Data Mining Project

MASTER DEGREE PROGRAM IN DATA SCIENCE AND ADVANCED ANALYTICS

Customer Segmentation

A2Z Insurance Company

Group U

Duarte Girão, number: 20220670

Sabeen Mubashar, number: 20220726

Wai Kong Ng, number: 20221384

January, 2023

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

INDEX

1. Introduction

The goal of this report is to segment the customer base of a fictional Portuguese company called *"AZ Insurance"*. We used data mining techniques to group our customers based on their consumer habits and preferences to provide the Marketing Department with a reliable characterization of our customer profile, which will lead to better future business decisions. In a nutshell, we aim to determine "*what is the value of each customer segment and which services are they most likely to purchase?"*

It is important to mention that "*AZ Insurance"* offers insurance services in various fields, including health, household, life, motor, and workers' compensation, and that our analysis was conducted with respect to the year 2016.

2. Initial Data and Variables Exploration

The initial dataset, which we named "*df"*, consists of 10,296 records and 14 variables. The variable "*CustID*" is a unique identifier for each record. As a precaution, we made a copy of the original dataset before starting our data preparation process. This is a standard practice that we followed throughout the project to ensure that we had a backup of the data at each stage and could work with a consistent version of the data. Even though the dataset is not particularly large, this measure helped us to maintain a high level of sensitivity and awareness about the "*DataFrame"* that we were working with.

3. Data Preparation

3.1 Coherency Checks

The Data Preparation phase was divided into four main topics. To begin, we conducted some coherency checks on the data and discovered that the only non-object variable, "*EducDeg*", has a float datatype (as shown **in table 1**). At this time, we chose not to transform "*EducDeg*", as we intended to encode it later on. Additionally, we confirmed that there were no duplicate values in our dataset (as shown **in table 2**).

3.2 Data Categorization

After coherency checks, we split our variables into two groups based on their nature: *metric* and *non-metric*. Besides "*EducDeg*", we also classified the variables "*GeoLivArea*" and "*Children*" as "*non-metric_features"*. The variable "*GeoLivArea*" is distributed across four areas codes but these codes have no relevant meaning and therefore we did not consider this variable as ordinal. The variable "*Children*" is a binary variable that takes on either a value of one or zero, indicating whether an individual has children or not. However, it does not discriminate regarding the number of children. The remaining eleven variables were labelled as "*metric_features".*

3.3 Missing Values

Moving on, we examined the missing values in our dataset (as shown **in table 3**). We found a total of 389 missing values across all variables, with "*PremLife*", "*PremWork*" and "*PremHealth*" having the most with 104, 86 and 43 respectively (as shown **in table 3**). Since this corresponds to nearly 4% of the entire dataset, we considered it very significant to discard them. So, we advanced to their imputation

according to their category:

- For the metric features, which corresponded of the majority missing values (around 90%), we decided to apply a more sophisticated method when compared to the median, in order to obtain the most accurate estimation for each of this missing values, "***MICE imputer***". Even though this method produces similar results as "*KNN Imputer*", on average, from our research this tends to be slightly more time efficient and to perform better when dealing with small datasets (see **reference [1]**). This imputer defines the missing value feature as the "*target*" and makes a regression model in order to estimate the value for the missing values, by taking the given values from the remaining variables as the "*training dataset*";
- For the remaining 39 missing values in the non-metric features, we did not considered them significant in comparison to the entire universe of our entire dataset and therefore, we chose to impute them using a less sophisticated method, the "***mode***".

3.3 Outliers Treatment

In the final subtopic of preprocessing, we employed two methods for removing outliers: manual and IQR. For the manual method, we defined thresholds per variable using the histograms and the boxplots (as shown in **graphs 1 and 2**). We adopted a gradual trial and error approach in order to determine the desired threshold with the exception of "*CustMonVal*", which required both a lower and upper threshold. Secondly, we applied IQR methods, which removed more than 15% of our dataset, more than acceptable in our view in this context. Finally, we combined both previous methods. In total, we deleted a total of 348 records, keeping 96,62% of our dataset, representing a new total number or records of 9948. To better understand the impact of this procedure on the dataset, we can compare the descriptive statistics before and after outlier removal (as shown in **tables 4 and 5**).

Additionally, we can also observe the histograms and boxplots before and after outlier removal (as shown in **graphs from 1 to 4**). We noticed that "*CustMonVal*" is slightly left skewed, "*Claims*" is slightly right skewed and among the premium variables, "*PremHousehold*", "*PremLife*" and "*PremWork*", are all significant left skewed. All the remaining variables are relatively normal distributed.

4. Feature Engineering

4.1. Variable Selection: Correlation Matrix

At this stage, we aimed either to add relevant or remove irrelevant ones in order to obtain a cleaner and more accurate dataset. In this sense, we found it useful to replace the variables "*BirthYear*" with "*Age*" and "*FirstPolYear*" with "*PolicyAge*" for easier interpretation in both cases.

Regarding feature removal, we took two main approaches:

- **Redundancy**: to avoid including highly correlated variables that explain a similar variation of the dataset, we created a Correlation Matrix (as shown in **graph 5**). From this, we can observe we observed a correlation of 0.92 between "*MonthSal*" and "*Age*", a correlation of -0.94 between "*CustMonVal*" and "*ClaimsRate*" and correlations above 0.65 between "*PremMotor*" and the remaining variables. We decided to keep all these variables despite this correlation values, as we believed they would be relevant for future costumer segmentation.

- **Relevancy**: requires a deep understanding of the relevant variables within the business industry in question and is subject to a higher degree of judgment. We adopted a conservative approach and the only irrelevant variable for our future analysis, was "*CustID*", so we dropped it.

After these procedures, we reevaluated the correlations again (as shown in **graph 6**).

4.2. One Hot Encoding

Moving on, we decided to encode all our non-metric variables to obtain a cleaner and uniform dataset. So, at this stage of our analysis, our dataset is composed by 20 variables, with the non-metric only taking binary values. As previously mentioned, we made a safety copy of the data before performing the encoding on the "***ohc_df***" dataset.

4.3. Data Normalization

Finally, before starting our clustering process, we also performed a last key data mining technique, scaling the data. Again, before applying change to our current DataFrame, we made another safety copy, saving it as "***df_before_scaling***".

From the next step onwards, we are going to focus our analysis on data clustering under different algorithms approaches, which measure the distance between points in their calculations. Once features naturally have different ranges of values and totally different scales, this step is essential to avoid a disproportionate and unrealistic impact from any features in the overall clustering output.

The two principal normalization methods learned throughout this course were "*MinMax*" and "*StandardScaler*". The first one, compresses the data in the range from 0 to 1, while the second, also known as Z-score, compacts the data, returning the average as 0 and the standard deviation as 1. Although both methods use to produce good results, on average, we decided to choose "*StandardScaler*" as long as it provided us more interesting results in later stages. Before moving on to data clustering, we checked the descriptive statistics table once again to ensure that all steps up to this point had been correctly implemented (as shown in **table 6**). We can confirm that the mean and the standard devation of all the metric features at this point are 0 and 1 respectively.

5. Clustering Algorithm | K-Means

After finishing all the data preparation and feature engineering procedures, we started our cluster analysis. We implemented six algorithms on top of our current dataset and an additional clustering by perspectives. Each of these algorithm's implementation was stored in an appropriate DataFrame which allowed us, in the end, to compare them in the end and present our conclusions based on the algorithm with best results.

5.1. Context and Procedures

By picking a predefined number of random observations, this Kmeans algorithm creates clusters and put the nearest observation inside them. After that the clusters calculate the mean of the observations and put in other observations continuously until all the observations are assigned to a cluster.

5.2. Key Conclusions

We discover that the most effective number of clusters is 4. We calculated the R2 and the silhouette score, which are **~0.45** and **~0.211**, respectively.

6. Clustering Algorithm | Batch K-Means

6.1. Context and Procedures

Instead of collecting observations after one another, this variation of K-means calculates a batch of observations on each iteration. By doing so the algorithm saves a lot of time and memory especially in a huge database.

6.2. Key Conclusions

We have used the Elbow Method, which lead us to the conclusion that the perfect number of clusters is 6, as long as it presents the lower SSW value. Even tough Silhouette score indicated us a perfect number of 4, as shown in graph, we decided to follow inertia conclusions.

7. Clustering Algorithm | Hierarquical Clustering

7.1. Context and Procedures

This algorithm initially considers every data point as an individual cluster and merges the nearest pairs of the cluster continuously until they form into one cluster. There are several ways of merging and we tested single, average, complete and ward. After comparing R2 scores, we decided to use ward. To determine the most effective number of clusters, we used Euclidean distance. As the result, we found that the best number of clusters is 4.

