NOVA IMS Information Management School

DATA MINING

MASTER DEGREE PROGRAM IN DATA SCIENCE AND ADVANCED ANALYTICS

CUSTOMER SEGMENTATION THROUGH CLUSTER ANALYSIS

INSURANCE SECTOR

Group BJ

Lucas Lopes, number: r20181111

Tiago Sousa, number: r20181077

January, 2021

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

INDEX

1. Introduction

This report addresses the final project in the course of Data Mining of the Master's Degree Program in Data Science and Advanced Analytics at Nova IMS. For its development, students were provided a fictional insurance company dataset, containing customer data, with datapoints ranging from the personal scope to product expenditure.

Throughout the development of the project, the team impersonated a consulting firm with the final goal of coming up with a "Customer Segmentation in such a way that it will be possible for the Marketing Department to better understand all the different Customers' Profiles".

2. Initial Data and Variables Description

Before anything else, we start by importing the data into out work environment to get an initial grasp of what we are dealing with. The initial dataset consists of **14 variables** and **10 296 observations**. We decided to name the initial dataset as ***insur***.

We immediately proceed to changing the datatypes of some variables in order to facilitate the analysis in the following stages of the process. *CustID* change from a float to a string, *EducDeg* changed from an object to a string, *GeoLivArea* changed from float to a string and *Children* from a float to a string. Table 1, in the appendix, shows all the initial variables as well as what they represent in context of the problem.

3. Data preparation

3.1. Coherence Checking

Moving on, we had to verify the integrity of the data being used. We started by looking for odd and unreasonable values in the variables and found an observation where the birth year of the client was registered as 1028, and another one where the date of the First Policy was registered as 53784. As these values make no sense, we removed both the observations.

Furthermore, we decided to check if there were any cases of extremely young people with very advanced qualifications for their age and found some observations. However, we did not delete these observations for a simple reason: the *BirthYear* variable issue.

When looking for incoherencies, we discovered that a great amount of First Policies had been signed prior to the birth of the client. In fact, this happened in 1997 observations, almost 20% of all the observations. Initially, and in order to preserve the variable, we decided that we would deal with this by defining these observations as NANs, with the objective of later on (in the Missing Values stage) imputing them using the k-nearest neighbors algorithm. And so, we did. However, after applying the algorithm, we doubled checked and realized that there were still generated observations with a First Policy date prior to the Birth Date of the client. In fact, in 765 observations we could still observe this.

This led to the conclusion that there was something wrong with either one of the variables and the best solution was the remove on of them. Our reasoning behind the decision was very simple: The *FirstPolYear* variable would be a value registered by the informatic system or even if manually, by a trained employee, therefore having a smaller chance of being a mistake. The *BirthYear*, on the other hand, being a value provided by the client, is more subject to mistakes either by distraction or handwriting that might not be clear. Therefore, we decided to remove the variable *BirthYear*.

Nevertheless, the code for the k-nearest neighbors algorithm can still be found commented in the Jupyter Notebook, in the Missing Values section.

3.2. Outliers

The next stage consisted of removing outliers present in the data. By analyzing the descriptive statistics table and Pairplot Grid, figures 1 and 2 of the annex, their presence is evident.

Given the small number of features, we decided to remove the outliers by hand by analyzing the histograms and boxplots of each variable individually. On table 2 of the annex, the number of observations removed from each variable can be found. All things together, 129 observations were removed from the ***insur*** dataset, just 1,26% of the total. It is important to mention that the observations pertaining to the outliers were kept in another separate dataset: ***outliers_df.***

The descriptive statistics table and Pairplot Grid, figure 3 and 4 of the annex, make it possible the see the clear difference when compared to before the outlier removal.

3.3. Missing Values Treatment

Regarding missing values, we had a lot 'na' values in our dataset. On figure 5 of the annex, it is visible how many missing values existed per variable.

We divided our imputation in two parts: the first one was all the straightforward and easier imputations, while the second one focused on more complex methods, such as KNN Classifiers and KNN Regressor.

In the first one, we did three different imputations:

- We removed the 30 missing values of *FirstPolYear*. We chose to do this because since this variable is created by the software, missing values would probably mean that happened some kind of problem with these records. And because there are only 30 observations in this situation, we decided to remove them.
- We imputed the mode in *GeoLivArea*. Because there is only one record with this missing value, we implemented the easiest way possible: replace it with the mode of that variable.
- Lastly, we looked at the 5 Premium variables at replace all the missing values with zero. This means that if we do not have any information about that specific premium, it makes sense to assume that that client did not spent any money on that service.

Secondly, we used KNN to impute values in the variables *EducDeg*, *Children*, and *MonthSal*:

- For *EducDeg*, since it is not a numerical variable, we used a KNeighborsClassifier with the following parameters (n_neighbors=7, weights='distance', metric='euclidean') to impute the two missing values.
- For the numerical variable *MonthSal*, we performed an econometric linear regression between that variable and all the other numerical ones. By analyzing its result, we were able to find which variables contributed more to explain to *MonthSal*. Having this information, we then used KNeighborsRegressor with the following parameters (n_neighbors=11, weights='distance', metric='euclidean'). By using a regressor, the 34 new imputed values are different from each other and result in the mean of the *MonthSal* variable of the 11 observations that are closer.
- Finally, we also used KNeighborsClassifier for the Children feature. We did in a very similar way to *EducDeg*, with the only difference that this time we used 9 neighbors. In the end, all the 13 missing values of this binary variable were imputed

4. Feature Engineering

4.1. Transform Variables

In order to extract more valuable information from the dataset to create the clusters for a precise customer segmentation, we decided to create a few new variables and add them to the ***insur*** dataset. The Variables created are shown in figure 6.

Adding to the creation of these new variables, we also opted for applying the square root to some of the original variables that were a bit skewed, as visible on figure 4, shown previously. These were *PremLife*, *PremWork* and *PremHousehold*. Following the same logic, we decided to also apply the square root to these variables after transforming them into ratios, given they were, once again, skewed.

5. Correlation Matrix

In the Data Mining process, a very important and crucial part lies in choosing the best variables to use for your analysis. And for assessing possible redundancy of features, the correlation matrix is an effective visualization. For this, we used the spearman method because we were also interested in nonlinear relationships between the variables.

With this matrix, we found out that a lot of the new variables were highly correlated between each other (see figure 7). Because of this, we removed two variables: *FirstPolYear* and *Annual*_Salaray. However, we decided to leave all the remaining variables and implement our models with different combinations of features. Only at the end, by evaluating which set of variables gave the best result, we chose the most important features for our analysis. Until then, this matrix was only a helpful guideline to check if the variables we were using at each time were not highly correlated among each other.

6. Scaling Methods

Feature scaling is essential for Data Mining algorithms that are performed by calculating the distances between data. However, our variables are not all on the same scale. And because we do not want the effect of one variable contributing more or less than the others, the range of all the features of our dataset should be normalized (that way, each feature contributes approximately proportionately to the final distance).

Having this in mind, we performed 3 different scaling methods: MinMaxScaler between 0 and 1, MinMaxScaler between -1 and 1, and StandardScaler (also known as Z-score). We know that the scaling used can have serious impact on the outputs obtained and also on the profiling of the final clusters, so we tried all our algorithms with these 3 different scalers and then chose the one that led to better and more interesting results: in our case, it was the **StandardScaler** (see table below for more details).

7. Segmentation Creation

Now that we had all the variables selected and were ready to start applying the clustering algorithms, it was time decide which perspectives we wanted to implement. We opted for 2 different perspectives: Value and Product.

Value: in this perspective, variables like *Client_Recency, MonthSal, CustMonVal, ClaimsRate* and *SalarySpent_Ratio* were included. With this segmentation, we intend the distinguish the clients according to their potential and dedication the company

Product: in this perspective, variables like *Total_Premiums, PremMotor, PremHealth, PremHousehold_sqrt, PremLife_sqrt, PremWork_sqrt, PremMotor_Ratio, PremHealth_Ratio, PremHousehold_Ratio_sqrt, PremLife_Ratio_sqrt and PremWork_Ratio_sqrt*. With this segmentation, we intend to distinguish the clients according to their favorite and most used products.

It is important to mention that in neither of the segmentations all of the variables were used simultaneously. Instead, subgroups were made given the high correlations between some of those variables. For instance, the variables *Total_Premiums* and *PremHousehold_sqrt* were never used together given their high correlation of 0.98. The different datasets created for each segmentation and their respective variables were can be found in the appendix, table 3.

8. Quartile Based Clustering

In order to get some extra insights into our data we decided to perform Quartile Based Clustering on the variables related to insurance consumption: *PremMotor, PremHealth, PremHousehold, PremLife* and *PremWork,* to show their potential in relation to the total amount of premiums.

By analyzing the quartile matrix (see figure 8), the correlation between *Total_Premiums* and *PremHousehold* is very evident. Furthermore, it is clear that those who spend a lot on *Prem_Health*, spend much less on the remaining categories together, and that those who spend a lot a in Motor, tend to spend much less in the remaining categories.

9. Clustering Algorithms

Once the previous steps were all concluded, we were ready to start performing the Clusters analysis. For this, we used seven different approaches, with a total of six different algorithms. For each segmentation we applied:

9.1. Hierarchical Clustering

This is an algorithm that groups similar objects into clusters, where the goal is to obtain a set of clusters different from each other, and the objects within each cluster are broadly similar to each other. In Agglomerative Hierarchical Clustering, the technique we have used, initially each data point is considered as an individual cluster and at each iteration cluster merge until k clusters are formed.

Regarding its implementation, we used the function **plot_r2_scores** to plot the r2 scores for K-means, K-medoids, and Hierarchical for each one of the linkages (single, complete, average and ward´s method). We calculated theses values for different numbers of clusters (from 1 to 5) so we could understand how the r2 scores are changing.

The second step was to plot the dendrogram (also with the help of a function) and try to understand what the ideal number of clusters should be. Note that, for the dendrogram, we used ward´s methods since it was the one with higher r2 scores.

With the output from the dendrogram, we would then decide the number of clusters to apply and even try out a few different possibilities, usually between 3,4 or 5 clusters, and see which one led to a better interpretability by looking at the means of the variables in each cluster as well as the r2 and silhouette scores. Finally, we would plot the resulting clusters with the respective variables' histograms, using a dedicated function.

