UNIVERSITÀ DI PISA

DATA MINING PROJECT REPORT

GROUP 24

Analysis of Marketing DataSet

Xiaoyan Li
Lanpei Li

Nov 12th, 2020

Contents

1 Data Understanding and Preparation

The dataset consists of **471910** customer purchasing records, appearing during the period from January 12th, 2010 to December 10th, 2011. This section will explain how we analyzed the data, assessed the data quality, and fixed the problems of missing values and outliers. Moreover, we also created some new indicators which help us to segment customers based on their purchasing behaviors, using RFM models.

1.1 Data semantics

Each row in the dataset contains 8 attributes, and their meanings are described as follows.

- *BasketID*: a unique identifier assigned to each transaction.
- *BasketDate*: a numerical discrete attribute which specifies the date when the transaction was made.
- *Sale*: a numerical attribute indicating the price per each unit of the product.
- *CustomerID*: a unique number to identify distinct customers.
- *CustomerCountry*: a categorical attribute, pointing out the nationality of the customer.
- *ProdID*: a unique discrete attribute given to each product.
- *ProdDescr*: some descriptions of a product.
- *Qta*: a numerical attribute pointing out the quantity of a product in a transaction.

1.2 Assessing data quality

This section checks duplicates, missing values and outliers, and treats them to avoid negative impacts they may cause on further analysis.

If the eight-column values in the dataset are the same, it is considered as duplicate data, and thus we removed the duplications.

While examining the dataset, we noticed there are missing values in the attributes CustomerID and ProdDescr. As our study focuses on the customer's purchasing behavior and product sales, we have decided to exclude records with missing CustomerID. Consequently, we also noticed that records with missing data in ProDescr were also removed simultaneously.

Meanwhile, we also found some records having zero values in the attribute Sale. After checking the prodDescr (e.g.,ROUND CAKE TIN VINTAGE GREEN) of the related product, we think these zeros are missing values, and therefore we handle them by replacing with the mean price over groups of products with the same ProdID.

Regarding outliers, we discovered that there are amount of records having quite large 'Sale' and'Qta' values (e.g., Sale: 38970, Qta: 90995), compared to the majority of the records that are below 4 £ and 12 units in the dataset. We could not label all of them as outliers as we thought this behavior is somewhat expected, and they are genuine transaction records anyway. Nevertheless, considering some clustering algorithms can be very sensitive to outliers' presence, we decided only to drop the extreme outliers that fall outside of the outer fences, and keep the mild ones. During this process, we used the **interquartile range** to create our outlier fences, in which lower outer fence and upper outer fence can be determined from the rules $Q1 - 3IQ$ and $Q3 + 3IQ$ respectively.

It is worth noting that attributes BasketID and Qta include some particular values, i.e., BasketID has values starts from the character 'C', and Qta has negative quantities. They should not be marked as outliers since they may represent cancelled orders and the corresponding quantity of returned products. As we can see in Figure 1, the scatter plot of records whose BasketID starts with 'C' and records having negative values in 'Qta' are identical. Therefore, we treat them as normal records for further usages.

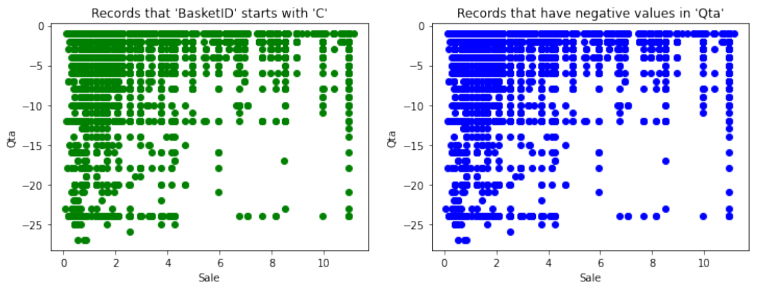


Figure 1: Scatterplot for Sale and Qta

1.3 Distribution of the variables and statistics

The plot in Figure 2 and Figure 3 shows the distributions and statistics of the attribute Sale and Qta. We can see that the distribution of 'Sale' is right-skewed, meaning products of lower prices (below 2 £) populates more than products of higher prices. Regarding the attribute Qta, most of the records in the dataset reside in the range between 0 and 12. Besides, the attribute Qta and Sale are negative weakly correlated since their correlation is -0.32573.

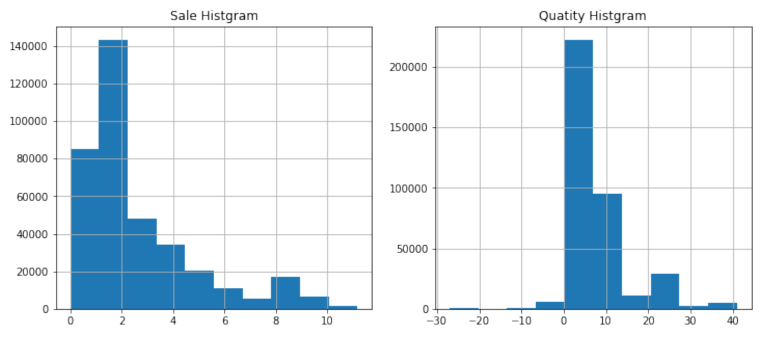
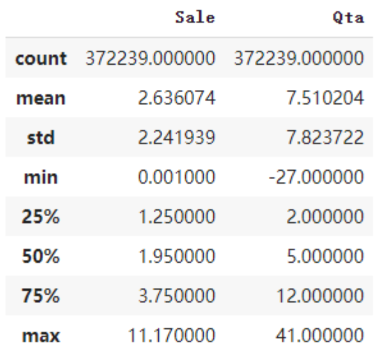


Figure 2: Histgrams for Sale and Qta



	Sale	Qta
count	372239.000000	372239.000000
mean	2.636074	7.510204
std	2.241939	7.823722
min	0.001000	-27.000000
25%	1.250000	2.000000
50%	1.950000	5.000000
75%	3.750000	12.000000
max	11.170000	41.000000

Figure 3: Statistics for Sale and Qta

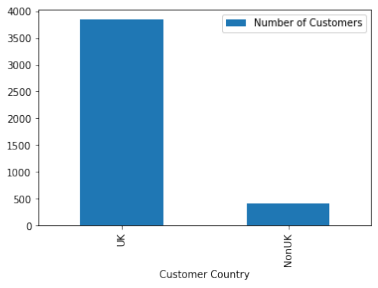


Figure 4: Customer Country

Another observation is that UK customers are much more than customers of Non-UK countries, as indicated in Figure 4.

1.4 Variables transformations and generation

First, we split 'BasketDate' into three features: 'Date', 'Year' and 'Month', to simplify their manipulations for further analysis. We also noticed the dataset is spanned more than one year, in order to understand better and analyze the dataset, we decided to focus on a full year transaction from 2010-11-30 to 2011-12-01. Also considering the Non-UK customers are the minority, we opted to limit our data to UK customers. Besides, we defined a new indicator 'Amount' by multiplying the value in the variable 'Sale' and 'Qta' for each record.

Aiming at characterizing customers' purchasing behavior in a more-depth level, we created a profile for each customer on the basis of the RFM model. RFM stands for Recency, Frequency and Monetary value. RFM analysis is a marketing technique used for identifying groups of customers to provide future personalization services.

Calculation of RFM indicators:

1. *Recency*: it is defined as time in days since the last purchases.
2. *First_Purchase*: it is defined as time in days since the first purchase.
3. *Frequency*: it is defined as the total number of purchase made during the period of observation.
4. *Monetary*: it is defined as the total amount spent by the customer within the year in analysis.

Finally we obtain a new data-frame which is composed of those four indicators and contains 3750 records.

1.5 Preliminary observations

Before proceeding further with our data mining analysis, we made a few initial observations about the dataset:

- the monthly sales and transactions are plotted as in Figure 5.

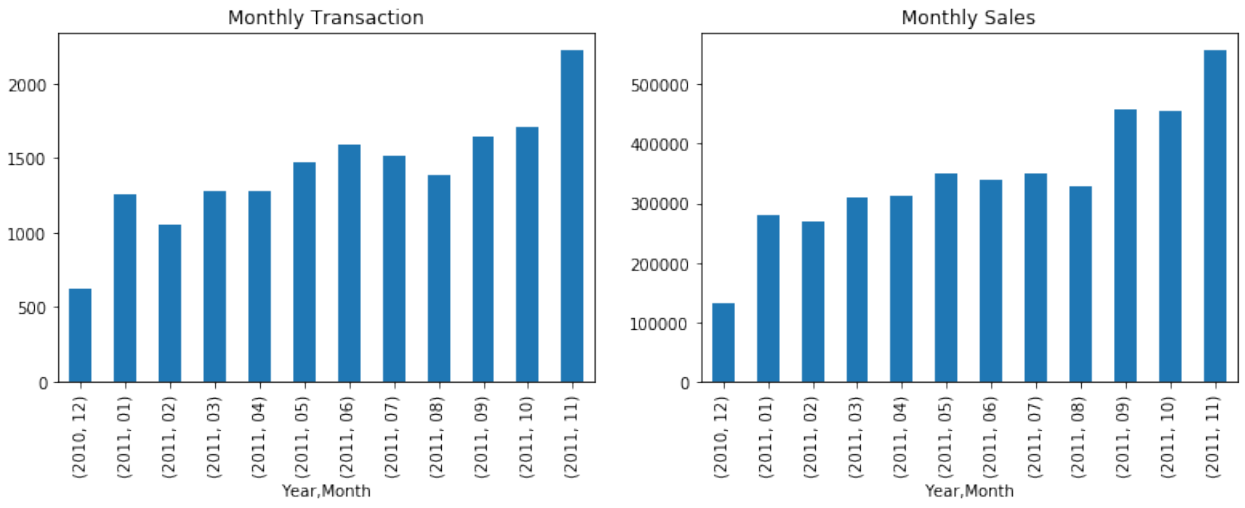


Figure 5: Monthly transactions and sales

  The calculation results show that the orders and sales are increasing each month in general, implying the business seems to be relatively healthy. There is an exception in August, which may be resulted from a summer holiday.

