RAM Vendor Analysis

Chenxiang Zhang
c.zhang4@studenti.unipi.it
Data Mining (309AA), Academic Year: 2020/2021
Date: 27/04/2021

Abstract

The quantity of data produced in the recent years led to the increase of usage and application of various data analysis techniques and algorithm. In this project, we analyze, study and process the dataset RAM Vendors. We create features that can be used to profile single vendors, and using them to experiment with clustering algorithms and classification machine learning models. Lastly, we mine few interesting frequent patterns and association rules providing additional general understanding of the data.

Contents

1 Introduction

The quantity of data produced in the recent years led to the increase of usage and application of various data analysis techniques and algorithm. Those tools allow to study the insight and extract valuable information from the raw data. In this work, we experiment with a dataset called RAM vendor, applying data mining techniques to analyze the transaction of the vendors.

In the Section 2, we analyze, study and process the data. Followed by the Section 3, where we apply and compare different clustering algorithms. The comparison are also performed in the Section 4, but with the machine learning classification algorithms. In the last Section 5, the Apriori algorithm is used to mine the frequent itemsets and association rules.

2 Data Understanding and Preparation

The provided dataset is composed of multiple tabular files. The main one that we are going to analyze is *sales_ram.csv*. All the remainings *geography.csv*, *ram.csv*, *time.csv* and *vendor.csv* contain tables that can be jointly combined with the main file to provide additional information. The dataset *sales_ram.csv* is described by multiple features. We start the analysis firstly by taking a quick glimpse to understand the semantics of the attributes, then we follow up by analyzing each attribute with more detail by plotting and interpreting the distribution. Once the analysis is completed, a check is performed to inspect the quality of the dataset. The duplicated, missing and outliers records are all removed. Finally in the Section 2.4, the dataset is ready to be used for creating additional features. We combine and transform the already existing features into new and more informative features that will help us to define a vendor profile.

2.1 Data Semantics

The main file *sales_ram.csv* has in total 3.412.331 records. Each record is a transaction consisting of one single RAM product being sold. The record is described by 8 different features reported in the Table 1. The table also describes the automatically inferred[1] types. The majority of the features, such as *ram_code*, *time_code*, *geo_code* and *vendor_code*, represent a finite domain of objects. For each one of these features it is reported the number of unique value it can assume in the Table 1.

The last column of the Table 1 describes an initial observable problem with the associated column. The feature *Unnamed: 0*, *Id* are both redundant columns. The first is an unique incremental value representing the number of the transaction with an

[1]Pandas library is used to elaborate the files

Column	Dtype	Unique	Fix
Unnamed: 0	int64	-	Redundant column
Id	int64	-	Redundant column
ram_code	float64	3119	Wrong type
time_code	int64	1840	-
geo_code	int64	75	-
vendor_code	int64	78	-
sales_usd	float64	-	-
sales_currency	float64	-	-

Table 1: Features of the main file *sales_ram.csv*

offset of +2.602.347. This feature is eliminated since we already have an unique index representing each record [2]. The second feature *Id* also is a redundant column for the feature *ram_code* with an offset of +3.718. This is also eliminated. Proceeding with the analysis, we observe that the feature *ram_code*, which represent a specific model of RAM, is inferred with a wrong type. It will be corrected and changed into int64. We continue by giving a description of the remaining features. The column *time_code* is an unique value indicating when the transaction is registered and it is created by concatenating the values of year, month and day present in the file *time.csv*, e.g. 20170301. The feature *geo_code* represents the region where the transaction happened, e.g. mid-west usa. The *vendor_code* is simply a code to identify an unique vendor. The last two remaining features *sales_usd* and *sales_currency* both represents the cost of the product. In the first one, the cost is expressed in american USD. The second one is in the local currency. We will only use the feature *sales_usd* for the analysis.

2.2 Data Distribution and Statistics

In order to grasp a better understanding of the variables, it is helpful to analyze the distribution of the variables. We first join the features from *sales_ram.csv* with the other files to derive additional information, such as the features country, year and month. These new information provide more ways to group transactions, e.g. group by year or country.

The Figure 1 displays temporal information created by grouping the records by months and years. The time span of all the sales is between 03/2013 and 04/2018. From the graph in the right, it is noticeable that there was a sudden increase in sell after the 03/2016. By analyzing the data and the vendors, we discover that this is due to the increase of RAM vendors. In fact, many of the vendors earliest transaction can be traced back in the year 2016.

[2]Generated automatically by Pandas library

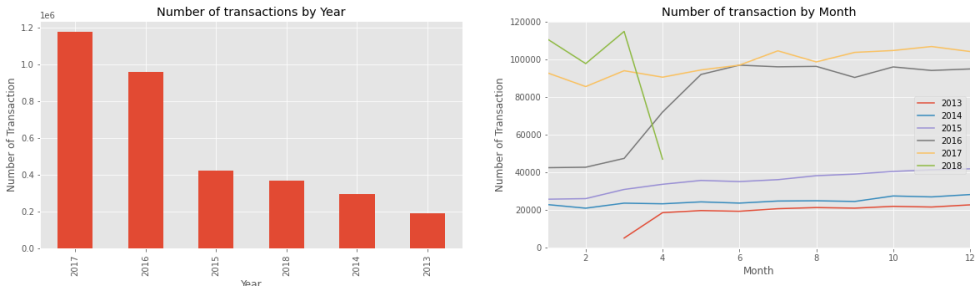

Figure 1: The distribution of the number of transaction grouped by year and month.

