Luleå University of Technology (LTU)

Department of Computer Science, Electrical and Space Engineering

Digital Services and Systems

COURSE: D0025E

Data Mining Assignment: K-MEANS

MEMBERS: Oscar Arandes Tejerina, Anders Dehlbom, Erik Vodopivec Forsman, Mattias Konnebäck, Nicole Aguiar Jakobsson,

LP1-2025

Abstract

This report applies the CRISP-DM framework to segment potential new customers for an automobile company. We begin with business and data understanding, then prepare the data by removing non-descriptive fields (ID, Segmentation, Var1), imputing missing values, one-hot encoding categorical variables (Gender, Ever Married, Graduated, Profession, Spending Score), and z-normalizing continuous variables (Age, Work Experience, Family Size). Using RapidMiner and KNIME, we model customer segments with k-means across a grid of k values and evaluate solutions with the elbow criterion (average within-centroid distance) and Davies–Bouldin. Results are not consistent across tools, RapidMiner indicates an elbow at k=5 and KNIME is not as clear. The result of a comparison of the existing k=4 segmentation in the company and a new k=5, splits the last cluster of k=4 into two behaviorally distinct clusters. From "established, older, married, graduated, mixed spenders" (k=4) to "average spenders versus older, lawyer-tilted low spenders"(k=5).

Contents

Chapter 1

The CRISP-DM Process

In this chapter, we provide a detailed overview of the CRISP-DM process and its key phases [She00].

1.1 Phase I: Business Understanding

The goal of the project is to classify potential new customers into an appropriate number of groups. This is something that any business owner would be inclined to prioritize doing. Classifying what purchasing trends your current customers have can be crucial to getting new customers. For example, you can get information about the types of products to advertise and what aspects of the product to highlight. Lower income shoppers may be interested in one thing, while higher income customers might want something completely different. Classifying customers is a huge help when it comes to boosting your business revenue.

To achieve this, one needs information about the potential customers (formated to csv). Another important part is developing a plan. The plan for this project involves using K-means clustering in order to find the different groups. This does not require a lot of resources, which makes the benefit of a greater understanding of the potential customer groups greater than the cost. Understanding what kinds of customer one might be able to bring in can greatly affect the outcome of an ad campaign.

1.2 Phase II: Data Understanding

The dataset contains information on the automobile company's current customers, such as age, profession and marital status (among other things). The values were both numerical and non-numerical (multiple choice only). None of the answers are text written by the customers themselves. There are also some values that can easily be converted to numerical values. An example of such a value is marital status, which only has two options. In terms of size, the data set has more than enough rows (8.068 individuals) to create a reliable representation of the customer base.

The data can be downloaded as a zip file in CSV format from the following link.

1.3 Phase III: Data Preprocessing

1.4 Data Preprocessing

In accordance with the CRISP-DM methodology [She00], data-preprocessing was used before clustering the dataset to prepare the data for analysis. The steps that were performed was attribute selection, replacing missing values, one-hot encoding of categorical attributes, normalization of numerical attributes and setting clustering parameters for the loop.

1.4.1 Attribute selection

Non-descriptive attributes were removed, specifically `ID` since this only served as a row index. The `Segmentation` column was also removed as this analysis aimed to perform an unsupervised clustering of data. `Var_1` was also removed as it was deemed irrelevant to customer segmentation. The remaining attributes were:

- Gender
- Ever Married
- Age
- Graduated
- Profession
- Work_Experience
- Spending_Score
- Family_Size

1.4.2 Replacing missing values

Cells that contained missing values were replaced with the average method to not distort the analysis when missing values occured. This was applied to all attributes and no distinction was made between numerical or categorical attributes.

1.4.3 One-hot encoding of categorical attributes

For an unsupervised clustering analysis to work the dataset needed to be converted into a nominal format. This can be done by various methods but our choice was to do a one-hot encoding of the categorical attributes where the different values of the attribute where redistributed into it's own column and valued as a binary. For example:

$$\text{Spending\_Score} \quad \rightarrow \quad \begin{cases} \text{Spending\_Score} = \text{Low} \\ \text{Spending\_Score} = \text{Medium} \\ \text{Spending\_Score} = \text{High} \end{cases}$$

1.4.4 Normalization of numerical attributes

Continuous attributes (`Family_Size`, `Age` and `Work_Experience`) were normalized using a z-transformation. This aims to stop variables with large ranges from being overrepresented in a distance-based clustering algorithm. Binary values were not normalized as they were already normalized to `0/1` by the previous step.

1.5 Phase IV: Modeling

In an article [Eda] on IBM:s website K-means clustering is described as "...an iterative, centroid-based clustering algorithm that partitions a dataset into similar groups based on the distance between their centroids". It is unsupervised learning, which means that the data is not classified beforehand.