9.2. K-Means/K-Medoids Algorithm

This is an iterative partition algorithm that splits the dataset into k predefined clusters and tries to make the intra cluster points as similar as possible while making the clusters as different as possible from each other. An important note is that K-means consistently showed better r2 scores when compared to K-medoids, so we decided to use K-means from this point of the project on.

Again, we wanted to decide what would be the ideal number of clusters to retain for the analysis, so we plotted the inertia values and the silhouette scores, keeping in mind that we want to minimize the inertia and maximize the silhouette score. Based on those result, we would define the number of clusters to be used, again trying several possibilities, and see which one led to a better interpretability. Just like before, in the end we would plot the resulting clusters with the respective variables' histograms, using the dedicated function.

9.3. Self Organizing Maps (SOM)

This is an algorithm based on unsupervised learning that provides dimensionality reduction by transforming high dimensional input spaces into two- or three-dimensional ones.

Here, we tuned the algorithm by changing various parameters. The most important parameters were the number of units to keep (a grid can be 10x10, 25x25, 40x40, 50x50, etc) and the size of the **train_rough_len** and **train_finetune_len** parameters (use to train our SOM instance). After multiple combinations, we settled a mapsize of 625 unit (25x25), a train_rough_len of 125, and a train_finetune_len of 110. To further analyse our SOM output, we plot the component planes, the U-Matrix and the Hit Map (all these representations can be seen in our jupyter notebook).

From then on, we would apply both **Hierarchical Clustering** and **K-Means/K-Medoids on the SOM Units**, following the same logic used before in the respective algorithms.

9.4. DBSCAN

This is an algorithm that groups together points that are close to each other based on a distance measurement, like Euclidean distance for instance, and close to a pre-defined minimum number of points. It also has the capacity to mark as outliers the points that are in low-density regions.

For this algorithm we also tunned some of the parameters, in particular the **min_samples,** which would always be the double of the variables present in that segmentation, and the **eps**, the maximum distance between two samples for one to be considered as in the neighborhood of the other. To help find out the right value for the eps, we would plot the K-distance graph. Moving on we would define the number of clusters, then compute both r2 and silhouette scores and then plot the resulting clusters with the respective variables' histograms, using the dedicated function.

9.5. Mean Shift Clustering

This is an algorithm that that assigns the data points to the clusters iteratively by shifting points towards the mode, the highest density of data points in the region.

For this algorithm we also tunned some of the parameters, by defining the bandwidth instead of using the one automatically estimated. Different bandwidths would lead to different numbers of clusters, so we adjusted the values several times to try out different outcomes. Through that we could then find out how many clusters to use, then compute both r2 and silhouette scores, compare them between different numbers of clusters, and then plot the resulting clusters with the respective variables' histograms, using the dedicated function.

9.6. Gaussian Mixture Model

This is an algorithm that assumes that there are a certain number of Gaussian distributions, and each of these distributions represent a cluster, creating the tendency to group the data points belonging to a single distribution together.

For this algorithm we also had to set of the parameters, such as the covariance type, either diag, spherical, full or tied, the number of initializations, as well as the number of components, which would be defined based on the AIC and BIC. Through that we could then find out how many clusters to use, then compute both r2 and silhouette scores, compare them between different numbers of clusters, and then plot the resulting clusters with the respective variables' histograms, using the dedicated function.

10. Clustering by Segmentation

10.1. Segmentation by Value

In the Value Segmentation, the most successful out of the 4 different datasets tested was value_df4, which contained the variables Client_Recency, SalarySpend_Ratio, and ClaimsRate. This group of variables consistently outperformed the other 3, both in terms of r2 and silhouette scores and cluster interpretability on every algorithm tested. For that reason, from now on, this will be the only mentioned dataset regarding value, as it was the one used for the remaining parts of the project.

As visible on table 4, several of the algorithms had a really good performance. Hierarchical, K-means, and both Hierarchical and K-means on SOM units had significantly better results than the other 3 remaining methods. From this point on we knew we could use any of those for further analysis. However, after some time of comparing the different algorithms and the respective outputs, we realized that, for the sake of interpretability it would be better to stick with either K-means or Hierarchical on SOM units. For that reason, we compared these two in more detail.

By looking at the figures with the averages, 9 and 11, and the figures containing the histograms, 10 and 12, we can conclude that the different clusters vary slightly. Also, the mean values for each variable do not give us a clear idea on the existence, or not, of discriminatory power of these variables. Again, because we do not have a significant difference when comparing these figures, we decided to keep the one that gave us the best R^2 and Silhouette scores (0.5942 and 0.3379, respectively).

10.2. Segmentation by Product

In the Product Segmentation, the most successful out of the 4 different datasets tested was **products_df,** which contained the variables *PremMotor, PremHealth, PremLife_sqrt, PremWork_sqrt, PremHousehold_sqrt*. This group of variables consistently outperformed the other 3, both in terms of r2 and silhouette scores and cluster interpretability on every algorithm tested. For that reason, from now on, this will be the only mentioned dataset regarding product, as it was the one used for the remaining parts of the project.

As visible on table 5, several of the algorithms had a really good performance. Hierarchical, K-means, and both Hierarchical and K-means on SOM units had significantly better results than the other 3 methods. From this point on we knew we could use any of those. However, after some time of comparing the several combinations possible, we realized that for the sake of interpretability it would be better to stick with only 4 clusters, so we decided to compare only K-means and K-means on SOM units.

By looking at the figures with the averages, 12 and 14, and the figures containing the histograms, 13 and 15, it is clear that these methods did a good job creating the clusters. In both, we can clearly identify different behaviours when moving from cluster to cluster in every single variable. Nevertheless, we must choose one of the algorithms to carry to next part of the project: the K-means. This choice is supported by the fact that it not only has a great interpretability, but also a better combination of R^2 and Silhouette scores, 0.6 and 0.2458 respectively.

11. Merging the Perspectives

In this stage, a final dataset was created with the best variables from both segmentations, value and products. To that same dataset, we added two columns with the respective labels of the best algorithms of each segmentation, mentioned in the previous step. By looking at figure 16, it is possible to understand how the observations were distributed between the different labels. As noticeable, there are some low values in some segmentations which means they should be clustered to nearest group. To do this, we used the dendrogram to verify which would be the ideal final number of clusters to have in our analysis. The results are shown in figure 17, from which we made the decision of keeping 4 final clusters. These 4 final clusters can be seen in figure 18. Note that the labels 0 and 1 of the value segmentation disappeared, meaning they were grouped together with the product segmentation labels.

12. The Final Clusters

Finally, as visible from figure 19, we are able to profile the final clusters. We can see that the 4 frequency vary quite a bit in size, cluster 3 is much larger than the rest, and cluster 1 has only a small portion of the observations. Furthermore, from the parallel coordinates plot we can visualize the mean values of each variable for each final cluster.

From figure 20, pertaining the non-metric variables, we can see the behavior of each cluster according to these variables. We can easily grasp that the variable *GeoLivArea* maintains the same distributions across the different final clusters, meaning it is probably not a very good discriminatory variable. The same happens for *Children*, where the distributions of the variable across the clusters is relatively the same, once again, not having a lot discriminatory power. *EducDeg,* on the other hand, depicts a different behavior across clusters. We can see that both clusters 1 and 4 have people slightly less educated that the other two (barely any clients have a PhD). Furthermore, clusters 2 and 3 have a much larger number of clients with Bachelors/Masters Degrees compared to the other two.

Moreover, we also used t-SNE in order to make it possible to visualize the clusters in 2 dimensions, as shown in figure 21.

12. Feature Importance

Furthermore, we decided to take a look at the feature importance of each variable with the resource of two different methods: by calculating the R^2 of each variable and by plotting a decision tree. The results shown in figure 22 and 23, indicate that variables such *Client_Recency* and *ClaimsRate* are not relevant for discriminating the final clusters. This corroborates the results from the parallel coordinates plot (figure 19) where the means of these variables are very close to zero in every cluster. Therefore, we decided to exclude these two variables from our final analysis.

13. Conclusion

Having analyzed the final clusters its time for final conclusions:

Cluster1: In this cluster, the smallest one with 626 observations, we can find clients with high values in *SalarySpent_Ratio*, which means they spent a large proportion of their annual salary on insurance premiums. Moreover, the expenditures on the various types of premiums are more predominant in Life, Work and Household. On the other hand, the value for *PremMotor* is lower when compared to other clusters. Curiously, this cluster is one of the least instructed ones.

Cluster2: In this cluster, with 2626 observations, we can find clients high values in *PremMotor*. On the other hand, on the remaining categories this is the cluster with the smallest expenditure. Regarding education level, in this clusters we can find mostly highly educated individuals.

Cluster3: In this cluster, with almost 5500 observations, we can find clients with higher values of expenditure in Health. Regarding the remaining categories, the expenditure of these clients does not substantially differ from the mean. Similarly to cluster 2, individuals in this cluster are mostly highly educated.

Cluster4: In this cluster, with 1386 observations, which is very similar to cluster 1, we can also find clients with high expenditure values on Life, Work and Household. The main difference concerns the proportion of the salary spent, which is much smaller in this cluster. This probably means that these clients earn more money than those in cluster 1. Lastly, also like in cluster 1, the majority of individuals in this cluster are less educated than the overall population of the dataset.

Regarding the possible marketing approaches for each cluster, the strategies are as follows:

- For both **cluster 1** and **4**, given their similarity, the approach should not vary a lot, with premiums focused on Life, Work and Household. Moreover, since these clusters are populated by people with lower levels of education, an interesting possibility to explore would be the promotion of workshops with educational content or provide a student discount. However, given the assumed disparity between the cluster's available budgets, we could try to explore different combinations of premiums, with higher quality and more expensive for cluster 4, and occasional promotions for cluster 1 in order to minimize the churn rate.

- For **cluster 2**, we could explore the possibility of cross selling the Motor premiums with other premiums in order to make them diversify their expenditure. This was, by investing more in Prem Motor, we could make them spend more in the remaining categories, and increase their percentage of salary spent in premiums, the lowest of all clusters.
- For **cluster 3**, probably the most important one given it comprises 54% of our client base, we should before anything, make sure these customers are satisfied. Furthermore, we could create a fidelity card for these customers and provide rewards based on the expenditure, in order to try to make them increase the amount of money they spend on other premiums apart from Health. It is important to mention that these fidelity cards would only apply to the clients with minimum levels of expenditure in all types of premiums. This way we can ensure that these 5500 customers would keep spending money on all premiums.

