ΟΙΚΟΝΟΜΙΚΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΑΘΗΝΩΝ

ATHENS UNIVERSITY
OF ECONOMICS
AND BUSINESS

ΣΧΟΛΗ
ΕΠΙΣΤΗΜΩΝ &
ΤΕΧΝΟΛΟΓΙΑΣ
ΤΗΣ
ΠΛΗΡΟΦΟΡΙΑΣ
SCHOOL OF
INFORMATION
SCIENCES &
TECHNOLOGY

ΤΜΗΜΑ
ΠΛΗΡΟΦΟΡΙΚΗΣ
DEPARTMENT OF
INFORMATICS

MASTER'S THESIS

"Customer Churn Rate/Attrition Rate: Prediction of customer loss"

Company: NBG

LAFAZANIDI ALEXANDRA

Master Thesis, AUEB (Athens University of Economics & Business),
Data Science, School of Information, Sciences and Technology,
Department of Informatics
Duration: July 1st - October 31st

MASTER'S THESIS

"Customer Churn Rate/Attrition Rate: Prediction of customer loss"

Academic supervisor:

Dr. Panagiotis Papastamoulis

NBG Supervisors:

Mr. Kalampokas Miltiadis

Mr. Termitzoglou Athanasios

Ms. Zevgaropoulou Georgia

Prepared by :

Lafazanidi Alexandra

Athens University of Economics and Business (AUEB),

Department of Informatics, MSc in Data Science

July 1st - October 31st

Description:

Build a machine learning model to predict the probability that a customer will cease his relationship with NBG within a month.

Churn rate refers to the proportion of customers who ‘’leave’’ during a given time period. It is probably the most important KPI in order to measure customer satisfaction. There are many methods to approach this problem. The case here, is to try to identify the signals of attrition using only transactional data. The candidate will have the chance to deal with a critical issue for the majority of the companies. The concept of attrition is not only to retain customers but to improve profitability and increase lifetime value. The candidate may adjust the project to his interests and he will have NBG’s guidance and support during the project period.

Data: 2 years transactional data from internet banking.

With the completion of this thesis I would like to thank kindly my academic supervisor Mr. Papastamoulis Panagiotis for all his tips and advice, as well as his willingness and knowledge that I got from him throughout the diplomatic process.

Abstract

Customer churn is commonly defined as 'the probability that a customer will voluntary cancel the existing contract within the next X number of months', where X is defined according to the type of business and the problem set in particular. It is also referred as loss of clients or customers. (Forecast Analytics, 2015). Customer churn is one of the most important metrics for a growing business to evaluate its customers' satisfaction and while it's not the happiest measure, it is a number that can give a company the hard truth about its customer retention. The higher the churn rate, the more many investors doubt the company's viability. From an analytics perspective, this becomes a classic binary classification problem as all customers either churned – or did not churn. There are different approaches to study customer churn via suitable predictive models but in many cases, when there are non-existent churners, the customer churn is not traceable. So, building an appropriate model is a procedure with high complexity.

In our study, we had to deal with this issue, so we developed a dual step building approach which consisted of clustering task and then the classification task. With this regard, firstly, customer segmentation was implemented on the bank's client base. The clients through K-means, were divided and depicted into three clusters based on their RFD related features with the aim of extracting a logical definition of churn and administering flags (churn- no churn) to the clients. Secondly, based on the above flags we proceeded to the second step – the model building phase. The utilized algorithms were the logistic regression and the Naïve Bayes classifier. Regarding the results, logistic regression had better performance on our data than Naïve Bayes.

Key words: Customer churn, Customer Relationship Management, Customer segmentation, Data Mining

Table of Contents

Description: ... 1
Abstract ... 3
Chapter 1 – Introduction ... 9
  1.1 Introduction ... 9
  1.2 Calculating customer churn rate. ... 11
  1.3 Problem definition ... 11
  1.4 Purpose of the project-research ... 12
  1.5 Research Questions ... 12
  1.6 Thesis structure ... 12
Chapter 2 – Literature Review ... 16
  2.1 Introduction ... 16
  2.2 Customer Relationship Management (CRM) ... 16
  2.3 Data mining in the churn management ... 19
    2.3.1 Big Data and Data Mining ... 19
    2.3.2 Data Mining and RFM analysis ... 21
    2.3.3 Data Mining and Clustering ... 22
    2.3.4 Data Mining and Classification ... 24
    2.3.5 Another approach ... 25
Chapter 3 – Strategy methodology ... 27
  Progress of the research ... 27
  3.1 Deep understanding ... 28
  3.2 Form of a general plan ... 29
  3.3 Data Collection (Selection, Process, Extraction) ... 30
  3.4 Data Pre-Processing ... 32
  3.5 Data Mining (Clustering) ... 34
  3.6 Evaluation – Interpretation ... 36
  3.7 Visualization ... 38
  3.8 Data Mining (Classification) ... 38
  3.9 Evaluation – Interpretation ... 41
Chapter 4 – Main analysis and Results ... 44
  4.1 Introduction ... 44
  4.2 Main Analysis ... 44

4.2.1 Variables and interpretation 44
4.2.2 The whole dataset (2 years) 45
4.2.3 20 months data (Observation period) 50
4.2.3.1 Preprocessing 50
4.2.3.2 Plots 58
4.2.3.3 Correlation 65
4.2.3.4 Clustering 67
4.2.3.4.1 1st approach 68
4.2.3.4.2 2nd approach 74
4.2.3.4.3 3rd approach 81
4.2.3.4.4 4th approach 88
4.2.3.5 Conclusions 95
4.2.3.6 Visualization of the best clustering 99
4.2.3.7 Some statistics about the clusters with the "optimal" k 101
4.2.3.7 Comments and interpretation 107
4.2.4 Data for the last four months (Lead period) 109
4.2.5 Whole data with labels 113
4.2.5.1 Summary and visualization 113
4.2.5.2 Classification 119
4.2.5.2.1 Logistic regression 119
4.2.5.2.2 Naïve Bayes 126
4.2.5.2.3 An additional try 128
4.2.5.3 Conclusions 128
4.2.5.4 Some extra notes 129
Chapter 5 Limitations & Extensions 130
5.1 Limitations 130
5.2 Extensions 130
References 132
Appendices 141

List of figures

Figure 1: Chapters based on semantic context 14
Figure 2: Chapters based on technical context 15
Figure 3: Customer Relationship Management (TECHONESTOP, 2019) 18
Figure 4: Big data parts 20
Figure 5: Evolution in the quest for information (Lejeune, 2001) 21
Figure 6: Before and after Clustering (Yoseph, Malim, AlMalaily, 2019) 23
Figure 7: Binary classification (Lucidworks, 2019) 24
Figure 8: Example of ECDF distribution for a sample of 20 clients and action level 95% (Forecast Analytics, 2019) 26
Figure 9: Research progress 28
Figure 10: Feature Engineering 31
Figure 11: Outliers (Rathi, 2018) 32
Figure 12: Detection of outliers using boxplots and IQR 33
Figure 13: Feature selection (Rathi, 2018) 34
Figure 14: K-means clustering (Lucidwords, 2019) 35
Figure 15: Elbow method (Yse, 2019) 36
Figure 16: Silhouette analysis (Vora, 2019) 37
Figure 17: Logistic Regression (Lucidwords, 2019) 39
Figure 18: Random Forest (Global Software Support, 2019) 40
Figure 19: Naive Bayes classifier (ThatWare, 2019) 41
Figure 20: Confusion matrix 42
Figure 21: Data variables 45
Figure 22: ibanknew spark data frame 45
Figure 23: Function count_results 46
Figure 24: Number of distinct clients 46
Figure 25: Plot of the variable start_trans_end_data (2 years data) 47
Figure 26: Statistical metrics for the variables 'start_trans_end_data' and 'average_trans_dur' for the 2 years period 48
Figure 27: Data with removing the clients who got to the internet banking the last 4 months 49
Figure 28: Number of clients after removing those who had first transaction in the last four months 49
Figure 29: data1 (until 4 months before) 50
Figure 30: Number of users in data1 50
Figure 31: Number of clients of freq_equp2 51
Figure 32: Scatter matrix 52
Figure 33: Boxplots for outliers 54
Figure 34: IQR score for numeric variables of the data 54
Figure 35: IQR score Boolean table 55
Figure 36: Number of users without outliers 55
Figure 37: Function for statistics 56
Figure 38: Table for Statistics 57

Figure 39: Histograms for countt 58
Figure 40: Histograms for recency 59
Figure 41: Histograms for frequency 60
Figure 42: Histograms for max_dist 61
Figure 43: Histograms for average_trans_dur 62
Figure 44: Histograms for diff_maxd_mind_days 63
Figure 45: Histograms for start_trans_end_data 64
Figure 46: Correlation table between the variables 65
Figure 47: Correlation matrix 66
Figure 48: Function for standardizing spark features 67
Figure 49: Elbow and Silhouette for seed =42 approach 1 log data 69
Figure 50: Elbow and Silhouette for seed = 650 approach 1 log data 71
Figure 51: Elbow and Silhouette for seed = 10 approach 1 log data 73
Figure 52: Descriptive elements in approach 1 and k =3 73
Figure 53: Descriptive elements in approach 1 and k =4 74
Figure 54: Elbow and Silhouette for seed =42 approach 2 log data 76
Figure 55: Elbow and Silhouette for seed =500 approach 2 log data 77
Figure 56: Elbow and Silhouette for seed =10 approach 2 log data 79
Figure 57: Descriptive elements in approach 2 and k =3 80
Figure 58: Descriptive elements in approach 2 and k =4 80
Figure 59: Elbow and Silhouette for seed =42 approach 3 initial data 82
Figure 60: Elbow and Silhouette for seed =346 approach 3 initial data 84
Figure 61: Elbow and Silhouette for seed = 12 approach 3 initial data 85
Figure 62: Descriptive elements in approach 3 and k =3 86
Figure 63: Descriptive elements in approach 3 and k =4 87
Figure 64: Descriptive elements in approach 3 and k =5 87
Figure 65: Elbow and Silhouette for seed = 42 approach 4 initial data 89
Figure 66: Elbow and Silhouette for seed = 350 approach 4 initial data 91
Figure 67: Elbow and Silhouette for seed = 13 approach 4 initial data 92
Figure 68: Descriptive elements in approach 4 and k =3 93
Figure 69: Descriptive elements in approach 4 and k =4 94
Figure 70: Descriptive elements in approach 4 and k =5 94
Figure 71: Summary of all approaches in clustering phase 96
Figure 72: Centroids of clusters (approach 2, k=3) 98
Figure 73: Explained variation per principal component 99
Figure 74: 2-D Visualization 100
Figure 75: 3-D Visualization 100
Figure 76: Cluster 0 102
Figure 77: Cluster 1 104
Figure 78: Cluster 2 106
Figure 79: Median and average values for each cluster 107
Figure 80: max_dist for studied users in the period of the last four months 110
Figure 81: Function for extracting labels 110
Figure 82: Number of churners/not churners in each cluster 112
Figure 83: Number of churners /not churners in total 112

Figure 84: Total dataframe with labels.......... 113
Figure 85: Histograms/Statistics for the whole data with labels 115
Figure 86: Percentage of churn/no churn in each cluster.......... 115
Figure 87: Percentage of churn/no churn in total.......... 116
Figure 88: Interactive histogram for Max_dist in total with groups 117
Figure 89: Histogram for max_dist in cluster 0 for each group.......... 118
Figure 90: Histogram for max_dist in cluster 2 for each group.......... 118
Figure 91: Summary of the model logit.......... 120
Figure 92: Correlation matrix (for the logit).......... 120
Figure 93: Summary of the modelnew logit.......... 121
Figure 94: performance of the modelnew logit 122
Figure 95: stepwise regression.......... 123
Figure 96: Summary of model1_log 124
Figure 97: Evaluation of logistic on log data 125
Figure 98: stepwise on log (data+1) 126
Figure 99: Performance of Naive Bayes 127
Figure 100: Predicted probabilities on test set 128
Figure 101: Final results on jupyter notebook 129

Chapter 1 – Introduction

1.1 Introduction

Customer churn, also known as customer attrition, customer turnover, or customer defection, is the loss of clients or customers of a company in a predefined time period. When a user, player, subscriber, decreases the business operations - during the partnership between customer and business - or releases the contract with the company, then this partner is considered to be a customer churn. (Netigate, 2019) Churn rate is defined as the rate at which churn occurs. Changes in a company's churn rate could be a signal for many things· Reduction of a company's churn rate is a sign that the company's viability is not at risk, whereas the opposite shows that the company should invest more on its improvement. (Gallo, 2014) In order to put the above in a practical sense ,the below example is helpful : Assuming that the company is a bank (as in our case) , from the side of clients, not renewing a service agreement, contract or even closure of an account of some sort and turning to another bank. Mainly, if the reason for leaving the company is the dissatisfaction, then the things get worse because this means word-of-mouth - bad reputation for the specific bank. As a result it seems crucial for the company's administration to prevent such behaviors. Analyzing the past history of the potential customers systematically, could decrease the churn rate. Therefore, the churn rate is too important by playing a dual role: with the churn rate not only could somebody understand by defining it, what happened in the last period, but also they could predict what's going to happen in the next one. For this purpose, large data concerning customers, is maintained and performing a proper analysis on them makes possible for the people in charge, to predict these one that might churn.

But it is a fact that for a company- organization retaining existing customers costs less than acquiring new customers. And the reason is following: The client - customer, who stays with a company over time in an active way, gives the people in charge the chance to develop more personalized content for him, gain referrals, and earn trust. The company could endorse a new product and earn positive word-of-mouth marketing. (Wonder, 2018) Clients want to cooperate and spend money with brands that make them feel special and therefore a strategy to eliminate churn rate would be the implementation of personalized services and experiences that leads to awareness of the customers' expectations, and satisfaction improvement. (Galetto, 2016)

And at a time when customers are looking for ways to fulfill their needs in faster, easier, and less expensive ways, the ability of people in charge to keep them feeling important via highly targeted, customized messaging and offers will be key to retaining their business. The big challenge although of the churn analysis is to identify and understand customers who are still making up their mind about the continuation of the contract with a business or not, because that is when you have the best chance of influencing them. And here comes the basic question: How do you know the exact time when customers are still making up their mind about churning or no churning? The answer is that you cannot know exactly. In order to succeed this, an observation of the customers, at a time when it is reasonable that they think about their next transaction, is vital. And that time is placed not immediately after the last renewal and not right before the upcoming renewal where they might churn. The exact interval of time depends on the cost and the duration of commitment of service, which means that the longer the commitment and the more expensive the service is, the longer the lead time is. According to some surveys, the lead time for large businesses is from 2 to 4 months. (Fighting Churn, 2019) In my case a 4 month lead period was used. (The reason is explained in the chapter 5 in main analysis (pages 43-44)).

Companies and talking more specifically about banks usually make a distinction regarding the term churn· the voluntary churn and the involuntary churn. Voluntary churn occurs due to a decision by the customer himself to turn to another company or service provider, whereas involuntary churn occurs due to circumstances which concern the cards (debit and credit cards) or the customers. Talking about the cards , we could have payment failures such as loss or cancellation of a card or not updating the information of the card which results in its expiration. As far as the clients are concerned we could take as example a presumptive customer's relocation to a distant location or even death. In most applications, involuntary reasons for churn are excluded from the analytical models. Only voluntary churn is at the center of analysts' interest, because it typically occurs due to factors of the company-customer relationship which companies control, such as how billing interactions are handled or how help is provided to client complaints. (Wikipedia, 2019) (In my case, I had no demographic elements about clients in my disposal, so there was not a manner to distinguish voluntary churn from involuntary one).

1.2 Calculating customer churn rate.

There is a variety of ways to measure-calculate the customer churn rate. Indicatively, churn rate may represent the total number of customers lost or the percentage of customers lost compared to the company's total customer count. For example if a company had 200 clients and lost 4 the last month its monthly churn rate is 2 percent. In addition, it may depict the value of recurring business lost, or the percentage of recurring value lost. Other organizations calculate churn rate for a certain period of time, such as quarterly periods or fiscal years. (Galetto, 2016) The most common and widespread method in order to calculate the customer churn is to subtract the number of customers that have been lost at a specified time period from the total number of clients that the company had at the beginning of that time and divide this difference by the total number of clients the company had at the beginning of that period. For example, if a business starts with 300 customers and at the end of the month it ends up with 200, then the customers churn is

$$\frac{(300-100)}{300}=\frac{200}{300}=0,6666=66,66\%$$ (Bonnie, 2017)

The following study aims at recognizing clients who tend to churn or not and finding an efficient and accurate predictive model for customer churn in a bank utilizing machine learning techniques.