K-means clustering is based on iterations, in which the centroids of the different groups are moved with the goal of minimizing the distance between data points and the centroids [Elr24; Awa20]. The method used to measure the distance is Euclidean distance. The distance is used to tell how different two values are [Ooy01]. To choose what amount of centroids to use, the elbow method was used. The elbow method uses the changes in the "Within Cluster Sum of Squares" to create a plot which is then used to decide how many centroids to use. A plot is supposed to be shaped like a bent arm and the part that would be the elbow represents the optimal amount of centroids [Eda].

The algorithm starts by creating the centroids. The number of centroids has been specified beforehand. After that is done, iterations begin. In each iteration the centroids are moved in an effort to find the spot where the Euclidean distance is minimal. They are performed either until the centroids have reached optimal placements or until the specified number of iterations have been performed [Eda].

1.6 Phase V: Evaluation

It is now time to evaluate the results obtained. We used two tools, RapidMiner and Knime, and did not arrive at exactly the same conclusions. To determine the optimal number of clusters, we applied the *elbow method* (see results for RapidMiner in Fig 1.2 and for Knime in Fig. 1.1).

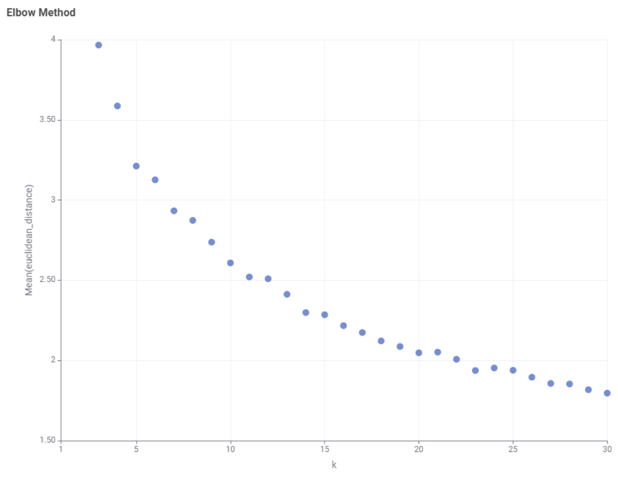


Figure 1.1: Scattering plot of the mean squared euclidean distances of the point to its cluster centroids as function of the cluster size k. Plot generated with Knime.

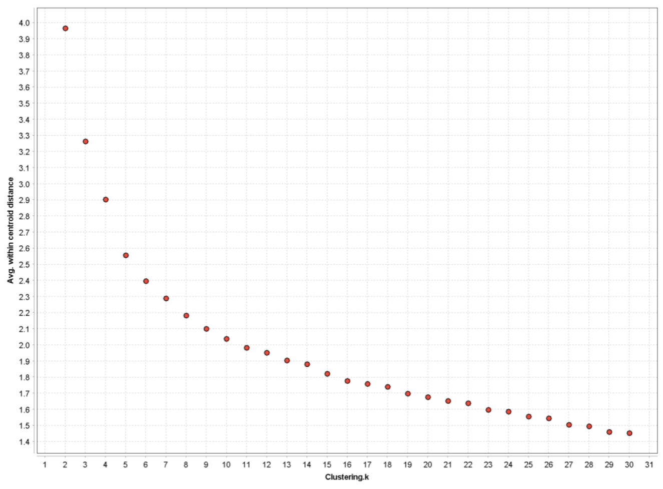


Figure 1.2: Scattering plot of the average within centroid distance in y-axis and x-axis the clustering k. Plot generated with RapidMiner.

There are several aspects to evaluate regarding how the process is carried out. One key aspect is how the categorical variables perform under the K-Means algorithm. We have used a naive approach, applying one-hot encoding to the categorical variables. However, one-hot encoding is not the only way to handle them. One alternative would be n-hot encoding, where n is the number of instances (see discussion in Ref. [McC24]). More refined methods exist for handling categorical variables in clustering, such as using *k-modes*.

1.7 Phase VI: Deployment

The deployment phase has been effectively completed through this report.

Chapter 2

Tools Insights

This chapter presents a comparative analysis of two data mining tools in terms of ease of use, efficiency, and cost to solve Assignment 3 in the course.

2.1 RapidMiner Insights

This section explains our insights while using RapidMiner (Altair AI Studio).

2.1.1 Ease of Use

Let us describe how to implement the K-means algorithm in RapidMiner (see Fig. 2.1).

After importing the dataset with the *Retrieve operator*, we replace missing values using the *Replace Missing Values operator*. A missing value is replaced by the mean value of the column; `default:average`.