14. References

[0] Akshara_416. (2021, July 26). Isolation Forest | Anomaly Detection with Isolation Forest. Analytics Vidhya https://www.analyticsvidhya.com/blog/2021/07/anomaly-detection-using-isolation-forest-a-complete-guide/

[1] Kelvin Salton do Prado. (2017, April 1). How DBSCAN works and why should we use it? https://towardsdatascience.com/how-dbscan-works-and-why-should-i-use-it-443b4a191c80

[2] ankurtripathi. (2019, May 16). ML | Mean-Shift Clustering https://www.geeksforgeeks.org/ml-mean-shift-clustering/

[3] Aishwarya Singh. (2019, October 31). Build Better and Accurate Clusters with Gaussian Mixture Models https://www.analyticsvidhya.com/blog/2019/10/gaussian-mixture-models-clustering/

[4] Tim Bock. What is Hierarchical Clustering? https://www.displayr.com/what-is-hierarchical-clustering/

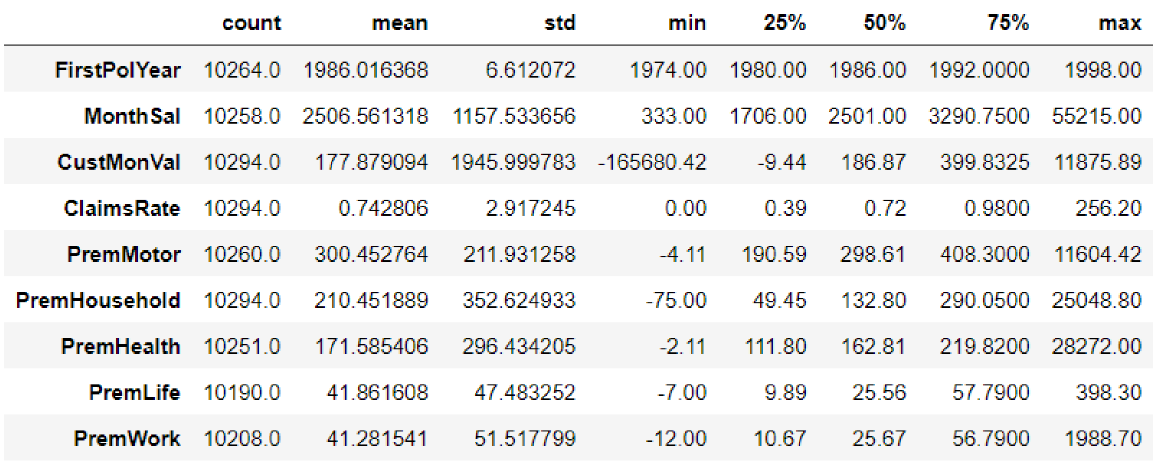
[5] Achraf KHAZRI. (2019, August 7). Self Organizing Maps https://towardsdatascience.com/self-organizing-maps-1b7d2a84e065

[6] Pulkit Sharma. (2019, August 19). The Most Comprehensive Guide to K-Means Clustering You'll Ever Need https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/

15. Appendix

VARIABLE	TYPE	DESCRIPTION
Cust_ID	string	Customer ID
FirstPolYear	float64	Year of the customer's first policy
BirthYear	float64	Customer's Birthday Year
EducDeg	string	Academic Degree Level (1-Basic, 2-High school, 3-BSc/Msc, 4-PhD)
MonthSal	float64	Gross monthly salary (€)
GeoLivArea	string	Living area
Children	string	Binary variable (Y=1) - customer has children (1) or not (0)
CustMonVal	float64	Lifetime value = (annual profit from the customer) X (number of years that they are a customer) - (acquisition cost)
ClaimsRate	float64	Amount paid by the insurance company (€)/ Premiums (€) Note: in the last 2 years
PremMotor	float64	Annual premiums in Motor (€)
PremHousehold	float64	Annual premiums in Household (€)
PremHealth	float64	Annual premiums in Health (€)
PremLife	float64	Annual premiums in Life (€)
PremWork	float64	Annual premiums in Work Compensations (€)

Table 1 - Variable Description



	count	mean	std	min	25%	50%	75%	max
FirstPolYear	10264.0	1986.016368	6.612072	1974.00	1980.00	1986.00	1992.0000	1998.00
MonthSal	10258.0	2506.561318	1157.533656	333.00	1706.00	2501.00	3290.7500	55215.00
CustMonVal	10294.0	177.879094	1945.999783	-165680.42	-9.44	186.87	399.8325	11875.89
ClaimsRate	10294.0	0.742806	2.917245	0.00	0.39	0.72	0.9800	256.20
PremMotor	10260.0	300.452764	211.931258	-4.11	190.59	298.61	408.3000	11604.42
PremHousehold	10294.0	210.451889	352.624933	-75.00	49.45	132.80	290.0500	25048.80
PremHealth	10251.0	171.585406	296.434205	-2.11	111.80	162.81	219.8200	28272.00
PremLife	10190.0	41.861608	47.483252	-7.00	9.89	25.56	57.7900	398.30
PremWork	10208.0	41.281541	51.517799	-12.00	10.67	25.67	56.7900	1988.70

Figure 1 - Descriptive Statistics

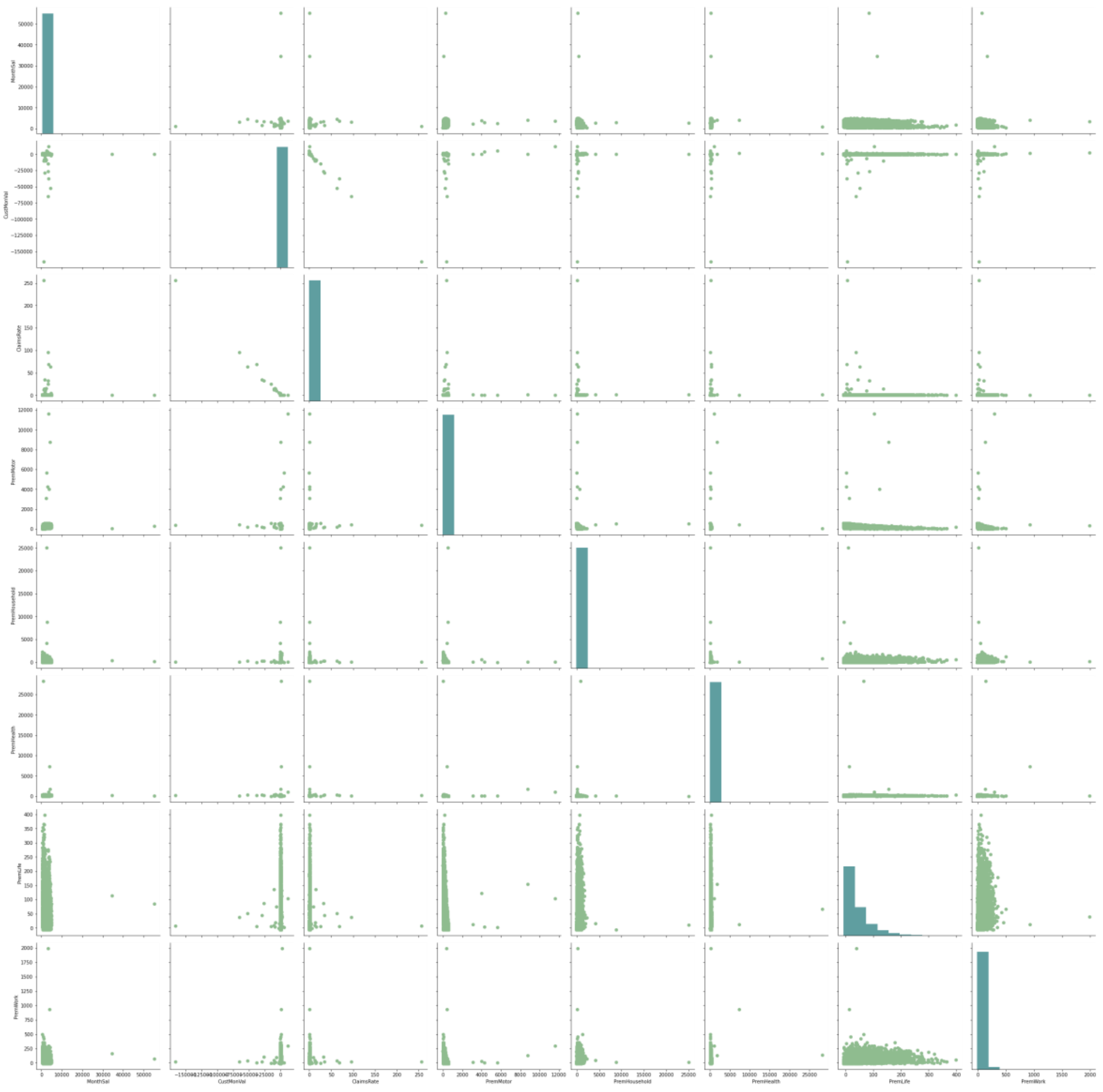
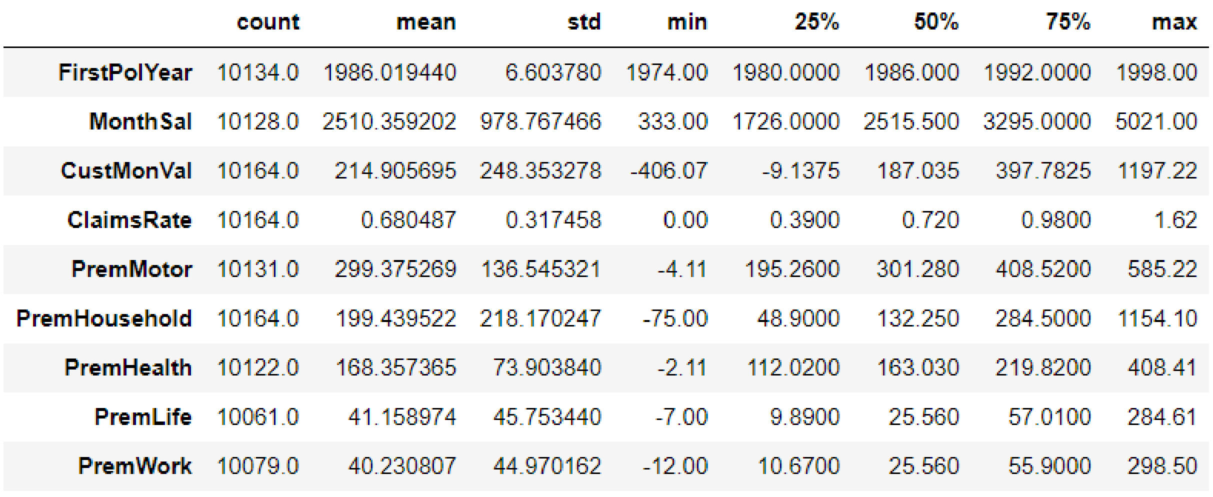