1.3 Problem definition

Not always can churn be so clearly defined. Any transaction might be the last interaction with the company, or could be part of continuing interactions with the company. For these non-contractual businesses, as in case of banks , if a client does not complete a critical event on the platform within a window of time (lead-period), then he could be considered as" churn" ,in other words, he seems to abandon the platform. (Forecast Analytics, 2019) As a result it seems vital to recognize the churners before they end up falling away from the bank. This model aims to recognize the customers who tend to churn in close future and predict the probability of this situation.

1.4 Purpose of the project-research

Purpose of my research is to build a machine learning model to predict the probability that a customer will cease his relationship with NBG within a month by separating the clients of the bank in first level into churners and no churners.

1.5 Research Questions

- ✓ Why studying the churn is so important for the companies?
- ✓ How the term "customer churn" could be defined in a non-contractual setting (in which there are no customers who have left the company so as to have (real-existing) labels for churn about them?
- ✓ How will we predict customer churn if we have only active clients?
- ✓ Which features should be extracted from the transaction data and be utilized as input in the clustering model?
- ✓ How long should be the window for the lead period? Is it fixed or not?
- ✓ Will the resulting clusters be meaningful? What may each cluster represent?
- ✓ Which model should I use for the classification phase?
- ✓ How does someone reduce their company's customer churn rate?

1.6 Thesis structure

To begin with, my thesis report starts with some definitions and explanations about the research problem –customer churn – and its context. (Chapter 1).Analyzing the related terms was the expedient so as to give prominence to the significance of the problem. Encountering deeper meanings was the means for providing the need for continuous thorough search and the building of efficient and accurate models for customer churn prediction. Subsequently, a variety of some methods is mentioned for measuring –calculating the churn based on the preference of many organizations (due to the complex nature of the problem itself). Then, leveraging the definition of the project problem and the purpose of this research, a series of questions which are come out of the topic "churn" is being shown. Chapter 1 is succeeded by Chapter 2 which is a literature review consisted of explanations of terms, closely related to the churn one as it is the “customer relationship management(CRM)” which is the fundament for any predictive model and “customer lifetime value (CLV) “. Subsequently, having mentioned and reviewed different models for churn prediction (data mining - machine learning approaches and not), I move on to the next chapter (Chapter 3) presenting my personal choices and describing my research methodology. Afterwards, seguing from the general to the more specialized, follows the detailed analysis of my project strategy which is an inseparable part of the previous section as well as the results coming from the implementation of my methodology and their interpretation (Chapter 4). After the “hard work”, as it is usual in every research, comes after the part with the conclusions about our project theme (Chapter 5). Ultimately, follows the section (Chapter 6) consisting of firstly the limitations that we faced with during the whole procedure and secondly some suggestions for further extension of the model and further research and then the last section (Chapter 7) with some appendings.

Note: If we wanted to be more explicit as far as the division of the project to chapters is concerned, focusing on technical part, we would say that there are nine (9) chapters instead of six (6). These are the previous six, with the addition of three additional chapters · one at the beginning, consisting of contents, list of figures and list of tables and the other two at the end, consisting of the references that were utilized and some appendings.

Here are two schematic figures that illustrate the above steps-chapters. The figure 1 focuses on the semantic context whereas the figure 2 focuses on the technical one of this report.

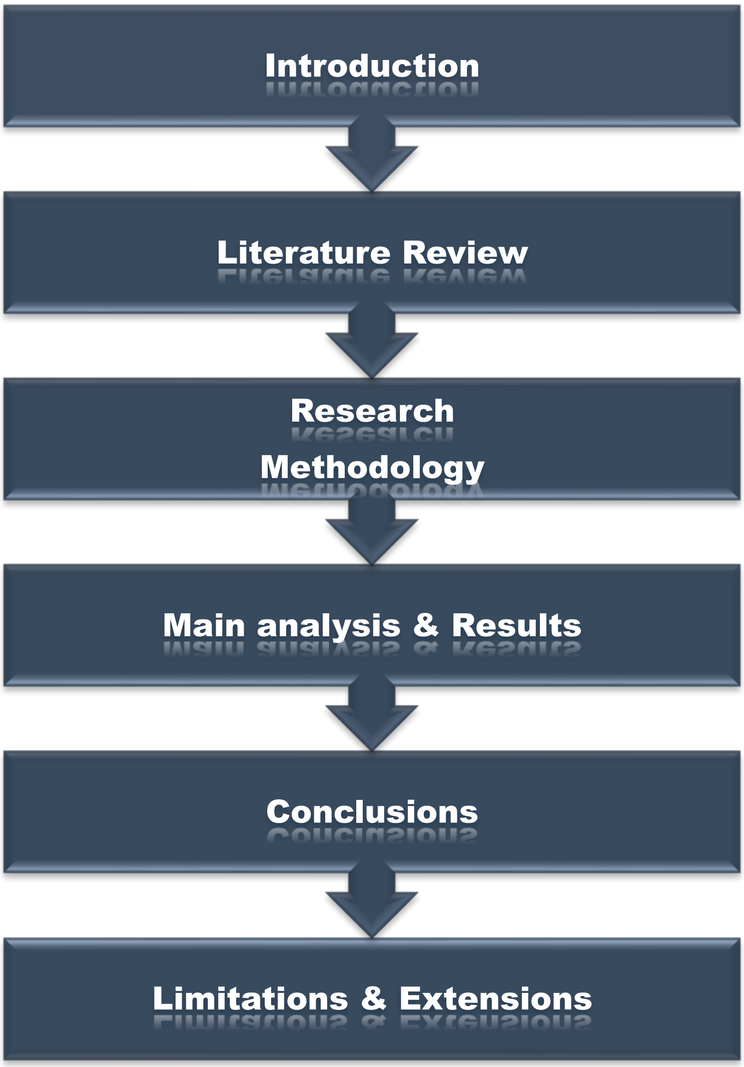


Figure 1: Chapters based on semantic context

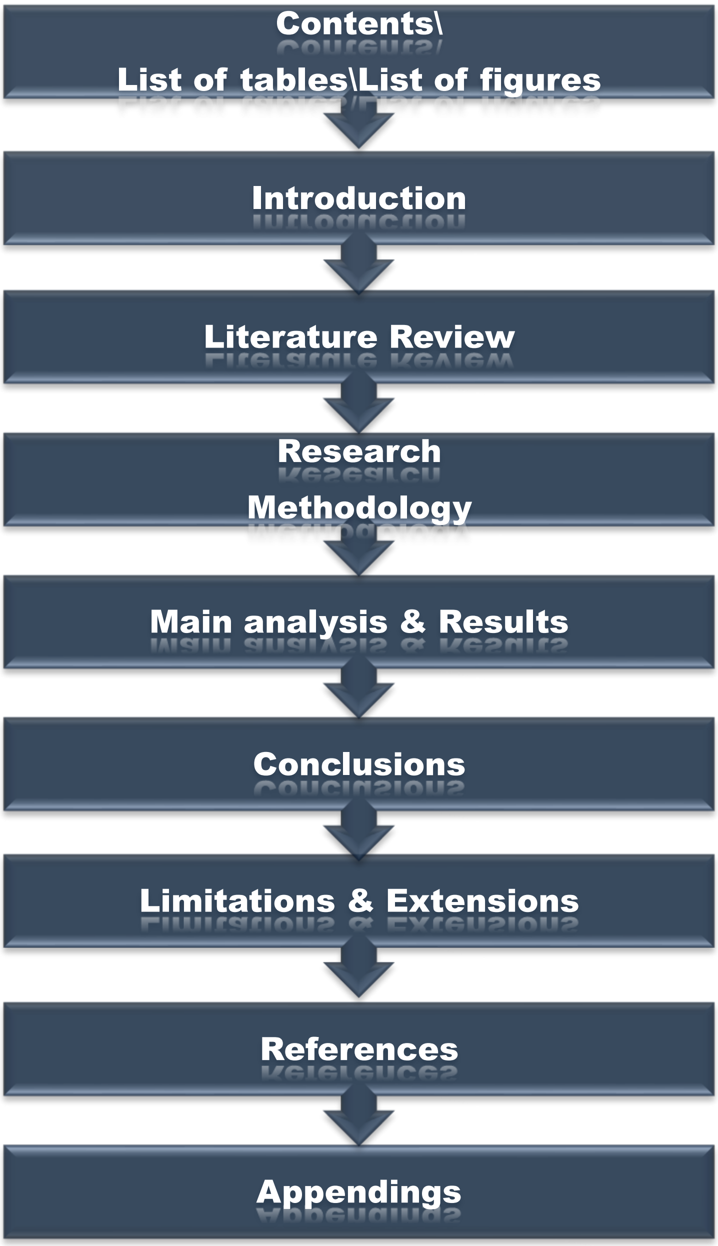


Figure 2: Chapters based on technical context

Chapter 2 – Literature Review

2.1 Introduction

This chapter consists of two separate parts. The first part introduces two important fundamental terms· the "customer relationship management (CRM)" in conjunction with the Information Technology (IT) and the "customer lifetime value (CLV)", two closely related terms to the churn prediction and client behavior in general. The second part is an introductory part to Data Mining –a machine learning technique – and includes -explains the connection-contribution of it to the above terms. Then, a quotation of some of the most common and applicable data mining techniques in CRM follows and ultimately this section ends with citing the ones I used in my project.

2.2 Customer Relationship Management (CRM)

CRM standing for Customer Relationship Management is a business strategy – philosophy which allows the banks and companies in general to understand better their clients and their behavior. CRM aims not only to the maximization of the winnings and takings of banks, but also to the satisfaction of their customers and consequently to a better performance of the financial institutions. (Αραμπατζής, 2008) Due to the fact that there is large competition among the banks it is vital to them knowing much more information about the "existing" clients so as to build stronger relationships with them and keep them happy. In addition, the potential of e-commerce and digital economy created huge range of options. As a result, clients are better informed than ever, which means that they discriminate more and have more preferences and demands. (Foss, Stone, 2002) These demands are the reason that many clients turn to another suppliers. This situation generated the notion of churn (Lejeune, 2001) and the need of creation of a customer oriented management from banks.

The possibility of clients' grouping into homogenous categories from banks, gives significant advantages to the business as they can develop more easily or improve a customer-oriented centric system and have close their 'best' loyal clients which offer profits. And here comes the term “Information Technology (IT)”. Information technology refers to any tool based on computer technology which gives the opportunity of gathering information and information –processing. (Aminuddin, 2011) CRM with the assistance of IT achieves acquisition, storage and analysis of knowledge about customers, aiming to the sale of goods, products or services with the most efficient manner.

CRM is separated to 2 basic types: Operational CRM and Analytical CRM (figure 3). These two parts are not independent, but instead we would say that one is a continuation of the other.

With the operational CRM (“front-office” part) every interaction with the client is automatically recorded in a database - contact history - (of the specific client).With every transaction of the client, the database is automatically enriched. In that manner, the client can come to contact with everyone from the business staff (one or more different people, through any channel) so as to solve his problem without explaining every time its transaction history. (Αραμπατζής, 2008)

After the Operational CRM, the Analytical CRM follows. The Marketing department by a logic use of the data from the previous part, analyses this information with tools of the Analytical CRM and extracts useful conclusions about the clients so as to improve business processes in Sales, Marketing and Service. With the use of Analytical CRM (“Back-office” part) companies can segment their clients, perform marketing campaigns, analyze clients’ profitability, predict future actions and even predict any action for customer defection (churn) and take the necessary steps. (TECHONESTOP, 2019, Αραμπατζής, 2008)

A CRM system helps businesses being constantly connected with their customers and improves their interactions. Thanks to clients’ connections with the web systems, the financial institutions through the IT have the ability to receive feedback without delay, receive data concerning clients’ preferences, interests and needs. (Lejeune, 2001) And the most important: companies can address issues before they become needs! (Foss, Stone, 2002)

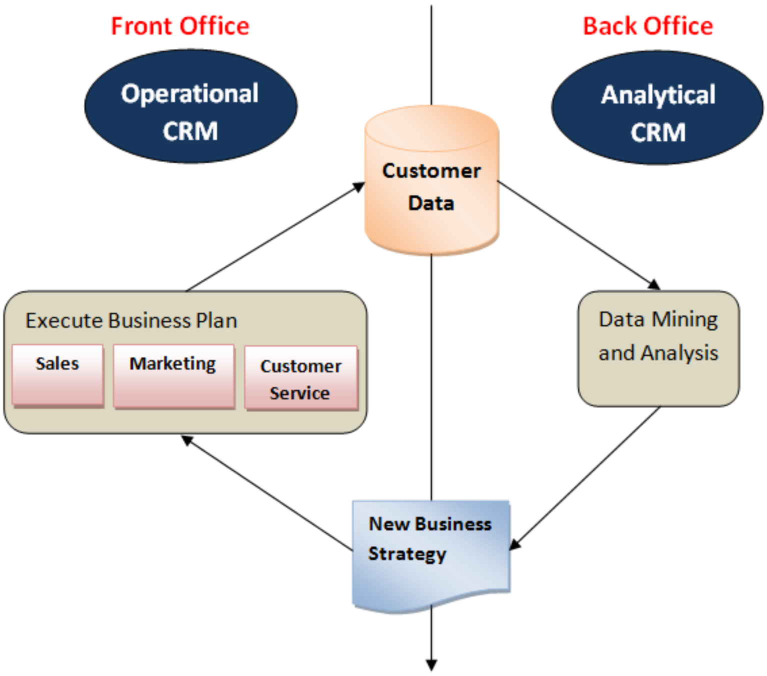


Figure 3: Customer Relationship Management (TECHONESTOP, 2019)

The powerful combination of CRM and IT is coming to enhance the notion "Customer Lifetime Value (CLV)". The CLV is one of the most important KPIs in the world of business and inextricably bound to the term "churn". It is a metric which indicates how valuable a customer is for a company in long term time based on his first purchase. (Grow Digital Team, 2018) Being acquainted with the CLV the banks and companies in general can develop methods –strategies so as to attract new customers but mainly improve their capability to "keep"-retain the existed customers. As the old verse goes, "Make new friends, but keep the old. One is silver, the other gold." As we said before in the chapter 1, it costs less for companies to develop a strong long-term connection with existed clients than acquiring new ones. There is no such reason to spend so much time to find a new client if you cannot keep the current ones happy and boost their loyalty · and the customer loyalty helps increase the CLV. Loyal customers are more likely to become brand advocates being positive in accepting new products and new services. (Rouse, 2019)

So, keeping the existed customers is valuable and engaging the right ones even more valuable and one of the key metrics in figuring out whether the company is retaining clients is the churn rate. (Gallo, 2014)

2.3 Data mining in the churn management

2.3.1 Big Data and Data Mining

According to Gartner analyst Doug Laney in his 3Vs concept in a 2001 MetaGroup research publication, 3D data management: “Controlling data volume, variety and velocity”, big data is data that are characterized by three (3) dimensions, known as “3Vs”: Volume, Velocity and Variety. (Laney, 2001) The first dimension (volume) represents the enormous breadth of data. It should be mentioned that the ninety per cent (90%) of all data was created in the past two years, IBM estimates. (Rijmenam, 2012) The second dimension (velocity) represents the increased speed at which the data is created, received, stored and analyzed. Lastly, the third one the Variety represents the wide different types of data (structured data, unstructured data, etc.). However, the definition of big data can be updated by four additional elements; the Veracity, the Variability, the Visualization and the Value. With the term Veracity we mean how truth and correct the data is. Due to the fact that too much information is automatically recorded without the existence of some control, we have to verify the validity. The term variability is as Brian Hopkins, a Forrester principal analyst mentions “the variance in meaning, in lexicon”. In other words, the same word among thousand ones that exist in big data could have a lot of meanings. Of course, we could not forget the part of visualization of the data (hence the characteristic ‘Visualization’). Nowadays it is unthinkable to talk about big data, that huge amount of data without making them easy understandable to everyone. Finally, the last term Value could be interpreted as the benefit (monetary and not) which big data offers to the companies and organizations. (Rijmenam, 2012)

Below in figure 4 is a schematic representation of the characteristics of big data:

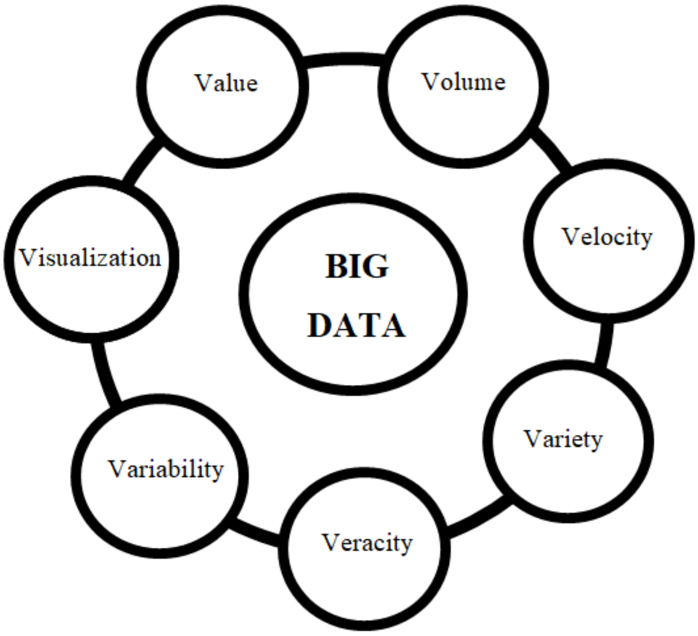


Figure 4: Big data parts

Because of the fact that there is a constant increase of so many big data sets in the scale of terabyte or even of petabyte, it is vital to find a way to analyze them. It is too complex and difficult to handle them in traditional methods. However, the Data Warehouse tool solves the above problem and makes it possible to collect, store, process and analyze the data through Data Mining techniques. Data mining methods such as Association (correlation between variables), Clustering (group items together in clusters based on similar characteristics-patterns), Classification (assign items into predefined categories), Decision Trees (represent decisions and decision making using a model of decisions like a tree.) are utilized so as to detect hidden information and assemble predictive models. (Datafloq, 2019)And after implementing statistic techniques and finding out probable patterns, all this data are represented graphically such as with scatterplots, histograms, barplots, tables etc. Data mining methods transform the raw data into useful and usable knowledge. (Lejeune, 2001)

As Michael J.A.Berry and Gordon S. Linoff mentioned in ‘Data Mining Techniques for Marketing, Sales and Customer Relationship Management’, second edition , ”Data Mining, as we use the term, is the exploration and analysis of large quantities of data in order to discover meaningful patterns and rules”. (Berry, Linoff, 2004) The figure 5 clarifies the stages involved in the constant search for more processed information.

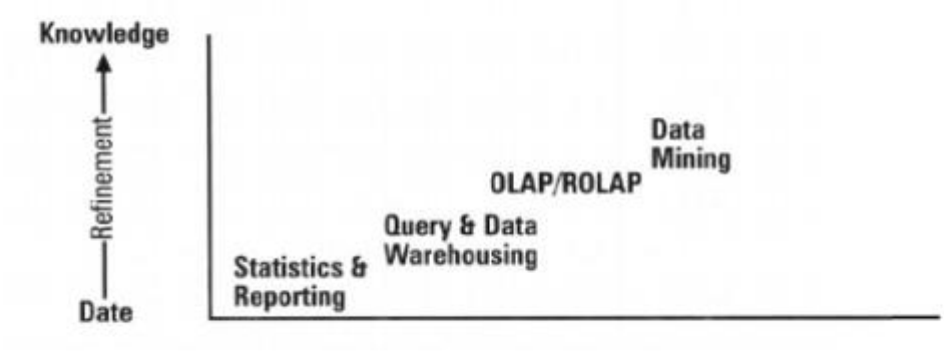


Figure 5: Evolution in the quest for information (Lejeune, 2001)

Talking about the e-commerce (electronic transactions) and Data Mining there is the English term "Web Data Mining". Web Data Mining is a technique developed because of the large amount of data on the Web environment so as useful information to be extracted from perplexed Web documents and Web sites. (Hongjiu, 2013)

The data mining can be used in the section of churn management. More specifically, it can be used to perform analysis about identifying valuable customers, preventing future actions and predicting even which customers are likely to switch bank (churn).

2.3.2 Data Mining and RFM analysis

Data warehouse tool supplies a customer database to us enabling us to deal with the data based on not only static but also dynamic features. RFM (recency, frequency and monetary) values have been utilized for many years to segment clients and identify the above categories of clients. (Lejeune, 2001) RFM analysis is a data mining model, maybe the most frequently adopted technique for segmentation that differentiates the clients by three variables·

1) Recency of purchases, which is the time period between the last transaction and present (as present we could assume any predefined date).
2) Frequency of purchases, which refers to the different number of transactions in a specific period and
3) Monetary which refers to the monetary value of the transactions in the above specific period.

As one could easily understand ,the smaller recency (it is more likely for client to repeat a purchase) ,the bigger frequency (it is more likely for client to purchase repeatedly) and the bigger monetary (it is more likely for client to contribute to profits) , the better clients are and more suitable for being brand advocates. It is being referred that the recency is the most important attribute among the three ones. (Mohammadian, Makhani, 2016) After defining the number of classes and assigning numeric labels from 1 to n where n is the number of classes to every client based on the significance (usually 1 is the best value whereas n is the worst) , then all of clients are presented by an arithmetic score which may be the concatenation of the individual RFM score numbers or the sum of them. For example, if we want to divide a customer base to four classes and we assume that one (1) is the highest value whereas the 4 is the worst, then a customer who has the score 111 is translated as a client who has purchased recently, is a frequent client and spends a lot of money. (PUTLER, 2019)

2.3.3 Data Mining and Clustering

Clustering or Cluster analysis is a Data mining method which divides a set of instances into groups without having a prior knowledge about the class labels(unsupervised learning method). With the use of clustering, analysts try to discover meaningful groups (clusters) which characterize the data in the best possible way. Optimal division is identified this one in which the data points within the same group are more similar to one another (maximization of intraclass similarity) and dissimilar to the data points in other groups (minimization of interclass similarity). (Han, Kamber, Pei, 2012) In other words the cluster analysis relies on the similarity and dissimilarity between unlabeled data points. (Priy, 2019) Because of the fact that there are no class labels, there are no criteria for characterizing a clustering as “good” and it depends on the analyst. The latter is the one who will decide if the clustering satisfies his needs. The figure 6 illustrates the role of clustering.

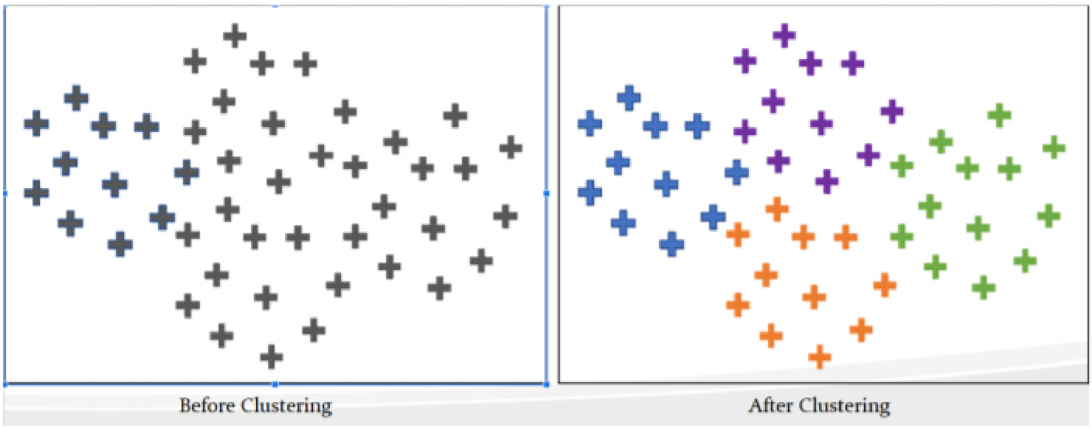


Figure 6: Before and after Clustering (Yoseph, Malim, AlMalaily, 2019)

There are enough clustering methods. To begin with, Density-Based Methods, as the name indicates itself, this method relies on the density of the space. It detects regions where data points are concentrated and separated by regions that are sparse or contain nothing. Some algorithms included in this category are DBSCAN, HDBSCAN, OPTICS etc. To continue, Hierarchical Based Methods, as the name indicates, is a clustering method which relies on the hierarchy. It creates a structure which has the type of tree and can be visualized using dendrograms. It assumes that each data point is a separate cluster, and repeatedly diagnoses the two clusters which have the closest distance between them and merge them. This procedure stops when all clusters have been merged together. (Bock, 2019) Some algorithms included in this category are CURE, BIRCH, DIANA etc. Another clustering method and the most common is Partioning Methods. This data mining clustering method forms k partitions of the data objects and each partition is assumed one cluster. Then, the former evaluates them, using some criterion for example minimizing the sum of square errors. (SlideShare, 2015) Some algorithms included in this category are K-Means, K-medoids, Clara etc. (SlideShare, 2015) Finally, there are Grid-based Methods. In this method the object space is quantized into a finite number of cells forming a grid structure. It is not very usual method but it is enough fast. Some algorithms included in this category are STING, WAVE CLUSTER, CLIQUE etc. (Priy, 2019)

2.3.4 Data Mining and Classification

Classification is another data mining task which allows customer segmentation. Here, the data instances are placed to predefined groups. This method handles data points with labeled response (whose categories are known) and tries to categorize unseen data points to existed set of groups. The classification method is a two-step procedure comprised the training phase, where the algorithm learns the input data and their behavior so as to be able to predict the class labels for the test set (unseen data) and ultimately its validity to be evaluated (classification phase). In the training phase, the data set is divided (split) into two parts (train, test) or three parts (train, development, test). The performance of chosen classifier is evaluated on the test set where all the labels are hidden. If the classifier assigned the test data instances correctly to a greater extent, then we could assume that this specific classifier is suitable for this case else we should choose another one. (GeeksforGeeks, 2019) The figure 7 illustrates the binary classification (binary means that there are only two (2) labels-classes).

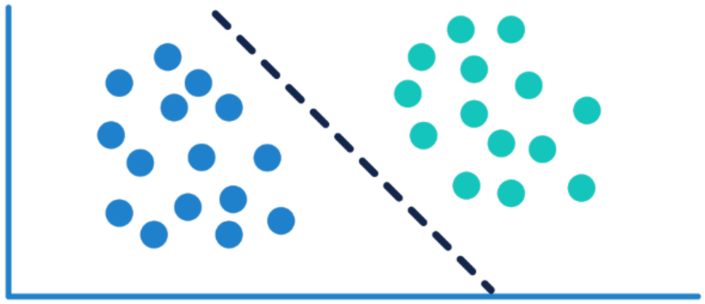


Figure 7: Binary classification (Lucidworks, 2019)

There are two considerable types of classifiers: The Discriminative classifiers and the Generative ones. The Discriminative classifiers try to assign each data point to just one class depending on the observed data towards. As a result, the assumptions for this type of classifiers are few. In the discriminative classifiers the quality of the data plays a key role. Some discriminative classifiers are the Logistic Regression , the

KNN (K-nearest Neighbour), Support Vector machines, Balanced Random Forest etc. (Joshi, 2018) On the contrast, the Generative classifiers focusing on a probabilistic description of the data, attempt to learn the model that generates the data behind the scenes estimating the distribution of the data and their assumptions. (Chioka, 2014) Some generative classifiers are Naïve Bayes, Gaussian Mixture Model, Bayesian networks etc. (Joshi, 2018) It should be marked that in the case we have a lot of data (as it is usual) discriminative classifiers perform better than the generative ones. (Chioka, 2014)

2.3.5 Another approach

There is another approach for dealing with the churn management without being a data mining tool. Indeed, it is a completely model free approach and uses only some simple data processing. It is an approach using the Empirical Cumulative Distribution Function (ECDF) and a threshold. This model studies the “usual” client behavior (for example what it goes nine times out of ten) and based on this situation, if a customer surpasses his atomic threshold ,then this behavior characterized as irregular-anomalous. For this method it should be calculated the distribution of time between the purchases-transactions and more specific the number of days between consecutive transactions. After computing the ECDF, the analysts define an action level such as ninety-five percent (95%) or ninety percent (90%) as a threshold for discriminating the regular from irregular behavior. (Forecast Analytics, 2019) The figure 8 illustrates in short the approach with the use of ECDF and with action level – threshold 95% for a sample of 20 clients.

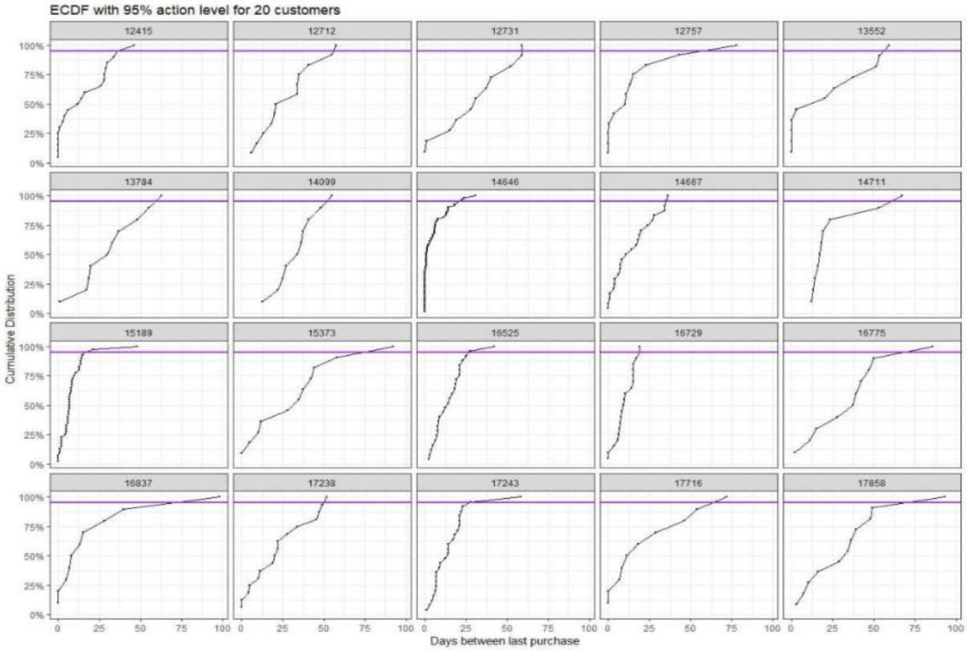


Figure 8: Example of ECDF distribution for a sample of 20 clients and action level 95% (Forecast Analytics, 2019)

Chapter 3 – Strategy methodology

Progress of the research

This chapter describes the approximate development – progress of my project. Guide for the construction of my research was the research questions that arised from the theme of the project, the churn. Base for the project before any action was the deep understanding of the project problem and then a creation of a theoretical plan about how we would work. After the completion of this step the practical process began. The first step was the data collection which involved data process and data selection so as to extract the final appropriate features and then the storage-extraction of them to csv files. After that, it followed the phase of data preprocessing and then the data transformation phase. Following through the previous tasks, the section of data mining took action (RFD analysis and clustering). Continuously, through visualization the interpretation - evaluation phase was achieved for the clustering. Afterwards, the data mining step was used again for the classification phase this time. Finally, as before, the practical part ended with the explanation of the results and the evaluation of them. The figure 9 illustrates graphically the research progress of my project.