Next, we use the *Select Attributes operator*, where we remove identifiers and irrelevant columns such as e.g. `ID`.

As we have a lot of categorical attributes, we use the *One-Hot Encoding operator*, which transforms the nominal attributes `Gender`, `Ever_Married`, `Graduated`, `Profession`, and `Spending_Score` into separate binary columns.

For the numerical attributes, we normalize with the *Normalize operator*. Here, you can choose from many alternatives, but the most interesting are `range transformation` (offering an adjustable range) and `Z-transformation`. Here we choose `Z-transformation` for continuous variables, while the binary columns from the *One-Hot Encoding* were left as they are. See discussion above for advantages and disadvantages with this approach.

Now, the preprocessing process is completed and the data is ready for clustering. Since we want to explore different values of cluster sizes (k), we place the whole clustering process inside a loop using the *Loop Parameters operator* (see Fig. 2.2). In this operator we can choose from a variety of different parameters that we would like to alter during the iteration. We here choose, of course, the k-value, and the range of k values is set in a `value list`, or, if you like, a grid. A suitable range could be k's from 2 to 30.

Now, every k in our value list is passed into the *Clustering operator* (K-Means). There we can adjust the process further with e.g. `cluster attribute`, starting value of k, `maximal runs` (calculation of new centroids), but, most important, we choose the drop-down meny `divergence` to be set to `Squared Euclidean Distance`.

Also inside the loop, we add the *Cluster Distance Performance operator* to get an evaluation of the clustered results. In this operator we set the `main criterion` to value `Average within centroid distance`, that is, the average distance of all points to their cluster centroids. In order to get even more data for analyzing the right value for k, we can also alter the `main criterion` to the index `Davies Bouldin`.

From the loop's output we now collect the results of each iteration of the loop, and in the tab *Results*, and then tab *Loop Parameters*, we get a table `Data` and a `Plot view`. For every k on the x-axis, we get a value for `the Avg.within centroid distance` (or, if chosen and processed, the `Davies Bouldin`) at the y-axis.

This plot allows us to detect a possible "elbow point" (if using `Davies Bouldin`, we look for a minimum value, lower values indicate more distinct and compact clusters).

2.1.2 Efficiency

Implementing the algorithm of K-means in RapidMiner is actually a very user-friendly process. Very few operators are required in the process and the building of the clusters and the performance evaluation can be done in only two operators. The operator Loop Parameters significantly simplifies the building of the loop. In the end, there are many and easily understood tools for the generated data to be analyzed in tables or plots. Here, RapidMiner really is the choice.

2.1.3 Costs

RapidMiner requires a license key and the process for installing it is complicated. Nevertheless, there are free-of-charge licenses for students.

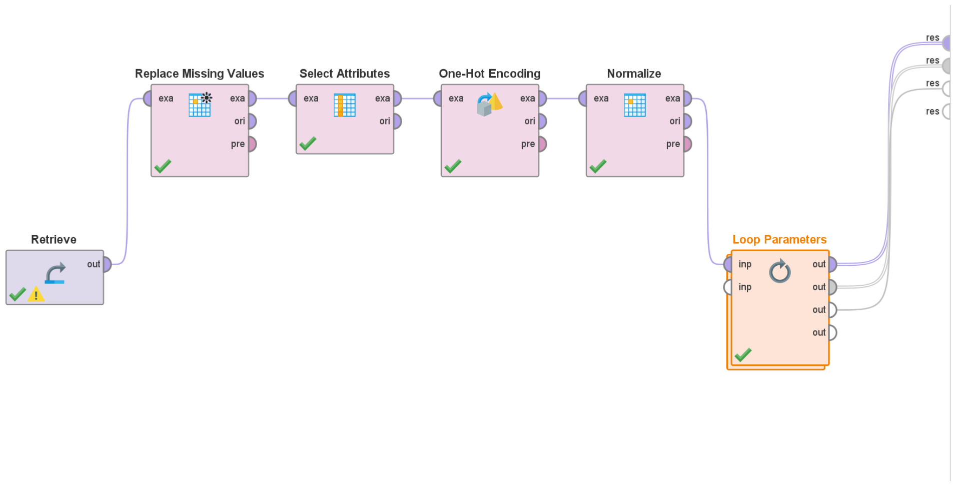


Figure 2.1: RapidMiner process for the K-means algorithm.