- Figure 6 shows the weekly sales and transactions.

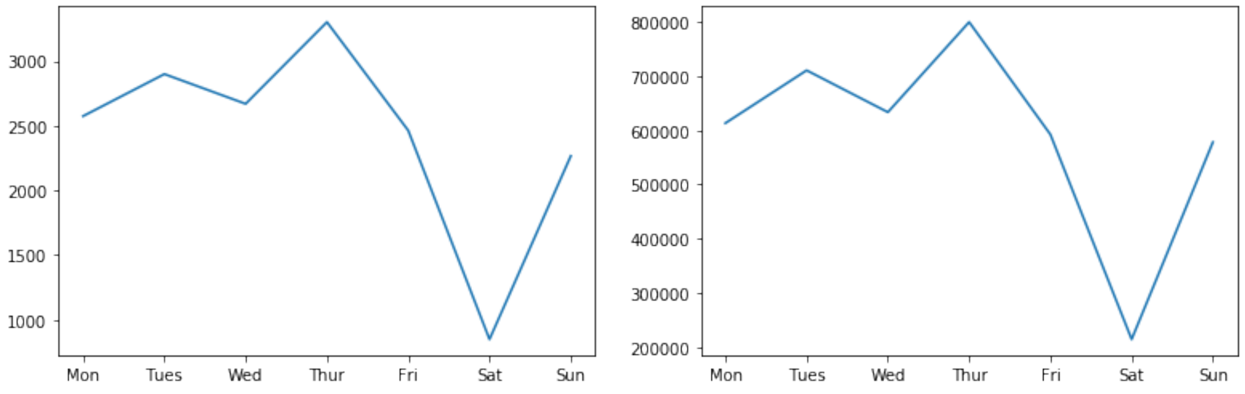


Figure 6: Weekly transactions and sales

  As we can see, the number of orders and the sales made was the highest on Thursday. While on the other hand, orders and sales on weekends are the lowest. It is suspected that there are company buyers who do not work on weekends.

- The number of items per each transaction is 132.3 units. This may suggest many of the customers were organizational customers rather than individual buyers.

- The product which got returned most frequently is ID22423. According to its product description, it is REGENCY CAKESTAND 3 TIER; thus the retailer should check the quality of this product.

1.6 Exploring the new features for a statistical analysis

1. Distribution of the variables and statistics, as shown in Figure 7 and Figure 8

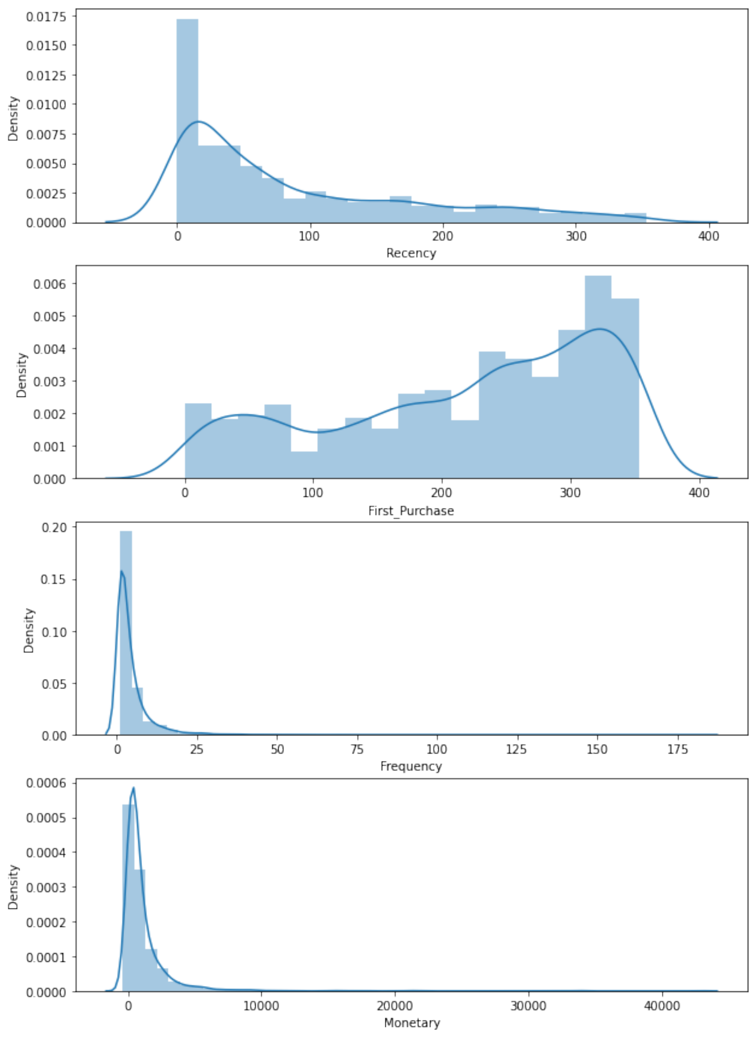
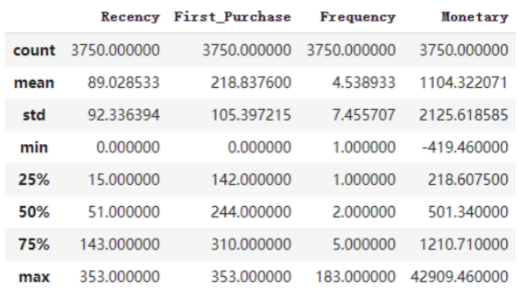


Figure 7: Distribution of Recency, First_Purchase, Frequency and Monetary



	Recency	First_Purchase	Frequency	Monetary
count	3750.000000	3750.000000	3750.000000	3750.000000
mean	89.028533	218.837600	4.538933	1104.322071
std	92.336394	105.397215	7.455707	2125.618585
min	0.000000	0.000000	1.000000	-419.460000
25%	15.000000	142.000000	1.000000	218.607500
50%	51.000000	244.000000	2.000000	501.340000
75%	143.000000	310.000000	5.000000	1210.710000
max	353.000000	353.000000	183.000000	42909.460000

Figure 8: Statistics of Recency, First_Purchase, Frequency and Monetary

We can see that:

- *Recency* is right skewed, meaning most customers went shopping recently(w.r.t the observation period).
- *First_Purchase* is left skewed, indicating most of customers are already regular clients. It is also worth to mention that 1000 new visitors emerged in the last two months of the observation period.
- *Frequency* is right skewed. It suggests that most customers went shopping within five times, but a decent number of customers shopped more often than that.
- *Monetary* is also right skewed. Although half of the customers only spent around 500 £ during the period of observation, the average amount is 1100 £, which seems to imply there are many heavy buyers in the dataset.

2. Handling outliers: similar to what we have discussed previously, we only sought to eliminate extreme outliers to keep more information about customers. To do so, we exploited the boxplot in Figure 9.

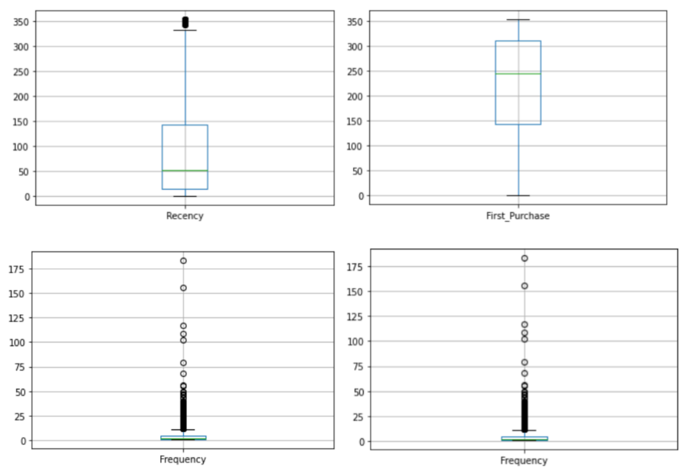


Figure 9: Boxplot of Recency, First_Purchase, Frequency and Monetary

3. Pairwise Correlations.

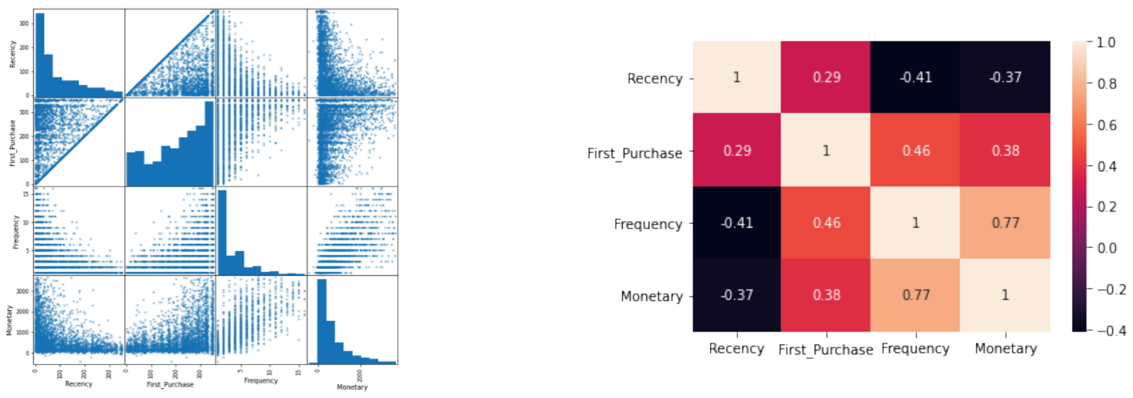


Figure 10: Correlations of attributes

The analysis of correlation is shown in Figure 10. It reveals a good dependency of the variable *First_Purchase* and *Frequency*, which can be explained in the way that since customers made their first purchase, they became loyal customers and shopped more often. Additionally, *Frequency* and *Monetary* are strongly correlated (0.77). However, we decided not to cut either of them out based on the rule of thumb.