7.2. Key Conclusions

We choose our final number of clusters from the dendrogram. There are several methods for determining the similarity between clusters in hierarchical clustering, including single linkage, complete linkage, average linkage, and centroid linkage. The choice of linkage method can significantly impact the resulting dendrogram and the final clusters obtained. We choose 'ward' as a linkage method and use Euclidean distance as distance metric, even though it is not the best performing linkage. We can observe that the differences between all the linkages are very insignificant and as long as 'ward' performed the best, we choose. In all of 4 clusters monthly salary contributed most followed by *CustMonthVal* and claims rate.

8. Clustering Algorithms | Using Different Perspectives

8.1. Segmentation Creation

At this moment, after applying K-means, Batch K-means and Hierarquical Clustering and before applying the density graphs, we decided to segmentate our variables according to each feature meaning. We divided our dataset into two groups:

- **Value demographic features**, including variables "*Age", "ClaimsRate", "CustMonVal", "MonthSal" and "PolicyAge*". We stored our current Data Frame filtered by these variables a

new dataset called "**df_value_dem".**

- **Premium features**, including PremHealth, PremHousehold, PremLife, PremMotor and PremWork. We stored our current Data Frame filtered by these variables a new dataset called "**df_premium".**

The first group pretends to segmentate our customers according to their purchasing power, to their loyalty and to their potential global value for AZ company. The second group pretends to characterize customers according to their favorite personal preferences. Both segments pretend to answer our initials two open questions: "*what is the value of each customer segment and which services are they most likely to purchase?"*

8.2. Context and Procedures

Firstly, we performed *K-Means* on both segments in order to understand what the most appropriate number of clusters for each group were. For this, we draw the graph showing us the estimated R square for each linkage method and achieved the same conclusion for both segments: 3 clusters. By looking at **table 13**, we are able to identify the distribution of the observations per label. However, as we can see by the results, we got still some disproportionate results, which means that the smallest groups can still be joined to a bigger close cluster. In this sense, we performed the dendrogram (from *Hierarquical Clustering method*), in order to merge both perspectives. As we can see in graph 10, cut the dendrogram with an Euclidean Distance, getting a final number of clusters equal to 3. This final 3 clusters results can be seen in **table 14**. By comparing both **tables 13 and 14**, we can notice that both clusters 0 and 1 from "*value demographic"* group disappeared, meaning that those observations were joined with a nearby cluster.

8.3. Key Conclusions

Regarding this merging perspective, we evaluated the quality of the final cluster output by using two methods. The first one, R2, we were able to achieve a final clustering solution of **~0.3393**. The second one, Decision Tree, we were able to correctly estimate, on average, 95% of costumers correctly.

9. Clustering Algorithm | Density | Mean Shift Clustering

9.1. Context and Procedures

Mean shift algorithm is a density-based algorithm that works by shifting points towards the mean of the points within a certain neighborhood around them and repeating this process until the points no longer move significantly. The points that end up in the same location form a cluster. The mean shift algorithm has one key parameter: the bandwidth, which determines the size of the neighborhood around each point.

9.2. Key Conclusions

The shift mean algorithm has one key parameter: the bandwidth and the estimated number of clusters are 3 with r-squared 0.139. Older Customers will have a high monthly salary with the highest contribution towards Premium Motor and the customer with low monthly salary will buy Premium Life insurance the most.

10.Clustering Algorithm | Density | DBSCAN

10.1. Context and Procedures

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a density-based algorithm which is used to find dense regions in a dataset. It makes a cluster from all the points in ε –neighborhood. It is a density-based algorithm; therefore, it does not matter whether data points are evenly distributed or dispersed randomly.

10.2. Key Conclusions

Eps and MinPts are two essential parameters for the DBSCAN algorithm. Eps radius defining the neighborhood.. The minimal number of points (MinPts) are necessary to establish a dense zone. For our algorithm we define eps=1.9 and MinPts = 20. The estimated number of clusters are 2. Cluster final solution with R square is 0.0258 with PremWork and Prem household contribute the most.

11.Clustering Algorithm | Density | Gaussian

11.1. Context and Procedures

A Gaussian mixture model (GMM) is a probabilistic model that assumes all the data points in a dataset come from a mixture of a finite number of Gaussian distributions with unknown parameters. The GMM algorithm extends this idea by assuming that each cluster is represented by a multivariate normal distribution (a Gaussian distribution). Each cluster is also associated with a weight that represents the proportion of the data points that belong to that cluster.

11.2. Key Conclusions

The parameter for Gaussian Mixture Models we choose number of components =3 , covariance type=full, n_init=10, init_params='kmeans'. Cluster final solution with R square is 0.2601 with Claim Rate and PremWork contributing the most..

12. Final Conclusions | Merging Perspectives | Profiling Analysis

Finally, after completing all the clusters, we were able to profile our final cluster solution in other to obtain a visual and easier interpretation of our results (as shown in **graph 7**). We chose Merging Perspectives clustering as the base for our analysis as long as it combines both *K-Means* and *Hierarquical Clustering* and provided interesting results.

In order to complement our analysis from the cluster, we computed the transpose of our values before scaling, allowing us to have a more real understanding from them (as shown **table 15**). So, from the **graph 7**, we can observe that ***cluster 0 and cluster 2*** are the biggest clusters by a significant distance when compared to ***cluster 2***. From **table 15**, we can have a really accurate understanding of each variable mean value before scaling, allowing us to have a better sensibility on data. We draw the main conclusions below on *Key Takeaways* section.

12.1. Final Conclusions | Merging Perspectives | Feature Importance

Beyond previous conclusions described above, we also computed the individual R2 for each variable (as shown in **table 15**) and plotted a decision tree. From this table, we can conclude that the variables that most contributed to our final cluster output were *PremMotor, PremHealth* and *PremHousehold,* with respective R2 of **~0.75, ~0.51** and **~0.46**. The variables "*PolicyAge* and *ClaimsRate* have the minimum impact on cluster final solution, meaning that they are not very good variables discrimating the final cluster output. This is in accordance with the results present on **graph 7**, as long as the mean is very close to zero for the three clusters.

12.2. Final Conclusions | Merging Perspectives | Key Takeaways

Finally, we reached the point where we make final considerations regarding each cluster:

Cluster 0: This cluster contains 4083 customers with an average *age* of 52 (the older cluster), represents the highest *monthly salary* and *PremHealth* value and has the lowest *customer monetary value*.

Cluster 1: This cluster contains 1399 customers with an average *age* of 30 (the younger cluster), and represents the lowest *monthly salary*, the highest *customer monetary value*, *PremHousehold, PremLife* and *PremWork* values and represents the less educated people from the overall dataset.

Cluster 2: This cluster contains 4466 customers (the biggest one) with an average age of 50, represents the highest *PremMotor* value and the highest *educated individuals* and tends to have *children*.

12.3. Final Conclusions | Merging Perspectives | Marketing Strategies

In order to address the **introductory questions from our report**, we would like to propose the following marketing strategies:

Marketing Strategy 1: This proposal is directed to **cluster 0** and consists in exploring the fact that it has a high Premium Health value. AZ Insurance could try to incorporate incentives with these clients to buy a wide diversity of services in the company, from other sectors. There could be created some discounts on services from other sectors for a certain level of Health-related expenditure on the company. Additionally, since this cluster has the highest *monthly salary,* we could try to promote high-quality (or even luxurious) services near this segment.

Marketing Strategy 2: This proposal is directed to **cluster 1** and tries to explore the age effect of this cluster, especially when compared to the remaining two clusters (since there is a twenty year gap between them). We suggest AZ Insurance to be increase its presence in student events and to promote itself near universities. Additionally, since this category also have the lowest *monthly salary* level, it must very likely reflect in a lower monthly budget. Then, we suggest directing special promotions periodically to this customers and publicity them the most accessible services in the company.

Marketing Strategy 3: This proposal is directed to **cluster 2** and seeks, similarly to cluster 1, to explore the fact that it has a high Premium Motor value. We propose *AZ* to promote fiscal incentives to reward the most loyal and valuable Motor sector clients, by offering special offers on the remaining services. Additionally, once the company has the highest educational level we could try to explore

the educational component by creating partnerships with museums, universities and libraries, offering special conditions for this segment. Furthermore, we could still explore children effect, by creating some special family offers for activities programs. Lastly, as long as this cluster represents the biggest cluster, we could promote some loyalty cards system, also providing the customers a wide range of possible discounts on AZ partnership companies. This would be important to satisfy the core segment of AZ´s costumers.

13. References

1. Zhang, Z. (2016). Multiple imputation with multivariate imputation by chained equation (MICE) package. Annals of translational medicine, 4(2).