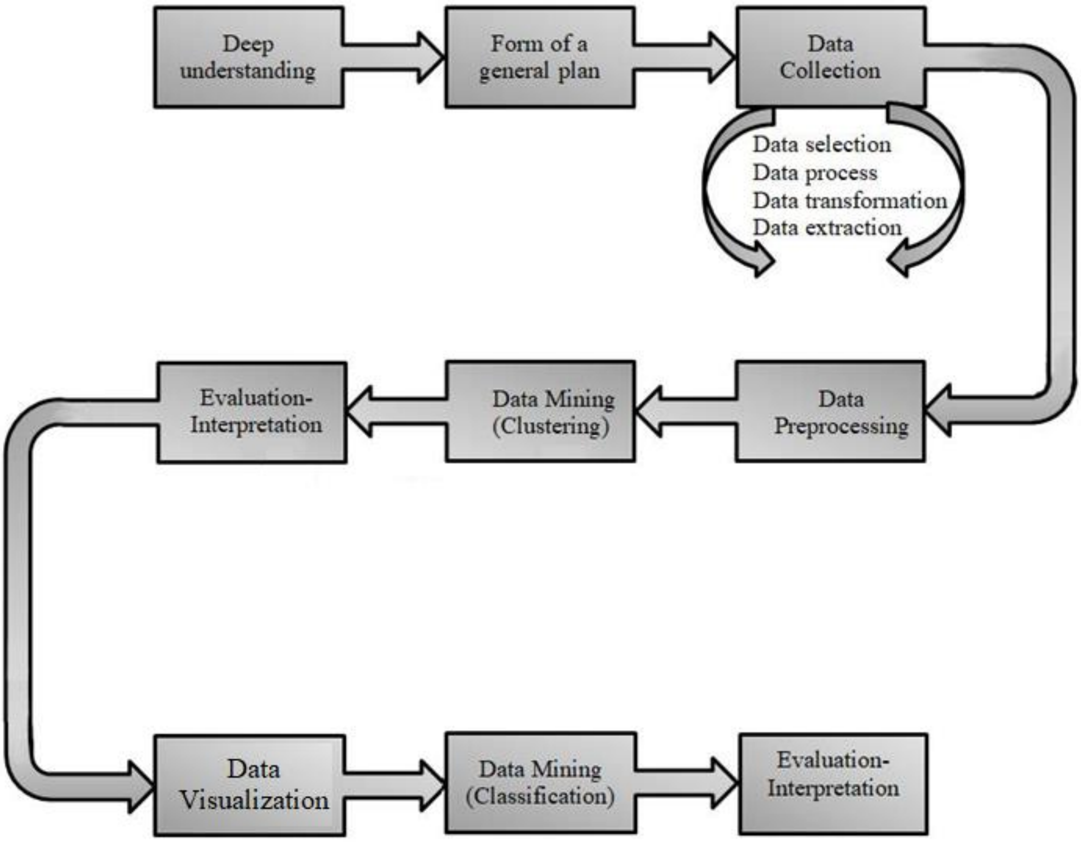


Figure 9: Research progress

3.1 Deep understanding

One of the most important and basic parts of a research is the deep understanding of the problem that should be dealt with and the form of a general theoretical plan. Due to the fact that there was no information for existed churners, we had to think how we could extract supposedly labels (churners/ no churners), in other words with which criterion someone should be targeted as "churner". After that, we would have to apply some classification algorithms so as to predict the required probability, the probability that a customer will cease his relationship with NBG within a month.

3.2 Form of a general plan

Every research requires the creation of a theoretical plan. For the procedure of extracting the labels we divided the time period of two years in two parts, the observation period and the lead period. The observation period was the period that we observed-studied the behavior-habits of clients so as to segment - categorize them to groups-clusters. On the other side, the lead period which succeeds the observation one, was the period in which we checked if there was existed any anomaly in users' behavior based on their historical background. In other words, if there was a different pattern in customers' habits. If the answer was positive then the specific user would be characterized as churner otherwise as no churner. Our first basic problem was how long this lead period-window would be. At first we thought that the lead period would be two months, so we divided the two years period to twenty-two (22) months as observation period and two (2) months as lead period. However the data results were unsatisfactory and showed us that we had to assume a longer space. So we assumed a four months (4) window for lead period and twenty (20) months as observation period. The results were adequate. As far as the second main problem is concerned, the criterion for the labels, our first thought was to check for each user if he/she made at least one transaction in the lead period. However, this approach was abandoned due to the fact that there are different types of customers. There are the good customers who have frequent contact with the bank and these who have rare contact. For example for a user who gets to the NBG system every month, never getting into the system in the lead period means that he "should" be characterized as churner whereas for a user who visits the NBG system once every five or six months, never getting into the system in the lead period means that he "should not" be characterized as churner. As a result, the scenario of getting to the internet banking at least once was not sufficient. On this point we relied on so as to implement clustering, that is to say, so as to segment the clients. For this reason we thought to use the variable – feature "max_dist" as the motivating power for targeting the clients. Having separated the clients, based on their habits, on the observation period, to groups-clusters, each cluster would have its own representative number as far the 'max_dist' is concerned. (To be reminded, the feature "max_dist" symbolizes the max duration between two consecutive transactions of each user). So, having extracted this characteristic number, the next move was to double this number and compare it with the max_dist of each user in the lead period. If the max_dist of a user in the lead period was bigger than double of the cluster's max_dist in which the specific user belongs to, then this client would be targeted as "churner" otherwise as "no churner". In this way we actually observe if there was a different pattern in users' behavior. With a few words each cluster would have its own individual groups as would be right.

3.3 Data Collection (Selection, Process, Extraction)

The data that utilized in my research-project, was web transactions of clients of National Bank of Greece and more specific web transactions with the NBG internet banking for a period of two (2) years from 01/08/2017 to 31/07/2019. With the term "web transactions" we mean any interaction of the client using any electronic device (personal computer, mobile etc.) through any channel (NBG internet banking, I-bank Pay app, I-bank pass, etc.). Some web transactions of a user might be the checking account, monetary transactions, updating information about the account, secure transactions, deposits, card payments even a simple login. The total number of clients was 4.645.949.

The whole project's data engineering workflow was completed on an NBG's Big Data cluster operating on Hadoop. Apart from the underlying Hadoop Distributed File System (HDFS) over which the cluster operates, the broader Hadoop Ecosystem provides a suite of various services needed to solve big data problems. Since Cloudera is NBG's platform distribution of choice for Hadoop, our working environment was accessed using HUE, a web-based interactive editor provided by Cloudera for browsing, querying and visualizing data. Through HUE, the underlying data was queried using Impala, a distributed SQL query engine for Hadoop.

Among various data types there was a variable called "error_text_data" which contained "NULL" or "Response is null!" when there was successful transaction and text of type json otherwise. From the countless clients we excluded the ones who had unsuccessful transactions and those who had empty name. The volume of the data was such that it was vital to use the command 'group by' in every query so that they can be loaded into memory. Utilizing the variables of the given tables and based on the

RFD analysis (a modified version of RFM analysis, but instead of the characteristic monetary in place it has the characteristic duration (Wikipedia, 2019), we built the needed and targeted features. Ultimately, we extracted the required data to csv files with the following variables: (the interpretation of them will be set out in the chapter 4 in the main analysis and results):

- client_username
- countt
- recency
- frequency
- max_dist
- average_trans_dur
- diff_maxd_mind_days
- start_trans_end_data

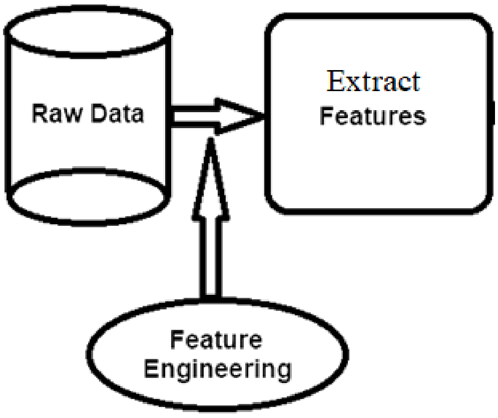


Figure 10: Feature Engineering

3.4 Data Pre-Processing

Due to the fact that the working data-features was produced and processed by the user itself (us) we did not face any missing values. However in both phases (clustering and classification) we had to deal with noisy data. Generally, there are various types of noise while working in data science which could garble models' performance, some of which are noise as an item, as a feature, as a record or noise in unsupervised methods. (Rathi, 2018) In our case, firstly, we had to 'fight' with the case of noise as an item and more specific, with the outliers. An outlier is an instance-data point which lies a peculiar distance from the majority of the data points in a population. (Nist Sematech, 2018) The figure 11 gives a schematic representation of outliers.

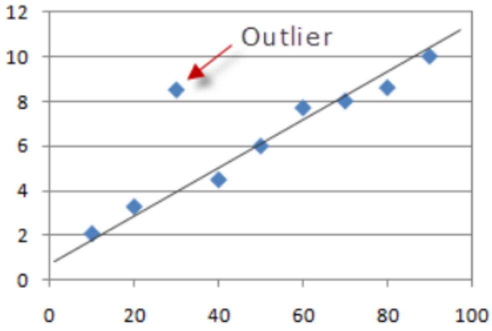


Figure 11: Outliers (Rathi, 2018)

The detection of outliers accomplished with the use of a visualization tool, the box plots. A box plot is an effective graphical way for describing the behavior of the data which uses the IQR method (interquartile range). (Nist Sematech, 2018, Sharma, 2018) A box plot, as its name says, draws a box between the upper (75th percentiles) (Q3) and lower (25th percentile) (Q1) quartiles and a vertical line across the box representing the median of the data. According to the Wikipedia definition, the IQR method also called as H-spread, is a measure of variability as measuring the statistical dispersion and is equal to the difference between the above percentiles (IQR =Q3-Q1). (Wikipedia, 2019) The data points that we are interested in are bounded between an area from Q1 - 1.5 * IQR to Q3 + 1.5 * IQR and every single point which lies outside this range is concerned as "outlier". The IQR defines somewhat a threshold in order to recognize the outliers. (Sharma, 2018) The figure 12 illustrates the detection of outliers using box plots.

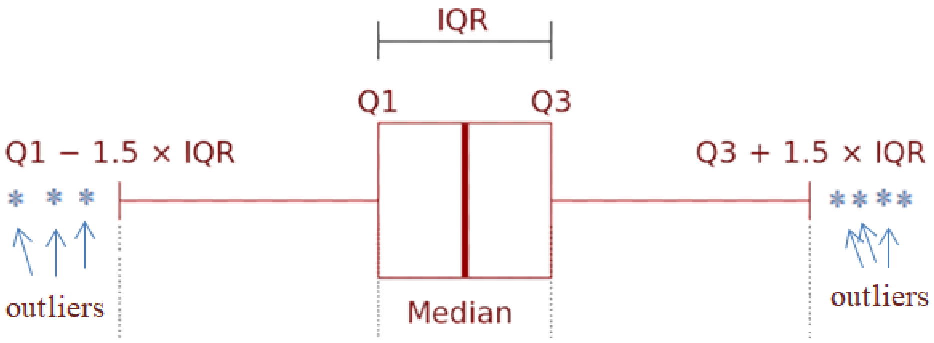


Figure 12: Detection of outliers using boxplots and IQR

The removal of outliers was held calculating the IQR score for each feature and then after this filtering, we kept the well founded values. (Sharma, 2018)

The next noise that we faced (in classification) was the noise as a feature. This type of noise occurs when one or more features are very highly or highly correlated with the target variable in the dataset and do not offer some additional information for our machine learning models. (Rathi, 2018) The method that used for the "feature selection" was the stepwise and in particular with the direction of both backward and forward for a more optimal result. The stepwise method or stepwise regression is a method which iteratively adds or\and removes the features-variables (called predictors) aiming to find a subset with the best predictors, i.e. the variables that contribute to the best performance of the model. The three types of the stepwise regression are the following:

- Forward selection
- Backward selection or backward elimination
- Stepwise selection or both forward and backward selection

The forward selection starts with an empty subset (no predictors) and gradually adds the most important features. This is an iterative procedure which stops until there is no longer statistically significant advance. On the other side, the backward selection is an iterative procedure too but it starts with the full model and gradually removes the least important features until it creates a subset where all predictors are statistically significant. Finally, the stepwise selection moves in both directions (forward and backward). It starts with the forward selection, but in every step where a new predictor is added in the model, the backward procedure is activated and removes any feature that no longer contributes to the model's improvement. (STHDA, 2018) The figure 13 illustrates the procedure of feature selection.

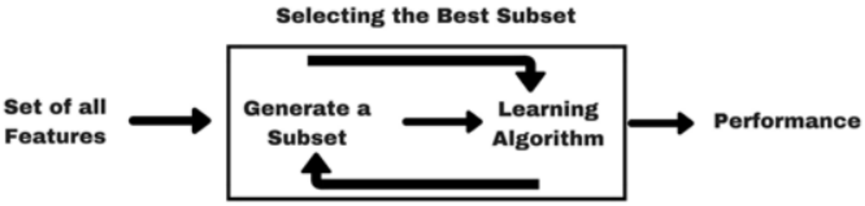


Figure 13: Feature selection (Rathi, 2018)

3.5 Data Mining (Clustering)

The data mining phase participated two times in our research project; once in clustering time and once in classification time. In this phase, in clustering, the customers were clustered based on their behavioral related to RFM analysis features. The algorithm that utilized for the clustering of client data base was the Expectation-Maximization (E-M) and more specific the K-means (Partioning method of unsupervised learning methods). (Github, 2018) K-means assumes to be the most common application of E-M algorithm for separating and characterizing data without knowing the classes in prior because it offers a lot of advantages. First of all, it offers a meaningful intuition of the data' structure. It is simple and easy to be implemented as it requires only one parameter (the number of clusters k) and it is very fast and can be used for large data sets (something that many other algorithms do not have). K means receiving a single parameter k which can be tuned, if the user gives a range of values for it, designates the number of clusters in which the user wants to group the data. The algorithm starts initializing k random points, called the "centroids" in the data space, with dimensions as many as the features' dimensions are. (DATA STUFF, 2019) The centroids or else "cluster centers", is the arithmetic mean of all the data observations belonging to the cluster. (Github, 2018) Afterwards, the K means assigns each data point to the cluster that is closest to it ("Expectation step" or "E-step"), reforms the centroids based on the point that lies on the average of those that have been assigned to each cluster("Maximization step" or "M-step") and repeats the above steps until it converges. (DATA STUFF, 2019) The final aim is each data point being closer to its own cluster center than to this of the other clusters. An advantage of K Means is that at least in the simple case the algorithm categorizes the data points to clusters in such a way that the human eye would separate them too. (Github, 2018) The figure 14 illustrates the K-means clustering.

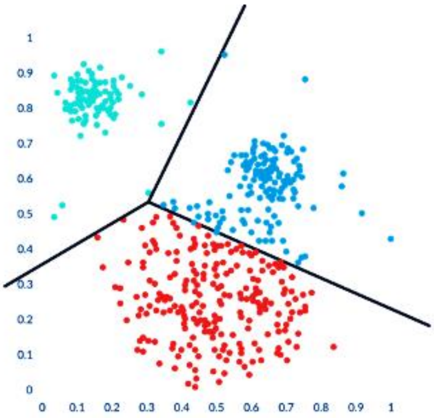


Figure 14: K-means clustering (Lucidwords, 2019)

K means as other clustering algorithms too, uses measurements based on distance so as to define the similarity between the data observations that takes as input. It uses Euclidean distance from data points to a centroid and more general the pairwise Euclidean distance between the data observations as the sum of squared deviations from the cluster centers is equivalent with the sum of pairwise squared Euclidean distances divided by the number of data points. (StackExchange, 2019) Finally, because of the above fact, it is strongly recommended to scale the data and more specific to standardize them so as the values lie in the same range of values. With the term standardization we mean that we want the values have mean equal to zero (mean = 0) and variance equal to one (variance = 1). Generally, the formula for standardizing data which have mean value as mean and standard deviation as std_dev is [(X - mean) / std_dev]. The goal of standardization is to make the features comparable. Finally, before the scaled data being added to the algorithm, they need to be transformed to a set of vectors as we did. (Dabbura, 2018)

Nevertheless, K means has some drawbacks. First of all, the choice of k or the choice of the range of k can affect the result. In addition, it assumes that the assigned data points in each cluster are located within a sphere around the cluster centroid. As a result, in the case of elongated groups, the K-means will fail. Finally, K –means is affected by outliers and noisy data. (Yse, 2019)

3.6 Evaluation – Interpretation

In supervised learning, since we have predefined classes (“target” variable) we have the ability to assure or not if the selected model performs well or not on the specific data. Antithetically, in unsupervised learning (clustering) we have not stable and strong evaluation metric so as to evaluate the output results. However there are two well-known ways- metrics that could give us some intuition about the optimal number of clusters. The first method is the Elbow method and the second is the Silhouette analysis. The Elbow method uses as criterion the sum of squared distance (SSE) between data points and their clusters’ centers which they are assigned into. (Dabbura, 2018) The detailed procedure is the following according to the Wikipedia: The algorithm calculates the distance from each observation to its centroid, the final centroid in each cluster, takes the square of that error and sums it up for the entire dataset. (Wikipedia, 2019) The “optimal” number of clusters (optimal k value) is the spot at which an increase of k will lead to a small decrease in the SSE while a reduction of k will lead to an abrupt change (increase) of the SSE. Graphically, we could say that the optimal number of clusters is that spot where an elbow is created, known as “elbow point”. (Yse, 2019) From this feature the method itself got its name. The figure 15 illustrates an example where the optimal number of clusters using the Elbow method is k = 3 .