Referring to the rule of thumb: if the correlation > 0.8, then severe multicollinearity may be present, and thus redundant variables should be discarded.

2 Clustering Analysis

In this section, we applied four different clustering algorithms (K-means, DBScan, Hierarchical, and K-Medoids) to the dataset to form groups of customers, and discussed their results. In preparation for this clustering task, we first normalized our data with the MinMax scaler.

2.1 K-means

1. Choice of attributes and distance function.

   As presented in the last section, we used four attributes *(First_purchase, Recency, Frequency and Monetary)* related to the RFM model to segment customers. Since our data are all discrete and numerical values, we adopted the Euclidean distance as our distance function.

2. Identification of the best value of k.

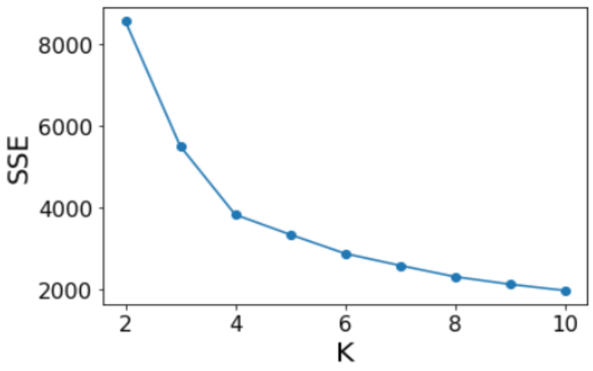


Figure 11: Selection of the best K

To find out the appropriate number of clusters (K), we plotted the corresponding SSE for $K \in [2, 10]$, shown in Figure 11. Using the Elbow method, K=3 is selected as the optimal one (the point where elbow bend locates).

3. Characterization of the obtained clusters

   - Analysis of the k centroids
     After running the algorithm, clusters we acquired can be visualized in Figure 12.

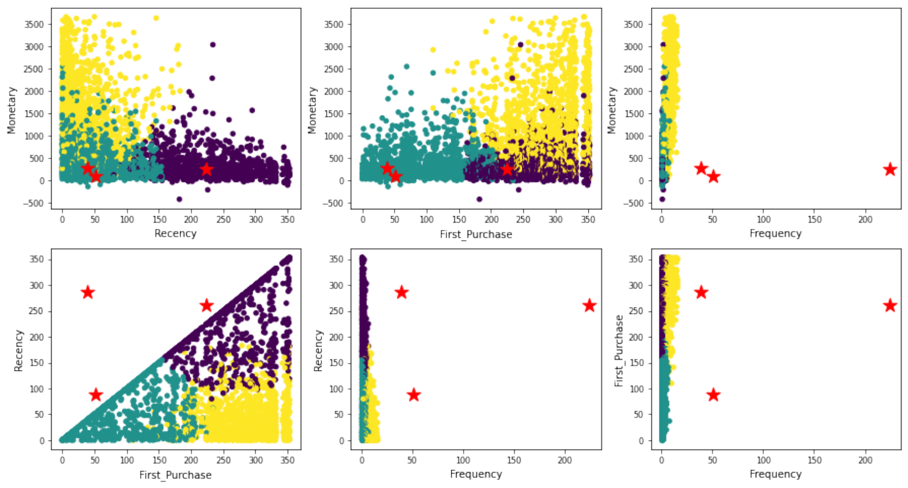


Figure 12: Clusters generated by K-means

While the cluster centers are plotted in Figure 13.

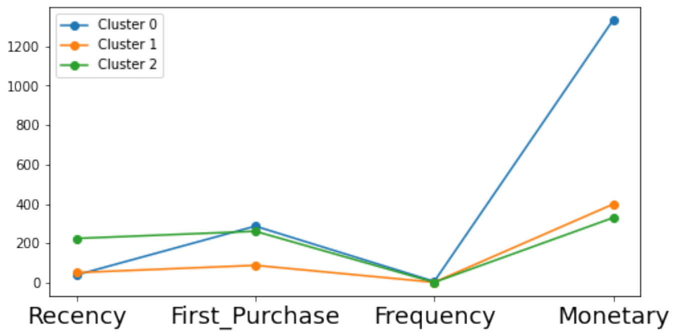


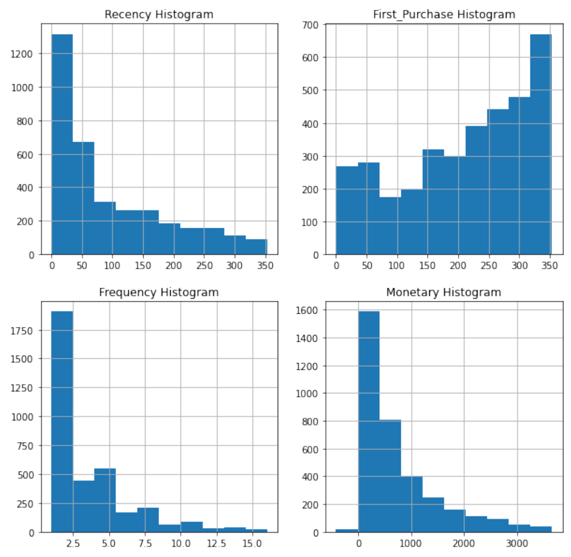
Figure 13: Visualization of Cluster Centers by Parallel Coordinates

     – **Cluster 0** (centroid:[38.85513673, 286.76053215, 5.83961567, 1332.92658025]) has 1353 customers, and it is the largest cluster. It also has the largest value of *Monetary*,*First_purchase*, and *Frequency*, which suggests customers in this group kept shopping throughout the whole observed year, and also shop very frequently. Therefore, this group should be considered as the most profitable group.

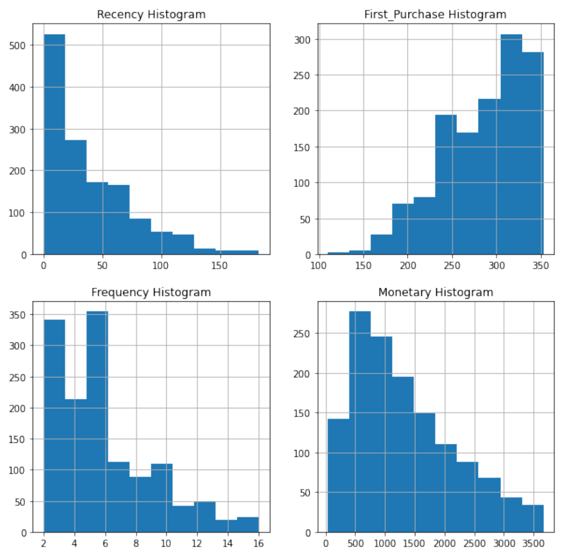
  - **Cluster 1** (centroid:[224.26577042, 260.67114788, 1.6401241 , 329.32842064]) relates to 967 customers, and it is the smallest one among the three. This group of customers has the largest number of *Recency*, suggesting they have not shopped for a longer time than others, so it is better to think of some promotion strategies to attract them back.
  - **Cluster 2**(centroid:[50.88255034, 87.68959732, 1.79614094, 398.16080705]) contains 1192 customers. Customers in this group have the lowest value of *Recency* and *First_Purchase*, implying that they are new registered customers, and they continued to shop to the end of the period in observation. This group of people has the potential to become highly profitable in the future.

- Comparison of the distribution of variables within the clusters and that in the whole dataset.

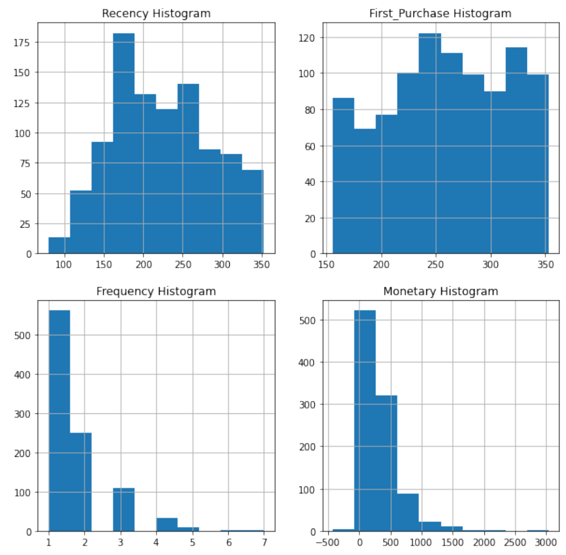
  The distributions of RMF attributes of the original dataset (after the treatment of missing values and outliers), Cluster 0, Cluster 1, and Cluster 2 are displayed in Figure 14, ordered from left to right, from top to bottom.



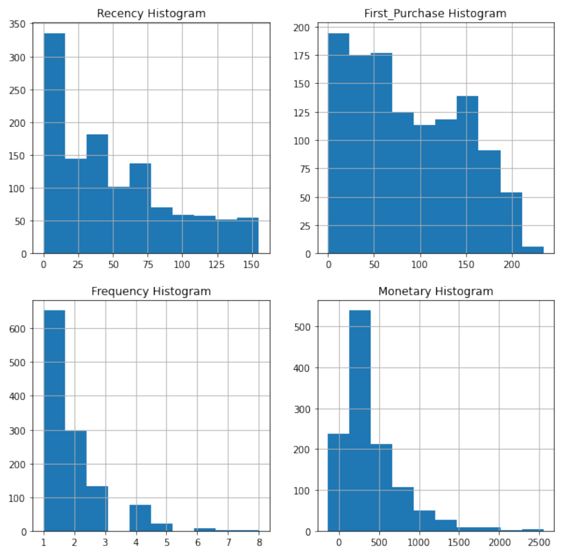
Whole dataset



Cluster 0



Cluster 1



Cluster 2

Figure 14: distributions of the whole dataset, Cluster 0, Cluster 1, and Cluster 2

As it has been shown:

  - **Cluster 0**: *First_Purchase* looks more left skewed than the whole dataset. On the opposite, the skewness of *Frequency*, *Recency* and *Monetary* are reduced, although still remains right skewed.