2. Lloyd, Stuart P. "Least squares quantization in PCM." Information Theory, IEEE Transactions on 28.2 (1982): 129-137

3. Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD'96). AAAI Press, 226–231.

4. Debomit Dey(2019). ML | Mini Batch K-means clustering algorithm. https://www.geeksforgeeks.org/ml-mini-batch-k-means-clustering-algorithm

5. Carreira-Perpinán, M. A. (2015). A review of mean-shift algorithms for clustering. arXiv preprint arXiv:1503.00687.

6. Nielsen, F. (2016). Hierarchical clustering. In Introduction to HPC with MPI for Data Science (pp. 195-211). Springer, Cham.

7. Johnson, S. C. (1967). Hierarchical clustering schemes. Psychometrika, 32(3), 241-254.

8. Pham, D. T., Dimov, S. S., & Nguyen, C. D. (2005). Selection of K in K-means clustering. Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science, 219(1), 103-119..

14. Appendix (Doesn’t count for the 10page limit)

14.1. Tables

```
# Check dataset data types.

df.dtypes
```

```
CustID           float64
FirstPolYear     float64
BirthYear        float64
EducDeg           object
MonthSal         float64
GeoLivArea       float64
Children         float64
CustMonVal       float64
ClaimsRate       float64
PremMotor        float64
PremHousehold    float64
PremHealth       float64
PremLife         float64
PremWork         float64
dtype: object
```

Table 1 – "Initial *Datatypes*"

```
# Check the duplicate values on our dataset.

df.duplicated().sum()
```

```
0
```

Table 2 – "Initial Duplicate check"

```
# Check the missing values on our dataset.

df.isna().sum()
```

```
CustID             0
FirstPolYear      30
BirthYear         17
EducDeg           17
MonthSal          36
GeoLivArea         1
Children          21
CustMonVal         0
ClaimsRate         0
PremMotor         34
PremHousehold      0
PremHealth        43
PremLife         104
PremWork          86
dtype: int64
```

Table 3 – "Initial *Missing Values*"

```
# Check descriptive statistics again before replacing missing values.

df.describe(include="all").T
```

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
CustID	10296.0	NaN	NaN	NaN	5148.5	2972.34352	1.0	2574.75	5148.5	7722.25	10296.0
FirstPolYear	10266.0	NaN	NaN	NaN	1991.062634	511.267913	1974.0	1980.0	1986.0	1992.0	53784.0
BirthYear	10279.0	NaN	NaN	NaN	1968.007783	19.709476	1028.0	1953.0	1968.0	1983.0	2001.0
EducDeg	10279	4	b'3 - BSc/MSc'	4799	NaN	NaN	NaN	NaN	NaN	NaN	NaN
MonthSal	10260.0	NaN	NaN	NaN	2506.667057	1157.449634	333.0	1706.0	2501.5	3290.25	55215.0
GeoLivArea	10295.0	NaN	NaN	NaN	2.709859	1.266291	1.0	1.0	3.0	4.0	4.0
Children	10275.0	NaN	NaN	NaN	0.706764	0.455268	0.0	0.0	1.0	1.0	1.0
CustMonVal	10296.0	NaN	NaN	NaN	177.892605	1945.811505	-165680.42	-9.44	186.87	399.7775	11875.89
ClaimsRate	10296.0	NaN	NaN	NaN	0.742772	2.916964	0.0	0.39	0.72	0.98	256.2
PremMotor	10262.0	NaN	NaN	NaN	300.470252	211.914997	-4.11	190.59	298.61	408.3	11604.42
PremHousehold	10296.0	NaN	NaN	NaN	210.431192	352.595984	-75.0	49.45	132.8	290.05	25048.8
PremHealth	10253.0	NaN	NaN	NaN	171.580833	296.405976	-2.11	111.8	162.81	219.82	28272.0
PremLife	10192.0	NaN	NaN	NaN	41.855782	47.480632	-7.0	9.89	25.56	57.79	398.3
PremWork	10210.0	NaN	NaN	NaN	41.277514	51.513572	-12.0	10.67	25.67	56.79	1988.7

Table 4 – "Descriptive Statistics before replacing missing values"

```
# Check descriptive statistics again after replacing missing values.

df.describe(include="all").T
```

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
CustID	10296.0	NaN	NaN	NaN	5148.5	2972.34352	1.0	2574.75	5148.5	7722.25	10296.0
FirstPolYear	10296.0	NaN	NaN	NaN	1991.06427	510.522485	1974.0	1980.0	1986.0	1992.0	53784.0
BirthYear	10296.0	NaN	NaN	NaN	1968.005833	19.698539	1028.0	1953.0	1968.0	1983.0	2001.0
EducDeg	10296	4	b'3 - BSc/MSc'	4816	NaN	NaN	NaN	NaN	NaN	NaN	NaN
MonthSal	10296.0	NaN	NaN	NaN	2506.645912	1156.237689	333.0	1706.0	2502.0	3290.0	55215.0
GeoLivArea	10296.0	NaN	NaN	NaN	2.709984	1.266293	1.0	1.0	3.0	4.0	4.0
Children	10296.0	NaN	NaN	NaN	0.707362	0.454996	0.0	0.0	1.0	1.0	1.0
CustMonVal	10296.0	NaN	NaN	NaN	177.892605	1945.811505	-165680.42	-9.44	186.87	399.7775	11875.89
ClaimsRate	10296.0	NaN	NaN	NaN	0.742772	2.916964	0.0	0.39	0.72	0.98	256.2
PremMotor	10296.0	NaN	NaN	NaN	300.375665	211.805964	-4.11	190.48	298.555	407.52	11604.42
PremHousehold	10296.0	NaN	NaN	NaN	210.431192	352.595984	-75.0	49.45	132.8	290.05	25048.8
PremHealth	10296.0	NaN	NaN	NaN	171.583737	295.7889	-2.11	111.91	162.92	218.9575	28272.0
PremLife	10296.0	NaN	NaN	NaN	41.721677	47.287491	-7.0	9.89	25.56	57.01	398.3
PremWork	10296.0	NaN	NaN	NaN	41.180808	51.334234	-12.0	10.67	25.67	56.12	1988.7

Table 5 – "Descriptive Statistics before after missing values"

```
# Check descriptive statistics again after normalization (rounded to 3 decimal places).

df[metric_features].describe().T.round(3)
```

	count	mean	std	min	25%	50%	75%	max
MonthSal	9948.0	0.0	1.0	-2.273	-0.794	0.006	0.799	2.576
CustMonVal	9948.0	-0.0	1.0	-2.361	-0.914	-0.112	0.751	3.829
ClaimsRate	9948.0	-0.0	1.0	-2.152	-0.916	0.130	0.954	2.730
PremMotor	9948.0	-0.0	1.0	-2.261	-0.761	0.016	0.795	2.103
PremHousehold	9948.0	0.0	1.0	-1.278	-0.690	-0.310	0.406	4.296
PremHealth	9948.0	-0.0	1.0	-2.329	-0.762	-0.065	0.697	2.876
PremLife	9948.0	0.0	1.0	-1.111	-0.702	-0.343	0.371	4.395
PremWork	9948.0	0.0	1.0	-1.231	-0.691	-0.314	0.370	4.439
Age	9948.0	0.0	1.0	-1.970	-0.854	0.028	0.850	1.908
PolicyAge	9948.0	-0.0	1.0	-1.822	-0.903	0.005	0.913	1.822

Table 6 – "Descriptive Statistics after Preprocessing"

```
# Compute the transpose (before scalling for the sake of an easier interpretation).

df_before_scaling_kmeansT = df_before_scaling_kmeans_mean.T
df_before_scaling_kmeansT
```

df_km_labels	0	1	2	3
MonthSal	3555.281265	1427.770087	2497.376482	2522.610814
CustMonVal	188.455131	247.197160	-7.112938	439.516133
ClaimsRate	0.729832	0.700354	0.977137	0.295929
PremMotor	230.356393	159.503005	412.865467	409.135895
PremHousehold	237.381543	375.357228	74.473171	97.650470
PremHealth	216.339634	206.525007	126.983136	124.872999
PremLife	48.231102	74.051393	16.783381	17.714202
PremWork	46.378901	73.908822	16.566695	17.358519
Age	67.173089	28.240665	47.989893	48.528699
PolicyAge	29.634554	29.885091	30.189687	30.173669
x0_b'1 - Basic'	0.107048	0.235394	0.047450	0.049408
x0_b'2 - High School'	0.409143	0.443497	0.268090	0.252348
x0_b'3 - BSc/MSc'	0.440381	0.303625	0.580071	0.585953
x0_b'4 - PhD'	0.043429	0.017484	0.104389	0.112291
x1_0.0	0.704000	0.163326	0.125346	0.140874
x1_1.0	0.296000	0.836674	0.874654	0.859126
x2_1.0	0.280381	0.306183	0.294187	0.303797
x2_2.0	0.106667	0.095949	0.106366	0.096366
x2_3.0	0.191238	0.200426	0.208383	0.202123
x2_4.0	0.421714	0.397441	0.391064	0.397713

