Project Report

K-Means Clustering Model Training

Student: Sudipta Kumar Das

Student ID: 20-43658-2

Course: Data Science

May, 2023 — Spring

Abstract

K-means clustering is a commonly used unsupervised learning algorithm for partitioning data into distinct groups. The algorithm is based on the idea of minimizing the within-cluster sum of squares (WCSS) and involves iteratively assigning data points to clusters and updating the cluster centroids until convergence. K-means has been widely applied in various fields such as image segmentation, customer segmentation, and anomaly detection. In this paper, we provide an overview of the K-means clustering method, including its algorithmic steps, initialization techniques, evaluation metrics, and practical considerations. We also discuss some of the limitations of K-means and potential extensions to overcome them. Overall, K-means clustering remains a popular and powerful tool for exploratory data analysis and pattern recognition. Additionally, we highlight some of the recent advancements in K-means clustering such as kernel-based variants, fuzzy K-means, and hierarchical K-means. These extensions offer more flexibility and robustness to the K-means algorithm in handling complex datasets and improving its performance.

Keywords: Unsupervised learning, Clustering algorithms, Data Science, K-Means Cluster

Contents

1. Introduction

K-means clustering is an unsupervised learning algorithm that groups similar data points into clusters based on patterns in the data. To train a K-means clustering model, we start by selecting an appropriate dataset and pre-processing it to prepare it for clustering. This typically involves standardizing and scaling the data and reducing its dimensionality using techniques such as PCA or t-SNE. A method for data clustering and pattern recognition in smart grid data is K-Means [1] clustering. It draws attention to its potential for enhancing energy management and the effectiveness of the integration of renewable energy sources into smart networks. This approach is useful for exploratory data analysis and gaining insights into the relationships among different variables in large datasets.

2. Background

Customer segmentation is a widely used marketing strategy that involves dividing customers into distinct groups based on their shared characteristics, preferences, and behaviors. It allows businesses to better understand their customers and tailor their marketing efforts to specific customer segments. The process of customer segmentation typically involves collecting data from various sources such as transactional records, website analytics, social media activity, and surveys. This data is then analyzed using statistical and machine learning algorithms to identify patterns and group customers with similar characteristics. There are several benefits to performing customer segmentation. For instance, it can help businesses develop targeted marketing campaigns that resonate with specific customer segments, improve customer experience by providing personalized recommendations, and increase customer loyalty and retention. To perform customer segmentation, various datasets have been created and analyzed. These datasets vary in size, complexity, and scope, and include attributes such as age, gender, income level, purchase history, and product preferences. Public datasets provided by organizations like Kaggle have enabled researchers and businesses to conduct more extensive analyses of customer segmentation. These datasets provide a wealth of information that can be used to uncover new insights into customer behavior and preferences. In conclusion, customer segmentation is a powerful marketing strategy that can provide valuable insights into customer behavior and preferences. By leveraging data and advanced analytical techniques, businesses can develop targeted marketing campaigns, improve customer experience, and drive revenue growth.

3. Data Description

The Customer Segmentation dataset available on Kaggle, created by user Vetrirah, contains customer data from an online retail store. The dataset includes 8,690 rows of data with 9 attributes for each customer.

3.1 Attribute : Gender

This attribute specifies the gender of the customer, with values of 'Male' or 'Female'. Gender is a categorical attribute with two unique values: '1' for male and '0' for female.

3.2 Attribute : Ever_Married

This attribute specifies the Marital Status of the customer, with values of 'Yes' or 'No'. Ever_Married is a categorical attribute with two unique values: '1' for Yes and '0' for No.

3.3 Attribute : Age

The age of the customer is recorded in years. It is a numerical attribute and not categorical.The data range for this attribute can vary depending on the age range of the customers in the dataset.

3.4 Attribute : Graduated

This attribute specifies the Graduation of the customer, with values of 'Yes' or 'No'. Graduated is a categorical attribute with two unique values: '1' for Yes and '0' for No.

3.5 Attribute : Profession

This attribute specifies the different Professions of the customers, with values of "Engineer", "Doctor","Executive","Artist","Lawyer","Healthcare","Entertainment","Marketing", "Homemaker". Profession is a categorical attribute with 9 unique values: '0' for Engineer, '1' for Doctor, '2' for Executive, '3' for Artist, '4' for Lawyer, '5' for Healthcare, '6' for Entertainment, '7' for Marketing, '8' for Homemaker.

3.6 Attribute : Work_Experience

The Work_Experience of the customer is recorded in years. It is a numerical attribute and not categorical. The data range for this attribute can vary depending on the Work_Experience years of the customers in the dataset.

3.7 Attribute : Spending_Score

This attribute specifies the how much a customer spends money, with values of 'Low', 'Average' and 'High'. Spending_Score is a categorical attribute with two unique values: '0' for Low , '1' for 'Average' and '2' for High.