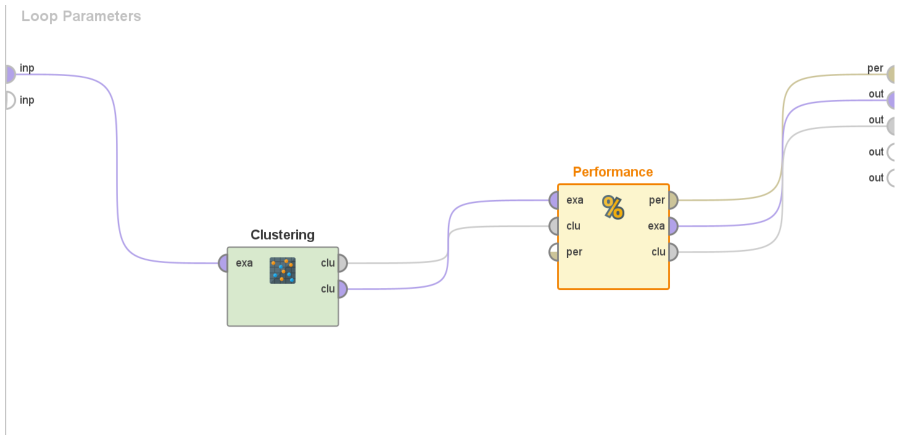


Figure 2.2: Inside the operator Loop Parameter.

2.2 Knime Insights

This section explains our insights while using Knime.

2.2.1 Ease of Use

Let us describe how to implement the K-means algorithm in Knime. After reading the CSV file with the *CSV Reader* node, we filter out the attributes `ID`, `Var1`, and `Segmentation`. We then remove missing values from the attributes `EverMarried`, `Graduated`, `Profession`, `WorkExperience`, and `FamilySize`. The node is configured to remove the entire row whenever a missing value is present.

Further preprocessing is performed with the *Normalizer* node, which standardizes the numerical attributes using z-normalization. For categorical variables, we apply one-hot encoding with the *One to Many* node, applied to the attributes `Gender`, `EverMarried`, `Graduated`, `Profession`, and `SpendingScore`.

At this stage, the processed data is ready for the *K-Means* node. Since we want to run the algorithm for different cluster sizes, we initialize a loop. To achieve this, we use the *Table Creator* node to generate a table containing the desired cluster sizes. It is important to configure the data type as integer, otherwise, the default string type will cause the process to fail. To ensure the process interprets these values as loop variables, we use the *Table Row to Variable Loop Start* node. Finally, we connect this to the *K-Means* node.

The node provides two outputs. The first is the data table with an appended column indicating the cluster assignment of each instance. The second output contains the centroid coordinates of each attribute for every cluster. To combine this information, we use the *Joiner* node: the top input is the table with the cluster column, and the bottom input contains as many rows as there are clusters. The result is a table where each row corresponds to the original data, with the coordinates of the centroid of the assigned cluster appended next to it. We can now compute, row by row, the squared Euclidean distance between each instance and its cluster centroid. This is implemented using a *Java Snippet* node. Important to click on Add in the Output section to determine what is the output of the node. In our case the name will be stored in column named `euclidean-distance`.

At this stage, we have a distance value for each row. To sum these values, we use the *GroupBy* node. We leave the Groups section empty and, in the Manual Aggregation section, select `euclidean-distance` with the aggregation function set to Mean (thus computing the average distance of all points to their respective cluster centroids).

Next, we aim to perform some data visualization. We want a table that contains the cluster size (in our process referred to as `k`) and a column with the averaged distances. To achieve this, we use the *Variable to Table Column* node before ending the loop with the *Loop End* node. Without this step, the output would instead contain a column `Iteration` with the iteration index (0, 1, 2,...) and the distances, rather than a table with the actual cluster sizes (k=3, k=4,...) paired with their corresponding distances. We then use the *Column Filter* node to remove the `Iteration` column before plotting the results with the *Scatter Plot* node.

2.2.2 Efficiency

In our experience, the efficiency of this process in Knime compared to RapidMiner is not very high. Unless there are better approaches, we found that Knime in general requires many more nodes to run the process than RapidMiner. On the other hand, RapidMiner can sometimes feel like a black box, where it is not always clear what each node is doing. In Knime every step is very transparent, though at the cost of being less automated.

2.2.3 Costs

An important advantage of Knime is that it is open-source and freely available, making it accessible for both academic research and professional use. Since no license key is required, it can be installed on multiple computers, unlike RapidMiner, where each license is tied to a specific computer.