Figure 2: The distribution of the number of transaction grouped by the country.

The Figure 2 displays spatial information created by grouping the records by country. It shows that most of the registered transaction are held by only few countries, such as Germany and USA. The right plot has an logarithmic heatmap to ease the visualization.

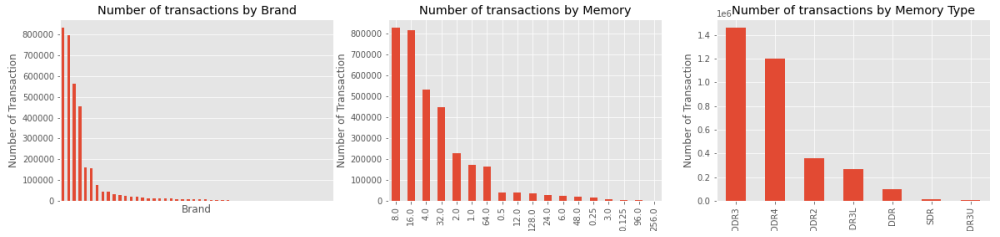

Figure 3: The distribution of the type of RAM.

We also plot the distribution highlighting the most popular type of RAM present in the market. From the Figure 3, we see that a few brands dominate the charts. The most sold memories have 8 or 16 Gigabytes with DDR3 or DDR4 memory technology.

The left Figure 4 presents the distribution of transaction grouped by the vendors. This shows that the top vendors registered the majority of transactions. Furthermore,

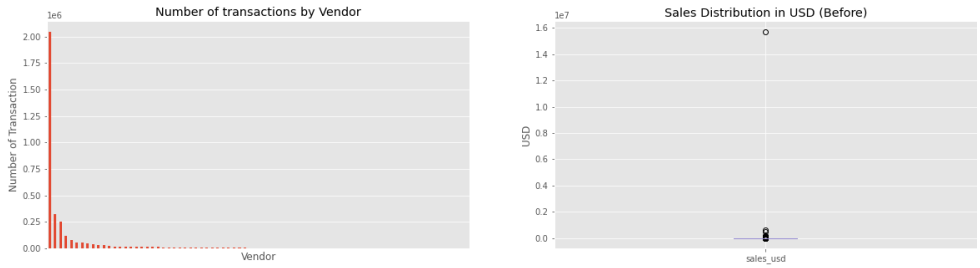

Figure 4: The distribution of the number of transaction grouped by vendors (left). The distribution of sales_usd (right), which presents outliers.

by checking the name of the most successful vendors, we notice that two of them contain the word *unknown*. Through a search on internet, we assess that these two vendors represent a collection of vendors that are not possible to trace back, hence they will be removed. The right Figure 4 is a boxplot for the feature *sales_usd*. It highlights a high number of outliers, with one in particular of several order of magnitude bigger. All these anomalies will be mitigated or removed from the dataset in the next step.

2.3 Data Cleaning

At this stage, we analyzed and have a good understanding of features. Now, the data will be cleaned by reducing the number of records. This operation involves in removing first the unknown vendors since they are a mixture of vendors. It is not possible to create a profile for the mixture since each vendor is different from the others. Then we proceed by removing the outliers data present in the feature *sales_usd*. The number of transactions before was 3.412.331. After the cleaning the number of transactions went down to 1.047.425.

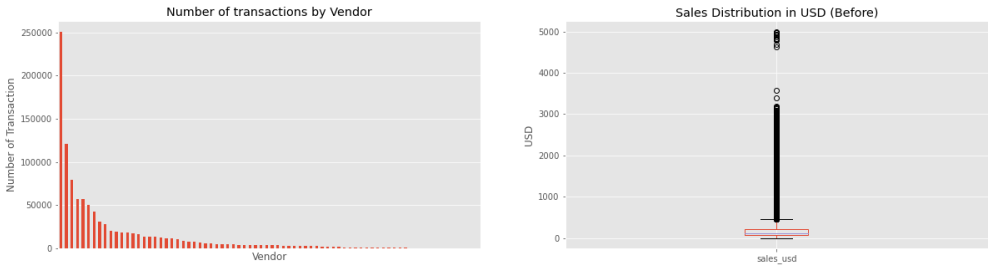

Figure 5: Cleaned version of the Figure 4.

2.4 Feature Creation

The data we have is cleaned and can be processed to create new additional features. Specifically, we want to create features that can be used to profile vendors. The desired profiles are two: {big, small}. The created feature are a transformation and combinations of the old ones. The features created are displayed in the Table 2. The first feature *unique_ram* is the number of unique RAM products that the vendor sold. This indicator can be seen as the product's variety of the vendor's magazine. The *total_units_sold* and *total_revenue* simply counts the number of transactions and sum the column *sales_usd* by aggregating the records via vendor. The *revenue_per_unit* are calculated by dividing the *total_revenue* by the *total_units_sold*. The feature *days* counts the number of days after the first registered transaction, indicating for how long the vendor has been in activity. Lastly *revenue_per_day* and *units_sold_per_day* are computed by taking respectively *total_revenue* and *total_units_sold* and dividing for the number of *days*.