3.8 Attribute : Family_Size

The Family_Size of a customer depends on how many members are present in his family. It is not a categorical attribute.

3.9 Attribute : Var_1

This attribute is quite unknown, with values of "Cat_1", "Cat_2","Cat_3","Cat_4","Cat_5","Cat_6", "Cat_7". Var_1 is a categorical attribute with 7 unique values: '0' for Cat_1, '1' for Cat_2, '2' for Cat_3, '3' for Cat_4, '4' for Cat_5, '5' for Cat_6, '6' for Cat_7.

3.10 Attribute : Segmentation

This attribute can be the target attribute with 4 categorical values. Those are '0' for A, '1' for B, '2' for C, '3' for D.

This dataset provides a comprehensive view of customer characteristics, including demographics, financial information, education level, occupation, and marital status. The age, annual income, and spending score attributes are particularly useful for clustering customers into different segments based on their purchase behavior. Overall, this dataset can be used for various applications such as customer segmentation, market research, and personalized marketing campaigns. It is a valuable resource for businesses looking to improve customer experience and drive revenue growth.

4. Approach

We will conduct data exploration for the dataset, and then we will conduct Multivariate exploration also for the dataset. We will also train the K-Means Clustering Algorithm .And to conduct the whole procedure We will use the R programming language to perform data analysis.

4.1 Data Preparation

Data preparation involves cleaning, transforming, and organizing raw data to make it suitable for analysis

4.1.1 Data Exploration

Data exploration involves examining and understanding the characteristics and patterns within a dataset, often using statistical methods and visualizations

4.1.1.1 Dataset Dimension

We need to know the dimensions of the dataset, to know how many records are available in the dataset as well as how many attributes are there in the dataset.

```
dim(data)
```

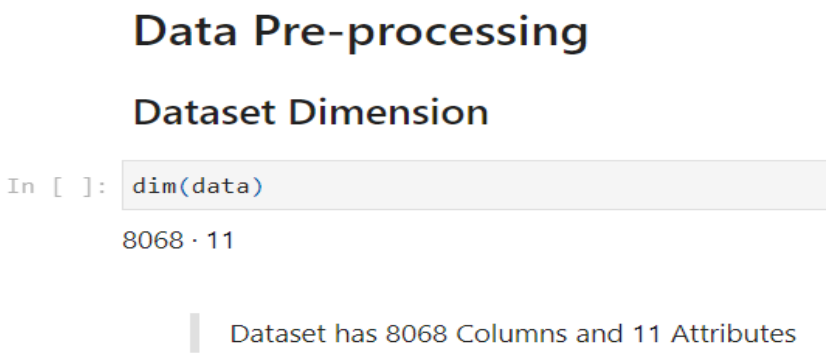

Figure 4.1.1.1: Dataset Dimension

After executing and analysis the dataset through the code, we have found that the dataset has 8068 records and 11 attributes.

4.1.1.2 Attributes Data Types

We also need to know the data types of the eac attribute in the dataset. Because we know that different algorithms supports different kinds of data types. That is the reason we need to know the data types of the attributes in the dataset. And to achieve this, we use structure function of R

```
str(data)
```

