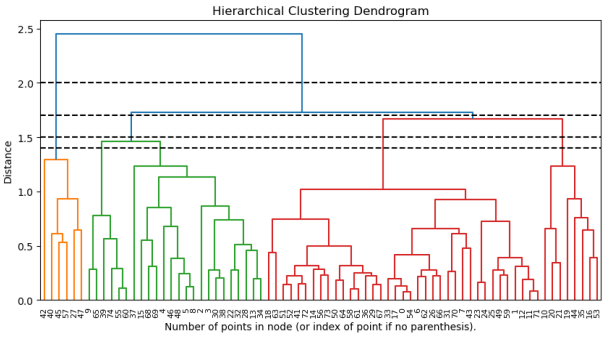
Single linkage



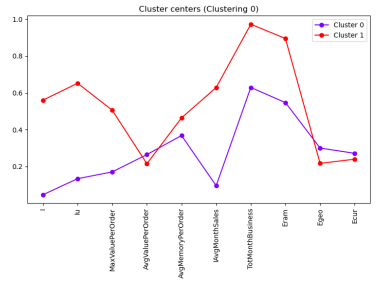
We can notice that, as a consequence of single linkage, all cuts result in a big cluster and a smaller one or a group of singleton clusters.
By post-processing the clustering and adding all singleton clusters of the lowest cut in a single "noise" cluster we obtain a clustering with better metrics compared to the higher cuts and that is also remarkably close to the cluster 0 of K-means.

The interpretation in this case is: **big** (label -1) and **small** (label 0).

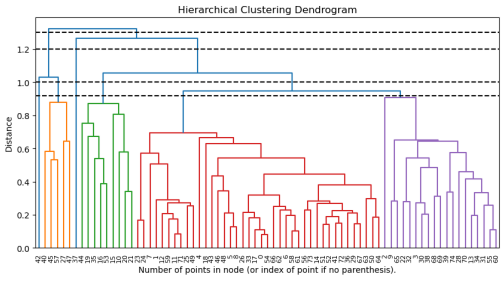
Complete linkage



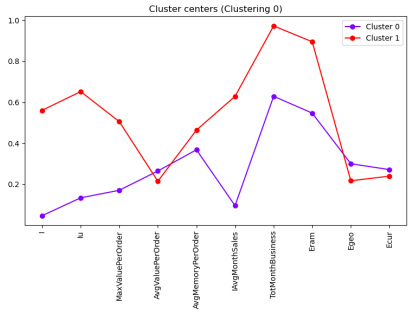
This time the cuts result in more diverse clusters, in particular in order to obtain a singleton cluster the cutting height has to be very low.
In this case the metrics favored the highest cut, which resulted in a clustering whose feature distribution is very similar to the previous one.
As a consequence, the interpretation is the same as the previous one.



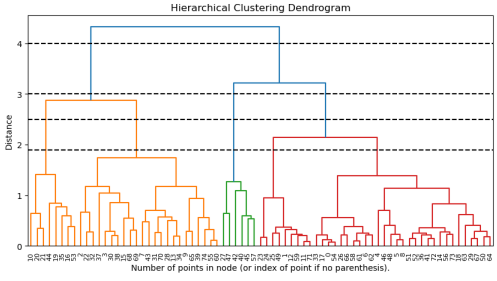
Average linkage



This linkage also presents many cutting heights that result in more balanced (in terms of number of vendors contained) clusters.
However all cuts, except the highest one, result in at least one singleton cluster.
These clusterings were post-processed in order to merge the singletons in the labels that showed the most similarity in terms of features, then a final metric evaluation took place.
The best results were obtained by the unaltered highest cut clustering, which also shows an identical distribution of the features to the complete linkage case.

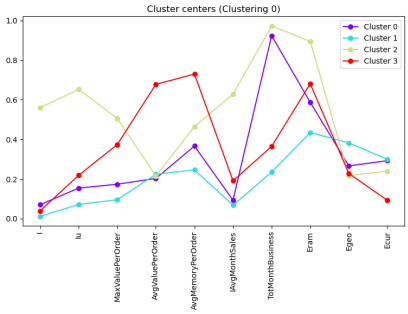


Ward linkage



The most promising cut this time is the second to lowest one.
This is somewhat surprising given that the objective of the Ward linkage is to minimise the SSE, same as K-means, however the clustering is drastically different.
The interpretation for the labels is always determined mostly by the same features,

however this time there are four labels: **old-small** (label 0), **new-small-cheap** (label 1), **old-big** (label 2) and **new-small-expensive** (label 3).
The use of the **expensive** due to AvgValuePerOrder being a distinctive feature of the label.