Column	**Dtype**
unique_ram	int64
total_units_sold	int64
total_revenue	float64
revenue_per_unit	float64
days	float64
revenue_per_day	float64
units_sold_per_day	float64

Table 2: Features to represent a vendor profile

By performing an initial correlation plot, it is possible to see that most of the features are skewed distributions. It is possible to apply a log transformation to the data, which helps to transform skewed distributions to appear more like a normal distribution. The new correlation pair plots are displayed in the Figure 6. By observing the correlation matrix at the bottom left of the Figure 6, we notice the presence of highly correlated features. This suggests that a dimensionality reduction is needed, selecting only a subset by eliminating one of the two highly correlated features. The obtained minimal feature set will be used to identify a vendor profile. The final set of variable is composed by the set of features {*unique_ram*, *revenue_per_unit*, *revenue_per_day*, *days*}. This dataset contains quantitative, qualitative and also temporal information. We believe it is a good representation of vendors and can be used for the clustering and predictive analysis in the following sections. The final correlation matrix can be checked in the Figure 6.

Figure 6: Pairwise correlations of all the vendor profile features (top). Pairwise correlations matrix before (left) and after (right) removing the highly correlated features of the vendor profile.

3 Clustering Analysis

In this section, we are going to analyze the data using clustering algorithms. Firstly we process even further the data by Normalization and Principal Component Analysis. Then with the prepared data we experiment with different clustering methods, such as center-based, density-based and hierarchical clustering. In the end, we compare quantitatively and qualitatively the results produced by the algorithms.

3.1 Data Preprocessing

Before starting with the clustering, we apply further processing to the data. Thanks to the operations performed in the Section 3.1, we have obtained a clean dataset. The additional processing step involves two more sequential steps: normalization and the PCA. The normalization is performed using the function StandardScaler[3], which scales the input by subtracting the mean and dividing by the standard deviation to shift the data distribution to be zero centered with an unit value standard deviation. The normalization is needed in order to apply PCA dimensionality reduction. PCA maximizes the variance and it is sensible to features with predominant high variance. This is why we want to have a unit variance for all the features. The Figure 7 displays the PCA components and their captured variance for each component. The red line represents the cumulative explained variance. We choose to use the top-2 components to feed the clustering algorithms, capturing the 0.812% of the total variance.

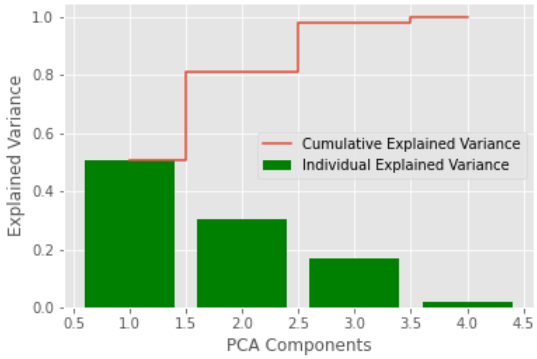

Figure 7: Explained variance for each component of PCA.

3.2 Density-Based Clustering

For the density-based clustering, the DBSCAN algorithm is used. The DBSCAN is controlled by two parameters value *Eps* and *MinPts*. The best empirical parameter for *MinPts* reported in the original paper [3] of DBSCAN is 4 for a dataset with two dimensional features. To find the optimal value of *Eps* we follow the approach taken by Rahmah et al. 2012 [6]. We compute the k-dist, which is the distance from a point to its k neighbors. Then we sort the distance in increasing order and plot the sorted values. The sharp change in the plot corresponds to the optimal value of *Eps*.

From the Figure 8, we can see that the clustering does not perform well. This is due to the nature of our data. The quantity of data is scarce and does not present well-behaved density property.

[3]From sklearn library

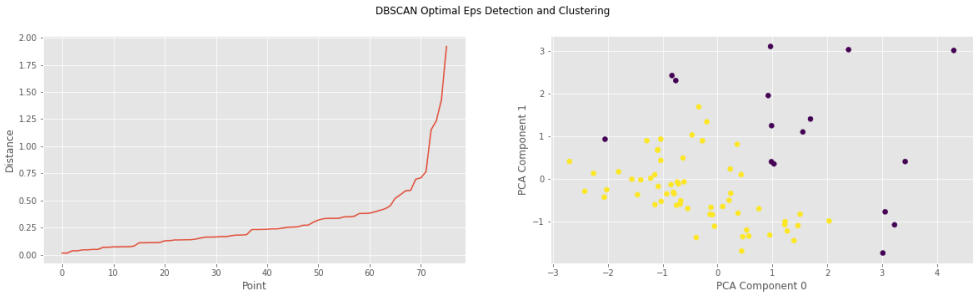

Figure 8: Detection of optimal value for the Eps (left). DBSCAN Clustering (right)

3.3 Center-Based Clustering

For this type of clustering, we choose the most widely known algorithm called K-Means clustering. First of all we want to determine the best value of K by plotting SSE and Silhouette score [7] for each different K value. The SSE score is the Sum of Squared Error, it measures the cohesion of a cluster, which is the sum of the distance for each point to its cluster center. We see from the first Figure 9 that the more K is increased, the lower is the SSE score. The Silhouette score is a more complete evaluation. It takes into account both the concepts of cohesion and separation, where separation indicates how well separated a cluster is from the others.