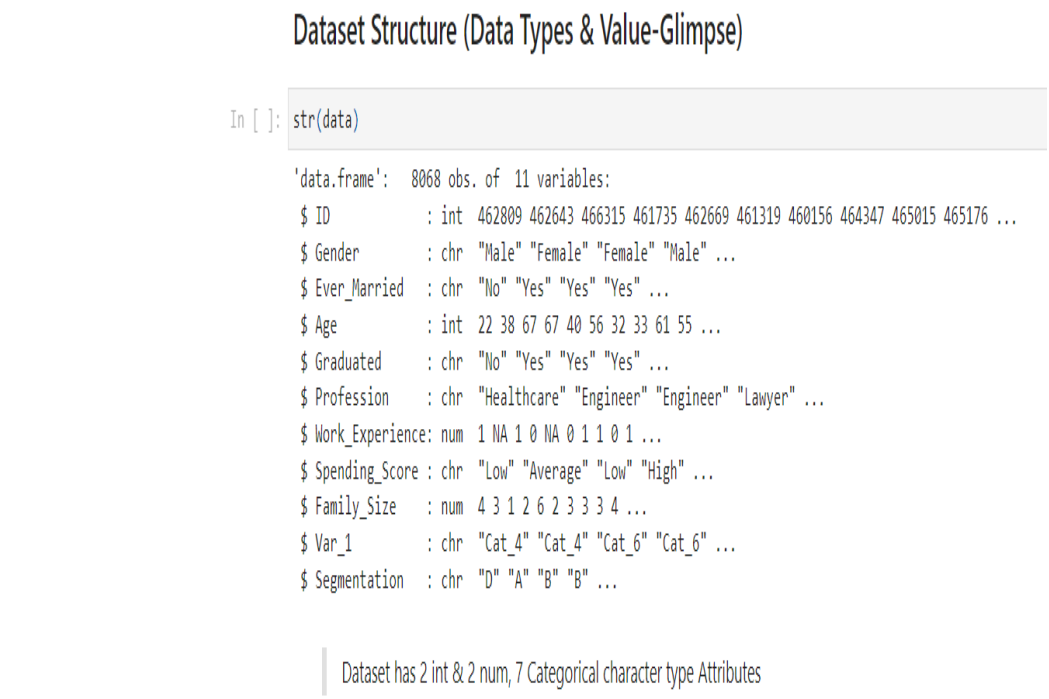

Figure 4.1.1.2: Attributes Data Type

After executing and analysis the dataset through the code, we have found that Only weight attribute is numeric blood attribute is character, rest of all are integer type.

4.1.1.3 Unique Values

After Observing the data-types and some glimpse of the values, we suspected some attributes could possess some categorical values. To find-out the categories and to labeling the categorical values to integer one, we observed the unique values first.

```
unique(data$Gender)
unique(data$Ever_Married)
unique(data$Graduated)
unique(data$Profession)
unique(data$Spending_Score)
unique(data$Var_1)
```

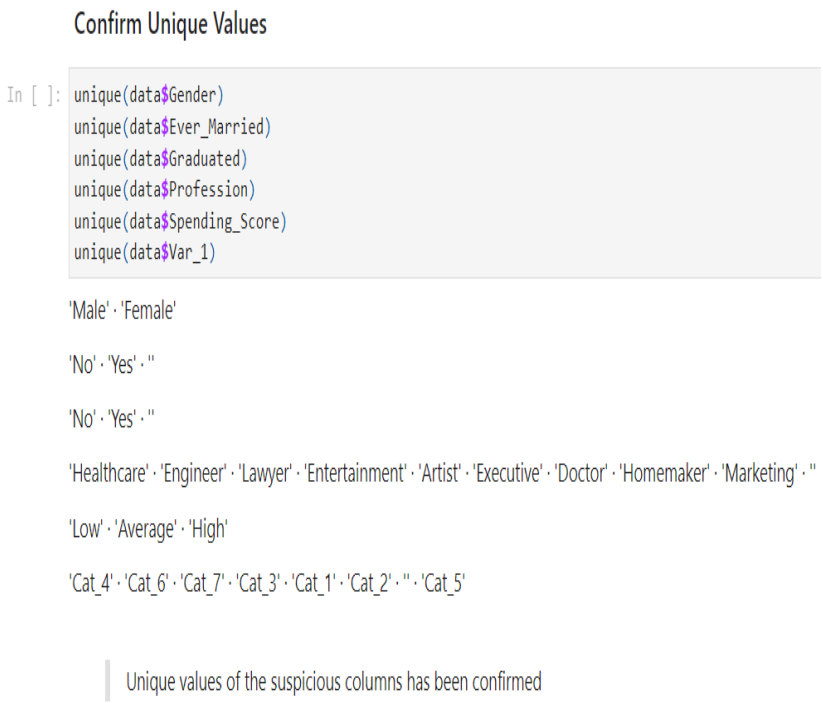

Figure 4.1.1.3: Unique Values

Here We found the unique values of those suspicious columns. We found Gender has 2 Type ”Male”, ”Female” then Ever_Married, And Graduated has 2 ”Yes”, ”No” Spending_Score has 3 ”Low”, ”Average”, ”High”. Finally Profession has 9 and var_1 has 7 categories.

4.1.1.4 Converting Data to Categorical Variables

Machine only understands Numerical values (More specifically Binary). That is the reason, we have to convert all the categorical strings to a category levels or digits.

```
data$Gender <- factor(data$Gender , levels <- c("Male", "Female"),
    labels <- c(1,0), exclude <- NULL)
data$Ever_Married <- factor(data$Ever_Married , levels <- c("Yes", "No
    "), labels <- c(1,0), exclude <- NULL)
data$Graduated <- factor(data$Graduated , levels <- c("Yes", "No"),
    labels <- c(1,0), exclude <- NULL)
data$Profession <- factor(data$Profession , levels <- c("Engineer", "
    Doctor","Executive","Artist","Lawyer","Healthcare","Entertainment"
    ,"Marketing","Homemaker"), labels <- c(0, 1, 2, 3, 4, 5, 6, 7, 8),
     exclude <- NULL)
data$Spending_Score <- factor(data$Spending_Score, levels <- c("Low",
    "Average", "High"), labels <- c(0, 1, 2), exclude <- NULL)
data$Var_1 <- factor(data$Var_1, levels <- c("Cat_1", "Cat_2", "Cat_3"
    ,"Cat_4","Cat_5","Cat_6","Cat_7"), labels <- c(0, 1, 2, 3, 4, 5, 6
    ), exclude <- NULL)
data$Segmentation <- factor(data$Segmentation, levels <- c("A", "B", "
    C","D"), labels <- c(0, 1, 2, 3), exclude <- NULL)
head(data)
```