  - **Cluster 1**: the distribution of *Recency* and*First_Purchase* became fairly symmetric, while distributions of *Frequency* and *Monetary* preserve the shape of the whole dataset with a bit of deformation due to selection of subsets of the ranges of the attributes values.
  - **Cluster 2**: the distribution of *First_Purchase* has changed a lot, switching from heavy right skewed to a very uniform shape. The others almost maintain the skewness of the whole dataset.

4. Evaluation of the clustering results:

   Silhouette: 0.4440431649824081
   SSE:285.1569139364326
   Separation:0.8312601408052895

2.2 Density based clustering

A low minPts means it builds more clusters from outliers; hence we should not choose a too-small value for it. One heuristic approach for deciding it is using $2 \times dimension$, where $dimension = 4$ here, and thus we set minPts $= 8$.

1. Choice of attributes and distance function. Same as in the analysis by K-means clustering algorithm.

2. Knee Method to estimate the best eps with k-distance plot, indicated in Figure 15:

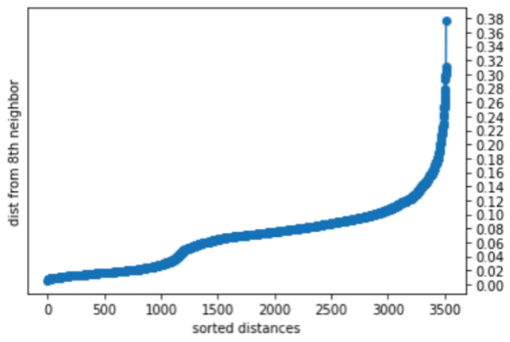


Figure 15: Searching the best eps

   The optimal value for epsilon is located at the point of maximum curvature. Therefore, we picked eps= 0.15 here.

3. Characterization and interpretation of the obtained clusters. After running this algorithm, we ended up with only one cluster (with 3464 records), while the other set is simply noisy data (with 48 records). This is not surprising to us, because in general DBScan does not perform well on high-density data like what we have here. However, it may serve very well as a way to identify outliers, demonstrated in Figure 16.

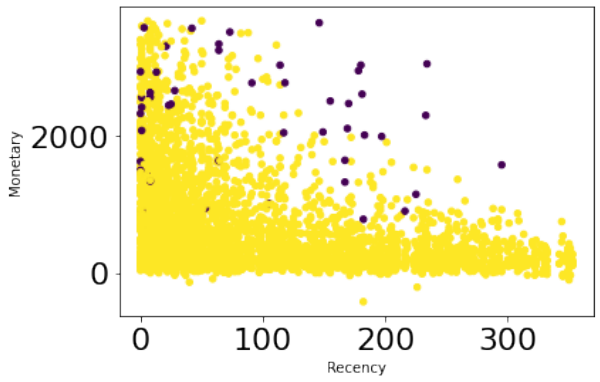


Figure 16: Clusters plotted with Recency and Monetary

2.3 Hierarchical clustering

1. Choice of attributes and distance function. Same vein as in the K-means algorithm.

2. Applying different algorithms and discussions about the derived dendrograms.

   We tested different connection criteria (single, complete, average, ward), and cut heights by virtue of the number of clusters (3) drawn from the k-means algorithm. To better understand which algorithm achieves the best result, we also computed each algorithm's obtained silhouette score. Figure 17 shows the results and dendrograms we obtained.

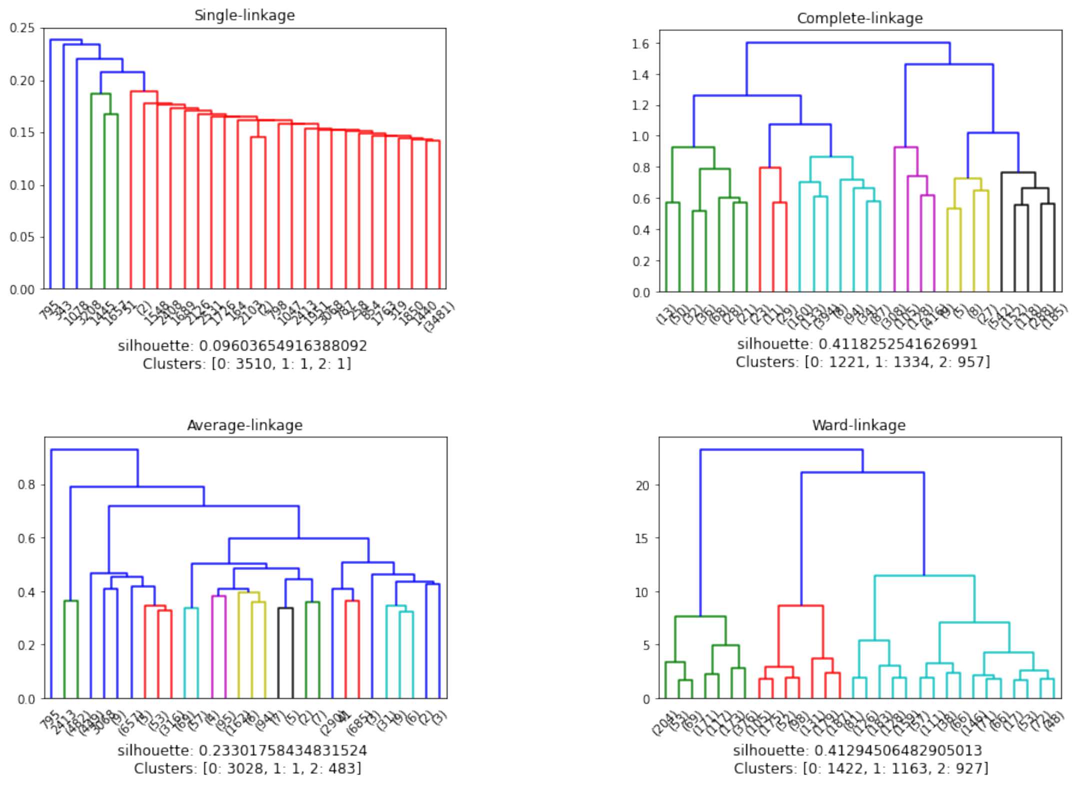


Figure 17: dendrograms produced by different algorithms

   As the result indicates, the methods 'complete-linkage' and 'ward-linkage' lead to a better result, which enjoys a higher silhouette score and more balanced groups.

2.4 K-Medoids (optional task)

This section explored the Pyclustering tool, and tried to apply another clustering algorithm — K-medoids. K-medoids is a partitioning clustering algorithm that uses the medoids instead of centers like K-Means algorithm, where a medoid is an object with the smallest dissimilarity to all others in the cluster.

Both k-means and k-medoids are partitional algorithms. K-means attempts to minimize the total squared error. In contrast, k-medoids minimizes the sum of dissimilarities between points labelled to be in a cluster and a point designated as the center of that cluster. What's more, unlike the K-means clustering algorithm, which is rather sensitive to outliers, K-medoids is much more robust thanks to the fact that it utilizes an actual point in the cluster to represent the center.

Following the result of the k-means algorithm, we used 3 clusters. The same logic applied to the distance function and chosen attributes.

After running the algorithm, we obtained three clusters illustrated in Figure 18, in which x_0, x_1, x_2, x_3 represents *Recency, First_Purchase, Frequency, Monetary* respectively. Figure 19 shows the the cluster medoids.

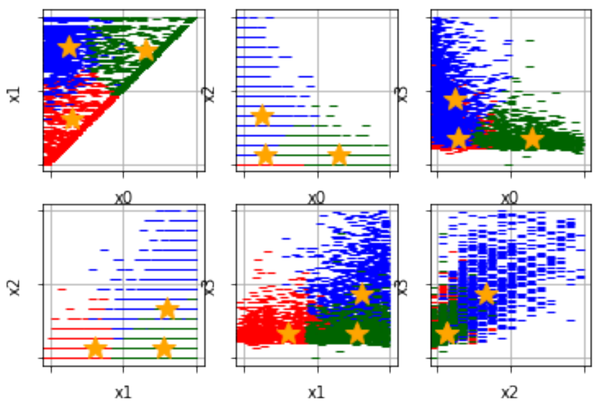


Figure 18: Cluster obtained by k-medoids

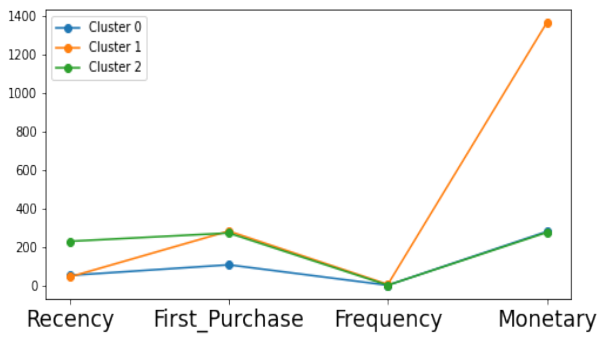


Figure 19: Visualization of Cluster Medoids by Parallel Coordinates

Similar to the k-means algorithms, we obtained groups with quite balanced numbers:0: 1288, 1: 1310, 2: 914. Moreover, the silhouette score is 0.6188306624686812.

Discussions about characteristics of each cluster and distributes are analogous to the logic of k-means; thus we omitted them here.