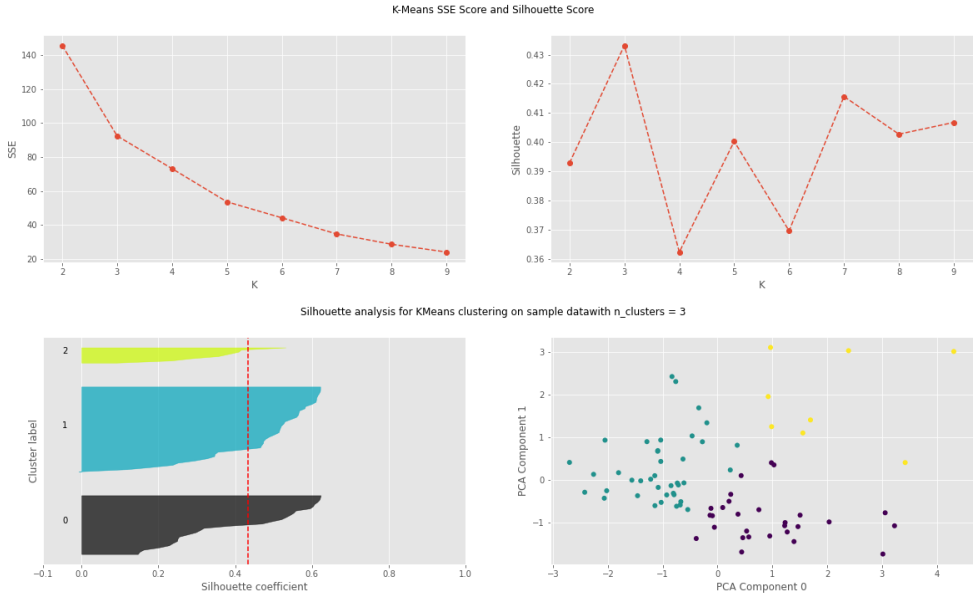

Figure 9: K-Means SSE and Silhouette score with varying K. The best value for the number of clusters determined by the search using the Silhouette score is K=3.

From the Silhouette score plot in the Figure 9, we can observe that the best value for

K is 3. But our objective is to find 2 clusters. This can be accomplished by applying a post-processing step at the end, merging two clusters into one.

3.4 Agglomerative Hierarchical Clustering

Hierarchical clustering is a general family of clustering algorithms that build nested clusters by merging or splitting them successively. This hierarchy of clusters is represented as a tree (or dendrogram). The idea is to compute a proximity matrix, then iteratively merge the closest two clusters and update the matrix until only one cluster remains. There are different metrics used to compute the proximate, called linkage criterion:

- Single link (min): compute the distance between the two closest points between two clusters. This type of linking handles non-elliptical shapes, but it is sensitive to noise and outliers.
- Complete link (max): compute the distance between the two furthest points between two clusters. This type of linking is less susceptible to noise and outliers, but it can break large clusters and it is biased towards globular clusters.
- Average link: compute the average distance between all points of two clusters. This type of linking is less susceptible to noise and outliers, but it is biased towards globular clusters.
- Ward link: minimizes the squared error when two clusters are merged. This type of linking is less susceptible to noise and outliers, but it is biased towards globular clusters.

We run the experiment by feeding the data to all the hierarchical clustering with different linkage criterion, observing the different ways in which the algorithm create the taxonomy. The results of the experiments are in the Figure 10. We can see that the best clustering is created by using the Ward Linkage. The plot at the bottom in the Figure 10 shows the Ward linkage clustering with two clusters. This is obtained by merging the top two branches in the dendrogram.

3.5 Affinity Propagation Clustering

Affinity Propagation [4] creates clusters by sending messages between pairs of samples until convergence. A dataset is then described using a small number of exemplars, which are identified as those most representative of other samples. The messages sent between pairs represent the suitability for one sample to be the exemplar of the other, which is updated in response to the values from other pairs. This updating happens iteratively until convergence, at which point the final exemplars are chosen, and hence the final clustering is given. Both time and space complexity for this algorithm is on the square