Converting Data to Categorical Variables using the factor() Function for Better Analysis

In [ ]:

```
data$Gender <- factor(data$Gender , levels = c("Male", "Female"), labels = c(1,0), exclude = NULL)
data$Ever_Married <- factor(data$Ever_Married , levels = c("Yes", "No"), labels = c(1,0), exclude = NULL)
data$Graduated <- factor(data$Graduated , levels = c("Yes", "No"), labels = c(1,0), exclude = NULL)
data$Profession <- factor(data$Profession , levels = c("Engineer", "Doctor","Executive","Artist","Lawyer","Healthcare","Entertainment","Marketing","Homemaker"), labels = c(0, 1, 2, 3
data$Spending_Score <- factor(data$Spending_Score, levels = c("Low", "Average", "High"), labels = c(0, 1, 2), exclude = NULL)
data$Var_1 <- factor(data$Var_1, levels = c("Cat_1", "Cat_2", "Cat_3","Cat_4","Cat_5","Cat_6","Cat_7"), labels = c(0, 1, 2, 3, 4, 5, 6), exclude = NULL)
data$Segmentation <- factor(data$Segmentation, levels = c("A", "B", "C","D"), labels = c(0, 1, 2, 3), exclude = NULL)
head(data)
```

A data.frame: 6 × 11

	ID	Gender	Ever_Married	Age	Graduated	Profession	Work_Experience	Spending_Score	Family_Size	Var_1	Segmentation
	<int>	<fct>	<fct>	<int>	<fct>	<fct>	<dbl>	<fct>	<dbl>	<fct>	<fct>
1	462809	1	0	22	0	5	1	0	4	3	3
2	462643	0	1	38	1	0	NA	1	3	3	0
3	466315	0	1	67	1	0	1	0	1	5	1
4	461735	1	1	67	1	4	0	2	2	5	1
5	462669	0	1	40	1	6	NA	2	6	5	0
6	461319	1	1	56	0	3	0	1	2	5	2

Converted to the Categorical Variable Successfully

Figure 4.1.1.4: Categorical Variables Conversion

After Conversation, we can do further Procedures

4.1.1.5 Convert Data Types to Numaric

The Main reason that we have to convert all the attributes data types to numeric, because it is mandatory to have all numerical data-type while generating co-relationships

```
data$ID <- as.numeric(data$ID)
data$Gender <- as.numeric(as.character(dadatata$Gender))
data$Ever_Married <- as.numeric(as.character(data$Ever_Married))
data$Age <- as.numeric(data$Age)
data$Graduated <- as.numeric(as.character(data$Graduated))
data$Profession <- as.numeric(as.character(data$Profession))
data$Work_Experience <- as.numeric(data$Work_Experience)
data$Spending_Score <- as.numeric(as.character(data$Spending_Score))
data$Family_Size <- as.numeric(data$Family_Size)
data$Var_1 <- as.numeric(as.character(data$Var_1))
data$Segmentation <- as.numeric(as.character(data$Segmentation))
corr_data = data
str(data)
```

Convert Data Types to Numaric

In [ ]:

```
data$ID <- as.numeric(data$ID)
data$Gender <- as.numeric(as.character(data$Gender))
data$Ever_Married <- as.numeric(as.character(data$Ever_Married))
data$Age <- as.numeric(data$Age)
data$Graduated <- as.numeric(as.character(data$Graduated))
data$Profession <- as.numeric(as.character(data$Profession))
data$Work_Experience <- as.numeric(data$Work_Experience)
data$Spending_Score <- as.numeric(as.character(data$Spending_Score))
data$Family_Size <- as.numeric(data$Family_Size)
data$Var_1 <- as.numeric(as.character(data$Var_1))
data$Segmentation <- as.numeric(as.character(data$Segmentation))
corr_data = data
str(data)
```

```
'data.frame':   8068 obs. of  11 variables:
 $ ID              : num  462809 462643 466315 461735 462669 ...
 $ Gender          : num  1 0 0 1 0 1 1 0 0 0 ...
 $ Ever_Married    : num  0 1 1 1 1 1 0 0 1 1 ...
 $ Age             : num  22 38 67 67 40 56 32 33 61 55 ...
 $ Graduated       : num  0 1 1 1 1 0 1 1 1 1 ...
 $ Profession      : num  5 0 0 4 6 3 5 5 0 3 ...
 $ Work_Experience: num  1 NA 1 0 NA 0 1 1 0 1 ...
 $ Spending_Score : num  0 1 0 2 2 1 0 0 0 1 ...
 $ Family_Size     : num  4 3 1 2 6 2 3 3 3 4 ...
 $ Var_1           : num  3 3 5 5 5 5 5 5 6 5 ...
 $ Segmentation    : num  3 0 1 1 0 2 2 3 3 2 ...
```

All Data Types Converted to Numeric Successfully

Figure 4.1.1.5: Convert Data Types to Numaric

As we converted all the string categorical to numerical category, the data types of some attribute got shifted during that process. So we need to convert all the attribute to numeric format.