Figure 2 - Pairplot Grid

Variable	Number of deleted observations
MonthSal	2
CustMonVal	51
CalimsRate	1
PremMotor	3
PremHousehold	35
PremHealth	6
PremLife	18
PremWork	13
Total	**129**

Table 2 - Number of deleted observations per variable



	count	mean	std	min	25%	50%	75%	max
FirstPolYear	10134.0	1986.019440	6.603780	1974.00	1980.0000	1986.000	1992.0000	1998.00
MonthSal	10128.0	2510.359202	978.767466	333.00	1726.0000	2515.500	3295.0000	5021.00
CustMonVal	10164.0	214.905695	248.353278	-406.07	-9.1375	187.035	397.7825	1197.22
ClaimsRate	10164.0	0.680487	0.317458	0.00	0.3900	0.720	0.9800	1.62
PremMotor	10131.0	299.375269	136.545321	-4.11	195.2600	301.280	408.5200	585.22
PremHousehold	10164.0	199.439522	218.170247	-75.00	48.9000	132.250	284.5000	1154.10
PremHealth	10122.0	168.357365	73.903840	-2.11	112.0200	163.030	219.8200	408.41
PremLife	10061.0	41.158974	45.753440	-7.00	9.8900	25.560	57.0100	284.61
PremWork	10079.0	40.230807	44.970162	-12.00	10.6700	25.560	55.9000	298.50

Figure 3 - Descriptive Statistics after outlier removal

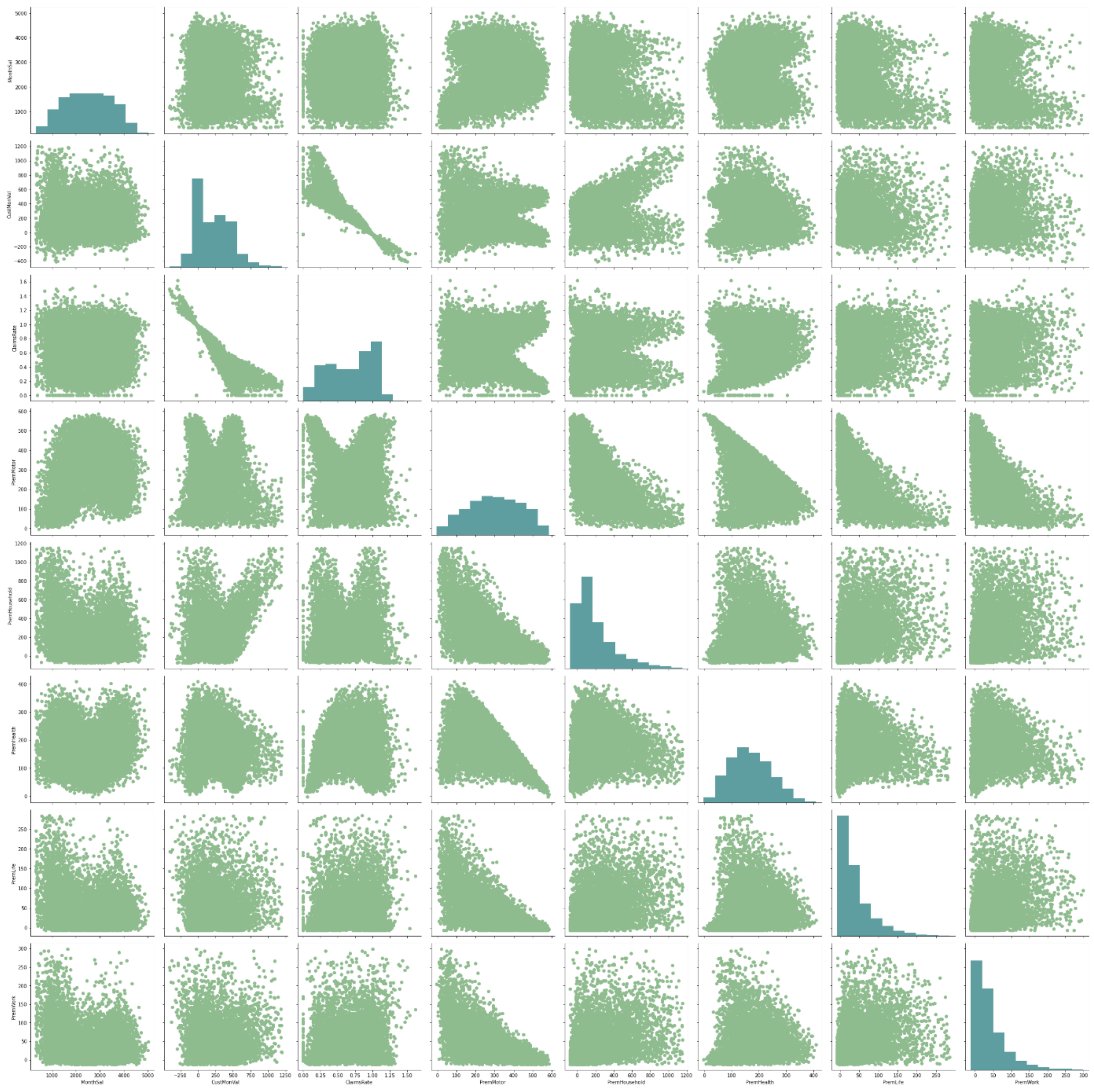
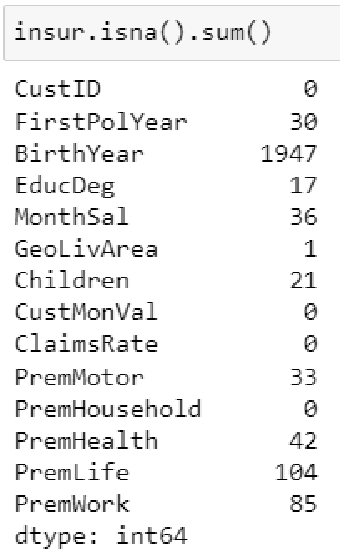


Figure 4 - Pairplot Grid after outlier removal



```
insur.isna().sum()
```

```
CustID             0
FirstPolYear      30
BirthYear       1947
EducDeg           17
MonthSal          36
GeoLivArea         1
Children          21
CustMonVal         0
ClaimsRate         0
PremMotor         33
PremHousehold      0
PremHealth        42
PremLife         104
PremWork          85
dtype: int64
```

Figure 5 - Missing Values per variable

Client_Recency: depicts the number of years since the customer first signed a policy with the customer (2016 – FirstPolYear).

Total_Premiums: depicts the total amount spent in premiums by every customer (PremMotor + PremHousehold + PremHealth + PremWork).

Annual_Salary: depicts the total salary earned by the customer for a period of 12 months (12 * MonthSal).

SalarySpent_Ratio: depicts the percentage of the total salary that the client has spent on insurance products (Total_Premiums / Annual_Salary).

PremMotor_Ratio: depicts the percentage of the amount spent in Motor related premiums, on the total expenditure of the customer (PremMotor / Total_Premiums).

PremHousehold_Ratio: depicts the percentage of the amount spent in Household related premiums, on the total expenditure of the customer (PremHousehold / Total_Premiums).

PremHealth_Ratio: depicts the percentage of the amount spent in Health related premiums, on the total expenditure of the customer (PremHealth / Total_Premiums).

PremLife_Ratio: depicts the percentage of the amount spent in Life related premiums, on the total expenditure of the customer (PremLife / Total_Premiums).

PremWork_Ratio: depicts the percentage of the amount spent in Work related premiums, on the total expenditure of the customer (PremWork / Total_Premiums).