2.5 Final evaluation of the best clustering approach and comparison of the clustering obtained

In conclusion, the best clustering results were obtained by the k-medoids approach. It gains the best silhouette score(0.6188306624686812), and provides us with groups of most balanced size. Next to it, also being a partitioning clustering algorithm, k-means offers similar clusters with a silhouette score of 0.4440431649824081.

On the other hand, the hierarchical clustering algorithm (Single linkage and complete linkage) gives the most intuitive and reasonable dendrograms from the dataset domain.

DBScan clustering algorithm performs the worst among the four clustering algorithms. However, it can be used to single out outliers of a dataset.

3 Classification

This section uses different models to predict if a customer is a high-spending customer, medium-spending customer or low-spending customer. Section 3.1 starts with labelling customers as three-level of spenders(i.e., high, medium, low), followed by the definition of a customer profile which enables the prediction task. In Section 3.2, we applied various classification algorithms on the dataset that we just obtained, and then report the results. Section 3.3 compares the performance of different algorithms and discusses the results.

3.1 Data preparation

Labelling customers: In order to split customers into three categories (high/medium/low spender), we divided them into three equal-sized bins, based on how much money a customer spends during the period of his first purchase date and his last purchase date. Formally it can be defined as

$$Monetary/(First\_Purchase - Recency + 1)$$

The definition of customer profile: Considering indicators *Monetary*, *First_Purchase*, *Recency* were used directly to define our prediction target, we should not use them as indicators for the prediction task any more(at least not all of them). Also indicators defined for the clustering task did not take into consideration of the temporal constraints, so we decided to redefine another set of indicators of the customer profile.

1. *I*: the number of items purchased by a customer per day during the time span between his first purchase and last purchase.
2. *Iu*: the number of distinct items bought by a customer.
3. *Imax*: the maximum number of items purchased by a customer during a shopping session.
4. *E*: the Shannon entropy on the purchasing behavior of a customer, i.e, the temporal varieties of his shopping sessions.
5. *Frequency*: the number of shopping sessions made by a customer per day during his whole shopping timespan.

3.2 Performing prediction using various different classification algorithms

Before starting, we partitioned the dataset into training set (75% of customers) and test set (25% of customers). To discover the optimal value of each classification algorithm's hyperparameters, we gridsearched a group of settings, shown in Table 1. Note in the neural network and KNN models, we applied normalization on the data, more detailed discussions are delayed to Section 3.3. Apart from this, one-hot encoding is used to convert the categorical data (customer's label: low, high, medium) to a numerical form to fit the neural network, and One-vs-Rest is used to handle the multi-class classification tasks.

The architecture of the neural network can be depicted as Figure 20.

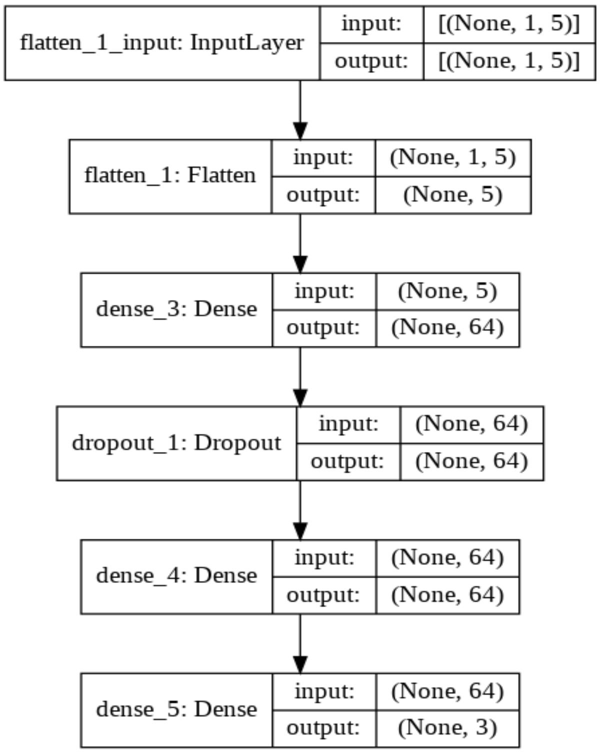


Figure 20: Visualization of Neural Network

Algorithm	Hyper-parameters
Decision Tree	*criterion* : 'gini', 'entropy' *max_depth* : 2, 5, 10, 15, None *min_samples_split*: 2, 5, 10, 20 *min_samples_leaf* : 2, 5, 10, 20
Naive Bayes(Gaussian)	None
Random Forest	*n_estimators*: 25, 50, 100, 200, 500, 1000 *max_depth*: 2,3,5,6,7,10,12,None *max_features*: sp_randint(1,6) *min_samples_split*: sp_randint(10, 51) *min_samples_leaf*: sp_randint(10, 51) *bootstrap*:True, False *criterion*:'entropy', 'gini' *class_weight*:'balanced', None, $\{0 : 0.2, 1 : 0.3, 2 : 0.5\}, \{0 : 0.1, 1 : 0.1, 2 : 0.8\}$
SVM	C: 1.0, 10000.0, 100000.0 *gamma*: 'scale', 'auto'
Rule based(RIPPER)	*prune_size*: 0.5, 0.6, 0.7 k : 1, 3, 5
KNN	*n_neighbors* : 2, 5, 10, 20 *weights*: 'uniform', 'distance'
Neural network	*optimizers* : 'Adam', 'Adamax' *init* : 'normal', 'uniform' *batches*: 32, 64

Table 1: Setup environment of the tested hyperparameters

To search the best combination of hyperparameters, we performed 5-fold cross validation per each group of hyperparameters. The best settings we found are reported in Table 2 As the best groups of hyperparameters are identified, we then train our classification models with those optimal hyperparameters using the whole training set. After that, we verify our model on the test set. Results are illustrated in Table 3 and Table 4, where 0 denotes low spending customer, 1 denotes medium spending customers, and 2 denotes high spending customers.

Algorithm	Hyper-parameters
Decision Tree	*criterion*: 'gini' *max_depth*:5 *min_samples_split*: 20 *min_samples_leaf* : 2
Naive Bayes(Gaussian)	None
Random Forest	*n_estimators*:1000 *max_depth*:6 *max_features*:2 *min_samples_split*:39 *min_samples_leaf*:14 *bootstrap*: False *criterion*:'gini' *class_weight*:None
SVM	C:10000.0 *gamma*:'scale'
Rule based(RIPPER)	*prune_size*: 0.7 k : 5
KNN	*n_neighbors* : 10 *weights*: 'distance'
Neural network	*optimizers* : 'Adam' *init* : 'normal' *batches*: 32

Table 2: The optimal hyperparameters

	Accuracy	Precision				Recall				F1-Score			
		0	1	2	*AVG*	0	1	2	*AVG*	0	1	2	*AVG*
Decision Tree	**0.93**	0.93	0.87	0.98	**0.93**	0.89	0.92	0.98	**0.93**	0.91	0.89	0.98	**0.93**
Naive Bayes	**0.81**	0.69	0.84	0.93	**0.82**	0.89	0.55	0.98	**0.81**	0.78	0.67	0.95	**0.80**
Random Forest	**0.93**	0.92	0.90	0.97	**0.93**	0.91	0.90	0.99	**0.93**	0.92	0.90	0.98	**0.93**
SVM	**0.92**	0.89	0.91	0.98	**0.92**	0.93	0.86	0.98	**0.92**	0.91	0.88	0.98	**0.92**
Rule based	**0.72**	0.89	0.80	0.60	**0.76**	0.65	0.59	0.92	**0.72**	0.75	0.68	0.73	**0.72**
KNN	**1.00**	1.00	0.99	1.00	**1.00**	1.00	1.00	0.99	**1.00**	1.00	1.00	1.00	**1.00**
Neural Network	**0.90**	0.91	0.84	0.97	**0.91**	0.85	0.88	0.98	**0.91**	0.88	0.86	0.98	**0.91**

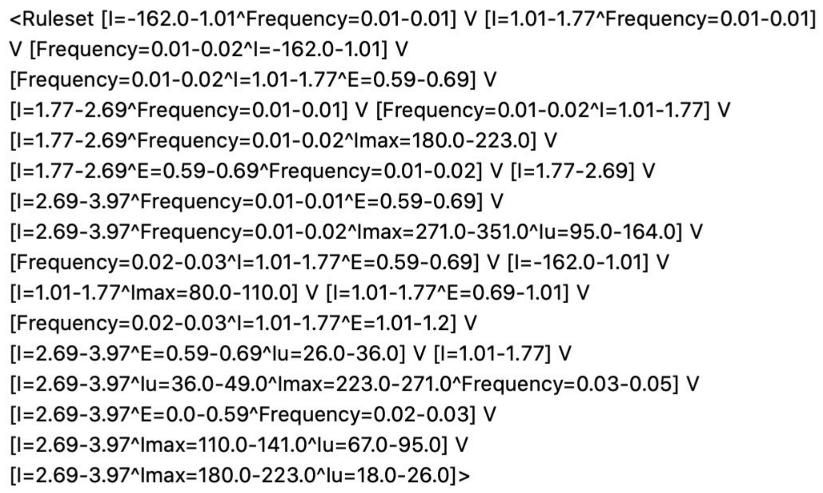
Table 3: Results over the training set

	Accuracy	Precision				Recall				F1-Score			
		0	1	2	*AVG*	0	1	2	*AVG*	0	1	2	*AVG*
Decision Tree	**0.93**	0.92	0.88	0.97	**0.93**	0.91	0.89	0.97	**0.93**	0.92	0.89	0.97	**0.93**
Naive Bayes	**0.81**	0.71	0.84	0.91	**0.82**	0.88	0.56	0.99	**0.81**	0.78	0.68	0.95	**0.80**
Random Forest	**0.91**	0.89	0.87	0.95	**0.90**	0.89	0.84	0.99	**0.91**	0.89	0.86	0.97	**0.91**
SVM	**0.90**	0.88	0.87	0.94	**0.90**	0.89	0.82	0.98	**0.90**	0.98	0.85	0.96	**0.90**
Rule based	**0.78**	0.89	0.75	0.72	**0.79**	0.77	0.73	0.83	**0.78**	0.83	0.74	0.77	**0.78**
KNN	**0.83**	0.75	0.79	0.96	**0.84**	0.84	0.69	0.97	**0.83**	0.80	0.74	0.97	**0.83**
Neural Network	**0.90**	0.89	0.84	0.96	**0.90**	0.85	0.85	0.99	**0.90**	0.87	0.85	0.98	**0.90**