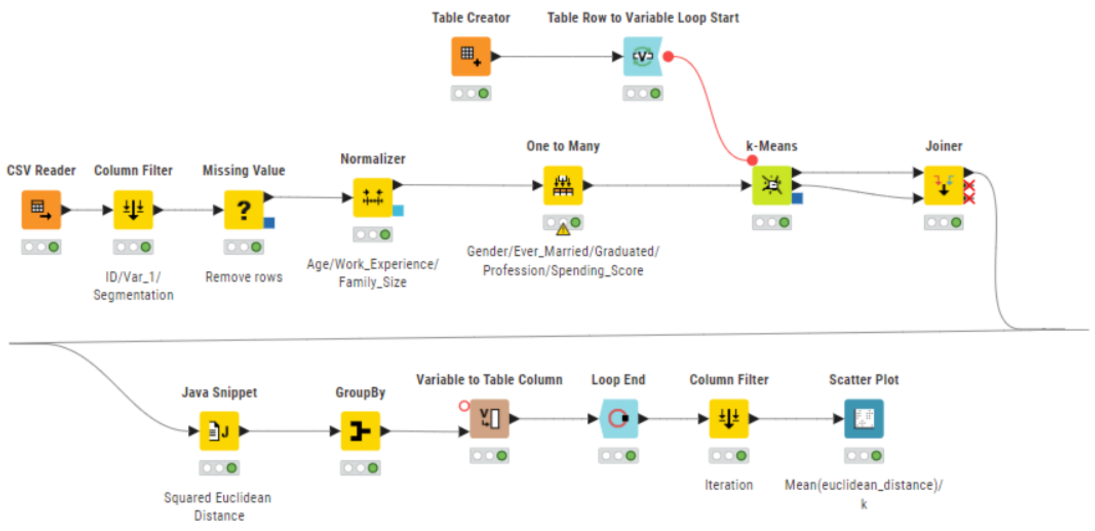


Figure 2.3: KNIME process for the K-means algorithm.

Chapter 3

Answers to Assignment Questions

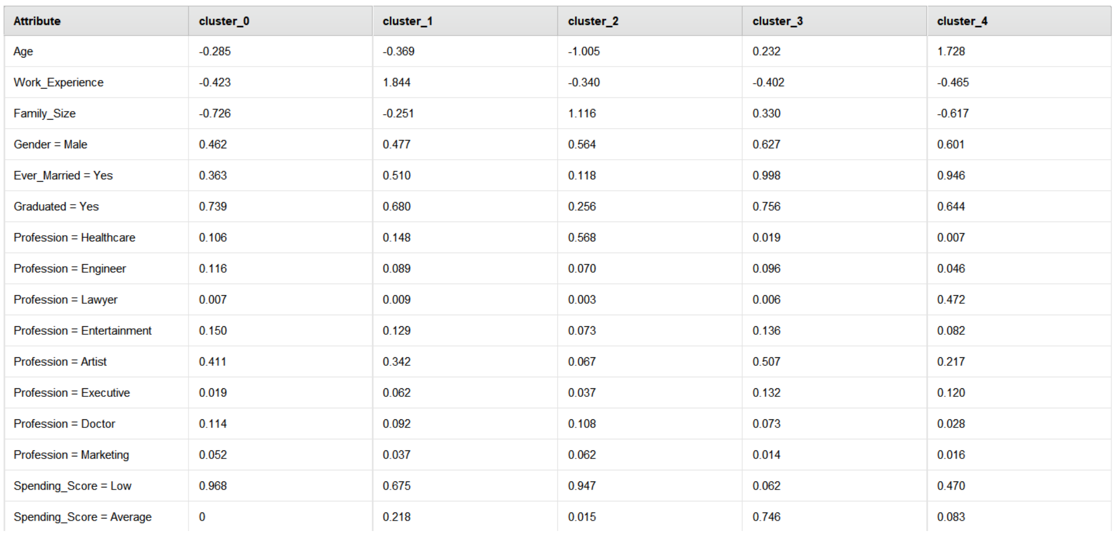
3.1 Discuss what is interesting about the clusters generated by the k-Means algorithm and describe what iterations of modeling you went through, such as experimentation with different parameter values, to generate the clusters

There was a lot of different experimentation through the work process, mainly in the data processing part. Deciding on whether to use the nominal-to-numerical node or the one-hot encoded node. We first normalized every attribute after the categorical data conversion with Z-transformation, then we learned that we should only normalize the numerical data with Z-transformation, as they are already on a 0-1 scale. All these steps contributed to different K-values in the elbow, first a k=10 (when normalizing every attribute), and in the last finished step (the one we chose) a k=5, in RapidMiner, in KNIME, the optimal K-value was not clear. We also tested to categorize the data through business rules; as we have 9 different professions, we tested k= 9. This test was not made with all attributes and was only for experimental purposes. We decided on behavior-driven clusters rather than just mirroring any single label.