Figure 6 – New Variables

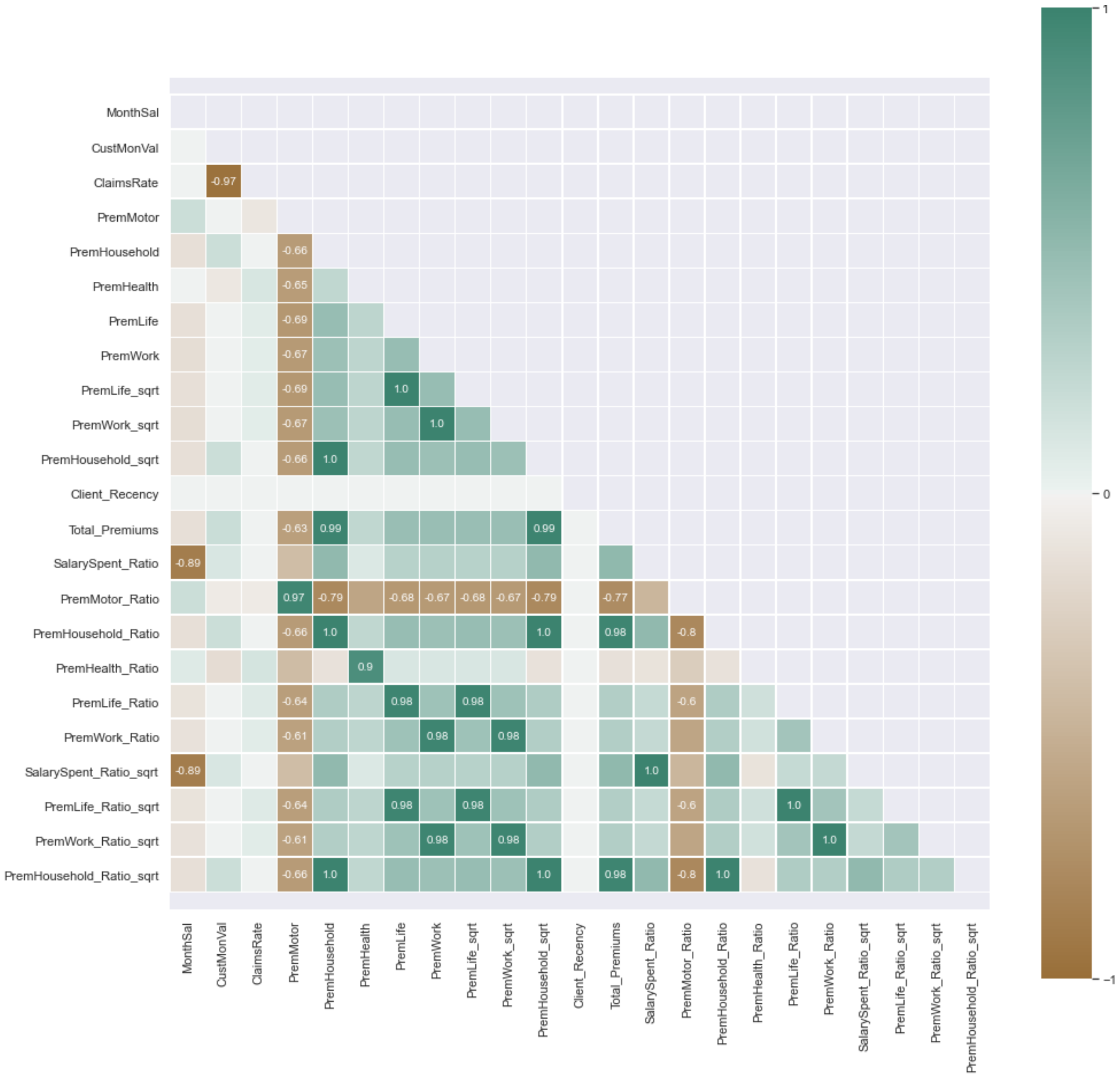


Figure 7 – Correlation Matrix

Dataset	**Variables**
value_df	*Client_Recency, MonthSal, CustMonVal*
value_df2	*Client_Recency, MonthSal, ClaimsRate*
value_df3	*Client_Recency, SalarySpent_Ratio, CustMonVal*
value_df4	*Client_Recency, SalarySpent_Ratio, ClaimsRate*
products_df	*PremMotor, PremHealth, PremLife_sqrt, PremWork_sqrt, PremHousehold_sqrt*
products_df2	*PremMotor, PremHealth, PremLife_sqrt, PremWork_sqrt, Total_Premiums*
products_df3	*PremMotor_Ratio, PremHealth_Ratio, PremLife_Ratio_sqrt, PremWork_Ratio_sqrt, PremHousehold_Ratio_sqrt*
products_df4	PremMotor_Ratio, PremHealth_Ratio, PremLife_Ratio_sqrt, PremWork_Ratio_sqrt, Total_Premiums

Table 3 – Segmentation Datasets and respective variables

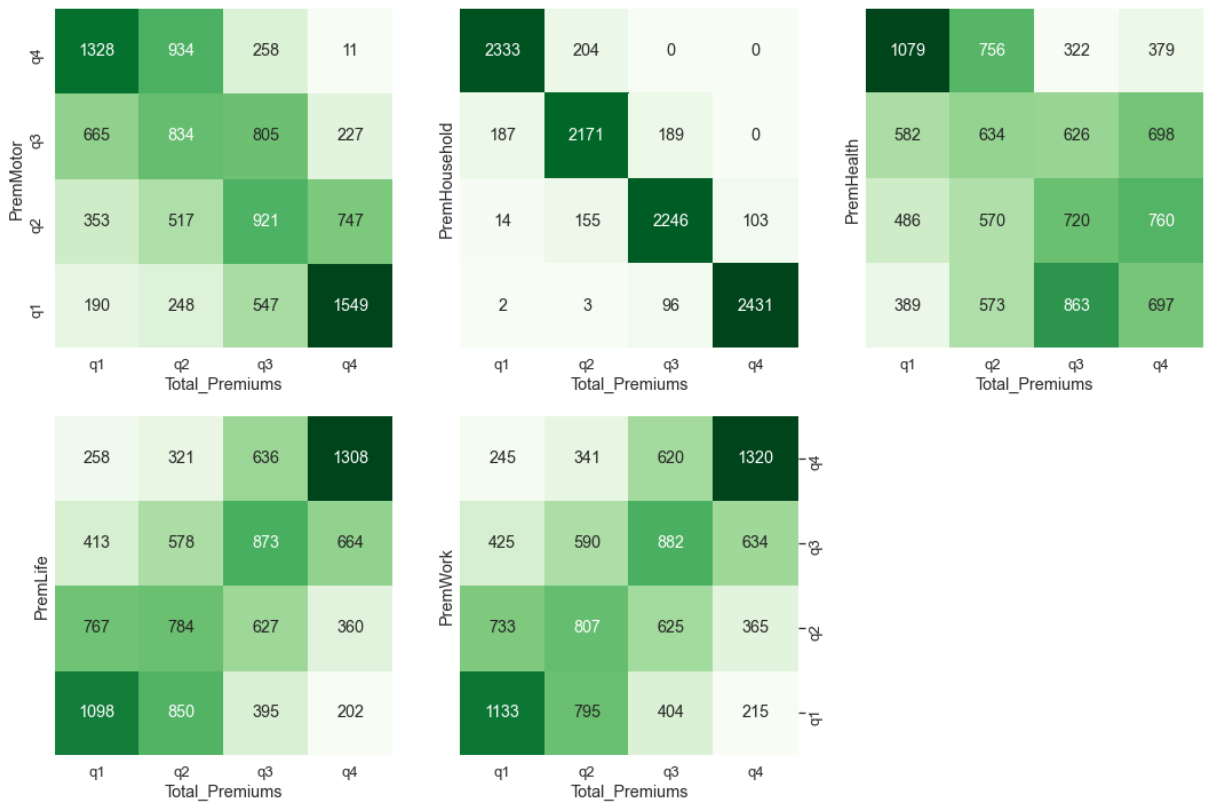
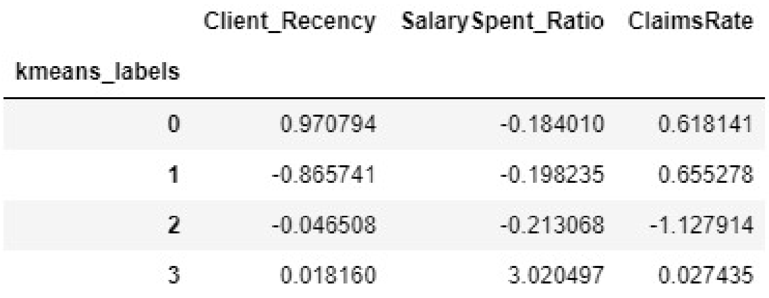


Figure 8 – Quartile Matrix

Method	Clusters	R^2	Silhouette
Hierarchical	4	0.5106	0.2631
K-means	4	0.5942	0.3379
Hierarchical on SOM Units	4	0.5236	0.2726
K-means on SOM Units	4	0.5781	0.3252
DBCSCAN	3	0.2012	0.404
Mean Shift	4	0.134	0.6631
Gaussian Mixture	4	0.4839	0.2406

Table 4 – Scores by algorithm and number of clusters: Value



	Client_Recency	SalarySpent_Ratio	ClaimsRate
kmeans_labels			
0	0.970794	-0.184010	0.618141
1	-0.865741	-0.198235	0.655278
2	-0.046508	-0.213068	-1.127914
3	0.018160	3.020497	0.027435

Figure 9 – Mean Values: Value – K-means

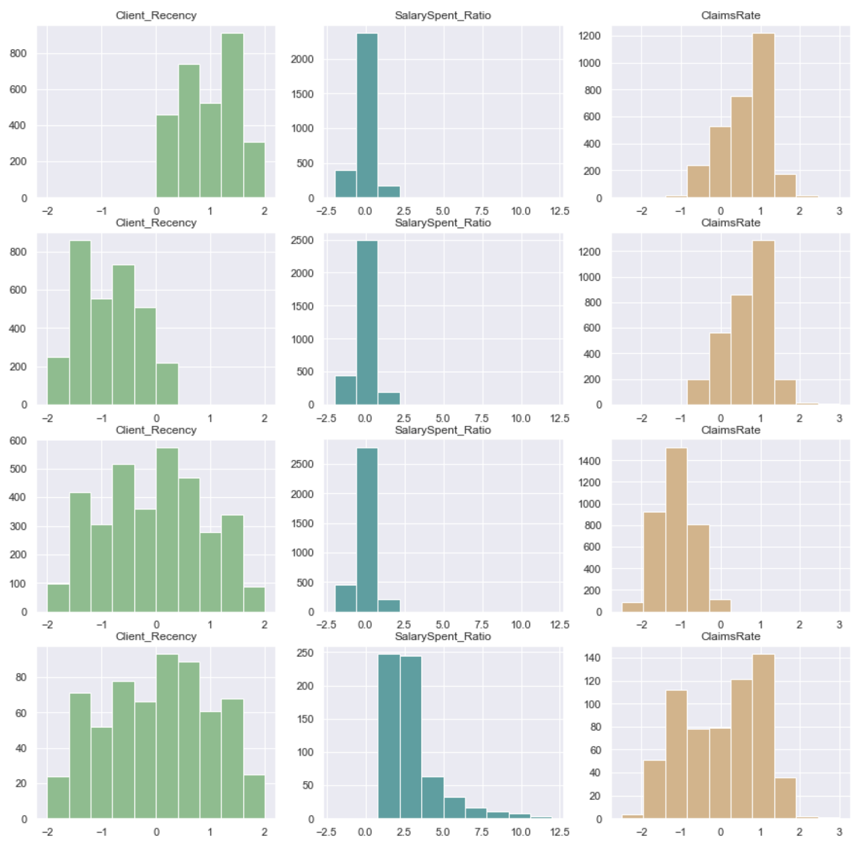
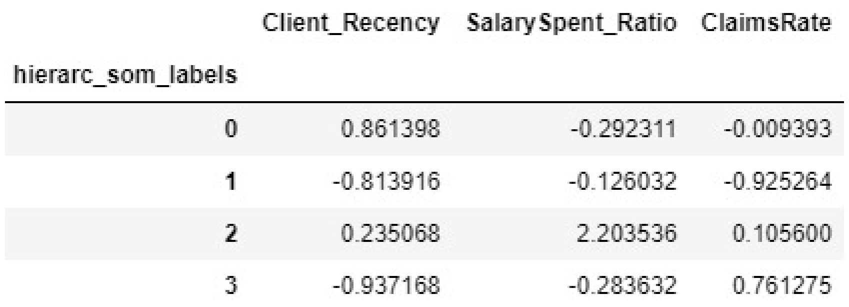