Table 4: Results over the test set

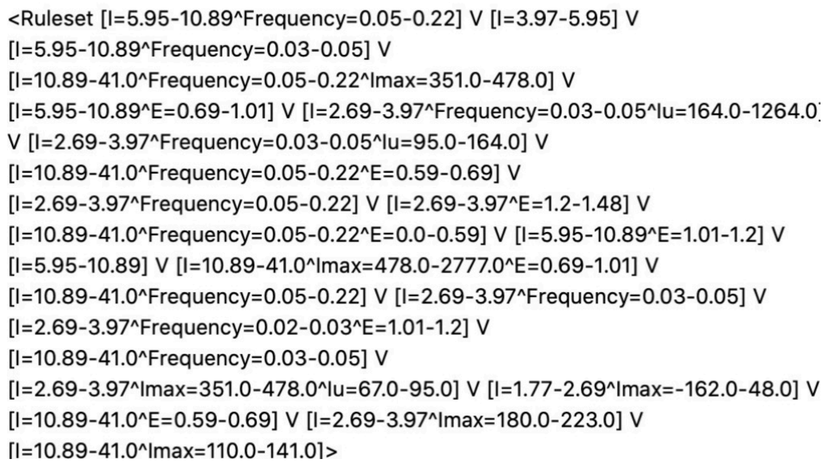
3.3 Model comparisons

Interpretability: Among all the models presented, decision tree, random forest, rule based models are able to provide us with some explanation of their predications: predications can be traced clearly from the decision tree (as in Figure 22) or rules (as in Figure 21) generated by rule based model. In addition, they can reveal the importance of features in making the decision as in Figure 22 and Figure 23.



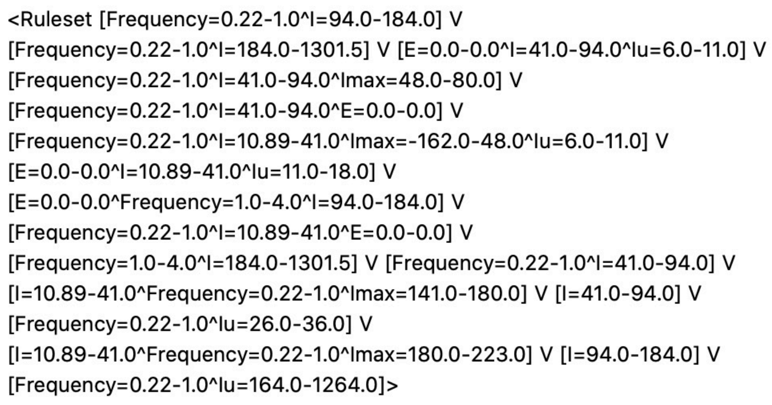
```
<Ruleset [I=-162.0-1.01^Frequency=0.01-0.01] V [I=1.01-1.77^Frequency=0.01-0.01]
V [Frequency=0.01-0.02^I=-162.0-1.01] V
[Frequency=0.01-0.02^I=1.01-1.77^E=0.59-0.69] V
[I=1.77-2.69^Frequency=0.01-0.01] V [Frequency=0.01-0.02^I=1.01-1.77] V
[I=1.77-2.69^Frequency=0.01-0.02^Imax=180.0-223.0] V
[I=1.77-2.69^E=0.59-0.69^Frequency=0.01-0.02] V [I=1.77-2.69] V
[I=2.69-3.97^Frequency=0.01-0.01^E=0.59-0.69] V
[I=2.69-3.97^Frequency=0.01-0.02^Imax=271.0-351.0^Iu=95.0-164.0] V
[Frequency=0.02-0.03^I=1.01-1.77^E=0.59-0.69] V [I=-162.0-1.01] V
[I=1.01-1.77^Imax=80.0-110.0] V [I=1.01-1.77^E=0.69-1.01] V
[Frequency=0.02-0.03^I=1.01-1.77^E=1.01-1.2] V
[I=2.69-3.97^E=0.59-0.69^Iu=26.0-36.0] V [I=1.01-1.77] V
[I=2.69-3.97^Iu=36.0-49.0^Imax=223.0-271.0^Frequency=0.03-0.05] V
[I=2.69-3.97^E=0.0-0.59^Frequency=0.02-0.03] V
[I=2.69-3.97^Imax=110.0-141.0^Iu=67.0-95.0] V
[I=2.69-3.97^Imax=180.0-223.0^Iu=18.0-26.0]>
```

Ruleset of low spending customers



```
<Ruleset [I=5.95-10.89^Frequency=0.05-0.22] V [I=3.97-5.95] V
[I=5.95-10.89^Frequency=0.03-0.05] V
[I=10.89-41.0^Frequency=0.05-0.22^Imax=351.0-478.0] V
[I=5.95-10.89^E=0.69-1.01] V [I=2.69-3.97^Frequency=0.03-0.05^Iu=164.0-1264.0
V [I=2.69-3.97^Frequency=0.03-0.05^Iu=95.0-164.0] V
[I=10.89-41.0^Frequency=0.05-0.22^E=0.59-0.69] V
[I=2.69-3.97^Frequency=0.05-0.22] V [I=2.69-3.97^E=1.2-1.48] V
[I=10.89-41.0^Frequency=0.05-0.22^E=0.0-0.59] V [I=5.95-10.89^E=1.01-1.2] V
[I=5.95-10.89] V [I=10.89-41.0^Imax=478.0-2777.0^E=0.69-1.01] V
[I=10.89-41.0^Frequency=0.05-0.22] V [I=2.69-3.97^Frequency=0.03-0.05] V
[I=2.69-3.97^Frequency=0.02-0.03^E=1.01-1.2] V
[I=10.89-41.0^Frequency=0.03-0.05] V
[I=2.69-3.97^Imax=351.0-478.0^Iu=67.0-95.0] V [I=1.77-2.69^Imax=-162.0-48.0] V
[I=10.89-41.0^E=0.59-0.69] V [I=2.69-3.97^Imax=180.0-223.0] V
[I=10.89-41.0^Imax=110.0-141.0]>
```

Ruleset of medium spending customers



```
<Ruleset [Frequency=0.22-1.0^I=94.0-184.0] V
[Frequency=0.22-1.0^I=184.0-1301.5] V [E=0.0-0.0^I=41.0-94.0^Iu=6.0-11.0] V
[Frequency=0.22-1.0^I=41.0-94.0^Imax=48.0-80.0] V
[Frequency=0.22-1.0^I=41.0-94.0^E=0.0-0.0] V
[Frequency=0.22-1.0^I=10.89-41.0^Imax=-162.0-48.0^Iu=6.0-11.0] V
[E=0.0-0.0^I=10.89-41.0^Iu=11.0-18.0] V
[E=0.0-0.0^Frequency=1.0-4.0^I=94.0-184.0] V
[Frequency=0.22-1.0^I=10.89-41.0^E=0.0-0.0] V
[Frequency=1.0-4.0^I=184.0-1301.5] V [Frequency=0.22-1.0^I=41.0-94.0] V
[I=10.89-41.0^Frequency=0.22-1.0^Imax=141.0-180.0] V [I=41.0-94.0] V
[Frequency=0.22-1.0^Iu=26.0-36.0] V
[I=10.89-41.0^Frequency=0.22-1.0^Imax=180.0-223.0] V [I=94.0-184.0] V
[Frequency=0.22-1.0^Iu=164.0-1264.0]>
```