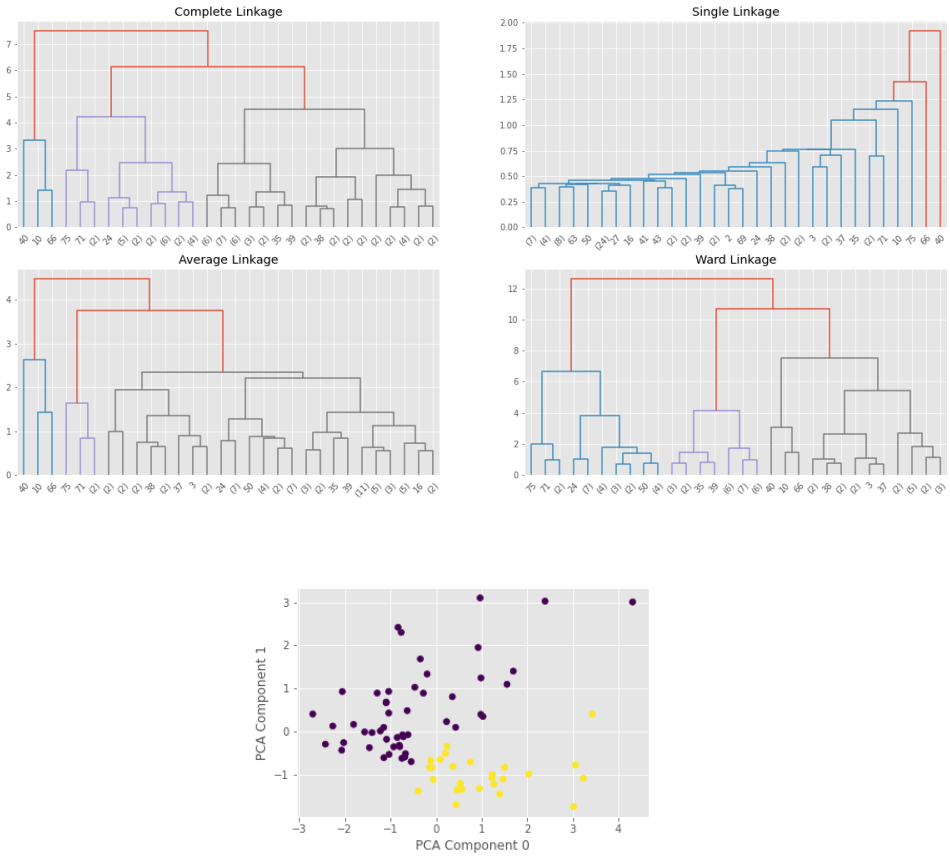

Figure 10: Agglomerative Hierarchical Clustering with four different types of linkage (top). Best linking method is Ward. Plot the Ward's clustering (bottom).

order of the number of input. In our case this is not a problem due to the small size of the dataset. The running of the algorithm and its relative clustering can be seen in the Figure 11.

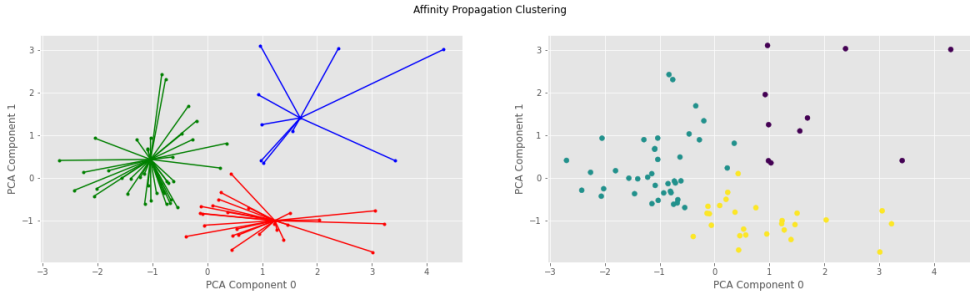

Figure 11: Affinity Clustering algorithm (left). The clustering without the linkage (right)

3.6 Clustering Comparison

In order to evaluate the performance of the clusters, we use the Separation [2] and Silhouette score [7] as our metric. A lower separation score indicates a better separation between clustering. On the other had, a higher Silhouette score indicates a well separated and cohesive cluster. For each point in the space, the Silhouette score assign it a score. Negative values indicate that the object is assigned to the wrong cluster. The maximum score is 1. For each of the clustering algorithm, we compute both the Separation and Silhouette score. A quantitative comparison of the algorithms can be observed in the Table 3.

Algorithm	**Separation**	**Silhouette**
K-Means	0.747	0.437
DBSCAN	1.287	0.409
Hierarchical (Ward)	1.076	0.346
Affinity Propagation	0.811	0.426

Table 3: Clustering algorithms comparison

As we can see from the Table 3, the worst algorithms are the DBSCAN and Hierarchical. The first badly separates due to the absence of well defined density property in the dataset. The latter algorithm achieves an almost reasonable Silhouette score, but the separation is too high. The most promising results comes from the K-Means and the Affinity Propagation clustering. Both the clustering created by these algorithms have a high Separation and Silhouette score. Furthermore, also their Figures 9, 11 seems similar. Using our small dataset and through the experiments conducted, we conclude enunciating that the most popular algorithm K-Means outperformed all his competitors both in quantitative and qualitative measures.

4 Predictive Analysis

In this section, we firstly need to complete the vendor profile in such a way that it can be used for the predictive analysis. This is done by adding a label to the already existing dataset profile created with the features created in the Section 2.4. After the dataset is created, we define the guidelines for the model selection, listing and describing a number of machine learning models for which we want to experiment with. Finally, we evaluate and provide a quantitative and qualitative comparison for all the results.