Figure 10 – Histograms: K-means – Value Segmentation



hierarc_som_labels	Client_Recency	SalarySpent_Ratio	ClaimsRate
0	0.861398	-0.292311	-0.009393
1	-0.813916	-0.126032	-0.925264
2	0.235068	2.203536	0.105600
3	-0.937168	-0.283632	0.761275

Figure 11 – Mean Values: Value – Hierarc on SOM Units

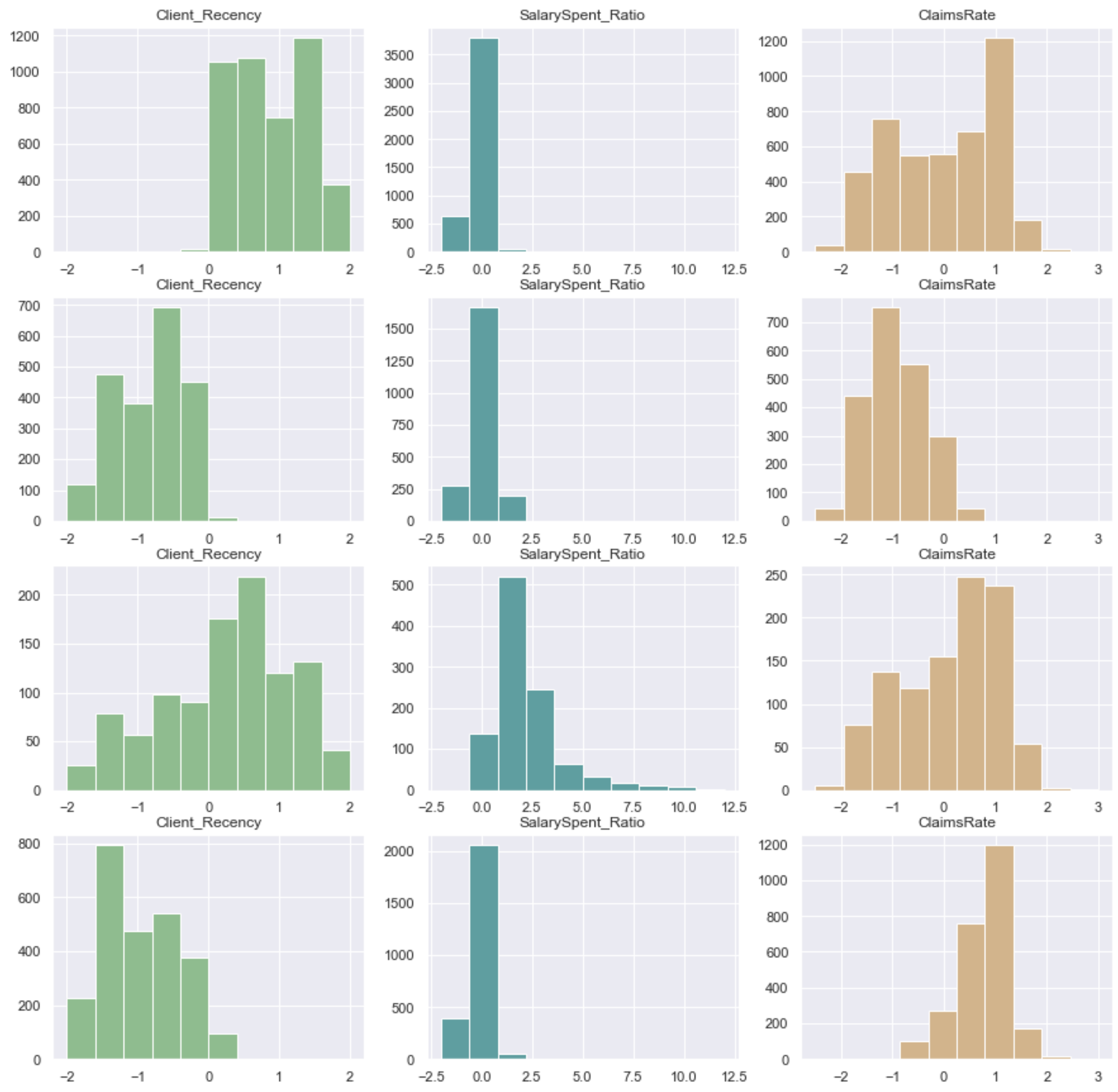
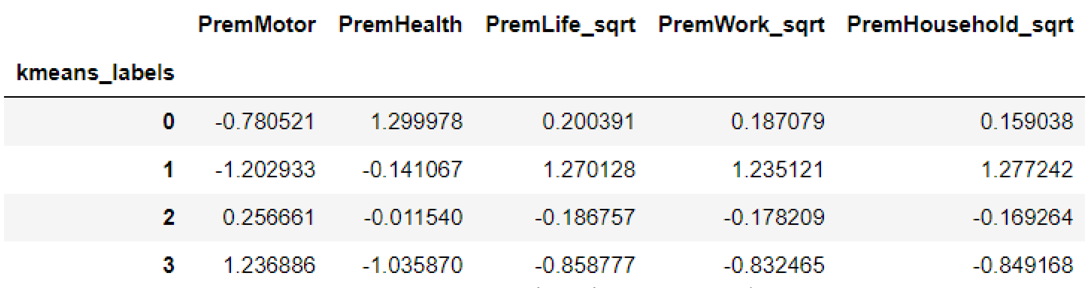


Figure 11 – Histograms: Hierarc on SOM Units – Value Segmentation

Method	Clusters	R^2	Silhouette
Hierarchical	4	0.5484	0.1764
K-means	4	0.6	0.2458
Hierarchical on SOM Units	5	0.6024	0.1785
K-means on SOM Units	4	0.5826	0.2357
DBCSCAN	4	0.1271	0.193
Mean Shift	2	0.005	0.2618
Gaussian Mixture	4	0.3382	0.2928

Table 5 – Scores by algorithm and number of clusters: Products



kmeans_labels	PremMotor	PremHealth	PremLife_sqrt	PremWork_sqrt	PremHousehold_sqrt
0	-0.780521	1.299978	0.200391	0.187079	0.159038
1	-1.202933	-0.141067	1.270128	1.235121	1.277242
2	0.256661	-0.011540	-0.186757	-0.178209	-0.169264
3	1.236886	-1.035870	-0.858777	-0.832465	-0.849168

Figure 12 – Mean Values: k-means - Product

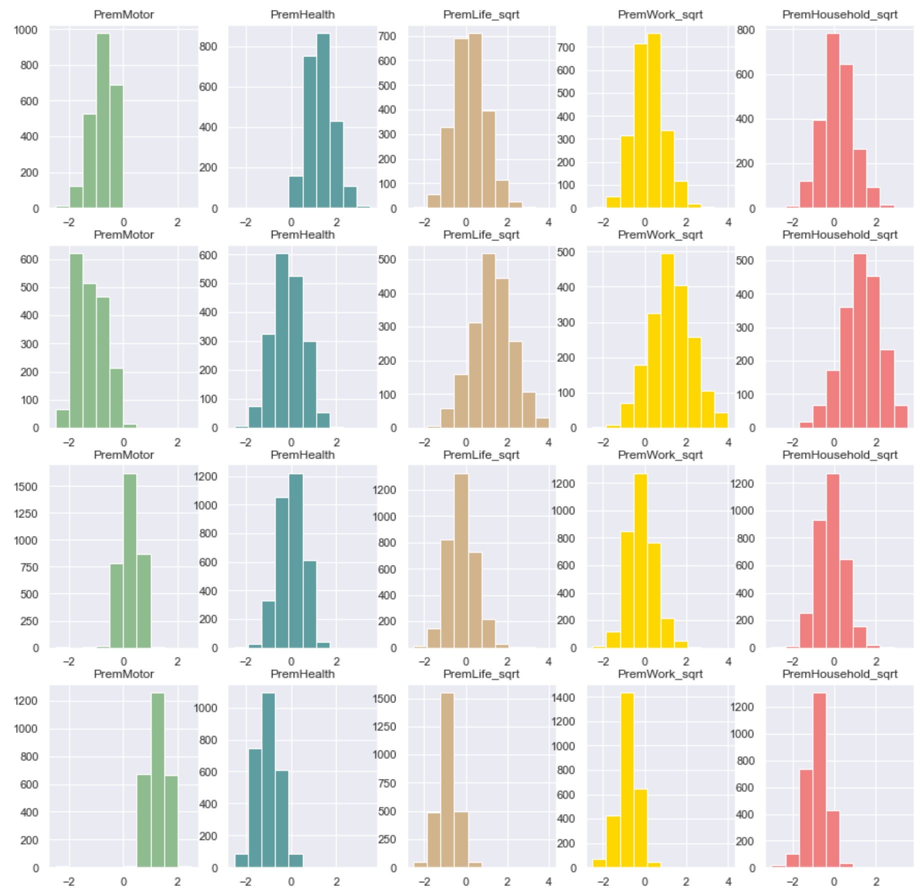


Figure 13 – Histograms: K-means – Product

kmeans_som_labels	PremMotor	PremHealth	PremLife_sqrt	PremWork_sqrt	PremHousehold_sqrt
0	0.123012	0.089935	-0.079444	-0.090430	-0.127815
1	-0.845880	1.371118	0.267323	0.284026	0.099771
2	1.158868	-0.939968	-0.812814	-0.774693	-0.783746
3	-1.160126	-0.162098	1.174719	1.111034	1.399708