Table 7 – "Kmeans means before scaling"

```
Kmeans Cluster final solution with R² of 0.4373

: PremMotor        0.676146
  Age              0.651695
  MonthSal         0.603409
  ClaimsRate       0.598067
  CustMonVal       0.427621
  PremHealth       0.343832
  PremWork         0.327722
  PremLife         0.323240
  PremHousehold    0.319617
  PolicyAge        0.001235
  dtype: float64
```

Table 8 – "Kmeans final R2 total and per variable"

```
# Characterize the final clusters (before Scalling).

df_before_scaling_bkmeans = pd.concat((df_before_scaling, pd.Series(df_bkm_labels, inde
df_before_scaling_bkmeans_mean = df_before_scaling_bkmeans.groupby('bkm_labels').mean(
df_before_scaling_bkmeans_mean.T
```

bkm_labels	0	1	2	3	4	5
MonthSal	1203.291446	2575.180084	1583.223758	2531.965558	3570.339238	3307.184015
CustMonVal	288.235639	436.124312	182.198289	-10.933333	106.679102	499.233342
ClaimsRate	0.702203	0.284760	0.728771	0.983165	0.813421	0.438523
PremMotor	97.172545	423.134098	221.867391	432.230669	245.591467	204.422979
PremHousehold	537.549201	78.739676	225.390336	66.658778	183.695771	437.810816
PremHealth	161.989464	118.430051	233.132614	113.090773	222.018044	172.783198
PremLife	111.626210	15.796619	44.364745	14.798946	42.566521	64.232204
PremWork	112.204917	15.465403	43.546671	14.731941	41.334406	61.484417
Age	23.712675	49.487339	31.206583	48.776535	67.454783	62.561966
PolicyAge	30.453811	30.181373	29.695339	30.159981	30.007814	28.816160
x0_b'1 - Basic'	0.434932	0.043399	0.102328	0.041199	0.081297	0.191710
x0_b'2 - High School'	0.440639	0.237095	0.408230	0.261236	0.373120	0.516839
x0_b'3 - BSc/MSc'	0.122146	0.602558	0.448295	0.584738	0.495301	0.275907
x0_b'4 - PhD'	0.002283	0.116948	0.041148	0.112828	0.050282	0.015544
x1_0.0	0.248858	0.150297	0.088793	0.117978	0.701128	0.567358
x1_1.0	0.751142	0.849703	0.911207	0.882022	0.298872	0.432642
x2_1.0	0.296804	0.297853	0.314023	0.294007	0.280075	0.292746
x2_2.0	0.086758	0.096848	0.099080	0.106742	0.106673	0.108808
x2_3.0	0.207763	0.210142	0.198701	0.203184	0.191729	0.185233
x2_4.0	0.408676	0.395158	0.388197	0.396067	0.421523	0.413212

Table 9 – "Batch Kmeans means before scaling"

```
Batch K-Means Cluster solution with R^2 of 0.5335

PremMotor        0.736857
Age              0.697373
MonthSal         0.642148
ClaimsRate       0.632377
CustMonVal       0.525166
PremHousehold    0.487690
PremHealth       0.483640
PremWork         0.459983
PremLife         0.451016
PolicyAge        0.003572
dtype: float64
```

Table 10 – "Batch Kmeans final R2 total and per variable"

```
# Characterize the final clusters (before scalling).

df_before_scaling_hc = pd.concat((df_before_scaling, pd.Series(df_hc_labels,
df_before_scaling_hc_mean = df_before_scaling_hc.groupby('df_hc_labels').mea
df_before_scaling_hc_mean.T
```

df_hc_labels	0	1	2	3
MonthSal	1504.464666	2321.242385	3426.996888	2694.626431
CustMonVal	207.581663	440.689766	243.824065	-8.498328
ClaimsRate	0.729449	0.283222	0.658382	0.978771
PremMotor	191.300238	414.844675	247.761963	426.169981
PremHousehold	318.440004	87.905831	239.585929	69.225257
PremHealth	206.084185	121.583022	200.703447	118.708363
PremLife	63.371407	17.737482	47.768326	14.268590
PremWork	63.977678	16.849123	45.239965	14.858009
Age	29.582728	44.837110	64.977150	51.554957
PolicyAge	28.844266	30.921338	30.255741	30.122360
x0_b'1 - Basic'	0.206578	0.046326	0.109690	0.037997
x0_b'2 - High School'	0.407982	0.253994	0.401983	0.255700
x0_b'3 - BSc/MSc'	0.358093	0.582535	0.441318	0.593205
x0_b'4 - PhD'	0.027347	0.117146	0.047010	0.113098
x1_0.0	0.143755	0.097977	0.613367	0.179705
x1_1.0	0.856245	0.902023	0.386633	0.820295
x2_1.0	0.305987	0.306177	0.287496	0.286097
x2_2.0	0.096083	0.095314	0.103294	0.110863
x2_3.0	0.200296	0.207668	0.193476	0.204291
x2_4.0	0.397635	0.390841	0.415734	0.398748

Table 11 – "Batch Kmeans means before scaling"

```
Hierarchical Clustering solution with R^2 of 0.3908

MonthSal         0.591927
CustMonVal       0.355899
ClaimsRate       0.508553
PremMotor        0.566382
PremHousehold    0.236690
PremHealth       0.312591
PremLife         0.240531
PremWork         0.243865
Age              0.646485
PolicyAge        0.012526
dtype: float64
```

Table 12 – "Batch Kmeans final R2 total and per variable"

premium_labels	0	1	2
value_dem_labels			
0	1174	441	1845
1	1153	770	1308
2	1756	629	872

Table 13 – "Merged perspectives labels before HC"

premium_labels	0	1	2
value_dem_labels			
2	4083	1399	4466

Table 14 – "Merged perspectives labels after HC"

```
Different Perspectives Cluster solution with R^2 of 0.3393
```

```
PremMotor        0.758329
PremHealth       0.515814
PremHousehold    0.460385
PremLife         0.424983
PremWork         0.423417
Age              0.186775
MonthSal         0.165299
CustMonVal       0.045514
ClaimsRate       0.018809
PolicyAge        0.000144
dtype: float64
```

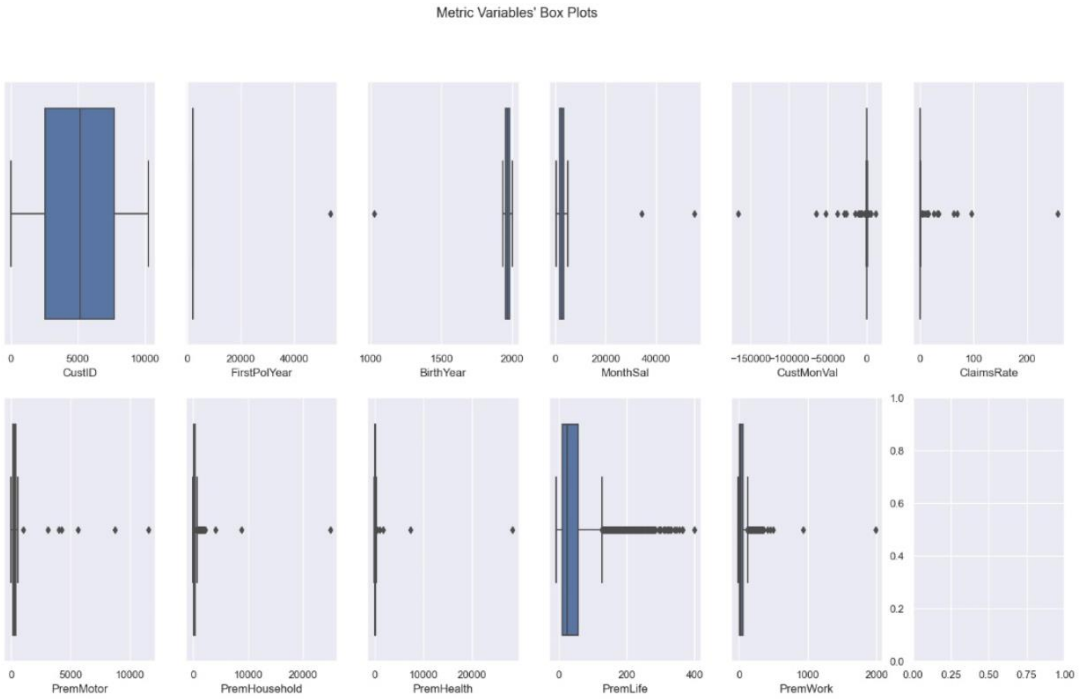
Table 15 – "Merged perspectives Kmeans final R2 total and per variable"

```
df_kmeans_r2 =  get_r2(df_kmeans, "df_km_labels")
print("Kmeans Cluster final solution with R² of %0.4f" % df_kmeans_r2);

df_bkm_r2 =  get_r2(df_bkm, "bkmlabels")
print("Batch K-Means Cluster solution with R^2 of %0.4f" % df_bkm_r2)

df_hc_r2 =  get_r2(df_hc, "df_hc_labels")
print("Hierarchical Clustering solution with R^2 of %0.4f" % df_hc_r2)

df_perspectives_r2 =  get_r2(df_, "merged_labels")
print("Different Perspectives Cluster solution with R^2 of %0.4f" % df_perspectives_r2);

df_msc_r2 =  get_r2(df_MSC_concat, "df_MSC_labels")
print("Mean Shift Cluster final solution with R² of %0.4f" % df_msc_r2);

df_dbscan_r2 =  get_r2(df_dbscan_concat, "df_dbscan_labels")
print("DBSCan Cluster final solution with R² of %0.4f" % df_dbscan_r2);

df_gmm_r2 =  get_r2(df_gmm_concat, "df_gmm_labels")
print("GMM Cluster final solution with R² of %0.4f" % df_gmm_r2);
```