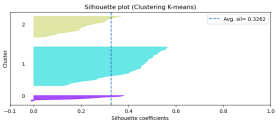


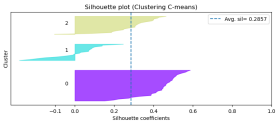
2.6: Final evaluation

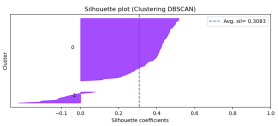
Internal metrics

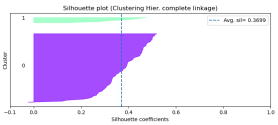
According to the Silhouette and Davies-Bouldin scores the best clusterings are the ones provided by either **hierarchical clustering with complete or with average linkage**.
These clusterings also present almost no negative Silhouette values on a per data point basis, which indicates that vendors are closer to entries in the same cluster than entries in other clusters (RECHECK).

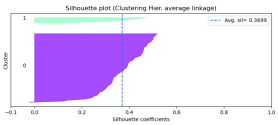
	Type	Clusters	SSE	Silhouette	Davies-Bouldin	Ideal corr. avg
0	K-means	3	19.604555	0.326206	1.222483	0.439060
1	C-means	3	20.075618	0.285702	1.411721	0.356643
2	DBSCAN	2	31.164668	0.308345	2.505490	0.447247
3	Hier. single linkage	2	32.294690	0.342998	2.110812	0.490579
4	Hier. complete linkage	2	28.158243	0.369885	1.040547	0.479302
5	Hier. average linkage	2	28.158243	0.369885	1.040547	0.479302
6	Hier. ward linkage	4	16.022487	0.328263	1.096212	0.454360











External metrics

The interpretation labels chosen have the interesting property of being mostly reducible to more general ones, we can then use the external metrics to compare the correspondence between the labels of the clusterings.
This analysis can be performed both by reducing the labels to the **old** and **new** or to the **big** and **small** ones.

	Model 1	Model 2	Homogeneity	Completeness	Mutual Info	Labels 1	Labels 2
0	K-means (2 labels)	C-means (2 labels)	0.611208	0.646168	0.628202	(old, new)	(old, new)
1	K-means (2 labels)	DBSCAN (2 labels)	0.021407	0.030522	0.025164	(old, new)	(big, small)
2	K-means (2 labels)	Hier. single linkage (2 labels)	0.001618	0.004141	0.002327	(old, new)	(small, big)
3	K-means (2 labels)	Hier. complete linkage (2 labels)	0.051791	0.116463	0.071698	(old, new)	(small, big)
4	K-means (2 labels)	Hier. average linkage (2 labels)	0.051791	0.116463	0.071698	(old, new)	(small, big)

Reducing the label to **old**/**new** didn't have good results, given that only the prototype-based clusterings allowed for such an operation.
There was no meaningful connection between the clusterings based on the **big**/**small** and **old**/**new** interpretations.

	Model 1	Model 2	Homogeneity	Completeness	Mutual Info	Labels 1	Labels 2
20	Hier. average linkage (2 labels)	Hier. ward linkage (2 labels)	1.000000	1.000000	1.000000	(small, big)	(small, big)
18	Hier. complete linkage (2 labels)	Hier. average linkage (2 labels)	1.000000	1.000000	1.000000	(small, big)	(small, big)
19	Hier. complete linkage (2 labels)	Hier. ward linkage (2 labels)	1.000000	1.000000	1.000000	(small, big)	(small, big)
3	K-means (2 labels)	Hier. complete linkage (2 labels)	0.775323	0.862691	0.816677	(big, small)	(small, big)
4	K-means (2 labels)	Hier. average linkage (2 labels)	0.775323	0.862691	0.816677	(big, small)	(small, big)
5	K-means (2 labels)	Hier. ward linkage (2 labels)	0.775323	0.862691	0.816677	(big, small)	(small, big)
0	K-means (2 labels)	C-means (2 labels)	0.462507	0.253480	0.327482	(big, small)	(big, small)

In the big/small reduction we can see that the scores are quite high for different combinations of clusterings.
In particular we can see a complete match between the different hierarchical clusterings (except for single linkage).
K-means also displays good results when compared to the hierarchical clusterings, while all the other combinations have medium to low scores.
According to these results, any of the aforementioned clusterings are equivalent when reduced to a more general label, they just offer a different partitioning of the data depending on the needs of the analyst.