Figure 14 – Mean Values: k-means on SOM Units- Product

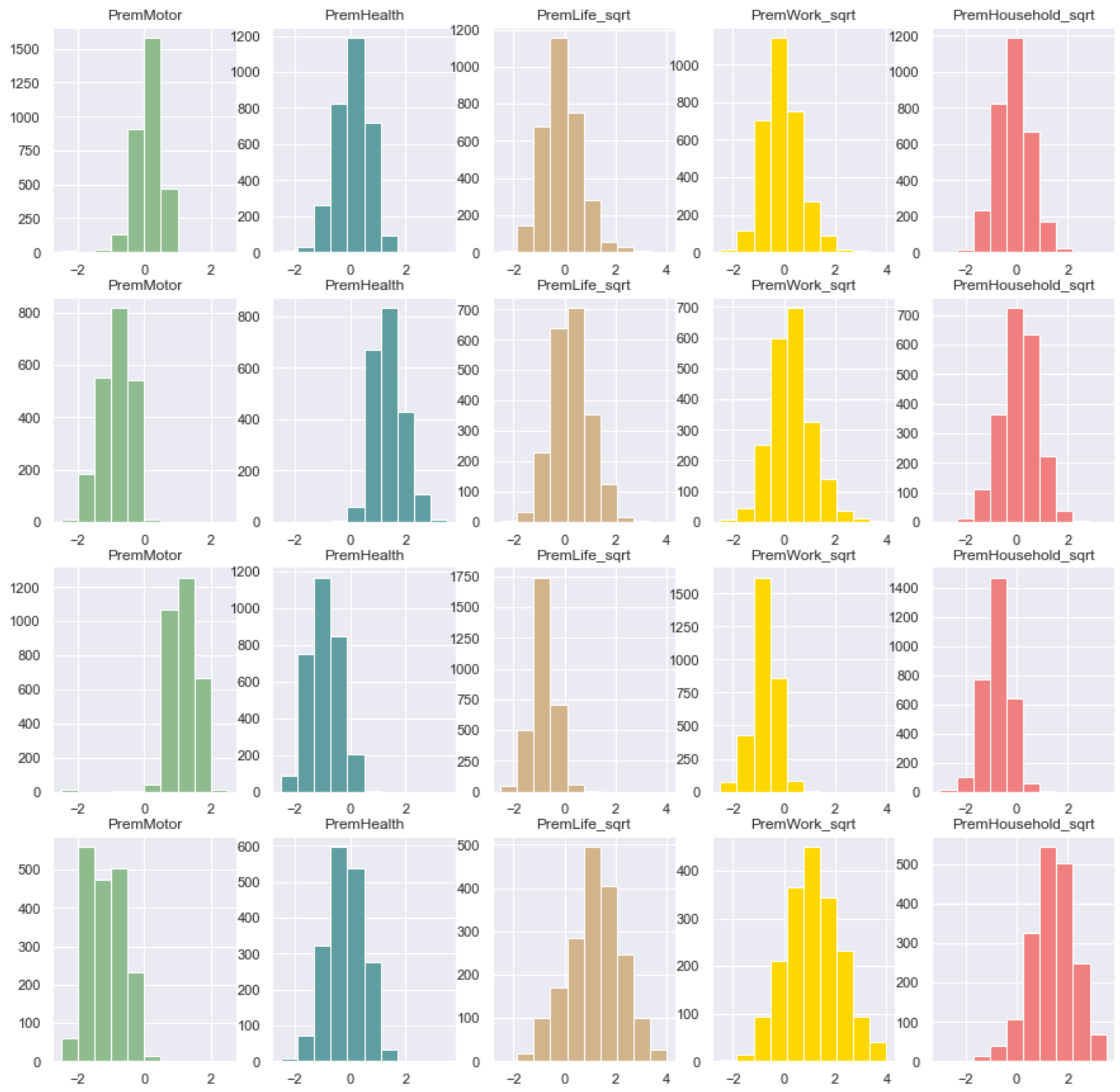
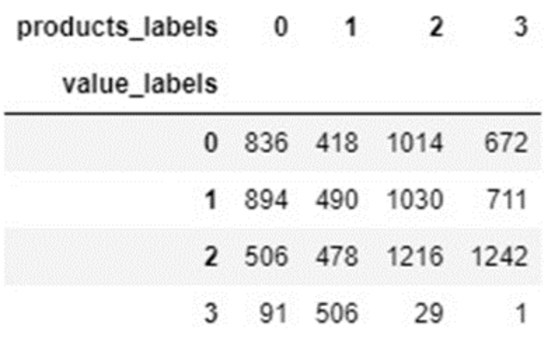


Figure 15 – Histograms: K-means on SOM units - Product



products_labels	0	1	2	3
value_labels				
0	836	418	1014	672
1	894	490	1030	711
2	506	478	1216	1242
3	91	506	29	1

Figure 16 – Cluster contingency table

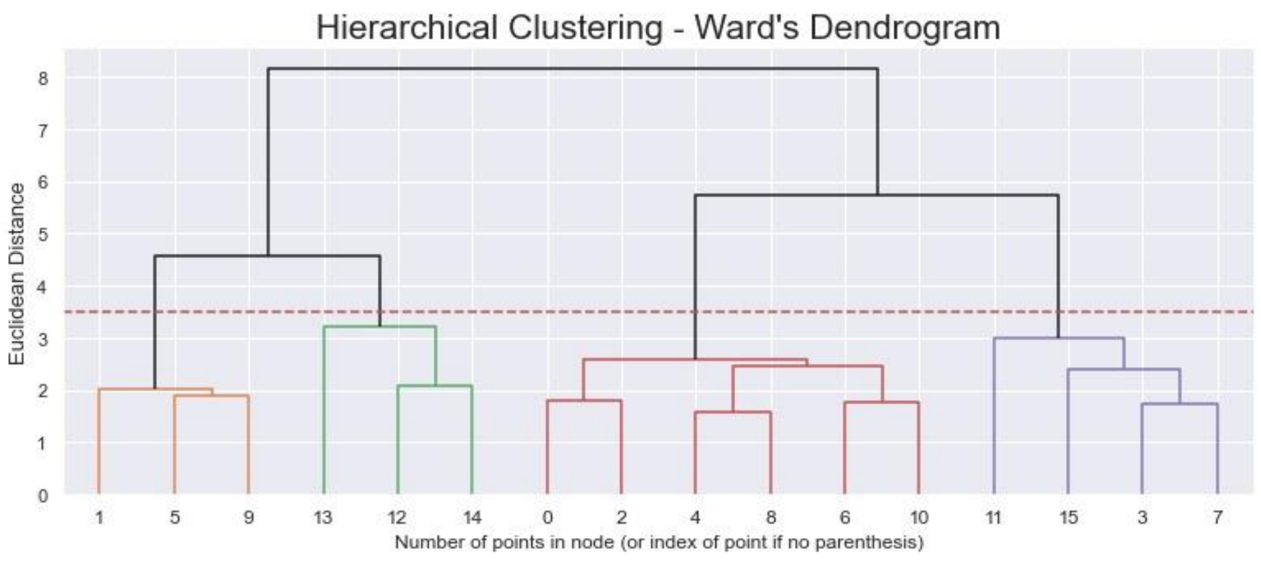
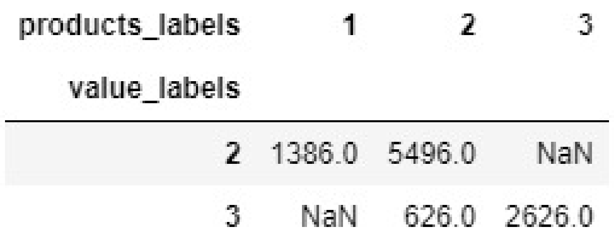