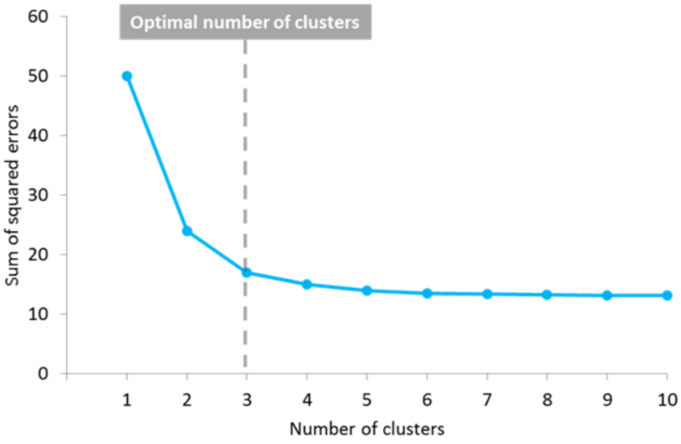


Figure 15: Elbow method (Yse, 2019)

The second method for evaluation is the Silhouette Analysis. Silhouette analysis could be used to examine degree of separation distance between the resulting clusters. In other words, it studies how close each data instance in one cluster is to data instances in the neighboring clusters. (Scikit-Learn, 2019) The Silhouette analysis computes the following coefficient: $\frac{a^i - b^i}{\max(a^i, b^i)}$ where a^i is the average distance of each data point i to all data points in the closest neighboring cluster and b^i is the average distance of the specific data point to all data points in the same cluster. The required coefficient can take values from -1 to +1. If the coefficient is equal to zero (0) ($a^i = b^i$) that means that the data instance is very close to the nearby cluster. If the coefficient is equal to negative one (-1), that means that the data instance is assigned to a wrong cluster. Finally, if the coefficient is equal to positive one (+1) this means that the data instance is assigned to the right cluster and it is quite far from the neighboring clusters. As a result, we assume as "optimal" number of clusters the spot where the Silhouette score is the higher and closer to the unit (+1). (Dabbura, 2018) The figure 16 illustrates an example where the optimal number of clusters using the Silhouette analysis is k = 4.

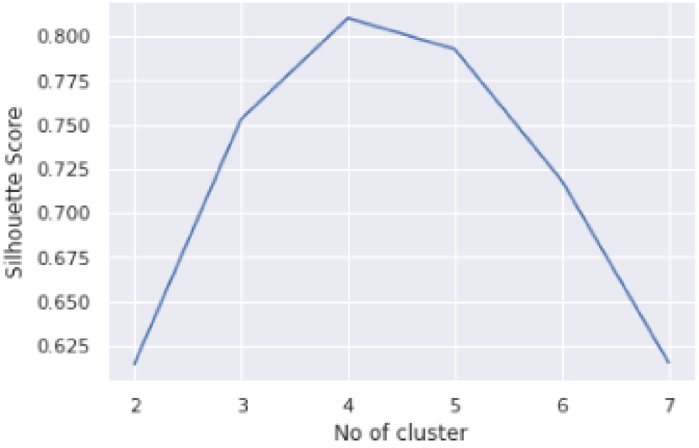


Figure 16: Silhouette analysis (Vora, 2019)

3.7 Visualization

After clustering the visualization section followed. One of the biggest problems nowadays due to the existence of big data is the ability to visualize the high number of dimensions of data. This ability is very important because an image offers to the people, as visual creatures, a better understanding of the problem. A graphically representation reduces the time within a human brain needs to receive the message. Therefore, it is vital for the analysts to be able to explore visually the data. For this purpose, there are some methods known as dimensionality reduction such as PCA and t-SNE. In our project the technique that was utilized was PCA (Principal Component Analysis). PCA tries to retain the minimum number of features (components) with the most information (maximum variance) using the correlation between some dimensions. (Derksen, 2016) The PCA is affected by scale, as K-means does the same. For example, in the case of not scaling the features, let us think that we have two components, e.g. the human weight and the human height. The feature weight has bigger variance in values than the feature height, because of their natural range of values. So, PCA may decide that the direction of max variance is closer to the axis concerning the variable weight so, a change in weight would be thought more significant than a change as far as the height is concerned which is clearly wrong. (Scikit-Learn, 2019) As a result, it is strongly recommended for users to scale-standardize the features in the data before applying PCA. This requirement is significant because it offers an optimal performance in many machines learning models. (Galarnyk, 2017)

3.8 Data Mining (Classification)

The second time that data mining is appeared is in the classification phase. After extracting the labels of churner and no churner to the clients we implemented classification algorithms. The first “work” was to split the data to train, development and test set. The algorithms that were used were both discriminative and generative and in particular, the logistic regression was used, the Random forests (but unfortunately with no success due to the large number of data) and finally the Naïve Bayes.

Logistic regression in general, is used to predict which class a data instance belongs and is used when the “target” variable, dependent variable is categorical. (Swaminathan, 2018) Logistic regression belongs to a family, named Generalized Linear Model (GLM), an extension of the linear regression models. Other equivalent synonyms are binary logistic regression, binomial logistic regression or logit model. The logistic regression predicts the probability a data point observation belongs to one of the two classes, so does not give directly the class of data instances. The user has to decide the threshold probability which separates the two classes. The default value is p = 0.5. However, in reality the user should consider the particularities of the project. (STHDA, 2018) There are the following types of logistic regression:

- Simple logistic regression, where there is the dependent variable and one independent variable and
- Multiple logistic regression, where there is the dependent variable and at least two independent variables (STHDA, 2018)

The figure 17 illustrates the Logistic Regression.

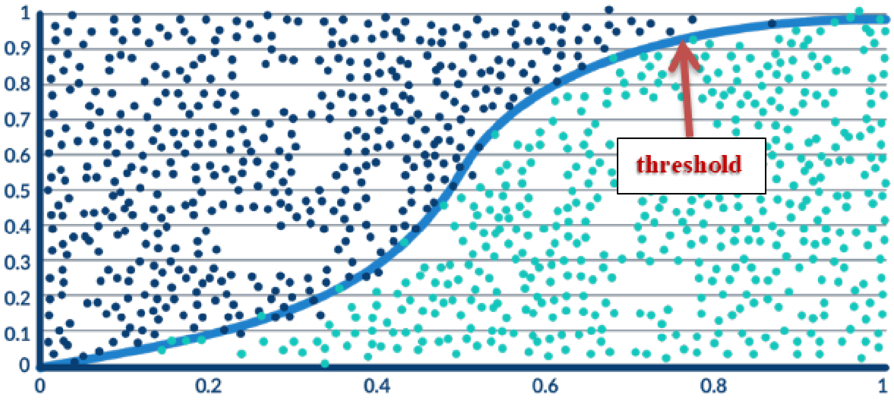


Figure 17: Logistic Regression (Lucidwords, 2019)

The Random forest classifier as its name states, consists of a forest, a large number of individual decision trees that work as one entity-ensemble. This classifier relies on the power of the majority. More specific, each of the generated independent members-tree in the random forest takes a different subset of the data features and gives a class prediction. (Global Software Support, 2019) Then, the class that has been voted by the majority of the trees becomes the requested model’s prediction. A strong advantage of this method is that members' individual errors are fallen through the net because although some trees might give wrong predictions, there is a group of other trees that will be right. As a result the predicted result will move to the right-correct direction. (Yiu, 2019) A drawback of this algorithm is that requires enough memory so as to be implemented. Unfortunately it did not work with our data on account of limited computer memory. The figure 18 illustrates an example of the Random Forest operation where the final predicted class is the class C.

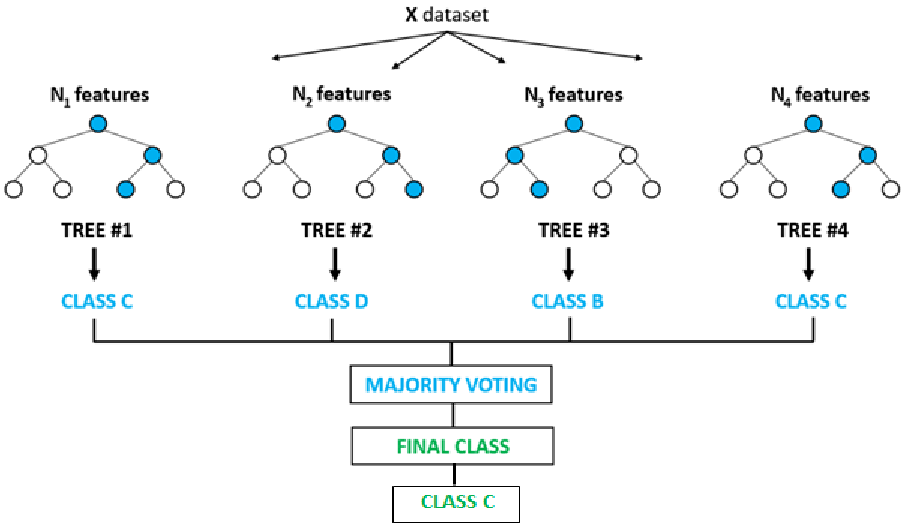
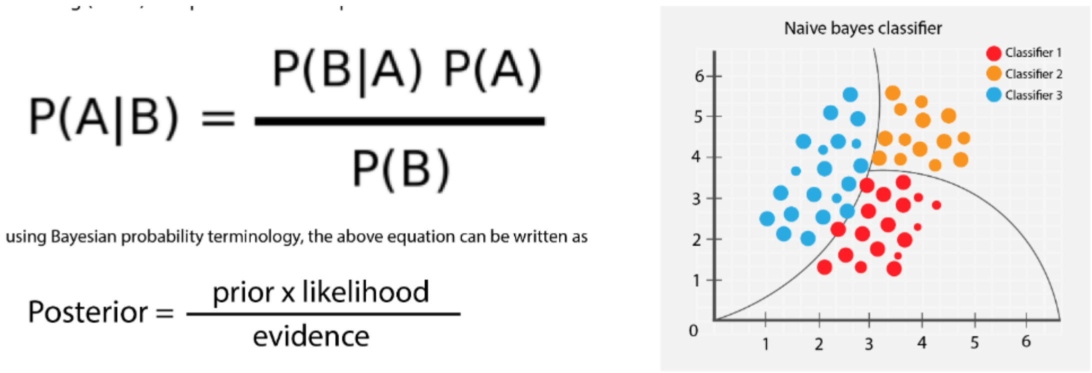


Figure 18: Random Forest (Global Software Support, 2019)

The last classifier that was used was Naïve Bayes classifier. Naïve Bayes as its name states is based on the Bayes theorem, so it is a probabilistic supervised machine learning model. The main assumption that Naïve Bayes classifier uses is that the predictors are independent, something that constitutes the main disadvantage of this classifier. (Gandhi, 2018) The Bayes theorem with mathematical terms is the following: $P(y \mid x_1,..,x_n) = \frac{P(x_1 \mid y) \cdot ... \cdot P(x_n \mid y) \cdot P(y)}{P(x_1) \cdot ... P(x_n)}$ where $x_1 ,..., x_n$ are the features-predictors and y is the target variable. In our project the variable y has only two outcomes, 0 or 1 where the number zero (0) represents the non-churners while the number one (1) matches with the churners. The aim of Naïve Bayes is to find the class y so that the probability $P(y) \cdot \Pi_{i=1}^{n} \cdot P(x_i \mid y)$ is maximized. . Two of the most important advantages of the Naïve Bayes are that it is easy and fast to be implemented. However, the problem of dependence of the predictors which exists in real life makes this classifier weak. (Gandhi, 2018)

Having the train set, the development (dev) set and the test set at our disposal we fit in each classifier our models. We trained them using the training data, (train set) and we studied their performance on the dev set. The aim of the whole procedure was not only to predict the required probability, but also to see-examine between the two utilized classifiers which of them performs better and ultimately to "investigate-assess" the theory that discriminative classifiers have better performance than the generative ones on large datasets. After resulting in the "best" classifier, we applied it on the test set. The following figure 19 illustrates the Naïve Bayes classifier:

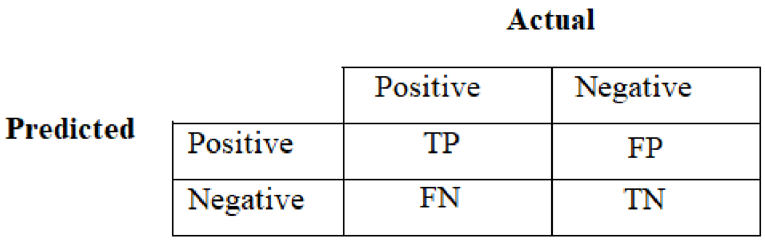


P(A|B) = P(B|A) P(A) / P(B)
using Bayesian probability terminology, the above equation can be written as
Posterior = prior x likelihood / evidence
Naive bayes classifier
Classifier 1
Classifier 2
Classifier 3
0
1
2
3
4
5
6
1
2
3
4
5
6

Figure 19: Naive Bayes classifier (ThatWare, 2019)

3.9 Evaluation – Interpretation

The predicted classes that a classifier gives after the classification operation usually called as estimated labels. A usual measure for evaluating the performance of a classifier, i.e. how well it achieved to identify the correct classes, is the accuracy. However, this metric can be very misleading in the case for example of imbalanced data. (Boyle, 2019) Some more representative metrics are the precision, recall, f1 score, the confusion matrix, the adjusted rand index (ARI), etc. In our research the tools that utilized were the f1 score, the confusion matrix, the ARI and the pseudo R squared. To be more understandable these metrics let us see the below table (figure 20) which called confusion matrix.



		Actual	
		Positive	Negative
Predicted	Positive	TP	FP
	Negative	FN	TN

Figure 20: Confusion matrix

where: TP = True positive,

TN = True Negative,

FP = False Positive,

FN = False Negative

Having the above confusion matrix at our disposal, we have the followings:

- Accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$ (Wikipedia, 2019)
- Precision = $\frac{TP}{TP+FP}$
- Recall = $\frac{TP}{TP+FN}$
- F1 score = $2 \times \frac{Precision \times Recall}{Precision + Recall}$

F1 score is the harmonic mean of the precision and recall and it is a better metric for evaluation than studying the precision and recall separately as it seeks a balance between them. A strong evidence for using F1 score is the existence of an uneven class distribution in the data, something that we had in our research project. (Shung, 2018)

Adjusted rand index (ARI) is a metric that takes values from zero (0) to one (1). The value 0 matches with the case of a random partition whereas the value 1 matches with the case of perfect agreement between the predicted values and the true ones. As a result, the closer the unit to this value, the better the algorithm performs. (MixGHD documentation, 2019)

Finally the McFadden's pseudo R squared (which is applied in logistic regression models) is a statistical tool that studies the goodness of fit for these models. McFadden's R squared is defined as $1-\frac{\log(L_c)}{\log(L_{null})}$ where log(Lc) is the log likelihood value for the fitted model and log(Lnull) is the log likelihood for the null model which includes only an intercept as predictor (so that every individual is predicted the same probability of 'success'). In particular, in case of glm the above type is equivalent with the type 1 - (Residual Deviance/ Null Deviance). (Bartlett, 15) The range of values of the McFadden's pseudo R squared is (0, 1). Values ranging from 0.2 to 0.4 indicate very good model fit. In case that McFadden's R squared is close to 1, indicates very good predictive ability whereas values close to 0 indicate no predictive ability.

Chapter 4 – Main analysis and Results

4.1 Introduction

Before analyzing the whole procedure of our research should be mentioned some general things.