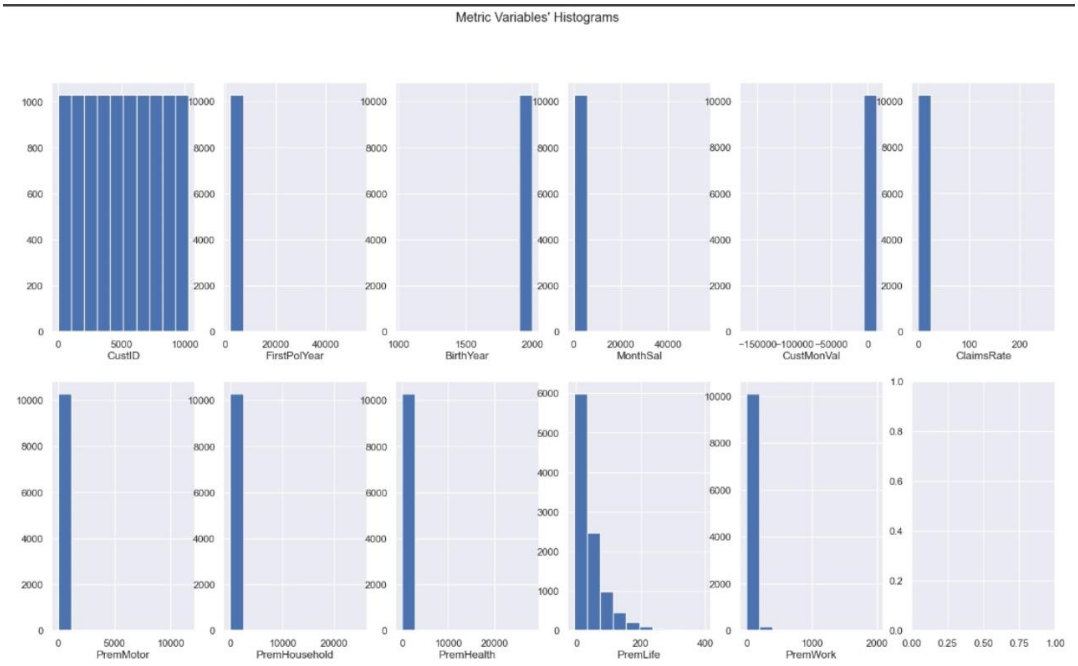
```
Kmeans Cluster final solution with R² of 0.4373
Batch K-Means Cluster solution with R^2 of 0.5335
Hierarchical Clustering solution with R^2 of 0.3908
Different Perspectives Cluster solution with R^2 of 0.3393
Mean Shift Cluster final solution with R² of 0.1395
DBSCan Cluster final solution with R² of 0.0258
GMM Cluster final solution with R² of 0.2601
```

Table 15 – "Final R2 summary per algorithm"

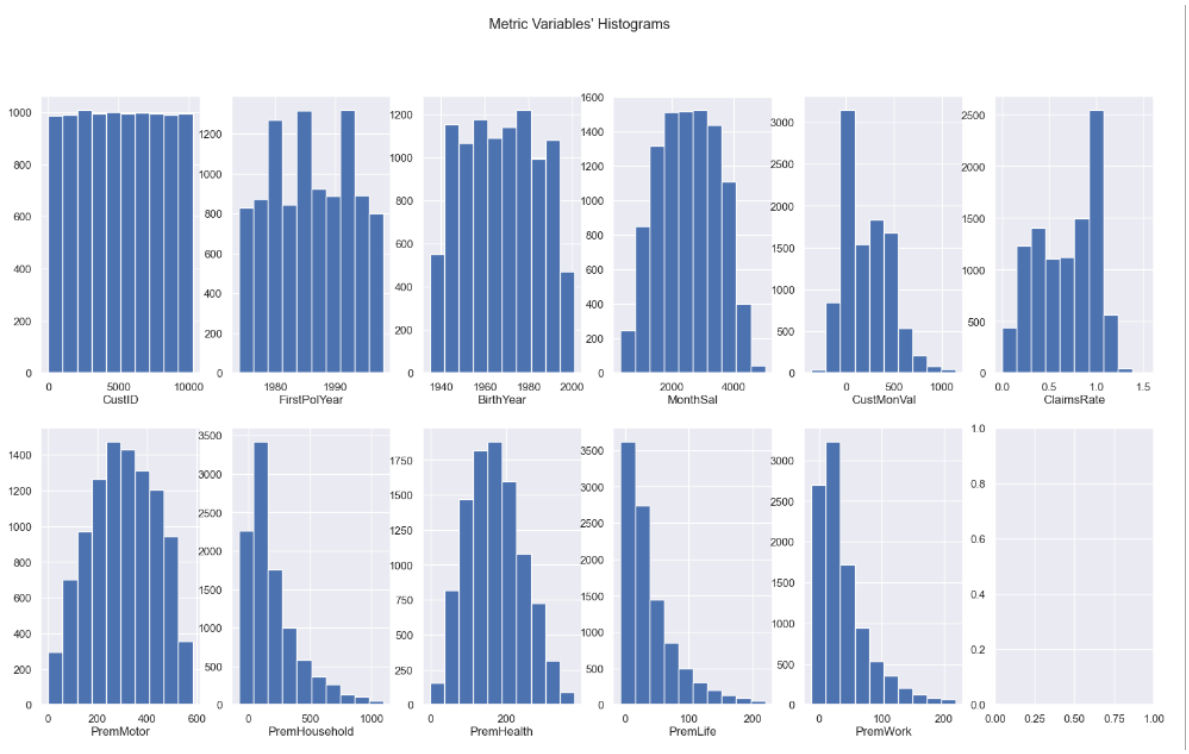
14.2. Graphs



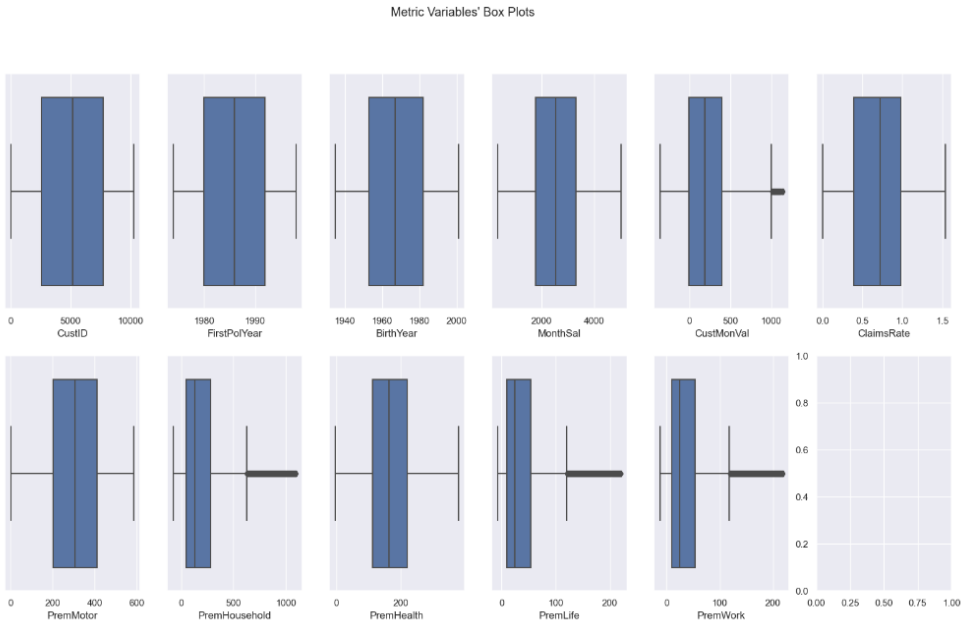
Graph 1 – "Boxplots before Outlier removal"