4.1.1.6 Attributes Names

We need to know if the attribute names are perfect or not. Because if there is any space or special characters between the attribute names, then it will create an issue while we perform any analysis.

```
names(data)
```

Figure 4.1.1.6: Attributes Names

After executing and analysis the dataset through the code, we have found that all the attributes name are useable. No problem found.

4.1.1.7 Data Summary

Data Summery is a very important step in data exploration. Because it gives us a brief idea about the dataset. It gives us the idea about the mean, median, mode, standard deviation, minimum value, maximum value, and quartiles of the dataset. And to achieve this, we use summery function of R

```
summary(data)
cat("Total Missing Values = ",sum(is.na(data)))
```

After executing and analysis the dataset through the code, we have found that Ever_Married(140), Graduated(78), Profession(124), Work_Experience(829), Family_Size(335), Var_1(76) attributes have missing values. Overall total 1582 missing values are available in this dataset. Based on the differences between Mean and Median, the Attribute -> Age, Work_Experience & Family_Size could have outliers.

Dataset Summery(Mean, Median, Min, 1st-Quadrant, 3rd-Quadrant)

Summery is used to find-out, which columns has Missing Values (How many) & Outliers(There is a big difference between MEAN & MEDIAN respect to the other columns differences)

In [ ]:
```
summary(data)
cat("Total Missing Values = ",sum(is.na(data)))
```

```
       ID            Gender       Ever_Married        Age        
 Min.   :458982   Min.   :0.0000   Min.   :0.0000   Min.   :18.00  
 1st Qu.:461241   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:30.00  
 Median :463473   Median :1.0000   Median :1.0000   Median :40.00  
 Mean   :463479   Mean   :0.5475   Mean   :0.5857   Mean   :43.47  
 3rd Qu.:465744   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:53.00  
 Max.   :467974   Max.   :1.0000   Max.   :1.0000   Max.   :89.00  
                                   NA's   :140                     
   Graduated       Profession    Work_Experience  Spending_Score  
 Min.   :0.0000   Min.   :0.000   Min.   : 0.000   Min.   :0.0000  
 1st Qu.:0.0000   1st Qu.:2.750   1st Qu.: 0.000   1st Qu.:0.0000  
 Median :1.0000   Median :3.000   Median : 1.000   Median :0.0000  
 Mean   :0.6218   Mean   :3.561   Mean   : 2.642   Mean   :0.5461  
 3rd Qu.:1.0000   3rd Qu.:5.000   3rd Qu.: 4.000   3rd Qu.:1.0000  
 Max.   :1.0000   Max.   :8.000   Max.   :14.000   Max.   :2.0000  
 NA's   :78       NA's   :124     NA's   :829                      
  Family_Size       Var_1        Segmentation
 Min.   :1.00   Min.   :0.000   Min.   :0.000  
 1st Qu.:2.00   1st Qu.:3.000   1st Qu.:1.000  
 Median :3.00   Median :5.000   Median :2.000  
 Mean   :2.85   Mean   :4.139   Mean   :1.562  
 3rd Qu.:4.00   3rd Qu.:5.000   3rd Qu.:3.000  
 Max.   :9.00   Max.   :6.000   Max.   :3.000  
 NA's   :335    NA's   :76                     
Total Missing Values =  1582
```

> Missing values found in Ever_Married(140), Graduated(78), Profession(124), Work_Experience(829), Family_Size(335), Var_1(76)
> AND
> Based on the differences between Mean and Median, the Attribute -> Age, Work_Experience & Family_Size could have OUTLIERS

Figure 4.1.1.7: Data Summary

4.1.2 Handling Missing Values

Missing values can lead any analysis to wrong conclusion. So we have to handle missing values. There are two ways to handle missing values. First is to discard the missing values and second is to use mean/media/mode of the attribute to fill the missing values. We have used both the methods to handle missing values.