Ruleset of high spending customers

Figure 21: Ruleset generated by rule based algorithm

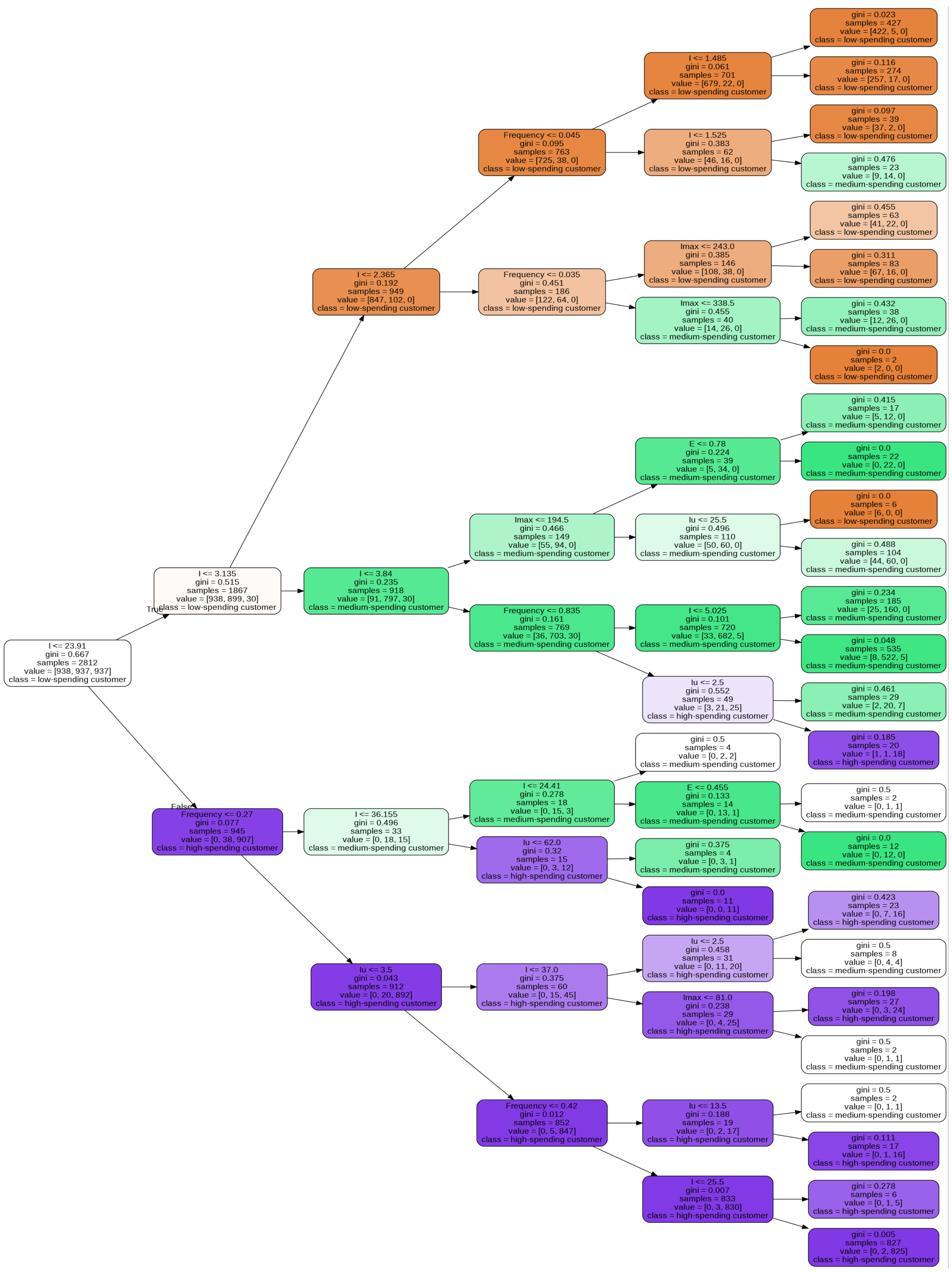

I <= 23.91 gini = 0.667 samples = 2812 value = [938, 937, 937] class = low-spending customer
True
False
I <= 3.135 gini = 0.515 samples = 1867 value = [938, 899, 30] class = low-spending customer
Frequency <= 0.27 gini = 0.077 samples = 945 value = [0, 38, 907] class = high-spending customer
I <= 2.365 gini = 0.192 samples = 949 value = [847, 102, 0] class = low-spending customer
I <= 3.84 gini = 0.235 samples = 918 value = [91, 797, 30] class = medium-spending customer
I <= 36.155 gini = 0.496 samples = 33 value = [0, 18, 15] class = medium-spending customer
Iu <= 3.5 gini = 0.043 samples = 912 value = [0, 20, 892] class = high-spending customer
Frequency <= 0.045 gini = 0.095 samples = 763 value = [725, 38, 0] class = low-spending customer
Frequency <= 0.035 gini = 0.451 samples = 186 value = [122, 64, 0] class = low-spending customer
Imax <= 194.5 gini = 0.466 samples = 149 value = [55, 94, 0] class = medium-spending customer
Frequency <= 0.835 gini = 0.161 samples = 769 value = [36, 703, 30] class = medium-spending customer
I <= 24.41 gini = 0.278 samples = 18 value = [0, 15, 3] class = medium-spending customer
Iu <= 62.0 gini = 0.32 samples = 15 value = [0, 3, 12] class = high-spending customer
I <= 37.0 gini = 0.375 samples = 60 value = [0, 15, 45] class = high-spending customer
Frequency <= 0.42 gini = 0.012 samples = 852 value = [0, 5, 847] class = high-spending customer
I <= 1.485 gini = 0.061 samples = 701 value = [679, 22, 0] class = low-spending customer
I <= 1.525 gini = 0.383 samples = 62 value = [46, 16, 0] class = low-spending customer
Imax <= 243.0 gini = 0.385 samples = 146 value = [108, 38, 0] class = low-spending customer
Imax <= 338.5 gini = 0.455 samples = 40 value = [14, 26, 0] class = medium-spending customer
E <= 0.78 gini = 0.224 samples = 39 value = [5, 34, 0] class = medium-spending customer
Iu <= 25.5 gini = 0.496 samples = 110 value = [50, 60, 0] class = medium-spending customer
I <= 5.025 gini = 0.101 samples = 720 value = [33, 682, 5] class = medium-spending customer
Iu <= 2.5 gini = 0.552 samples = 49 value = [3, 21, 25] class = high-spending customer
gini = 0.5 samples = 4 value = [0, 2, 2] class = medium-spending customer
E <= 0.455 gini = 0.133 samples = 14 value = [0, 13, 1] class = medium-spending customer
gini = 0.375 samples = 4 value = [0, 3, 1] class = medium-spending customer
gini = 0.0 samples = 11 value = [0, 0, 11] class = high-spending customer
Iu <= 2.5 gini = 0.458 samples = 31 value = [0, 11, 20] class = high-spending customer
Imax <= 81.0 gini = 0.238 samples = 29 value = [0, 4, 25] class = high-spending customer
Iu <= 13.5 gini = 0.188 samples = 19 value = [0, 2, 17] class = high-spending customer
I <= 25.5 gini = 0.007 samples = 833 value = [0, 3, 830] class = high-spending customer
gini = 0.023 samples = 427 value = [422, 5, 0] class = low-spending customer
gini = 0.116 samples = 274 value = [257, 17, 0] class = low-spending customer
gini = 0.097 samples = 39 value = [37, 2, 0] class = low-spending customer
gini = 0.476 samples = 23 value = [9, 14, 0] class = medium-spending customer
gini = 0.455 samples = 63 value = [41, 22, 0] class = low-spending customer
gini = 0.311 samples = 83 value = [67, 16, 0] class = low-spending customer
gini = 0.432 samples = 38 value = [12, 26, 0] class = medium-spending customer
gini = 0.0 samples = 2 value = [2, 0, 0] class = low-spending customer
gini = 0.415 samples = 17 value = [5, 12, 0] class = medium-spending customer
gini = 0.0 samples = 22 value = [0, 22, 0] class = medium-spending customer
gini = 0.0 samples = 6 value = [6, 0, 0] class = low-spending customer
gini = 0.488 samples = 104 value = [44, 60, 0] class = medium-spending customer
gini = 0.234 samples = 185 value = [25, 160, 0] class = medium-spending customer
gini = 0.048 samples = 535 value = [8, 522, 5] class = medium-spending customer
gini = 0.461 samples = 29 value = [2, 20, 7] class = medium-spending customer
gini = 0.185 samples = 20 value = [1, 1, 18] class = high-spending customer
gini = 0.5 samples = 2 value = [0, 1, 1] class = medium-spending customer
gini = 0.0 samples = 12 value = [0, 12, 0] class = medium-spending customer
gini = 0.423 samples = 23 value = [0, 7, 16] class = high-spending customer
gini = 0.5 samples = 8 value = [0, 4, 4] class = medium-spending customer
gini = 0.198 samples = 27 value = [0, 3, 24] class = high-spending customer
gini = 0.5 samples = 2 value = [0, 1, 1] class = medium-spending customer
gini = 0.5 samples = 2 value = [0, 1, 1] class = medium-spending customer
gini = 0.111 samples = 17 value = [0, 1, 16] class = high-spending customer
gini = 0.278 samples = 6 value = [0, 1, 5] class = high-spending customer
gini = 0.005 samples = 827 value = [0, 2, 825] class = high-spending customer

Figure 22: Decision tree interpretation

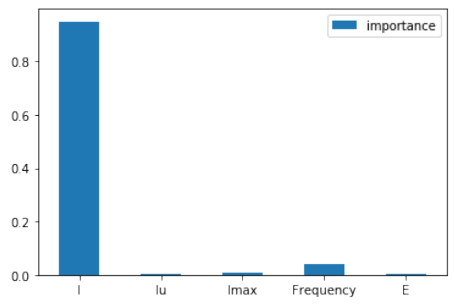


Figure 23: Decision tree feature importance

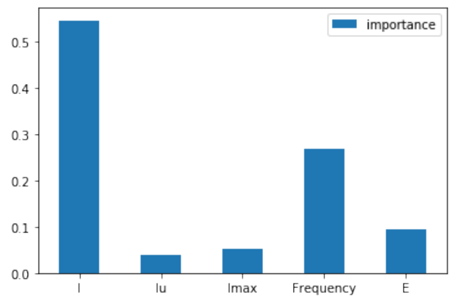


Figure 24: Random forest feature importance

As we can see from the pictures, I (the quantities of products bought by a customer) take the most prominent role in both the decision tree and rule based models. It is actually non-surprising at all, as the price of each product varies little in our dataset.

Efficiency: neural network, rule-based , random forest are much more time-consuming than other classification algorithms. Note Here we are just using a simply constructed neural network and a relatively small epoch in order to finish running in a short time.

Preprocessing: In KNN and neural network, we normalized the data before fitting the algorithm, because Knn relies on majority voting based on class membership of 'k' nearest samples for a give test point, while nearest point is typically measured by distance, and these distances are highly influenced by the measurement unit. Therefore it will impact the accuracy of the classification if not normalized. For neural network, it is not necessary to normalize the data in theory. However, the practice has shown neural network training can be more efficient when data are normalized.

Performance: From the results shown in Table 3 and Table 4, decision tree performs the best among all the models. However, hyperparameters of neural network can be more fined tuned to improve its performance.

4 Sequential Pattern Mining

In this section, we dig into the dataset by extracting interesting sequential patterns to better understand the information hidden.