The whole research implemented on a personal computer with the following characteristics:

Processor: Intel ® Core ™ i7-7500U CPU @ 2.70GHz 2.90 GHz
RAM: 8,00 GB (7,87 GB is the possible use)
System Type: Operating system 64 bit, processor of technology x64

The operating systems that were used were Ubuntu and Windows. At Ubuntu we used Jupyter notebook, a software in which we created-started a spark session and connected Python 3 with it (Pyspark). As far as the Windows software is concerned, we used Anaconda Jupyter notebook and the programming language that was used was Python 3. In addition, RStudio was used, an integrated development environment (IDE) for R, a programming language for statistical computing and graphics. (Wikipedia, 2019)

4.2 Main Analysis

4.2.1 Variables and interpretation

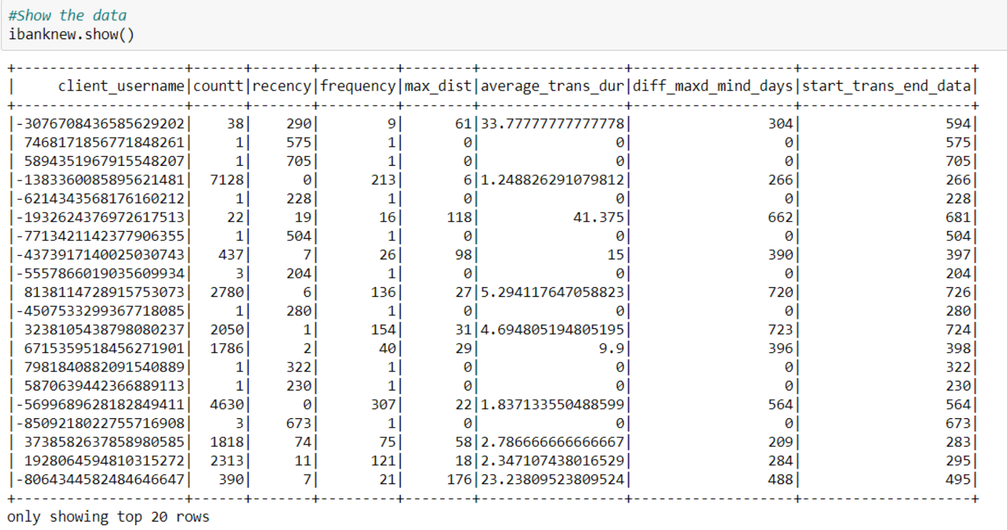
After executing the appropriate queries on the Hue (illustratively, the code for the total period of two years is presented in chapter 7 appendings), removing clients who had null names and keeping those who had successful transactions with the internet banking of NBG, we stored the data and extracted it to a csv file with title (‘"churn_results_3_merged.csv”) .This csv fie concerned the whole data with dates from 01/08/2017 until 31/07/2019 and had the following variables (figure 21):

client_username	the username of each user
countt	the total number of transactions of each user in the studied period
recency	the duration (number of days) between the last transaction of each user and present (as present we assume the more recent date of the period of two years i.e. the date 31/07/2019)
frequency	the total times (different days) that a user got to the system in the studied period (frequency of transactions of each user)
max_dist	the maximum duration between two consecutive transactions of each user in the studied period
average_trans_dur	the average duration between two consecutive transactions of each user in the study period
diff_maxd_mind_days	the duration between the first and the last transaction of each user in the studied period
start_trans_end_data	the duration between the first transaction of each user and present (31/07/2019)

Figure 21: Data variables

4.2.2 The whole dataset (2 years)

Using the Jupyter notebook on Ubuntu software and after importing the appropriate libraries, we read in spark the csv file with the whole two years data. The created Pyspark data frame with tittle ‘ibanknew’ had the following form (figure 22):

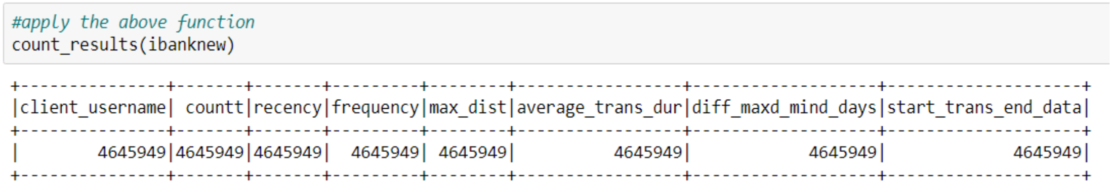


```
#Show the data
ibanknew.show()
```

```
+--------------------+------+-------+---------+--------+-----------------+-------------------+--------------------+
|     client_username|countt|recency|frequency|max_dist|average_trans_dur|diff_maxd_mind_days|start_trans_end_data|
+--------------------+------+-------+---------+--------+-----------------+-------------------+--------------------+
|-3076708436585629202|    38|    290|        9|      61|33.77777777777778|                304|                 594|
| 7468171856771848261|     1|    575|        1|       0|                0|                  0|                 575|
| 5894351967915548207|     1|    705|        1|       0|                0|                  0|                 705|
|-1383360085895621481|  7128|      0|      213|       6|1.248826291079812|                266|                 266|
|-6214343568176160212|     1|    228|        1|       0|                0|                  0|                 228|
|-1932624376972617513|    22|     19|       16|     118|           41.375|                662|                 681|
|-7713421142377906355|     1|    504|        1|       0|                0|                  0|                 504|
|-4373917140025030743|   437|      7|       26|      98|               15|                390|                 397|
|-5557866019035609934|     3|    204|        1|       0|                0|                  0|                 204|
| 8138114728915753073|  2780|      6|      136|      27|5.294117647058823|                720|                 726|
|-4507533299367718085|     1|    280|        1|       0|                0|                  0|                 280|
| 3238105438798080237|  2050|      1|      154|      31|4.694805194805195|                723|                 724|
| 6715359518456271901|  1786|      2|       40|      29|              9.9|                396|                 398|
| 7981840882091540889|     1|    322|        1|       0|                0|                  0|                 322|
| 5870639442366889113|     1|    230|        1|       0|                0|                  0|                 230|
|-5699689628182849411|  4630|      0|      307|      22|1.837133550488599|                564|                 564|
|-8509218022755716908|     3|    673|        1|       0|                0|                  0|                 673|
| 3738582637858980585|  1818|     74|       75|      58|2.786666666666667|                209|                 283|
| 1928064594810315272|  2313|     11|      121|      18|2.347107438016529|                284|                 295|
|-8064344582484646647|   390|      7|       21|     176|23.23809523809524|                488|                 495|
+--------------------+------+-------+---------+--------+-----------------+-------------------+--------------------+
only showing top 20 rows
```

Figure 22: ibanknew spark data frame

Defining a function ‘count_results’ we checked if the count results are the same (in each column) in the spark dataframe, in other words, if there are null-missing values as we can see below(figure 23):

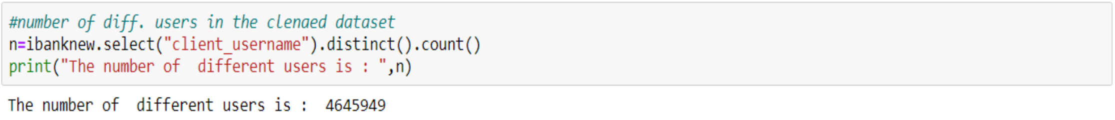


```
#apply the above function
count_results(ibanknew)
```

```
+---------------+-------+-------+---------+--------+-----------------+-------------------+--------------------+
|client_username| countt|recency|frequency|max_dist|average_trans_dur|diff_maxd_mind_days|start_trans_end_data|
+---------------+-------+-------+---------+--------+-----------------+-------------------+--------------------+
|        4645949|4645949|4645949|  4645949| 4645949|          4645949|            4645949|             4645949|
+---------------+-------+-------+---------+--------+-----------------+-------------------+--------------------+
```

Figure 23: Function count_results

Fortunately, as we could see in the figure 23, we did not face any null values. In addition, from the figure 23 it seems that the total number of rows is 4.645.949. To assess the number of (different) clients too, in order to be sure that there are no duplicates, we executed the following command (figure 24):

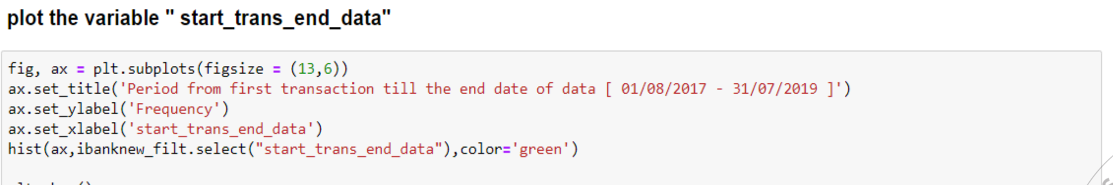


```
#number of diff. users in the clenaed dataset
n=ibanknew.select("client_username").distinct().count()
print("The number of  different users is : ",n)
```

```
The number of  different users is :  4645949
```

Figure 24: Number of distinct clients

Before starting the analysis we had to examine how many clients we are going to take into account because some of them did their "first" transaction close to the end date of data collection (31/07/2019). As a result there is not enough time to study their behavior in such a short period so as to come to a conclusion for their pattern. The main question was how far back in time we have to go so as to drop these specific clients? An approach was to see the distribution of the parameter 'start_trans_end_data' and some statistical metrics for all clients and make assumptions. Plotting a histogram for the variable ‘start_trans_end_data’ we got the following results (figure 25):



plot the variable " start_trans_end_data"

```
fig, ax = plt.subplots(figsize = (13,6))
ax.set_title('Period from first transaction till the end date of data [ 01/08/2017 - 31/07/2019 ]')
ax.set_ylabel('Frequency')
ax.set_xlabel('start_trans_end_data')
hist(ax,ibanknew_filt.select("start_trans_end_data"),color='green')

plt.show()
```

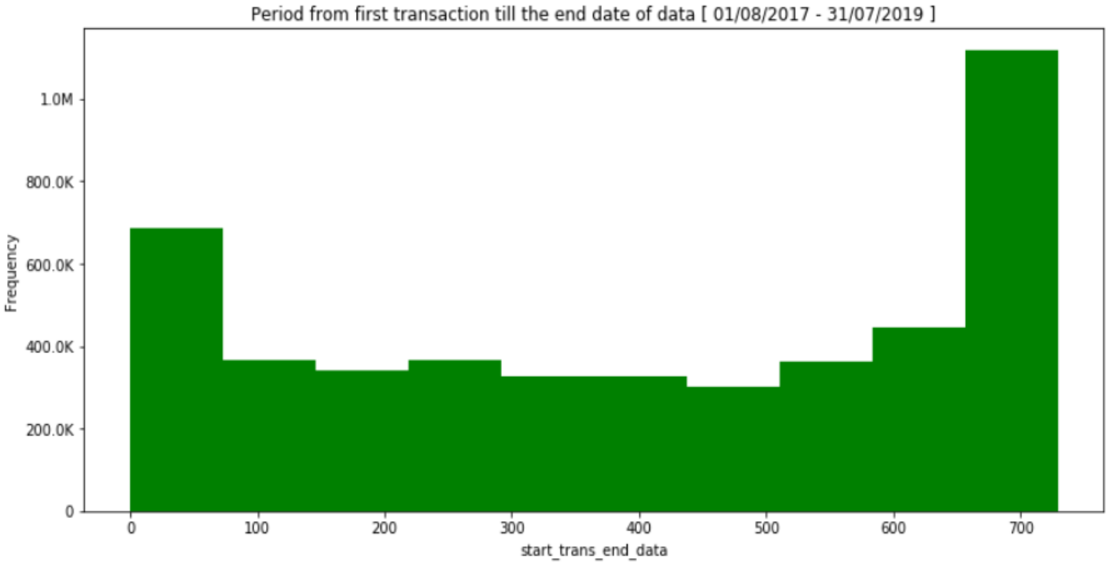
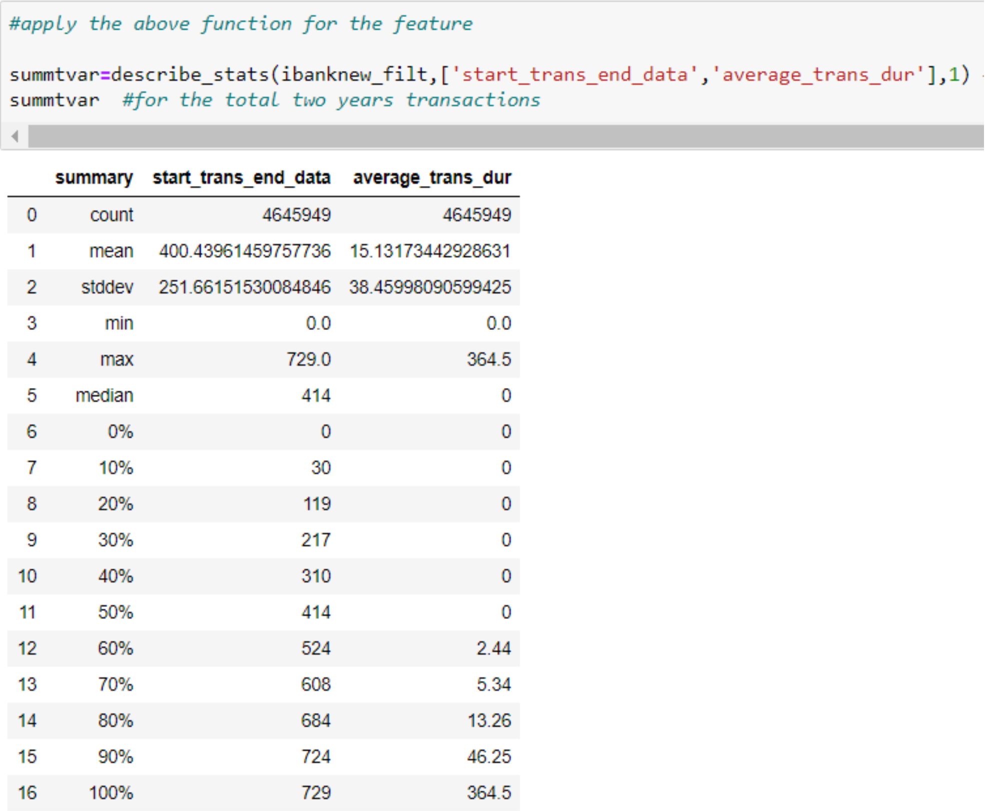


Figure 25: Plot of the variable start_trans_end_data (2 years data)

At a glance we could say that the right part (mainly) and a little the left part of the graph have a great deal of weight of clients. There are many clients (over 1.000.000) who did their first transaction approximately 650 days before the end date of collection data (31/07/2019) and over and a sufficient amount of clients (approximately 700.000 clients) who did their respective one close to the collection end date (approximately the last 2 months).

Defining a function “describe_stats” for getting the statistical data for both features ‘start_trans_end_data’ and ‘average_trans_dur’ we got the following results (figure 26):



```
#apply the above function for the feature

summtvar=describe_stats(ibanknew_filt,['start_trans_end_data','average_trans_dur'],1)
summtvar  #for the total two years transactions
```

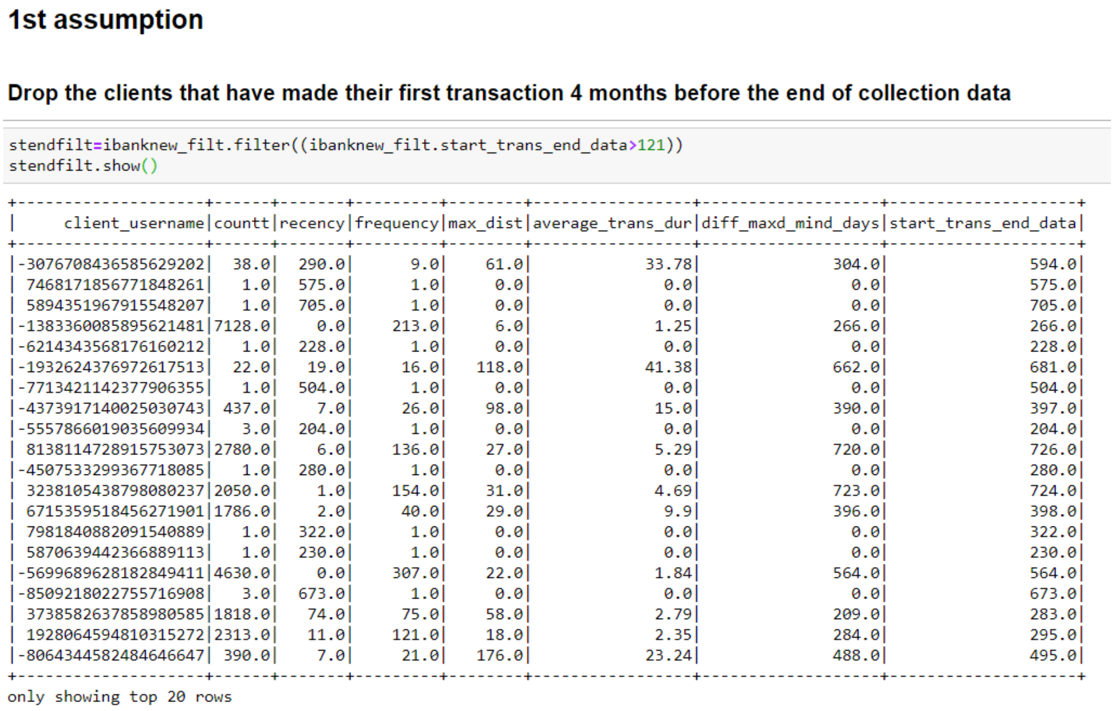
	summary	start_trans_end_data	average_trans_dur
0	count	4645949	4645949
1	mean	400.43961459757736	15.13173442928631
2	stddev	251.66151530084846	38.45998090599425
3	min	0.0	0.0
4	max	729.0	364.5
5	median	414	0
6	0%	0	0
7	10%	30	0
8	20%	119	0
9	30%	217	0
10	40%	310	0
11	50%	414	0
12	60%	524	2.44
13	70%	608	5.34
14	80%	684	13.26
15	90%	724	46.25
16	100%	729	364.5

Figure 26: Statistical metrics for the variables ‘start_trans_end_data’ and ‘average_trans_dur’ for the 2 years period

As far as the variable 'start_trans_end_data' is concerned ,someone could observe that there is at least one user that firstly got to the internet NBG banking at the first date of collection data (01/08/2017) and at least one user that entered to the NBG system at the last date of collection data. (31/07/2019).