4.1.2.1 Discard Missing Values

Sometimes the missing values are very less in number. So we can discard the missing values. And to achieve this, we use na.omit function of R

```
data <- na.omit(data)
cat("Total Missing Values = ",sum(is.na(data)))
```

Fix the Missing values By Dropping them

In [ ]:
```
data <- na.omit(data)
cat("Total Missing Values = ",sum(is.na(data)))
```

```
Total Missing Values =  0
```

> Missing Values has been Removed Successfully

Figure 4.1.2.1: Handling Missing Values

After executing and analysis the dataset through the code, we learn that there are total 1582 missing values in the dataset. And after discarding the missing values, we have found that there are 0 missing values in the dataset. And the dimension of the dataset is 8068 x 11.

4.2 Co-Relations

We know that one or many things depends also on one or many other things. This dependency is called co-relation. We have to find out the strongly positive and strongly negative relationship between the attributes. It will help us to understand if we want a particular result, then exactly which attributes we have to manipulate.

```
cor_matrix <- cor(corr_data, use = "pairwise.complete.obs")
corrplot(cor_matrix, method = "circle")
corrplot(cor_matrix, method = "number")
```

From the relationship graph, the attributes Age and Weight are strongly positively related. And the attributes Age and blood are negatively correlated. The attributes Caesarin and heart are positively correlated. The attributes Caesarin and Delivery_time are negatively correlated. Finally The attributes Delivery_number and heart are positively correlated.

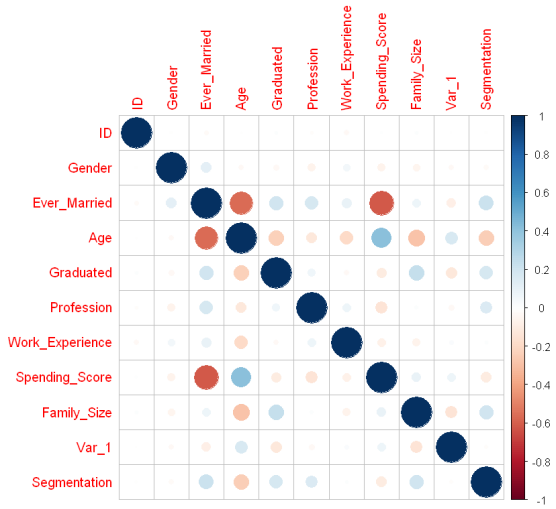

Figure 4.2.0.0: Co-Relation Heat Map Circle

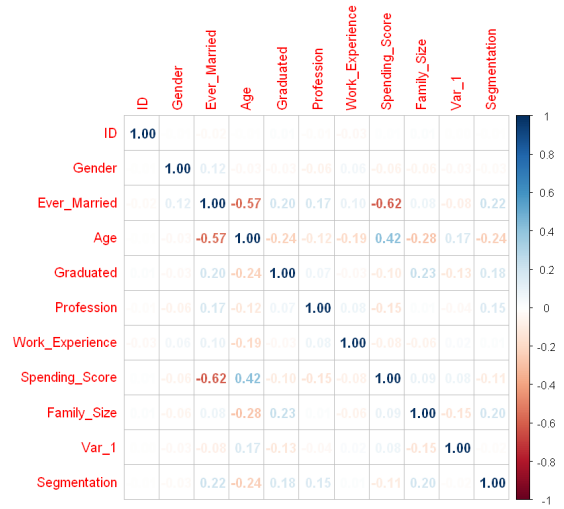

Figure 4.2.0.0: Co-Relation Heat Map Number

Figure 4.2.0.0: Co-Relationship Diagram

Selected Features : Ever_Married, Age, Graduated, Profession, Spending_Score, Family_Size

4.3 Dataset Split

To build a machine learning model, we have to split the dataset into two parts. One is training dataset and another is testing dataset. We have to train the model with training dataset. And to achieve this, we use some functions of R

```
set.seed(123) # for reproducibility
train_indexes <- sample(nrow(data), round(0.8 * nrow(data)))
train_data <- data[train_indexes,]
test_data <- data[-train_indexes,]
```

Split the dataset into training (80%) and testing sets (20%)

```
In [ ]: set.seed(123) # for reproducibility
train_indexes <- sample(nrow(data), round(0.8 * nrow(data))) # Splitting 80% For Training Purpose and 20% for Testing Purpose
train_data <- data[train_indexes, ]
test_data <- data[-train_indexes, ]
```

Split the Dataset into Training set and Testing set Successfully

Figure 4.3.0.0: Data Split

4.4 Feature Selection

Feature selection is the process of choosing relevant and significant features from a dataset to improve model performance and reduce complexity. The goal of feature selection is to eliminate irrelevant or redundant features that may negatively impact model accuracy and slow down processing time. This process can also help to identify important variables and relationships between features, providing valuable insights for further analysis.

```
features = c("Ever_Married", "Age", "Graduated", "Profession", "
    Spending_Score", "Family_Size")
X_train = train_data[, features]
y_train = train_data$Segmentation
```