Figure 17 – Final Cluster Dendrogram



products_labels	1	2	3
value_labels			
2	1386.0	5496.0	NaN
3	NaN	626.0	2626.0

Figure 18 - Cluster contingency table

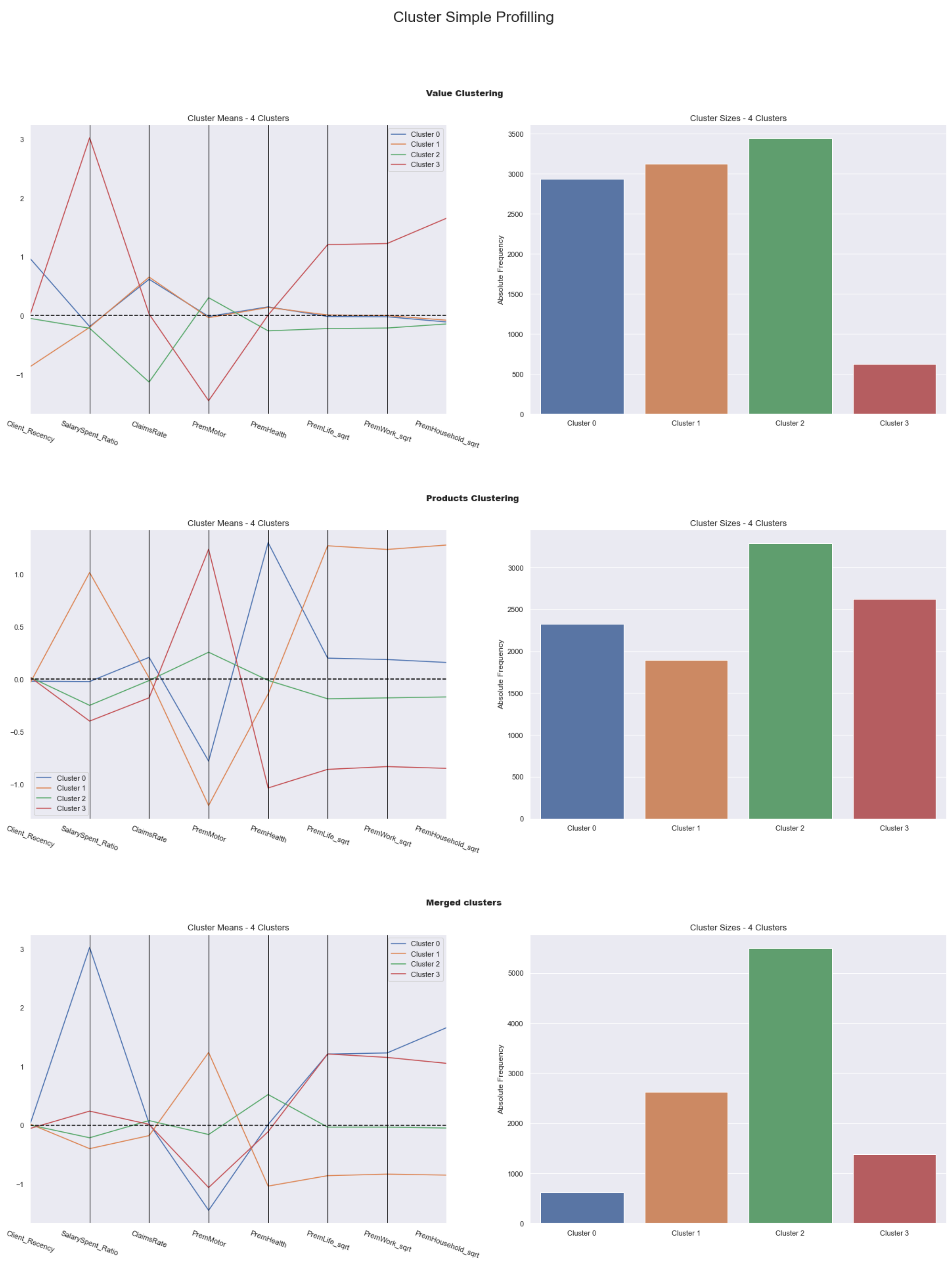


Figure 19 – Cluster Profile

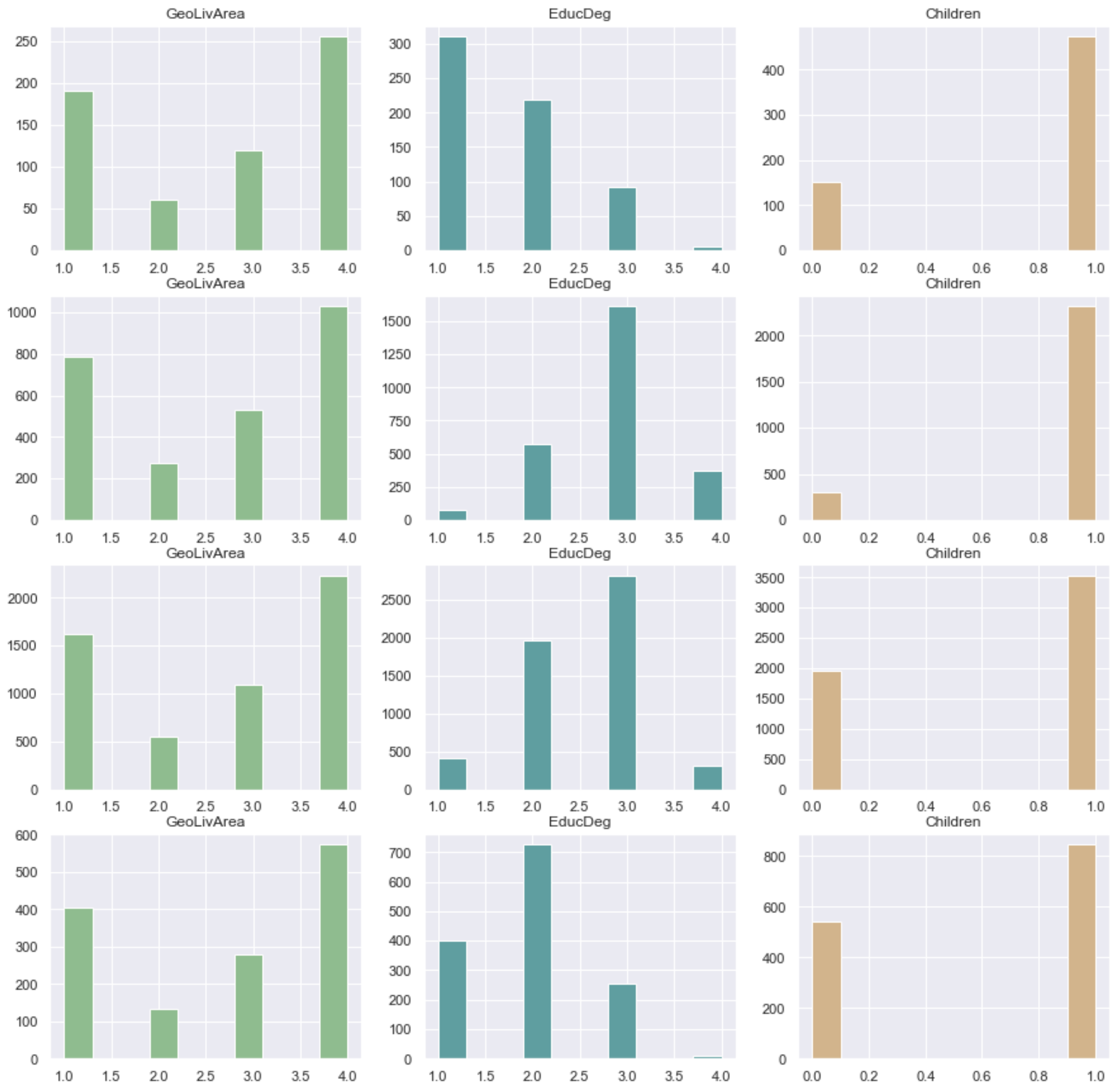


Figure 20 – Non-Metric Variable

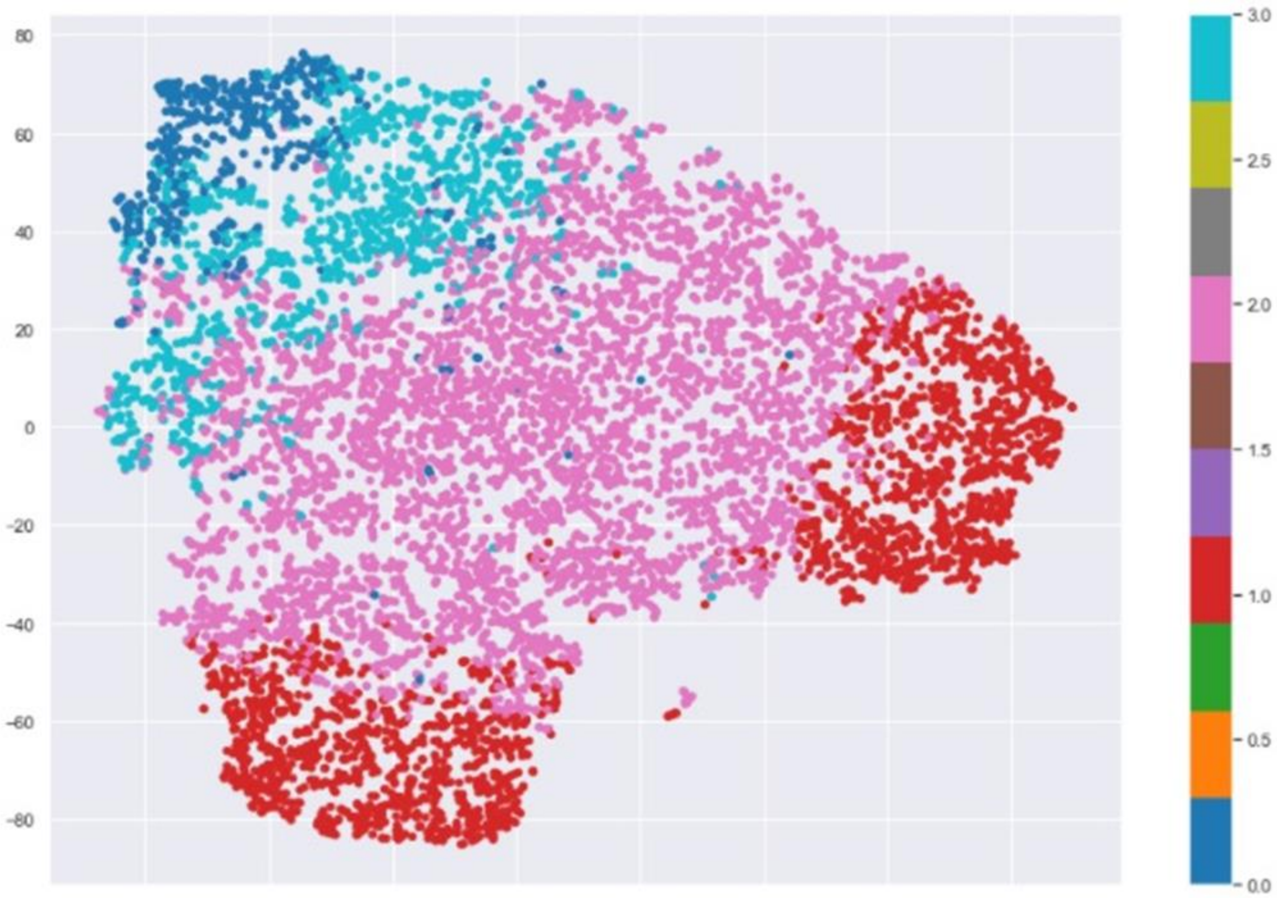
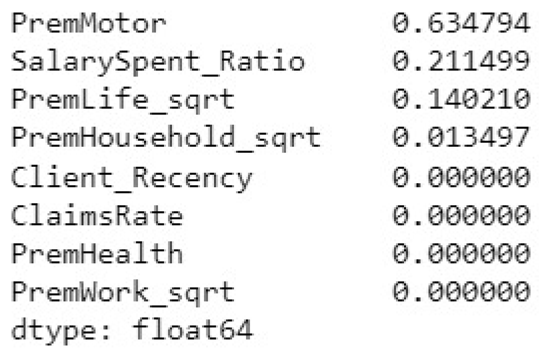


Figure 21 – t-SNE visualization

```
PremMotor              0.693457
SalarySpent_Ratio      0.638406
PremHousehold_sqrt     0.509434
PremLife_sqrt          0.482650
PremWork_sqrt          0.455096
PremHealth             0.427151
ClaimsRate             0.011530
Client_Recency         0.000552
dtype: float64
```

Figure 22 – Feature Importance with r^2



```
PremMotor              0.634794
SalarySpent_Ratio      0.211499
PremLife_sqrt          0.140210
PremHousehold_sqrt     0.013497
Client_Recency         0.000000
ClaimsRate             0.000000
PremHealth             0.000000
PremWork_sqrt          0.000000
dtype: float64
```

Figure 23 - Feature Importance with decision trees