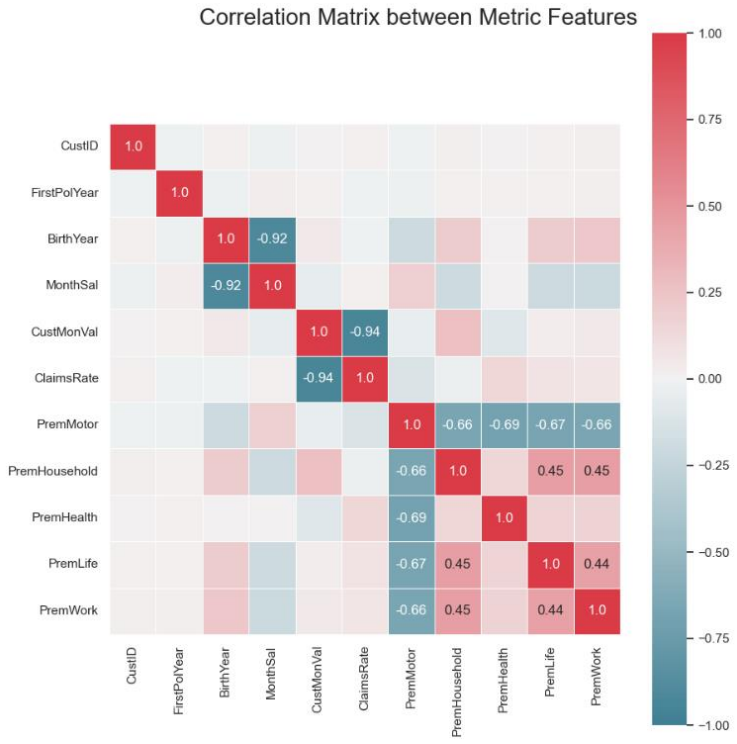
Graph 2 – "Histograms before Outlier removal"



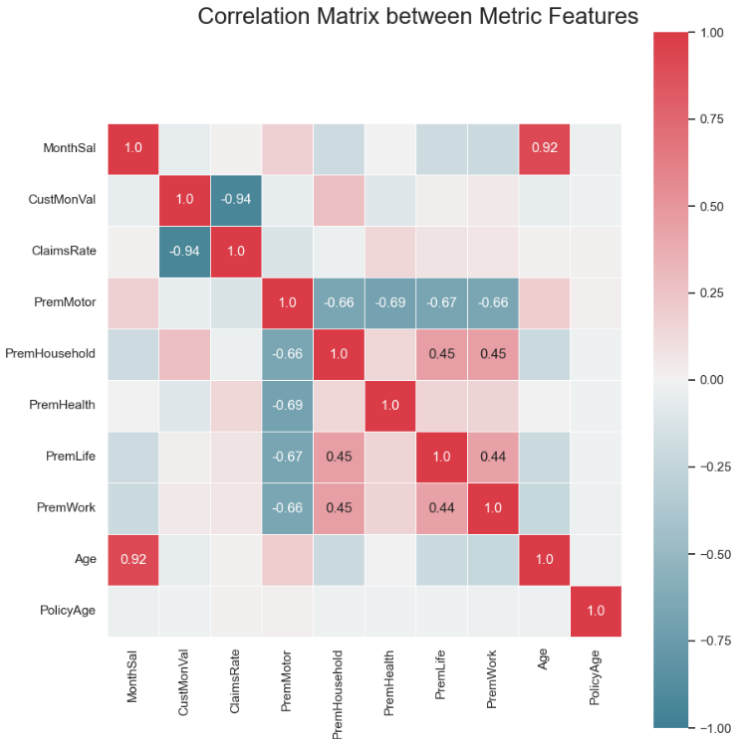
Graph 3 – "Histograms after Outlier removal"



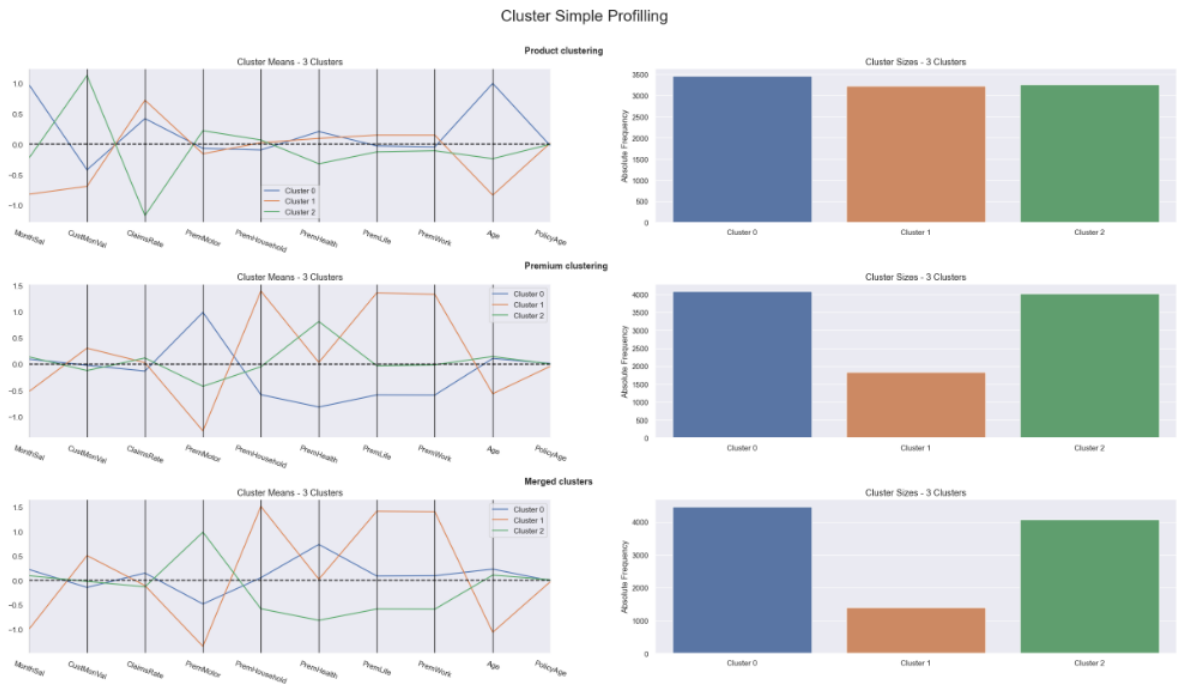
Graph 4 – "BoxPlots after Outlier removal"



Graph 5 – "Initial Correlation Matrix"