As far as the variable 'average_trans_dur' is concerned, the average-mean of average duration between two consecutive transactions of users is 15.13 and with one standard deviation goes to the 15.13+38.46 = 53,59 < 60 days (= 2 months). This was the stimulus so as to take 2 months (60 days) as lead period-window. Now, we had to study the clients' behavior from the start date of collection until 2 months before the end collection date and run clustering for the clients in this period (observation period [01/08/2017, 31/05/2019]. Unfortunately, the results of the clustering showed us that the window of two months was not enough for our study and we had to take a larger time period. As a result, based on the above results and on some documentations (page 7) we considered a four month lead period. So, this plan was implemented then.

Due to the fact that we kept the last four months for lead period, it was immediate consequence to filter the data dropping the clients who got to the internet banking of NBG for first time within the period of the last four months. The new dataframe that was created had the new name “stendfilt”. The result of this procedure was the following (figure 27):



1st assumption

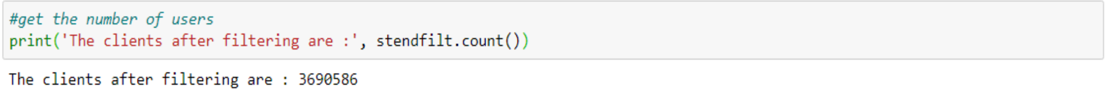
Drop the clients that have made their first transaction 4 months before the end of collection data

```
stendfilt=ibanknew_filt.filter((ibanknew_filt.start_trans_end_data>121))
stendfilt.show()
```

```
+--------------------+------+-------+---------+--------+-----------------+-------------------+--------------------+
|     client_username|countt|recency|frequency|max_dist|average_trans_dur|diff_maxd_mind_days|start_trans_end_data|
+--------------------+------+-------+---------+--------+-----------------+-------------------+--------------------+
|-3076708436585629202|  38.0|  290.0|      9.0|    61.0|            33.78|              304.0|               594.0|
| 7468171856771848261|   1.0|  575.0|      1.0|     0.0|              0.0|                0.0|               575.0|
| 5894351967915548207|   1.0|  705.0|      1.0|     0.0|              0.0|                0.0|               705.0|
|-1383360085895621481|7128.0|    0.0|    213.0|     6.0|             1.25|              266.0|               266.0|
|-6214343568176160212|   1.0|  228.0|      1.0|     0.0|              0.0|                0.0|               228.0|
|-1932624376972617513|  22.0|   19.0|     16.0|   118.0|            41.38|              662.0|               681.0|
|-7713421142377906355|   1.0|  504.0|      1.0|     0.0|              0.0|                0.0|               504.0|
|-4373917140025030743| 437.0|    7.0|     26.0|    98.0|             15.0|              390.0|               397.0|
|-5557866019035609934|   3.0|  204.0|      1.0|     0.0|              0.0|                0.0|               204.0|
| 8138114728915753073|2780.0|    6.0|    136.0|    27.0|             5.29|              720.0|               726.0|
|-4507533299367718085|   1.0|  280.0|      1.0|     0.0|              0.0|                0.0|               280.0|
| 3238105438798080237|2050.0|    1.0|    154.0|    31.0|             4.69|              723.0|               724.0|
| 6715359518456271901|1786.0|    2.0|     40.0|    29.0|              9.9|              396.0|               398.0|
| 7981840882091540889|   1.0|  322.0|      1.0|     0.0|              0.0|                0.0|               322.0|
| 5870639442366889113|   1.0|  230.0|      1.0|     0.0|              0.0|                0.0|               230.0|
|-5699689628182849411|4630.0|    0.0|    307.0|    22.0|             1.84|              564.0|               564.0|
|-8509218022755716908|   3.0|  673.0|      1.0|     0.0|              0.0|                0.0|               673.0|
| 3738582637858980585|1818.0|   74.0|     75.0|    58.0|             2.79|              209.0|               283.0|
| 1928064594810315272|2313.0|   11.0|    121.0|    18.0|             2.35|              284.0|               295.0|
|-8064344582484646647| 390.0|    7.0|     21.0|   176.0|            23.24|              488.0|               495.0|
+--------------------+------+-------+---------+--------+-----------------+-------------------+--------------------+
only showing top 20 rows
```

Figure 27: Data with removing the clients who got to the internet banking the last 4 months

After this filtering, the number of customers reduced, so the new number of them was 3.690.586 as we can see below (figure 28):



```
#get the number of users
print('The clients after filtering are :', stendfilt.count())
```

```
The clients after filtering are : 3690586
```

Figure 28: Number of clients after removing those who had first transaction in the last four months

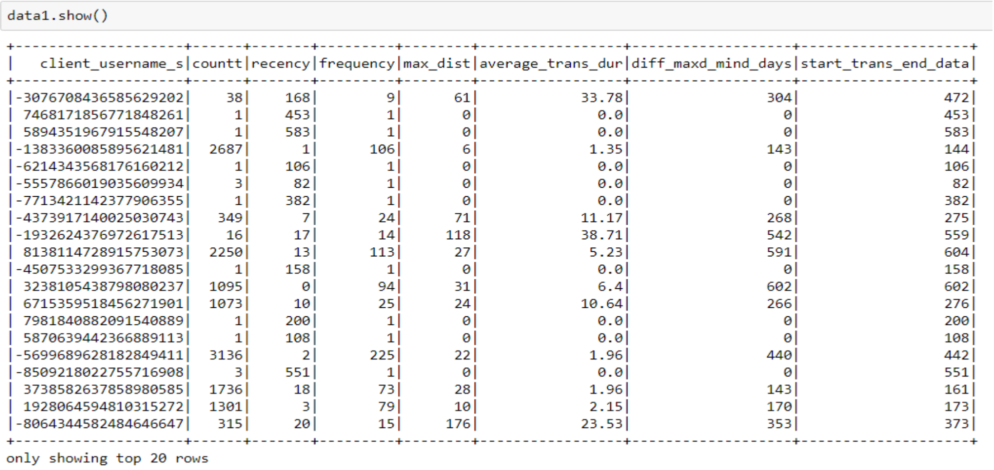
In other words, for the studying period of two years, within the last four months, there were 955.363 clients that “firstly” entered to the internet banking of NBG.

Leaving this Pyspark dataframe “stendfilt” aside for a while, (we will use it again at the end of the clustering phase), we had to turn our attention to the period from 01/08/2017 to 31/03/2019). Again, we had to execute the same query we executed before but this time for these specific dates. The new extracted csv file was called “churn_results_5_merged.csv”.

4.2.3 20 months data (Observation period)

4.2.3.1 Preprocessing

After reading in spark the file, renaming the columns and rounding the column “average_trans_dur”, our spark dataframe, named “data1” had the following form (figure 29):

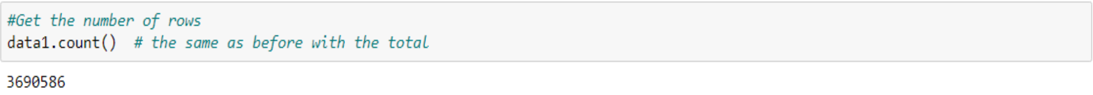


```
data1.show()
```

```
+--------------------+------+-------+---------+--------+-----------------+-------------------+--------------------+
|   client_username_s|countt|recency|frequency|max_dist|average_trans_dur|diff_maxd_mind_days|start_trans_end_data|
+--------------------+------+-------+---------+--------+-----------------+-------------------+--------------------+
|-3076708436585629202|    38|    168|        9|      61|            33.78|                304|                 472|
| 7468171856771848261|     1|    453|        1|       0|              0.0|                  0|                 453|
| 5894351967915548207|     1|    583|        1|       0|              0.0|                  0|                 583|
|-1383360085895621481|  2687|      1|      106|       6|             1.35|                143|                 144|
|-6214343568176160212|     1|    106|        1|       0|              0.0|                  0|                 106|
|-5557866019035609934|     3|     82|        1|       0|              0.0|                  0|                  82|
|-7713421142377906355|     1|    382|        1|       0|              0.0|                  0|                 382|
|-4373917140025030743|   349|      7|       24|      71|            11.17|                268|                 275|
|-1932624376972617513|    16|     17|       14|     118|            38.71|                542|                 559|
| 8138114728915753073|  2250|     13|      113|      27|             5.23|                591|                 604|
|-4507533299367718085|     1|    158|        1|       0|              0.0|                  0|                 158|
| 3238105438798080237|  1095|      0|       94|      31|              6.4|                602|                 602|
| 6715359518456271901|  1073|     10|       25|      24|            10.64|                266|                 276|
| 7981840882091540889|     1|    200|        1|       0|              0.0|                  0|                 200|
| 5870639442366889113|     1|    108|        1|       0|              0.0|                  0|                 108|
|-5699689628182849411|  3136|      2|      225|      22|             1.96|                440|                 442|
|-8509218022755716908|     3|    551|        1|       0|              0.0|                  0|                 551|
| 3738582637858980585|  1736|     18|       73|      28|             1.96|                143|                 161|
| 1928064594810315272|  1301|      3|       79|      10|             2.15|                170|                 173|
|-8064344582484646647|   315|     20|       15|     176|            23.53|                353|                 373|
+--------------------+------+-------+---------+--------+-----------------+-------------------+--------------------+
only showing top 20 rows
```

Figure 29: data1 (until 4 months before)

Computing the number of users (equivalent to number of rows) on this dataset we assessed that it is the same with the number of users in figure 28 as we can see below (figure 30):

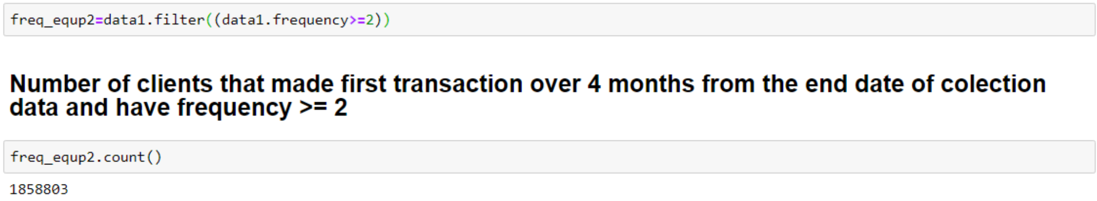


```
#Get the number of rows
data1.count()  # the same as before with the total
```

```
3690586
```

Figure 30: Number of users in data1

Owing to the fact that we had to study the behavior – pattern of the clients so as to do segmentation, we should have had at least two transactions for each client. As a result, in this moment, we had to make a second assumption. We filtered the data and kept those clients who had frequency >= 2 creating the spark dataframe, called “freq_equp2”. The new number of customers was 1.858.803 as we could see below (figure 31):



```
freq_equp2=data1.filter((data1.frequency>=2))
```

Number of clients that made first transaction over 4 months from the end date of colection data and have frequency >= 2

```
freq_equp2.count()
```

```
1858803
```

Figure 31: Number of clients of freq_equp2

We observed that there were 1.831.783 ~ 2.000.000 customers who had only one transaction in the studying period of 22 months, a large amount of clients, if we consider that the initial number was 3.690586. In other words, the 0,496 ~ 0,5 ~ 50% of the total clients made only one transaction in the studying period.

After converting the Pyspark data frame “freq_equp2” to a pandas dataframe with the name “pd__freq_equp2”, we executed a scatter matrix for all the numeric variables of the data. The output was the following (figure 32):

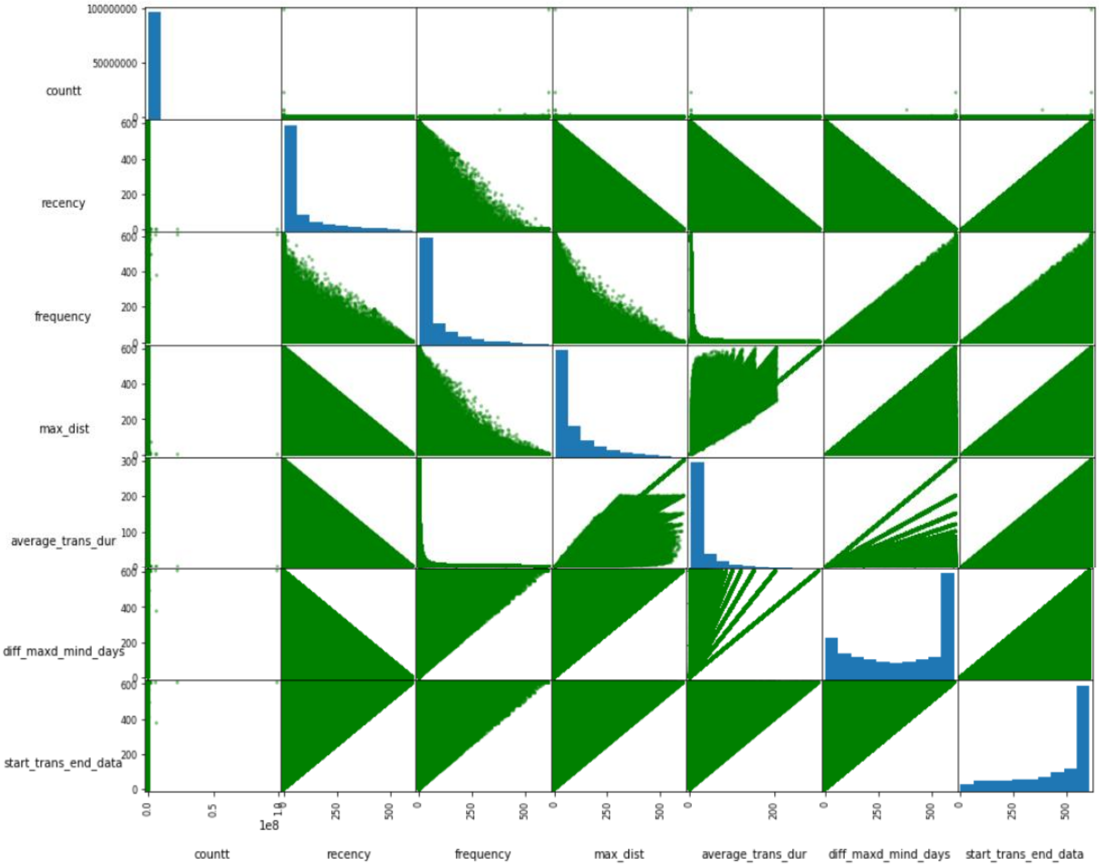
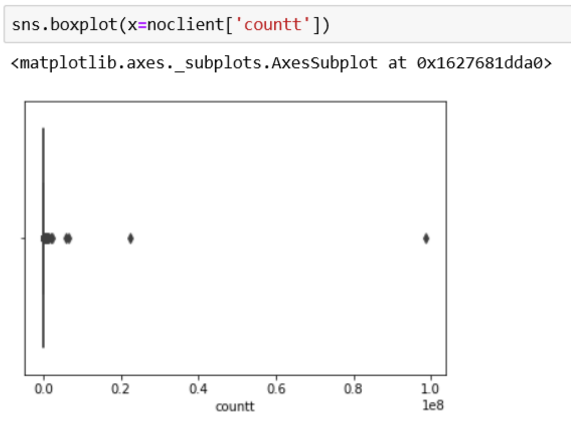


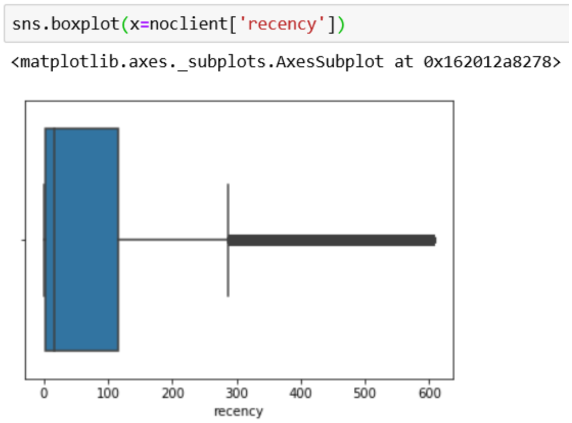
Figure 32: Scatter matrix

The scatter_matrix () function helps in plotting the preceding figure. It takes as input (numeric) data and it has as output a square table with dims n * n where n = number of variables. In the diagonal graphs, where the variable is against itself, a histogram is plotted. From the above scatterplot we could see that there are many outliers in our dataset (e.g. in the plot between average_trans_dur and max_dist and in the plot between diff_maxd_mind_days and average_trans_dur). As a result, we exported the pandas dataframe (pd__freq_equp2) to a csv file with the name “outlierscheck4fromubuntu.csv ” so as to check for outliers on the windows anaconda Jupyter notebook.