While we were testing the different k-values before doing an elbow, we looked at the performance node, which exhibits the average within centroid distance and Davies Bouldin. The higher the k, the lower the average within centroid distance value. But while testing the different parameters, we found that the same rule did not apply for Davies Bouldin, a higher k-value does not necessarily mean lower values. We also experimented with the maximize checkbox in the performance node, which made the results appear in positive values instead of negative, which made the scatter plot better for visualization purposes. The same value was shown whether it was maximized or not; the difference was only in that while not maximizing, the result appeared negative, and while maximizing, positive, so either -1 or 1, for example.

3.2 Explain how your findings are relevant to your original question.

As the company already had segmented the customers into four different groups, we wanted to see how many segments we would get as optimal, using the elbow method. As we got k=5, it only added one more cluster, compared to the company. This finding is relevant because the number of clusters is not significantly different, and if the company can act on their business design with one more cluster, they should, as they can gain a better understanding from separating the last two clusters. We will explain this further by analyzing k=5 and k=4.



Attribute	cluster_0	cluster_1	cluster_2	cluster_3	cluster_4
Age	-0.285	-0.369	-1.005	0.232	1.728
Work_Experience	-0.423	1.844	-0.340	-0.402	-0.465
Family_Size	-0.726	-0.251	1.116	0.330	-0.617
Gender = Male	0.462	0.477	0.564	0.627	0.601
Ever_Married = Yes	0.363	0.510	0.118	0.998	0.946
Graduated = Yes	0.739	0.680	0.256	0.756	0.644
Profession = Healthcare	0.106	0.148	0.568	0.019	0.007
Profession = Engineer	0.116	0.089	0.070	0.096	0.046
Profession = Lawyer	0.007	0.009	0.003	0.006	0.472
Profession = Entertainment	0.150	0.129	0.073	0.136	0.082
Profession = Artist	0.411	0.342	0.067	0.507	0.217
Profession = Executive	0.019	0.062	0.037	0.132	0.120
Profession = Doctor	0.114	0.092	0.108	0.073	0.028
Profession = Marketing	0.052	0.037	0.062	0.014	0.016
Spending_Score = Low	0.968	0.675	0.947	0.062	0.470
Spending_Score = Average	0	0.218	0.015	0.746	0.083

Figure 3.1: Centroid table k=5

Although there are some differences in amount, there is no great significance in the 5 different clusterings regarding gender majority; for that reason, I will mostly not include gender while providing the information about the clusters.

In cluster 0 the group is relative young (≈-0,29), with a low work experience and the group with the smallest family size. The gender is mostly female but not a significant difference (male 0,462). The majority hasn't been married but have graduated. The spending score is very low, and there is a mix of professions most being artists.

Cluster 1 have age slightly below avg (≈-0.37), but very high work experience (≈+1.84), smaller families, have been/is married and mostly graduated. Low spend is still dominant and with a profession mix, but lightly more artist represented.

Cluster 2 are the youngest, and they have low work experience (but not the lowest of the clusters). They have the largest families, have not been married, and have a lower graduation rate. They are also mostly low spenders, with the profession of healthcare being most represented in the cluster.

Cluster 3 are slightly older (≈+0.23), with low work experience. They have a slightly bigger family size, and almost all have been/are married. Most have graduated and have an average spending score, with a big group of artists representing the cluster in comparison to the other professions.

Lastly cluster 4 in which we find the oldest group (≈+1.73), with low work experience and small families, almost all have been/are married, and most have graduated. The spending score is on the lower side, with a clear lawyer tilt in the profession representation.