Choose the attributes for training the model

```
In [ ]: features = c("Ever_Married", "Age", "Graduated", "Profession", "Spending_Score", "Family_Size")
X_train = train_data[, features]
y_train = train_data$Segmentation
```

Attributes has been chosen Successfully

Figure 4.4.0.0: Feature Selection

4.5 Model Train : K-Means Clustering

K-means clustering is an unsupervised learning technique that segments data into K distinct clusters based on their similarity. The process of training a K Means Clustering model involves randomly initializing K cluster centroids and iteratively updating their positions until convergence.

```
kmeans_model <- kmeans(X_train, centers = 7)
```

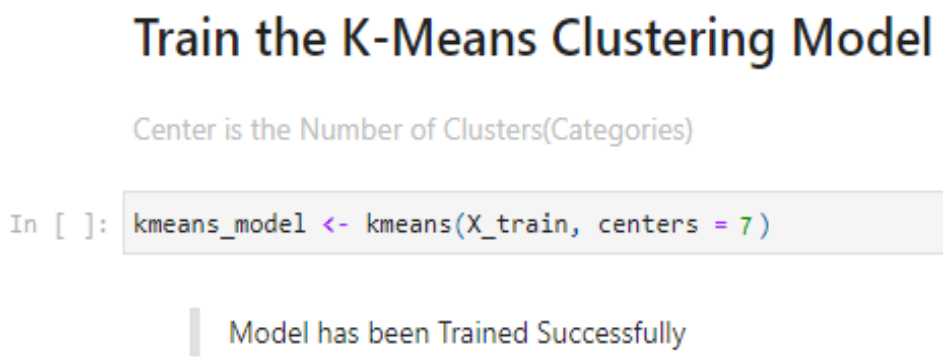

Figure 4.5.0.0: K-Means Clustering Model Training

4.6 Visualize the Clusters

In this K-Means Clustering model, we have used center=7 that means there will be 7 clusters. Those 7 clusters is highlighted through 7 different colors.

```
fviz_cluster(kmeans_model, data = X_train)
```