Task 3: Predictive Analysis

Here are included the results from the classification task.
The analyses were performed using different models with varying results, the main difficulty being that the dataset is extremely small (78 vendors) resulting in models that can easily overfit.
In order to avoid this problem and to better understand the models behaviour, different explainability techniques are used throughout the notebook.

3.1: Pre-processing

The pre-processing phase consisted of:

- Encoding the categorical features using a 1-hot encoding and the labels using ordinal encoding
- Splitting the data in training/test set (70-30 split)
- Creating a balanced training set (using ADASYN) since the original dataset is unbalanced with respect to the **big-seller** label, potentially affecting the performance of some models
- Re-introduction of the outlier vendors from task 1 (any amount of additional data is welcome)
- Standardisation of the datasets (preferred due to the presence of outliers)

3.2: Vendor profile definition

The **big-seller/small-seller** vendor labels are defined by taking inspiration from the clustering task: **I** and **IAvgMonthSales** as defining features and using a quantile-based threshold.

The vendor profile changes, compared to the clustering task, consist of the removal of the features used to define the labels and of the strongly correlated **Iu** feature and the addition of the 1-hot encoded features from the pre-processing step.

3.3: Models evaluation

All the model's parameters have been chosen with either a grid search or randomised search (depending on the size of parameter space) with cross validation.
Following are the models tested with this labeling/dataset (taken from sklearn, unless specified otherwise):

- Interpretable models:
    - Logistic regression
    - Decision tree
    - Naive bayes
    - KNN
    - Ripper (from the wittgenstein python library)
- Black-box models:
    - Random forest
    - SVM with linear and RBF kernels
    - Multi-layer perceptron

Each of these models has been trained with the following datasets/parameters:

- Unbalanced train set
- Balanced train set
- Unbalanced train set with model's class_weight parameter equal to "balanced" (the score of the model is weighted according to the size of each label in the set, not every model supports this option)

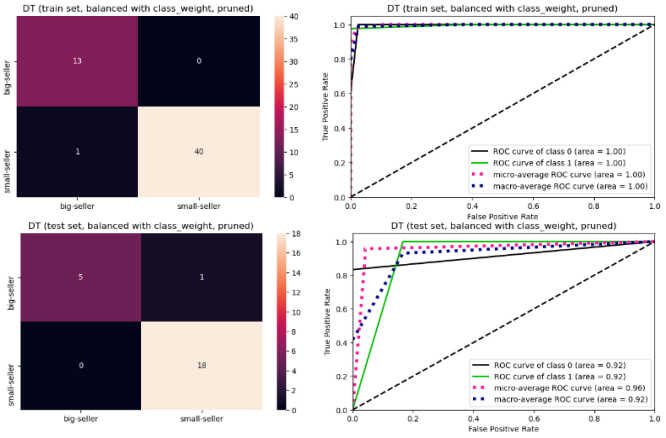
In this report are included only the top 3 models according to the following evaluation methods used:

- Standard classification metrics: accuracy, precision, recall, etc.

- Confusion matrix
- ROC curve

Decision tree

This model's unbalanced variant almost managed to correctly classify all instances presented, both in the train and test set.
This model in particular also went through a process of post-pruning due to the presence of empty nodes in the tree.