4.1 Define Vendor Profile

Consider the problem of predicting for each profile a label that defines if it is a big-seller or small-seller vendor. We need to define a vendor profile that enables the vendor

classification. One approach is to use the clusters discovered by the best clustering algorithm in the previous section 3.6. However, due to the small number of the data we decide to follow and experiment with another approach. We decide to discretize every feature of the Table 2 using 2-quantiles. Each feature will have a binary value {0, 1}. To assign a label for each vendor we create a new feature *type*, which is the sum of all the binary features. The domain for *type* is {0, 1, 2, 3, 4}. Since we want only two values for label, we binarize the domain by mapping {0, 1, 2} to *small* and {3, 4} to *big*. The process to create the feature *type* can be seen in the Table 4.

vendor_code	**unique_ram**	**revenue_per_unit**	**revenue_per_day**	**days**	**type**
1	0	0	0	1	small
2	1	0	0	1	small
...	...	...	...	...	...
80	1	0	1	1	big
81	0	0	0	0	small

Table 4: Vendor Profile Labeling with Quantile base approach

Now we provide a comparison between this approach and the K-Means results. Firstly, a quantitative comparison between this method and the clusters found by K-Means is analyzed. The Silhouette score for K-Means is 0.392 and for Quantile-based is 0.286. Based solely on this value, one would conclude that K-Means labels are more accurate. By analyzing the vendor's distribution from the data understanding section 2, we found that the distribution is skewed and most of the vendors are small. This result lead us to believe that the Quantile-based approach is a more accurate representation of the vendors clusters. In fact, the Quantile-based approach results in fewer number of big-vendors than the K-Means approach, this can be visually seen in a qualitative comparison in the Figure 12.

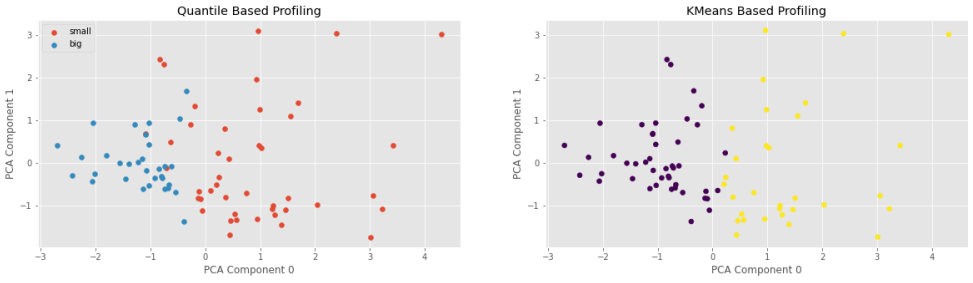

Figure 12: Comparison between the K-Means and Quantile-based clustering.

4.2 Model Selection

Before running the machine learning models, we need to define the guidelines used for the training of the models. We split the full dataset into train set (70%) and test set (30%), respectively. The split is done in a stratified way to preserve the label's distribution in both train and test set. For model selection we decide to use the Randomized search to navigate through the space looking randomly for a good set of hyperparameters. This approach is easier to setup and can explore a wider space in less time [1]. Furthermore, we also use the cross-validation with K=4.

4.3 Models Experiment

Now we choose the kind of classifiers we want to experiment with and give a brief description for all of them:

- Support Vector Machine: searches an hyper plane that can linearly separate the data points in a higher space using the kernel trick.
- Gaussian Naive Bayes: estimates the class conditional probability for each item using the Bayes theorem. It assumes that all the features distribution are Gaussian and conditionally independent between each other.
- K-Nearest Neighbors: find a defined number of training samples closest in distance to the new point, and predict the label from these.
- Decision Tree: builds a tree where nodes are decision rules learned from the training data. These rules linearly cut the space until a leaf is reached.
- Random Forest: ensemble model, it uses bagging with decision trees. It combines the predictions made by multiple decision trees and outputs the class that is the mode of outputs.
- Neural Network: multilayer neural network which learns a new representation of data with each layer. The last layer classifies the original data transformed with all the hidden layers.

By feeding the dataset of the created vendor profiles to all the models following the guidelines in the Model Selection Section 4.2, we report the results in the quantitative Table 5. This allows the comparison in terms of accuracy.

However, we believe that in order to assess the best classifier, one does not decide the final judgement based purely on the quantitative value of the accuracy. This is even more suggested in our case where the dataset is very small. We proceed in the next section to provide an additional comparison.

Model	Accuracy (TR)	Accuracy (TS)
SVM	0.982 ± 0.012	0.868 ± 0.094
Gaussian-NB	0.916 ± 0.025	0.855 ± 0.057
K-NN	0.947 ± 0.012	0.881 ± 0.068
Decision Tree	0.982 ± 0.012	0.907 ± 0.068
Random Forest	0.991 ± 0.008	0.934 ± 0.068
Neural Network	0.934 ± 0.022	0.868 ± 0.058

Table 5: Results to compare all the best classifier obtained by the model selection.