4.1 Data preparation

In this task, we use the whole dataset, filtering out any rows with non-purchase transaction codes/product codes(e.g., *BasketIDs* starting with 'C', *ProdIDs* labelled by 'M'). What's more, for the results to be interesting, there must be at least two elements in a pattern, so we eliminate infrequent customers who only shopped once, i.e., who had only one *BasketID*. From the 4335 customers, only 2829 had at least two visits.

To prepare for the sequential pattern mining task, we need to **model the customer as a sequence of baskets**: all transactions(baskets) of a customer can together be viewed as a sequence, where each basket corresponds to a set of product items(*ProdIDs*), and list of baskets corresponds to a sequence, ordered by increasing BasketDate. In the end, a customer will be represented like $< \{ProdID1\}, \{ProdID2, ProdID3\}, \{ProdID4\} >$, where $\{ProdID2, ProdID3\}$ means $ProdID2$ and $ProdID3$ were bought together in one single basket.

4.2 Mining sequential patterns

Here we are using PrefixSpan algorithm from the **SPMF** library(`https://www.philippe-fournier-viger.com/spmf/`), because it not only runs very efficiently when the minimal support is set to a very small value, but also it gives us more long sequential patters, compared to other algorithms (e.g., GSP).

With a minimum support threshold of 0.05 (or 5%) we found 231 frequent sequences with two baskets, 11 with three baskets, and 1 with four baskets, resulting in a total of 243 patterns with at least two baskets.

The patterns that appear more often are:

- **two baskets' patterns**
  - $< 85123, 85123 >$ 407 times, 14.4%
  - $< 85099, 85099 >$ 379 times, 13.4%
  - $< 22423, 22423 >$ 330 times, 11.7%
  - $< 47566, 47566 >$ 304 times, 10.7%
  - $< 84879, 84879 >$ 291 times, 10.3%
  - $< 20725, 20725 >$ 276 times, 9.8%
  - $< 85099, 23203 >$ 251 times, 8.9%
- **three baskets' patterns**
  - $< 85099, 85099, 85099 >$ 232 times, 8.2 %
  - $< 85123, 85123, 85123 >$ 224 times, 7.9 %
  - $< 85099, 85099, 23203 >$ 168 times, 5.9%
  - $< 85099, 23203, 85099 >$ 163 times, 5.8%
  - $< 22423, 22423, 22423 >$ 163 times, 5.8%
  - $< 47566, 47566, 47566 >$ 161 times, 5.7%
  - $< 20725, 20725, 20725 >$ 159 times, 5.6%
- **four baskets' patterns**
  - $< 85099, 85099, 85099, 85099 >$ 151 times, 5.3 %

4.3 Results discussion and mining sequential patterns through categorizing products

From the patters we have obtained above, we can see there are two types of patterns.

1. Same products repeating: Customers repeatedly buy the same items periodically. For example, the pattern $< 85099, 85099 >$
2. One after another: a customer buys different items sequentially. Example: $< 85099, 23203 >$

As we can see, products $85123, 85099, 22423, 47566, 84879, 20725$ are repeatedly purchased by customers, so these products should be sufficiently stock up to support customers buying them.

Another aspect we have noticed is that there are no patterns mined above the support 15%, and patterns above support 10 % are all about repeated products, and with no pattern containing three baskets. Considering in commercial stores, each digit of a product's code is often related to a product's category. That is to say, the first three digits represent more general product's category compared to the code containing five numbers. We think this principle is also valid here after looking into product codes and their corresponding product descriptions in our dataset. Therefore, we decided to map a 5-digital ProdID to a 3-digital ProdID by picking up only the first three digits(one or two would be too general, while four or five would be too specific), representing a **class** of products. By doing so, we get 914 patterns with at least two baskets with a $min\_support = 30\%$.

The patterns that appear more often are:

- **two baskets' patterns**
  - $< 224, 224 >$ 1609 times, 56.9%
  - $< 226, 226 >$ 1578 times, 55.8%
  - $< 221, 221 >$ 1539 times, 54.4%
  - $< 224, 226 >$ 1527 times, 54.0%

  - $< 224, 221 >$ 1518 times, 53.7%

- **three baskets' patterns**
  - $< 224, 224, 224 >$ 981 times, 34.7 %
  - $< 226, 226, 226 >$ 960 times, 33.9 %
  - $< 221, 221, 221 >$ 939 times, 33.2%
  - $< 224, 224, 226 >$ 930 times, 32.9%
  - $< 224, 221, 221 >$ 926 times, 32.7%

Based on the above results, products that were frequently bought are 224, 226, and 221. Therefore the retailer should increase the number of stocks for these products in order to make more profit. The results also show customers' behavior in purchasing in the store. For example, customer who bought the product of class 224 also bought the product of class 226 or class 221 at a latter time. This patterns appears both in the two baskets' sequences (with a support more than 50%) and also three basket's sequences (with a support more than 30%). Based on these sequences, the retailer may provide customers who purchased product class 224 recommendations to try the products class 226 and 221. Based on the above results, products that were frequently bought by customers are 224, 226, and 221. Therefore the retailer should increase the number of stocks for these products in order to make more profit. The results also show customers' behavior in purchasing in the store. For example, customer who bought the product of class 224 also bought the product of class 226 or class 221 at a latter time. This pattern appears both in the two baskets' sequences (with a support more than 50%) and three basket's sequences (with a support more than 30%). Based on these sequences, retailer may provide customers who purchased product class 224 recommendations to try the products class 226 and 221. In addition, retailer may locate products class 224, 226, and 221 close to each other to provide easy access and convenient for customers in searching these products.

4.4 Sequential pattern mining with temporal constraints

In the following content, we will explore sequential patterns that contain some temporal constraints, using Hirate-Yamana algorithm from **SPMF** library. To do so, we need to extend our data with time information, so we annotate each basket with an integer timestamp, representing time elapsed in days since the starting date. For example, a customer (sequence) could be described as $< \{0, ProdID1\}, \{5, ProdID2, ProdID3\}, \{12, ProdID4\} >$, where $\{5, ProdID2, ProdID3\}$ means $ProdID2$ and $ProdID3$ were bought at the same time on the fifth day, and after 7 days, he also bought ProdID3.

With a $min\_support = 0.5\%, min\ time\ interval = 1, max\ time\ interval = 30, max\ whole\ timeinterval = 90$, 131 sequential patters ($pattern\_length \geq 2$) are mined. The first ten patterns are:

- $< \{0, 85099\}, \{29, 85099\} >$ 29 times, 1.0%
- $< \{0, 85099\}, \{22, 85099\} >$ 29 times, 1.0 %
- $< \{0, 85099\}, \{21, 85099\} >$ 27 times, 0.95 %
- $< \{0, 85099\}, \{6, 85099 >\}$ 26 times, 0.91 %
- $< \{0, 85099\}, \{10, 85099 >\}$ 26 times, 0.91 %
- $< \{0, 85099\}, \{12, 85099 >\}$ 26 times, 0.91 %
- $< \{0, 85099\}, \{14, 85099 >\}$ 26 times, 0.91 %
- $< \{0, 85099\}, \{28, 85099 >\}$ 26 times, 0.91 %
- $< \{0, 85123\}, \{21, 85123 >\}$ 26 times, 0.91 %
- $< \{0, 85099\}, \{13, 85099 >\}$ 25 times, 0.88 %

Note in these sequences, the time annotation means time intervals (in days) between two consecutive orders, which is different from what we have seen in the input data. The resulting patterns tell us that not merely product 85099 (JUMBO BAG) was very regularly bought by customers; moreover, it also indicates how regular would a customer repurchase it.

To discover more interesting patterns, we also apply this algorithm on product category representation mentioned previously. With $min\_support = 3\%, min\ time\ interval = 1, max\ time\ interval = 30, whole\ time\ interval = 90$, we ended up with 47 patterns ($pattern\_length \geq 2$) , the first ten are:

- $< \{0, 226\}, \{28, 226\} >$ 121 times, 4.3%

- $< \{0,221\},\{28,221\} >$ 106 times, 3.7 %
- $< \{0,226\},\{28,224\} >$ 103 times, 3.6 %
- $< \{0,224\},\{28,224 >\}$ 101 times, 3.6 %
- $< \{0,226\},\{28,232 >\}$ 99 times, 3.5 %
- $< \{0,226\},\{28,221 >\}$ 99 times, 3.5 %
- $< \{0,232\},\{28,232 >\}$ 96 times, 3.4 %
- $< \{0,221\},\{21,221 >\}$ 95 times, 3.4 %
- $< \{0,221\},\{28,226 >\}$ 95 times, 3.4 %
- $< \{0,224\},\{21,226 >\}$ 95 times, 3.4%

As the patterns suggest, products class 226, 221, 224, 232 have been repurchased regularly. Besides, within a month time, customers who bought 226 would buy 224, 232 and 221 subsequently. As the patterns suggest, products class 226, 221, 224, 232 has been repurchased very regularly. Besides, within a month time, customers who bought 226 would buy 224, 232 and 221 subsequently, also customer who bought products of class 221 or 224 would buy 226 as well.

5 Conclusion

We used different data mining techniques to segment customers, predicate whether customer is a high/medium/low spender, and mine interesting sequential patterns for the analysis of this transaction data.

In the Data Understanding section, we addressed issues related to data qualities, fixing missing values and outliers, and created a customer profile using the RFM model.

In the Clustering analysis task, we applied various clustering algorithms to the dataset to segment customers. We found that k-medoids performs the best, and k-means also works very well in this case.

In the Classification phase, we built a new set of indicator to serve this task better and also addressed the temporal constraints. Among all the classification models, decision tree was the best approach, considering both its performance, efficiency, and interpretability.

Finally, in the Sequential Pattern Mining section, we tried to discover the dataset's hidden information by mining sequential patterns. Additionally, we extended sequences with the time information, and in this way, we are able to take care of some temporal constraints.