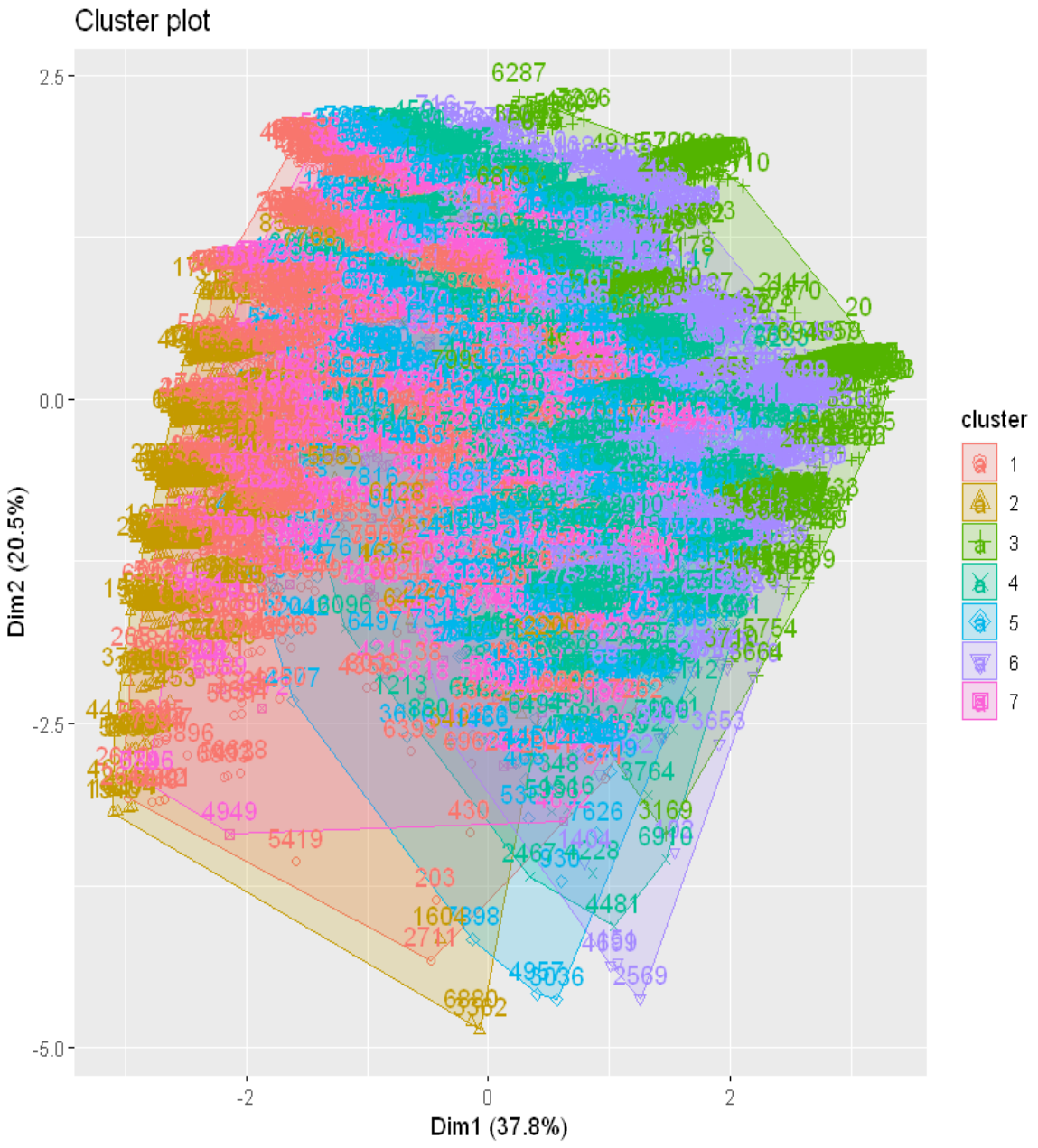

Figure 4.6.0.0: Cluster Visualization

5. Results

We applied K-means clustering to our dataset and obtained two clusters with sizes 1915 and 3417. The cluster means show that Cluster 1 has higher values for Ever_Married, Age, Graduated, Profession, and Spending_Score, but lower values for Family_Size compared to Cluster 2. This suggests that there are distinct differences in demographic and spending behavior between the two groups. The within-cluster sum of squares (WSS) shows that Cluster 1 has higher variability than Cluster 2, indicating that this group is less homogeneous. We present the centers of the attributes for each cluster, which represent the typical profile of individuals in each group. Our analysis suggests that these profiles can inform targeted marketing or communication strategies. The total number of iterations was one, implying that the algorithm converged quickly.

View the Trained Result

In [ ]:
```
cluster_display = X_train
cluster_display$cluster <- as.factor(kmeans_model$cluster)
head(cluster_display)
```

A data.frame: 6 × 7

	Ever_Married	Age	Graduated	Profession	Spending_Score	Family_Size	cluster
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<fct>
2985	0	26	1	3	0	4	1
3037	1	49	0	3	0	1	4
2692	1	45	1	3	1	2	5
646	0	31	0	1	0	4	1
5177	1	66	1	4	2	2	6
3607	0	28	0	5	0	8	1

In [ ]:
```
print("Centers of the Attributes for each Cluster")
kmeans_model$centers
cat("Total Number of Iteration = ", kmeans_model$iter)
```

```
[1] "Centers of the Attributes for each Cluster"
```

A matrix: 7 × 6 of type dbl

	Ever_Married	Age	Graduated	Profession	Spending_Score	Family_Size
1	0.22780749	27.85882	0.52513369	3.901604	0.18181818	3.296257
2	0.03469388	20.49796	0.01020408	4.665306	0.04081633	4.034694
3	0.94472362	79.75879	0.60552764	3.708543	1.15075377	1.884422
4	0.84241824	50.25768	0.81169475	3.104063	0.74430129	2.777998
5	0.64364364	41.07508	0.76376376	3.112112	0.56056056	2.583584
6	0.93241379	63.41655	0.74068966	3.208276	0.91448276	2.510345
7	0.46262887	34.53222	0.69072165	3.449742	0.39046392	2.693299

```
Total Number of Iteration =  4
```

Figure 5.0.0.0: View Trained Clusters

Here, we can that the model has defined each row as a cluster based on the learnings. Now, for the 7th Row, the centroids of those columns are the closest to the mean values of those attributes. This is the reason, we can say that the centroids of the 7th row are called the actual centroids for those attributes.

Overall, our K-means clustering results provide insights into the structure of our dataset and highlight meaningful differences and similarities between groups of individuals. Future research could investigate the underlying factors driving these differences and explore potential implications for business strategy.

6. Conclusion

We may infer from the K Means clustering algorithm's output that the method was successful in clearly separating comparable data points into various groups. This implies that a number of applications where data has to be separated based on similarities might benefit from the technique. The exact training process parameters, such as the number of clusters and distance metric, can have a big impact on how well the algorithm performs. Therefore, depending on the particular type of data being studied, it is crucial to properly select these factors.

K Means clustering is a strong method for data analysis that may be used in a variety of contexts. It is crucial to remember that the caliber and organization of the input data will have a significant impact on how accurate the outcomes are.

Bibliography

[1] Kayalvily Tabianan, Shubashini Velu, and Vinayakumar Ravi. K-means clustering approach for intelligent customer segmentation using customer purchase behavior data. *Sustainability*, 14(12):7243, 2022.