4.4 Models Comparison

In this paragraph, we aim to give a visual comparison plotting the decision boundary created by the classifiers. We do so by training all the classifiers proposed previously with the two dimensional data that are preprocessed by the PCA in the section 3.1. Doing so, the decision boundary can be visualized easily in a two dimension plot. The hyperparameters of the classifiers are the best hyperparameters returned by the Randomized search and reported in the Table 6.

Model	Best Hyperparameters
SVM	{C: 77, kernel: rbf}
Gaussian-NB	{-}
K-NN	{K: 3}
Decision Tree	{criterion: gini, max_depth: 2, max_features: log2 }
Random Forest	{criterion: gini, max_depth: 7, n_estimators: 6}
Neural Network	{hidden_layer_size: 43, learning_rate: 0.005, iter: 291}

Table 6: Best Hyperparameters found by the Randomized search.

In the Figure 13, it is possible to notice that all the models successfully create a boundary separating the two cluster of data. The Decision Tree and Random Forest models create a rigid rectangular shaped boundary, with the latter that seems to overfit the dataset in the region (-1, 0) where the noisy blue dots are separated from the red region. The SVC and Gaussian Naive Bayes traces a similar almost circular shaped boundary managing to separate properly the data. The K-NN also separates well the dataset, but it also seems to slightly overfit. The most generalized decision boundary is created by the Neural Network model, but it seems to be missing a region of red dots in the region (0, -1).

As we see, there are some great decision boundaries traced by models such as SVC, Gaussian Naive Bayes and NN. These models seems able to generalize even with the small

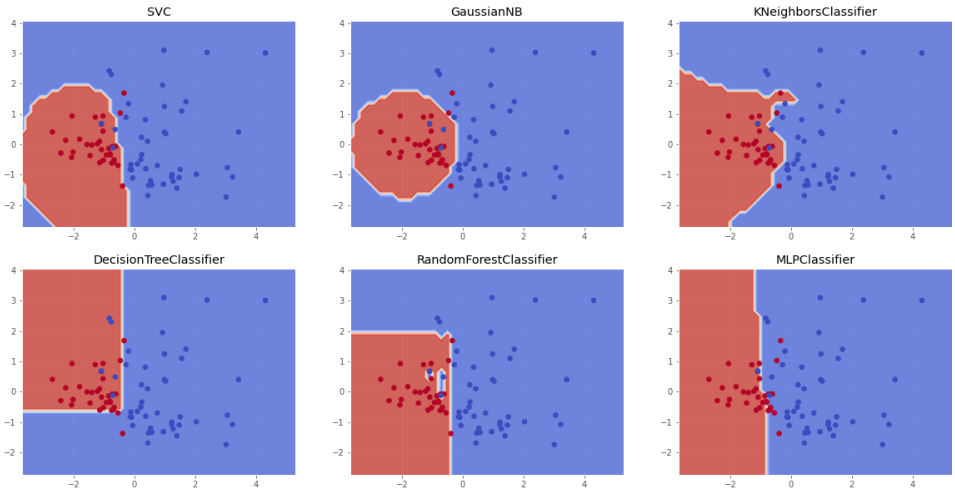

Figure 13: Qualitative comparison between all the classifiers via the decision boundary.

number of data. There's no sharp winner that rules on both quantitative and qualitative measures. We believe this is due to the bad nature of our data.

5 Patterns and Rules Mining

In order to perform the Patterns and Rules Mining, we would like to have a dataset where multiple records are associated with a distinctive transaction ID. Unfortunately, in our case we don't have such a property. In fact each record in our dataset is associated with a single product and identified by its own unique ID. Despite this inconvenience, we decided to create a transaction by taking the assumption where we group and consider all the purchases in a day as a single transaction. In this way, each transaction is identified by the feature *time_code* containing all the RAM sold in a single day by each vendor.

In our dataset we have multiple vendors. As in real world, each vendor sale's distribution does not influence the others and thus is independent from the others. In order to conduct the analysis, we pick four different vendors and analyze their most frequent and interesting patterns. The chosen vendors are two big-sellers and two small-sellers. For both the mining tasks we use the Apriori algorithm.

5.1 Frequent Patterns Mining

In this task, the goal is to find groups of frequently items which are purchased together for each vendor. The first two vendors, Mindfactory and Vuugo, are among the biggest sellers. Since that we consider a transaction as all the products sold in a single day, these two big-seller vendors have their transaction filled with almost all the existing unique RAM product. This means that even if we set the support to a very high value, the Apriori algorithm will still return many frequent set. This property is applicable also to the second vendor Vuugo, although with a smaller support.