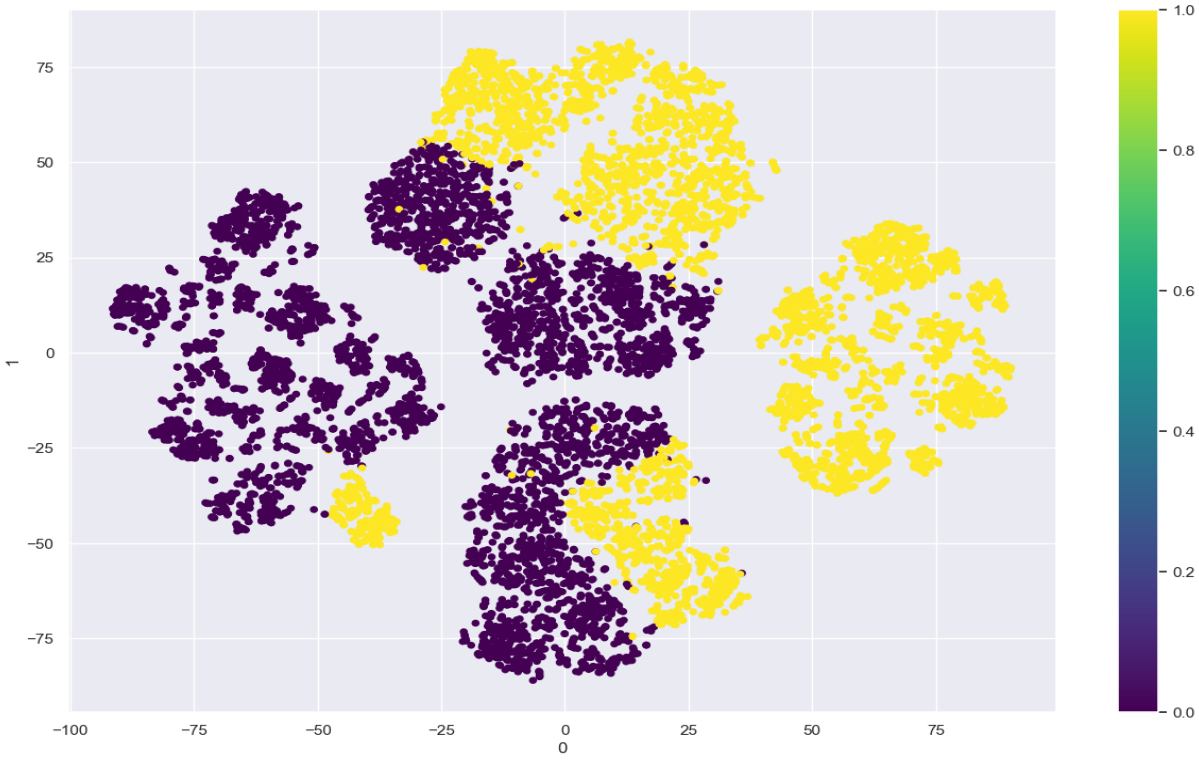


Graph 6 – "Final Correlation Matrix"



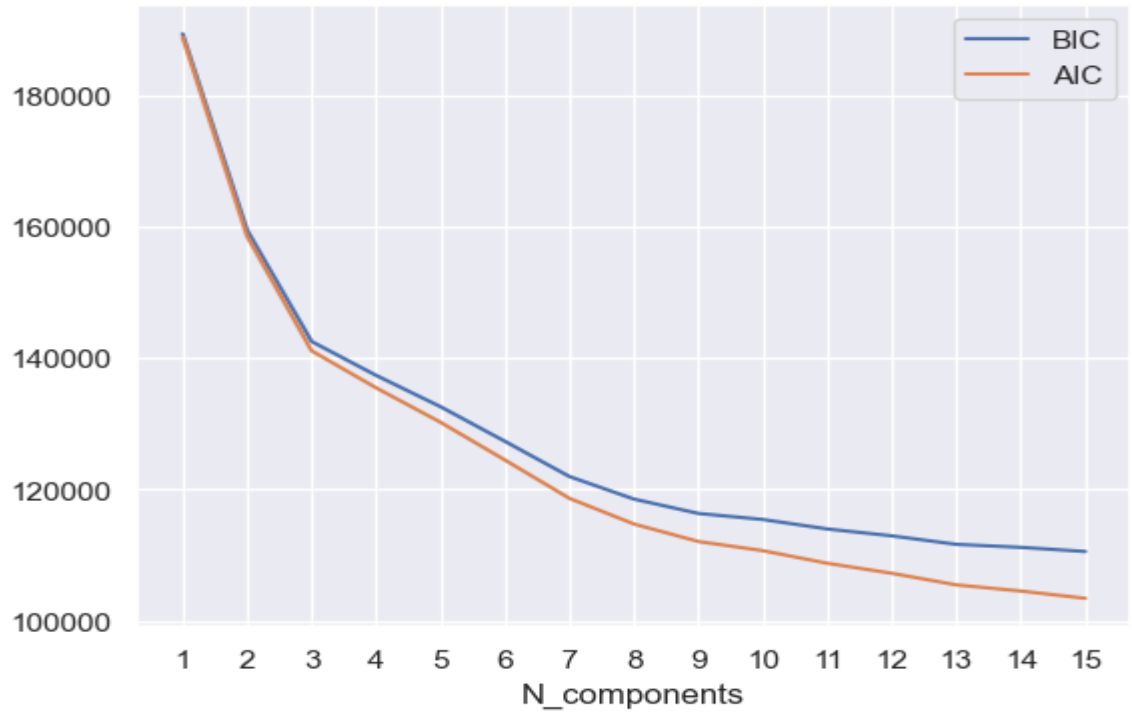
Graph 7 – "Customer Profiling"

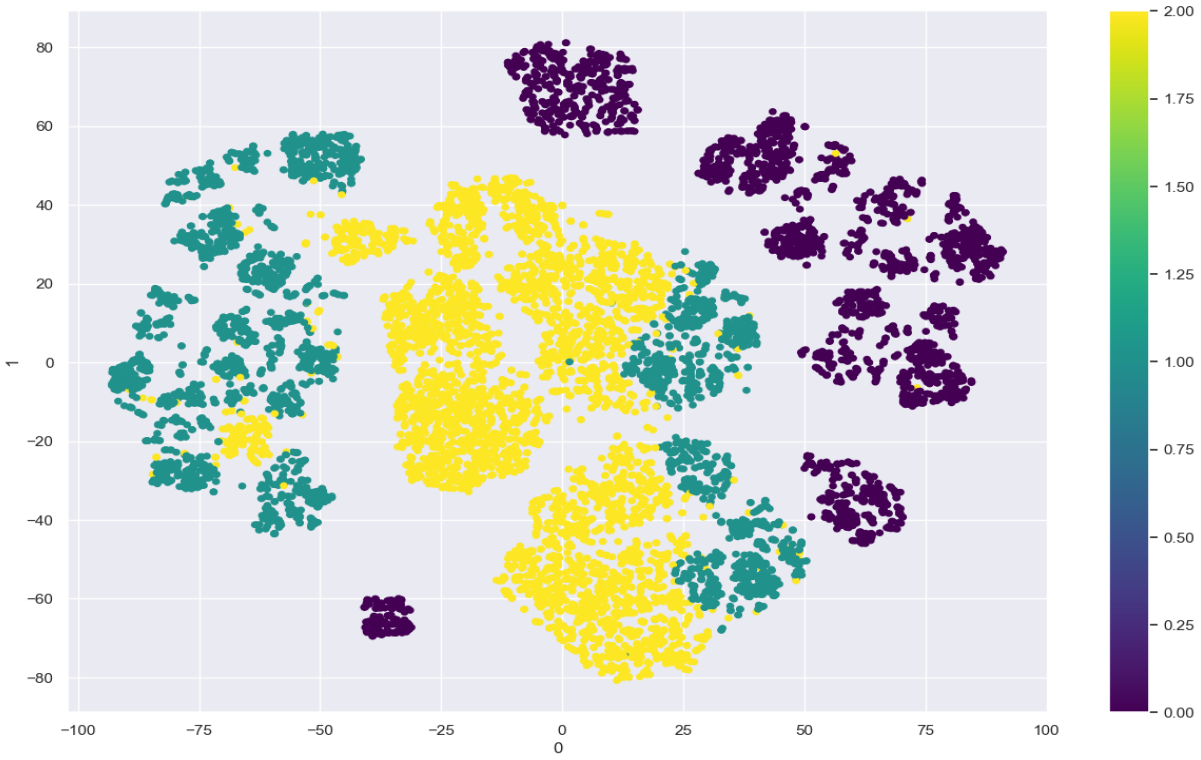
11. Mean Shift:



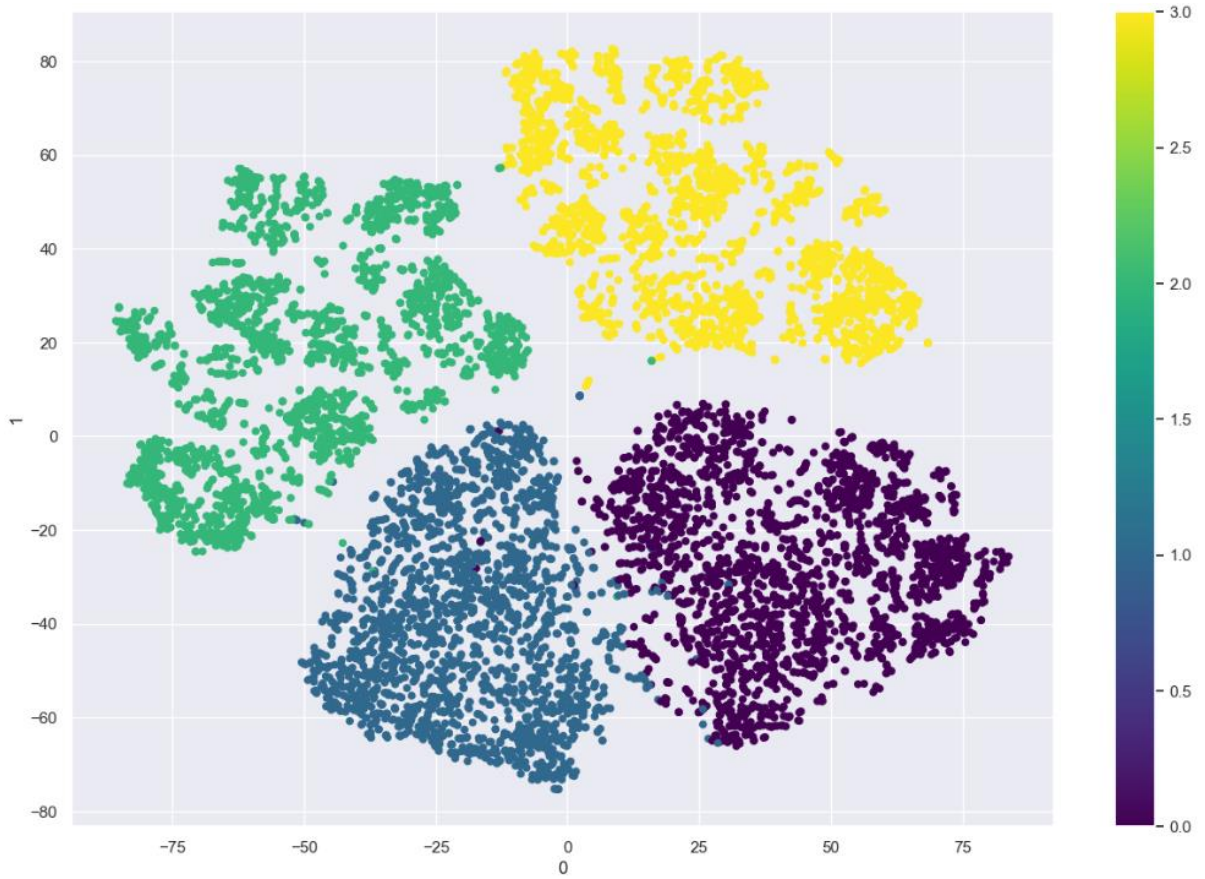
10. DBSCAN:

11. GuassianMixture:

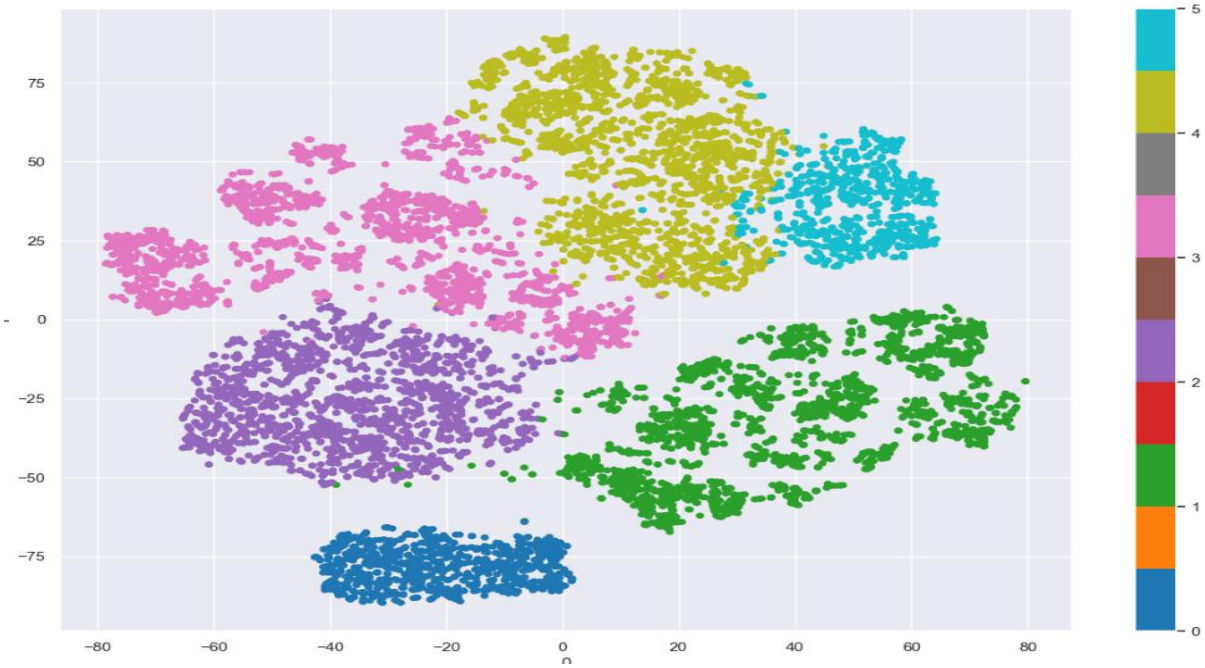

BIC
AIC
180000
160000
140000
120000
100000
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
N_components


80
60
40
20
0
−20
−40
−60
−80
1
−100 −75 −50 −25 0 25 50 75 100
0
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00

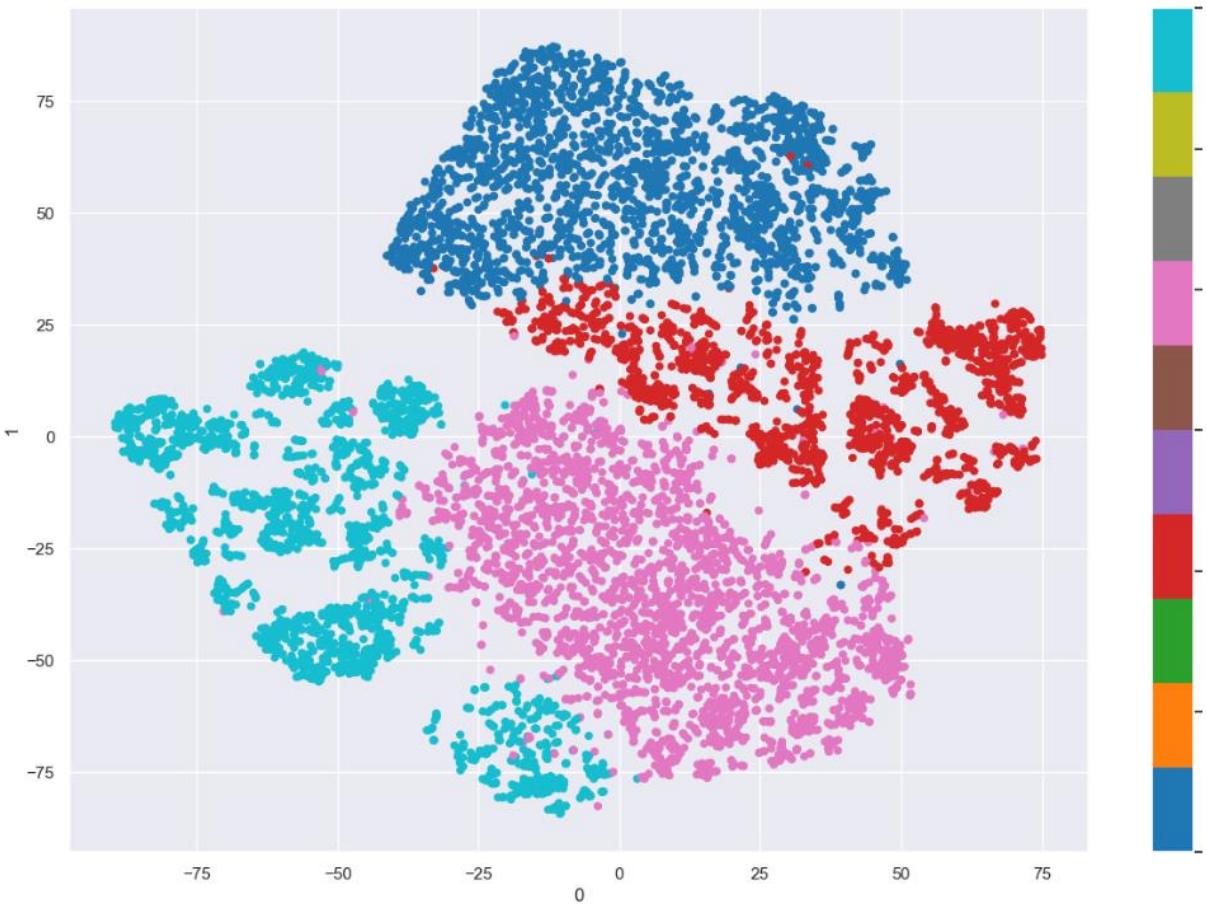
Kmeans



Mini batch



Hierarchical


75
50
25
0
−25
−50
−75
1
−75
−50
−25
0
25
50
75
0