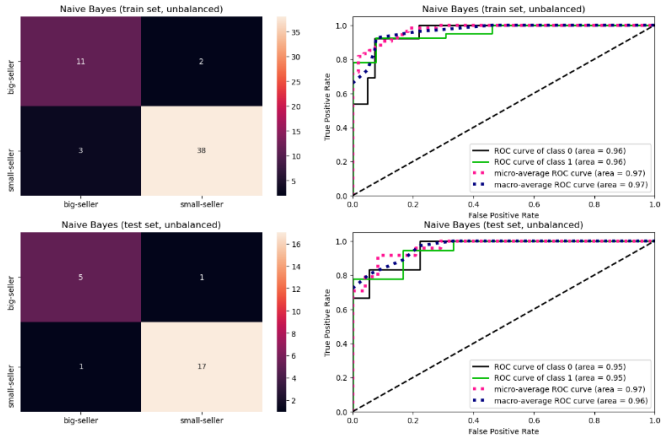


Naive bayes

This model is a composition of two separate Naive Bayes models:

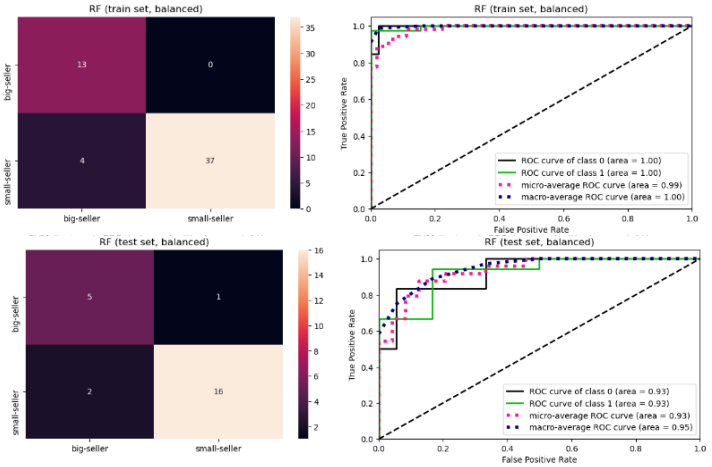
- GaussianNB: used for numerical features
- CategoricalNB: used for categorical features (in this case the features are ordinal encoded)

The probabilities of these two models are combined to obtain the final classifier.

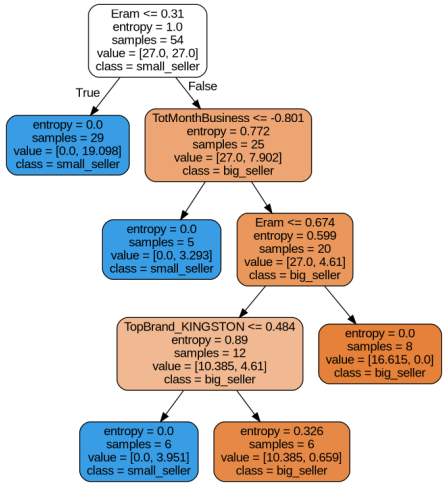


Random forest

This is the best performing black-box model for this dataset according to the notebook's results. One downside is that it is comparatively slow to train.



3.4: Conclusions



The best model, according to the results, is clearly the Decision Tree one which also has the additional perk of being easily interpretable with all the benefits that this entails.

The Naive Bayes model also shares this property, however it seems plausible that, especially in contexts in which the explanation is to be given to non-expert/consumer users, the DT is easier to understand.

Task 4: Frequent Pattern mining and Association Rule Mining

Due to the computational cost of this task the majority of the tests have been conducted on sampled datasets, the size was chosen to guarantee a small runtime for each test.

4.1: Pre-processing

This time the pre-processing consisted of:

- Removing irrelevant features (ie IDs)
- Discretisation of the numerical features into intervals, some with hand-crafted ranges others with quantile-based approaches

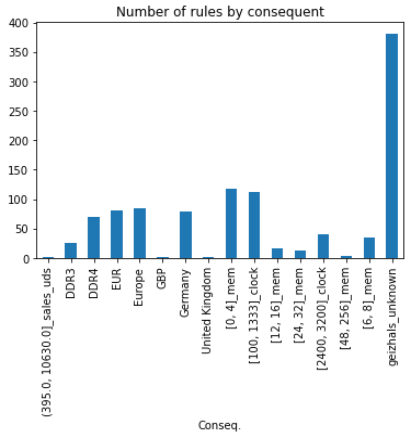
4.2: Association rules

All trials were conducted using the following thresholds:

- Support >= 5%
- Confidence score >= 60%
- Lift >= 1

Two main approaches have been tried to find meaningful association rules:

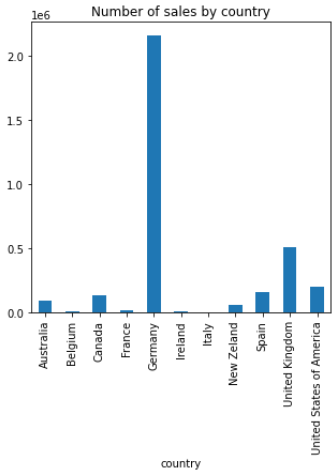
- ram_code level
    - Find interesting rules by using the ram_codes as elements of both the premises/consequents and grouping them by vendor
    - The idea being that this way it could be possible to find associations, shared across vendors, between different modules
    - The single ram_codes were then substituted by a selection of the features of the premises/consequent and a brief analysis was conducted.
    - Unfortunately no interesting properties/correlations were found with this approach
- Dataset level
    - Apply the apriori algorithm on the whole set of entries of the sales dataset, considering each row's features as a set of items and see if some interesting rules is found
    - The resulting rules are then passed through various custom filters in order to exclude the ones with premises containing redundant information (ie country and currency used only in that country)
    - The remaining rules are then grouped by consequent and analysed one group at a time
    - The findings are mostly rules connecting the different features together according to the product tier to which they, on average, belong (ie a module with memory type DDR2 is a low-end product nowadays, it's associated with having a low clock range [100, 1333] and being sold mostly in euros)



4.3: Comparison of association rules between countries

Here the objective is to find association rules from a small selection of top selling countries (in our case Germany, the UK and the US) and compare the results.

The approach taken closely follows the dataset level one, including the filtering and, in this case, the removal of country specific features from the premises (ie currency, local vendor, etc).



Following are some of the differences found between these countries:

- Germany lacks a ([48, 256]_mem, DDR4) -> (395.0, 10630.0] price range rule due to the low volume of sales for modules with those features (low support)
- Germany lacks any strong association between brands and DDR3 modules due to the fact that the sales of DDR3/DDR4 grouped by brand are more uniformly distributed compared to the UK/US, this results in a low confidence score
- The UK lacks associations between the lowest price range (0.999, 27.0] and the lowest memory size range [0,4] (gigabytes) due to the low amount of sales for these modules
- The US lacks associations between DDR2 and any low-end module features due to the low volume of sales for this memory type in the US.
  These sales take place mainly in Europe

4.4: Frequent sequences

The apriori thresholds this time are less standardised due to the computational complexity of the search.
The parameters were changed on a case by case basis in order to keep the time required to perform the analysis reasonable, while also avoiding too small of a sample size.

Multiple approaches have been tried for this section, mainly revolving around trying different combinations of features of the sales to group the single events (sales/features) into elements and sequences:

- Events defined as ram_codes:
  - Time-based: sequences defined by grouping the sales by week/month/year and each element as a single sale
  - Location-based: sequences defined by grouping the sales by region and each element as a single sale
  - Location/time-based: sequences defined by grouping the sales by location (region/country) and the elements defined by grouping the sales in a same location by time (week/month)
- Events defined as an entire row of the discretized sales dataset:

    - Time-based: sequences defined by grouping the sales by week/month/year and each element as an entire sale row
    - Location-based: : sequences defined by grouping the sales by region and each element as an entire sale row

Unfortunately none of these approaches yielded interesting results.

One possible reason is that the sales dataset, by its very nature, contains entries that have no apparent connection between one another.
This is because a ram module is a product that is bought once in a while, therefore it seems pointless to collect data on repeat purchases by the same customer.
This results in the absence of a natural grouping for the events into elements (such as what we can find in a supermarket setting, where a customer purchases can be grouped by shopping session) and makes it so that:

- In the first case (ram_codes as events), the apriori algorithm returns a list of frequent subsequences (containing elements of size 1) containing the most popular (in terms of sales) modules sold across time/location
- In the second case (entries' features as events), it returns a list of frequent combinations of features similarly to the association rules section

In light of these considerations and the results obtained, frequent sequence analysis doesn't seem to be the right tool for this dataset.