Vendor	**Support**	**Itemset**
Mindfactory	1	{Corsair Vengeance, G.Skill Trident Z}
Vuugo	0.90	{Kingston, Kingston Hyperx Fury}
Vuugo	0.85	{Crucial, Kingston Hyperx Fury}
Centre Com	0.43	{Kingston Hyperx Fury, Corsair Vengeance}
Centre Com	0.30	{Crucial, Corsair Vengeance}
PC Force	0.53	{G.Skill Ripjaws V, Corsair Vengeance}
PC Force	0.42	{Kingston Valueram, Kingston Hyperx Fury}

Table 7: Vendor frequent itemsets

The Apriori algorithm is run with the parameter *zmin=2*, which represents the minimum of items we want in a itemset. The support is the fraction of transactions containing the itemset. The maximum is 1 when the itemset is contained in every transaction. The Table 7 reports the interesting and minimal itemsets found per vendor. The first vendor Mindfactory has an astonishing total number of 250.000 sales, while the second vendor Vuugo has 30.000 sales and the remaining two vendors Centre Com and PC Force have around 2000/3000 sales, individually. The number of sales explains why for big-sellers vendor a higher number of support is needed to find the minimal itemset. We can see that the most popular products that is contained in 3 sellers out of 4 are {Corsair Vengeance} and {Kingston Hyperx Fury}.

5.2 Association Rules Mining

In this task, the goal is to find all the association rules satisfying a minimal support and a minimal confidence. An association rule is implication expression of the form X $\rightarrow$ Y, where X and Y are disjoint itemsets. Support indicates the fraction of transaction containing both X and Y. While, confidence is how often items in Y appear in transactions containing X. In order to mine the association rules, we are once again using the Apriori algorithm.

From the analysis conducted, we noticed that the first vendor, Mindfactory, has no interesting rules. Even if we set the support and confidence values to a high threshold,

Vendor	Supp.	Conf.	Rule
Mindfactory	0.99	0.99	{Team Group Dark → Kingston Valueram, G.Skill Trident Z}
Vuugo	0.90	0.95	{Kingston → Kingston Hyperx Fury}
Vuugo	0.83	0.99	{Kingston → Kingston Hyperx Savag}
Centre Com	0.41	0.95	{Corsair Vengeance → Corsair ...}
Centre Com	0.20	0.93	{Corsair Vengeance → Kingston Hyperx Fury}
PC Force	0.97	0.97	{Kingston Valueram → Kingston Hyperx Fury}
PC Force	0.95	0.96	{Crucial → Kingston Hyperx Fury}

Table 8: Vendor association rules

the Apriori algorithm will still return a large number of rules. This is again due to the high quantity of sales that the vendor Mindfactory registers for each day. From the rules discovered from the second vendor, Vuugo, we can derive that this seller's most sold RAM brand is {Kingston} and the products of this brand are very often bought together. For the third vendor, Centre Com, most of the association rules involve {Corsair → Corsair ...}, which indicates that this vendor sells primary Corsair products. The only association rule involving a different product is {Corsair → Kingston}, which has a low support of 20%. The last vendor, PC Force, despite its small number of total sale 2000, the mined association rules present a high value of support and confidence. This indicates that this vendor mostly sell {Kingston} and {Crucial} RAM brands.

5.3 Association Rules Mining By Country

Lastly, we want to conduct an analysis comparing the association rules of different geographical regions. We decide to group by countries. The country analyzed, starting with the one recording the most total sales are Germany: 270.000, followed by USA and Australia: 220.000, France: 15.000 and Italy: 1.300. The analysis is conducted with the Apriori algorithm.

Vendor	Supp.	Conf.	Rule
Germany	1	1	{Crucial → Kingston Hyperx Fury, Corsair Vengeance}
USA	0.98	0.99	{Corsair Vengeance → Kingston, Crucial}
Australia	0.88	0.99	{Corsair Vengeance → Crucial, Kingston}
France	0.35	0.84	{Corsair Vengeance → G.Skill Trident Z, Kingston Hyperx Fury}
Italy	0.60	0.87	{Crucial → Kingston}

Table 9: Association rules by country

From the Table 9, it is noticeble that the first 4 countries all contain the same itemset with a different permutation. This tells us that the set of items {Crucial, Kingston, Corsair Vengeance} can probably be the best RAM products desired in the market in

every region. The last country, Italy, has few total number of sales but it still include a subset of the most frequent itemsets, indicating an trend agreement with the other analyzed countries.

6 Conclusion

In this work we have processed and analyzed the dataset RAM vendor. In the first section, we elaborated the data and grasped the meaning of each feature. Then we created new features to profile vendors. This profile dataset allowed us to also experiment with clustering algorithms and classification machine learning models. We found out that for our dataset, the K-Means clustering was the best. While for the classification models, none was clearly superior with respect to others. At the end, we mined and analyzed few interesting patterns and rules providing additional general understanding of the data. We believe that this whole practical and complete exercise strengthened our ability to create a fully complete pipeline for the data elaboration and analysis.

References

[1] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(10):281–305, 2012.

[2] David L. Davies and Donald W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):224–227, 1979.

[3] Martin Ester, Hans-Peter Kriegel, Jiirg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. *KDD-96 Proceedings*, 1996.

[4] Brendan J. Frey and Delbert Dueck. Clustering by passing messages between data points. 315(5814):972–976, 2007.

[5] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[6] Nadia Rahmah and Imas Sukaesih Sitanggang. Determination of optimal epsilon (eps) value on dbscan algorithm to clustering data on peatland hotspots in sumatra. *2016 IOP Conf. Ser.: Earth Environ. Sci.*

[7] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.