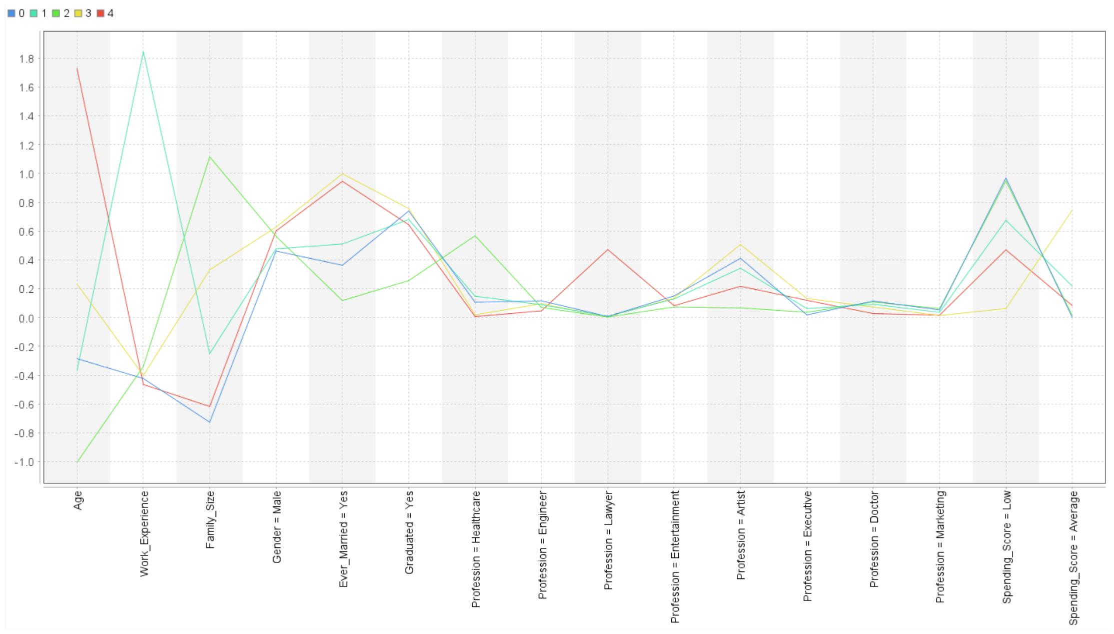
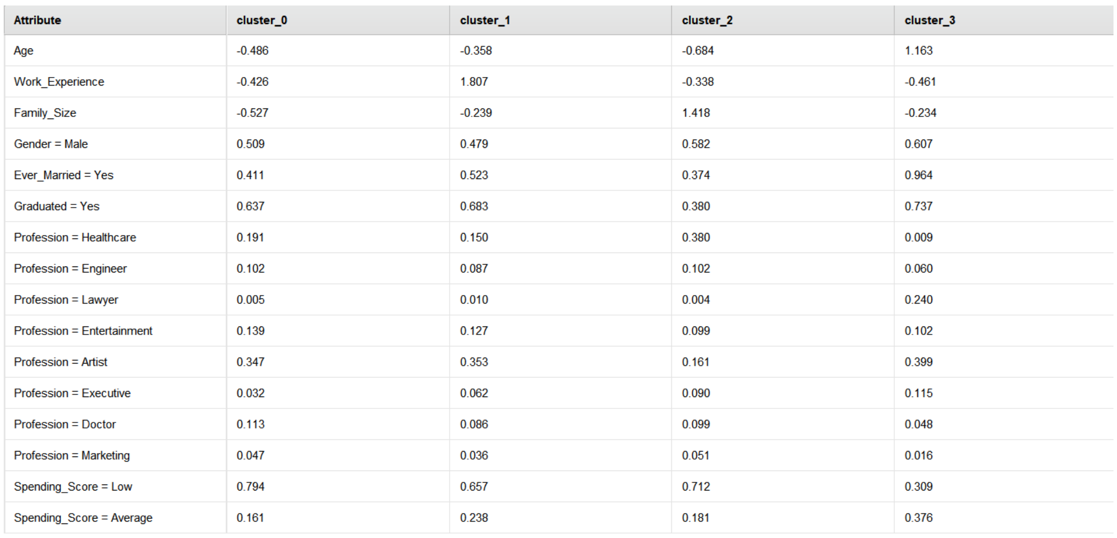


Figure 3.2: Cluster model k=5 plot view.



Attribute	cluster_0	cluster_1	cluster_2	cluster_3
Age	-0.486	-0.358	-0.684	1.163
Work_Experience	-0.426	1.807	-0.338	-0.461
Family_Size	-0.527	-0.239	1.418	-0.234
Gender = Male	0.509	0.479	0.582	0.607
Ever_Married = Yes	0.411	0.523	0.374	0.964
Graduated = Yes	0.637	0.683	0.380	0.737
Profession = Healthcare	0.191	0.150	0.380	0.009
Profession = Engineer	0.102	0.087	0.102	0.060
Profession = Lawyer	0.005	0.010	0.004	0.240
Profession = Entertainment	0.139	0.127	0.099	0.102
Profession = Artist	0.347	0.353	0.161	0.399
Profession = Executive	0.032	0.062	0.090	0.115
Profession = Doctor	0.113	0.086	0.099	0.048
Profession = Marketing	0.047	0.036	0.051	0.016
Spending_Score = Low	0.794	0.657	0.712	0.309
Spending_Score = Average	0.161	0.238	0.181	0.376

Figure 3.3: Centroid table k=4.

The clusters produced by k=4 have a similar persona as k=5 in cluster 0 and 1. Cluster 2 persona is also very similar to cluster 2, k=5, young with a big family, low work experience, with slightly more people ever married and graduated, slightly fewer healthcare workers, and a slightly bigger spending score (still being on the very low end). Cluster 3 have also similar demographic as cluster 3 k=5 (older, married, graduated), but mixed spending (low $\approx$ 0.31; average $\approx$ 0.38). So, it is a blended "established" group at k=4.

By having a cluster with different types of spending and not segmenting the different types, as in cluster 4, the company risks sending the same campaign to groups with different willingness to spend. The blending of the clusters makes targeting not as effective as having the clusters separate, as with k=5, which is why the company should consider having 5 segmentations instead of 4.

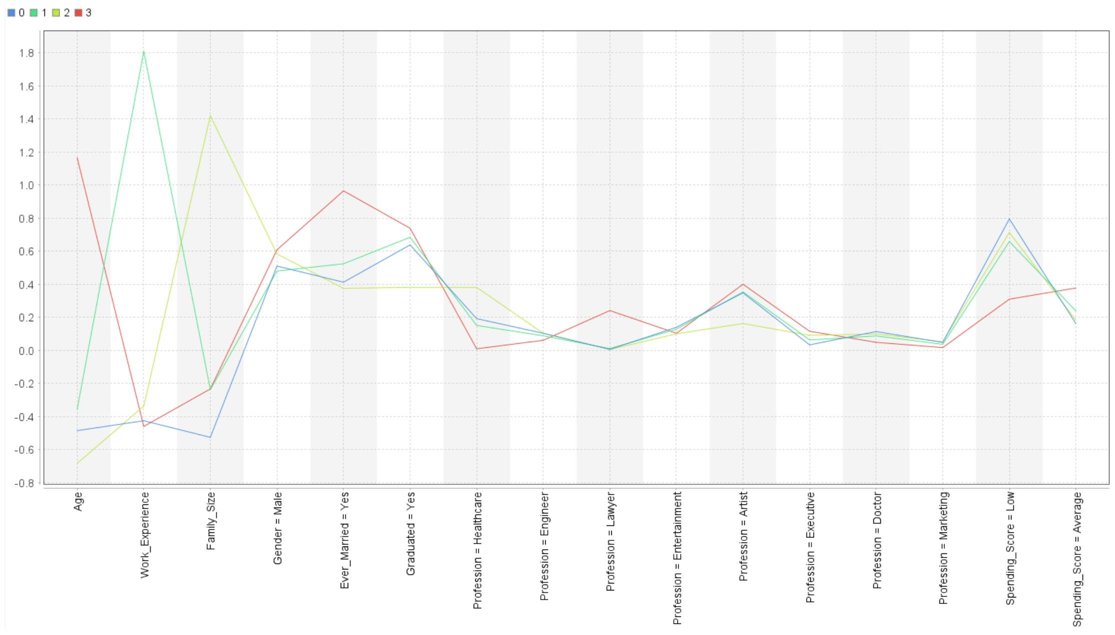


Figure 3.4: Cluster model k=4 plot view.

Chapter 4

Conclusion

Our goal was to see whether unsupervised clustering of the new-market prospects would reproduce the company's existing four segments and, if not, whether an alternative K would be more actionable. After cleaning the data, one-hot encoding categorical fields, and z-normalizing the continuous variables, we ran k-means in a loop, in RapidMiner and KNIME and inspected the elbow (avg. within-centroid distance) together with Davies–Bouldin. Through the process we had a hard time with KNIME producing different results from RapidMiner, as KNIME is not as beginner-friendly as RapidMiner. In the end we got a final K of 5 in RapidMiner and an unsure result in KNIME. Upon analyzing either k=5 or k=4, k=5 is preferable, as it adds a clear, actionable segment without disrupting the other personas. Which should help the company make better targeted campaigns, only adding one more cluster than they already have.

Bibliography

[Awa20] Alaa Awad. *Introduction to Data Science*. Wiley, 2020.

[Eda] Vanna Winland Eda Kavlakoglu. *What is k-means clustering?* https://www.ibm.com/think/topics/k-means-clustering. Accessed: 2025-10-03.

[Elr24] Ahmed Elragal. *Introduction to Big Data*. Pearson, 2024.

[McC24] James McCaffrey. *Clustering Mixed Categorical and Numeric Data Using k-Means with C#*. https://visualstudiomagazine.com/articles/2024/05/15/clustering-mixed-categorical-and-numeric-data.aspx. Accessed: 2025-10-1. May 2024.

[Ooy01] Arjen van Ooyen. *New Approaches for the Generation and Analysis of Microbial Typing Data*. Elsevier Science Ltd, 2001.

[She00] Colin Shearer. "The CRISP-DM Model: The New Blueprint for Data Mining". In: *Journal of Data Warehousing* 5.4 (2000), pp. 13–22.