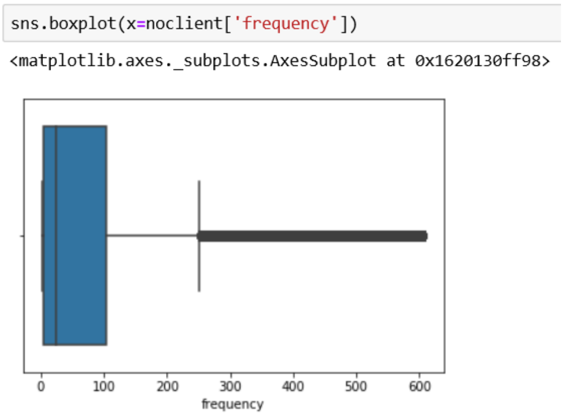
After importing again the appropriate libraries on windows Jupyter this time, we read the csv file as pandas data frame, called “dfclient_feat”. Selecting the numeric variables, we “searched” for outliers firstly with a visual way, using the boxplots (of seaborn library), so as to have a better immediate understanding of them and then applying the method of removing the outliers through IQR score. The boxplots assessed our intuition for the existence of outliers as someone could see in the following figures (figure 33):



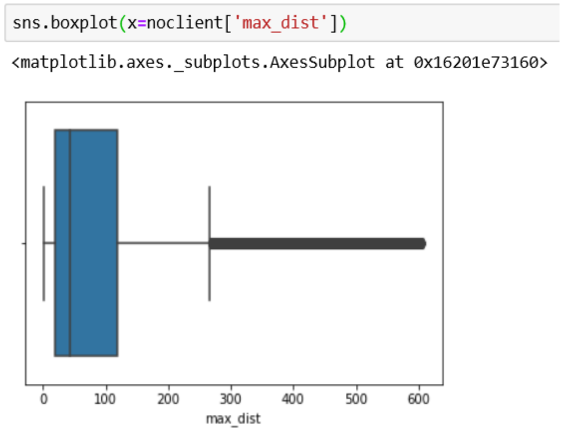
```
sns.boxplot(x=noclient['countt'])
<matplotlib.axes._subplots.AxesSubplot at 0x1627681dda0>
```



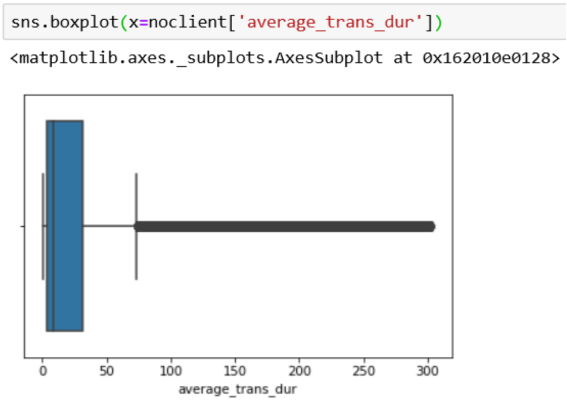
```
sns.boxplot(x=noclient['recency'])
<matplotlib.axes._subplots.AxesSubplot at 0x162012a8278>
```



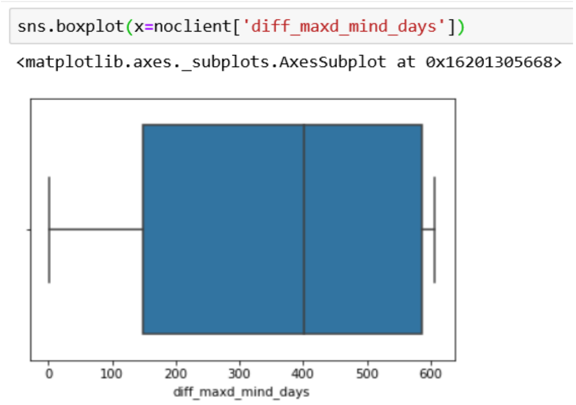
```
sns.boxplot(x=noclient['frequency'])
<matplotlib.axes._subplots.AxesSubplot at 0x1620130ff98>
```



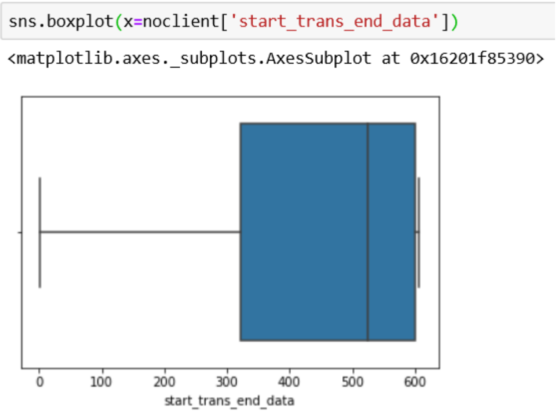
```
sns.boxplot(x=noclient['max_dist'])
<matplotlib.axes._subplots.AxesSubplot at 0x16201e73160>
```



```
sns.boxplot(x=noclient['average_trans_dur'])
<matplotlib.axes._subplots.AxesSubplot at 0x162010e0128>
```



```
sns.boxplot(x=noclient['diff_maxd_mind_days'])
<matplotlib.axes._subplots.AxesSubplot at 0x16201305668>
```



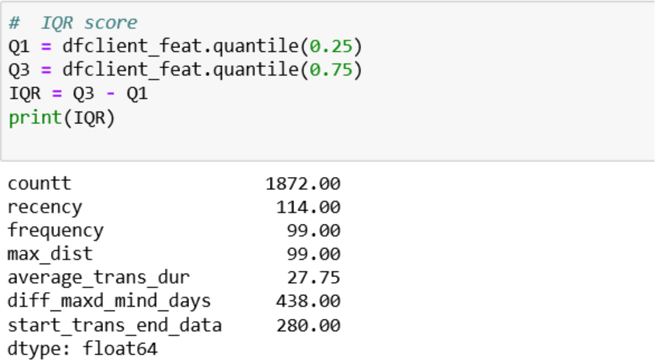
```
sns.boxplot(x=noclient['start_trans_end_data'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x16201f85390>
```

Figure 33: Boxplots for outliers

Observing the boxplots we could say that as far as the variables diff_maxd_mind_days and start_trans_end_data no outliers are detected.

As far as the removing of outliers is concerned through IQR score, we computed the IQR score for each column as follows (figure 34):

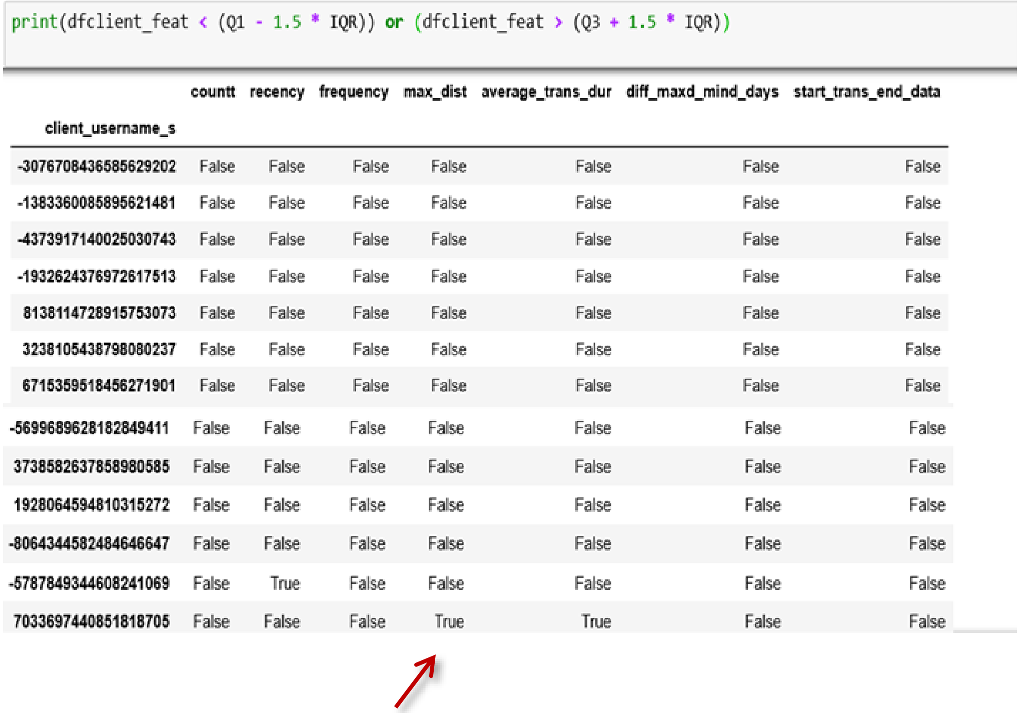


```
#  IQR score
Q1 = dfclient_feat.quantile(0.25)
Q3 = dfclient_feat.quantile(0.75)
IQR = Q3 - Q1
print(IQR)
```

```
countt                  1872.00
recency                  114.00
frequency                 99.00
max_dist                  99.00
average_trans_dur         27.75
diff_maxd_mind_days      438.00
start_trans_end_data     280.00
dtype: float64
```

Figure 34: IQR score for numeric variables of the data

As the theoretical background supports, we dropped any value that lied left from the Q1 - 1.5 * IQR value and right from the Q3 + 1.5 * IQR and we kept the data instances that had the intermediate values. More specific, we firstly printed a pandas data frame analogous to our original data frame with the difference that except for the numeric values it had boolean ones (False\True). The Boolean value False indicated that the data point was in our desirable range of values whereas the boolean value True indicated that the data point was not (and by extension it should be removed). The figure 35 illustrates a sample of the above data frame:



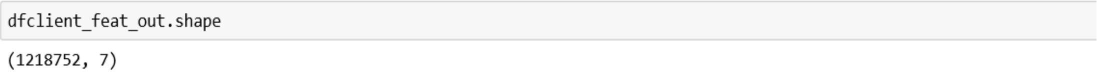
```
print(dfclient_feat < (Q1 - 1.5 * IQR)) or (dfclient_feat > (Q3 + 1.5 * IQR))
```

client_username_s	countt	recency	frequency	max_dist	average_trans_dur	diff_maxd_mind_days	start_trans_end_data
-3076708436585629202	False	False	False	False	False	False	False
-1383360085895621481	False	False	False	False	False	False	False
-4373917140025030743	False	False	False	False	False	False	False
-1932624376972617513	False	False	False	False	False	False	False
8138114728915753073	False	False	False	False	False	False	False
3238105438798080237	False	False	False	False	False	False	False
6715359518456271901	False	False	False	False	False	False	False
-5699689628182849411	False	False	False	False	False	False	False
3738582637858980585	False	False	False	False	False	False	False
1928064594810315272	False	False	False	False	False	False	False
-8064344582484646647	False	False	False	False	False	False	False
-5787849344608241069	False	True	False	False	False	False	False
7033697440851818705	False	False	False	True	True	False	False

Figure 35: IQR score Boolean table

<u>Comment</u>: Observing the above table we could see for example that the client with username “**7033697440851818870**” has an “anomaly” value as far as the variable max_dist since it has the value True in this sell.

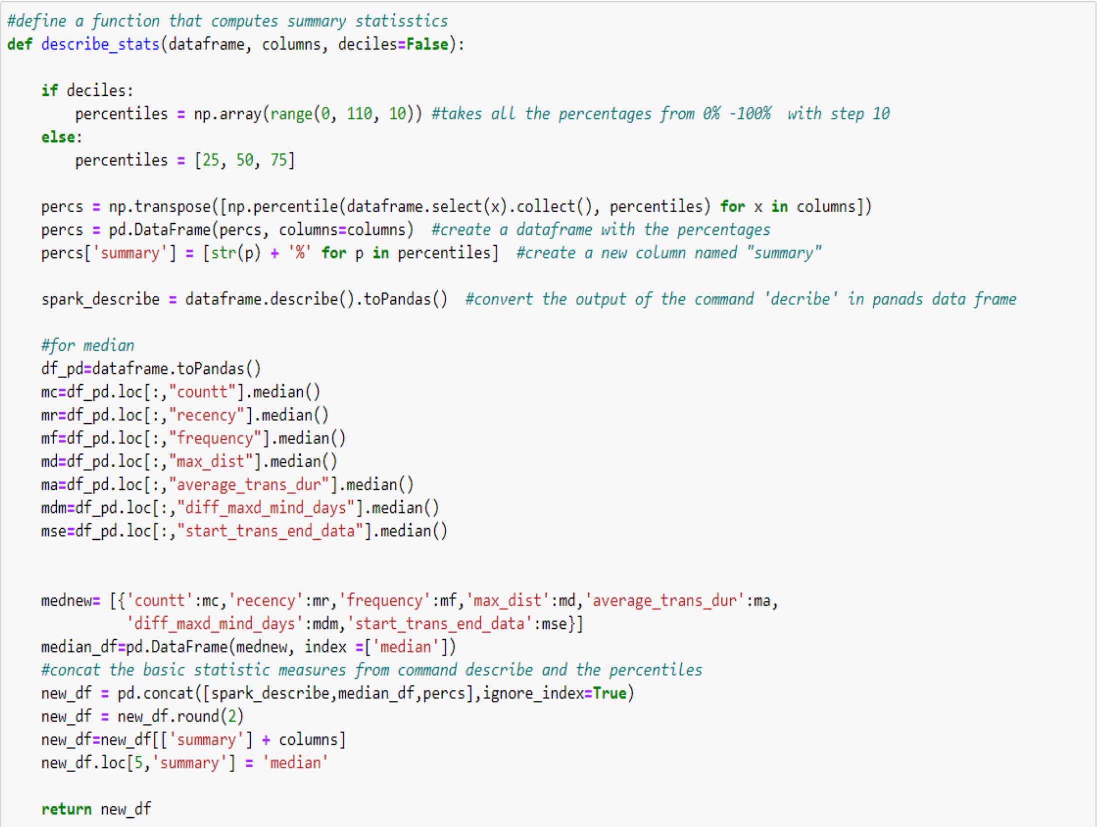
Having removed the outliers, we extracted the “cleaned” pandas dataframe (dfclient_feat_out) to a s csv file (dfclient_feat_out4) so as to be read then in Ubuntu jupyter notebook again. It should be mentioned that, using the IQR score method we found that there were 640.051 outliers since the new number of clients was 1.218.752 as we can see in the figure 36



```
dfclient_feat_out.shape
(1218752, 7)
```

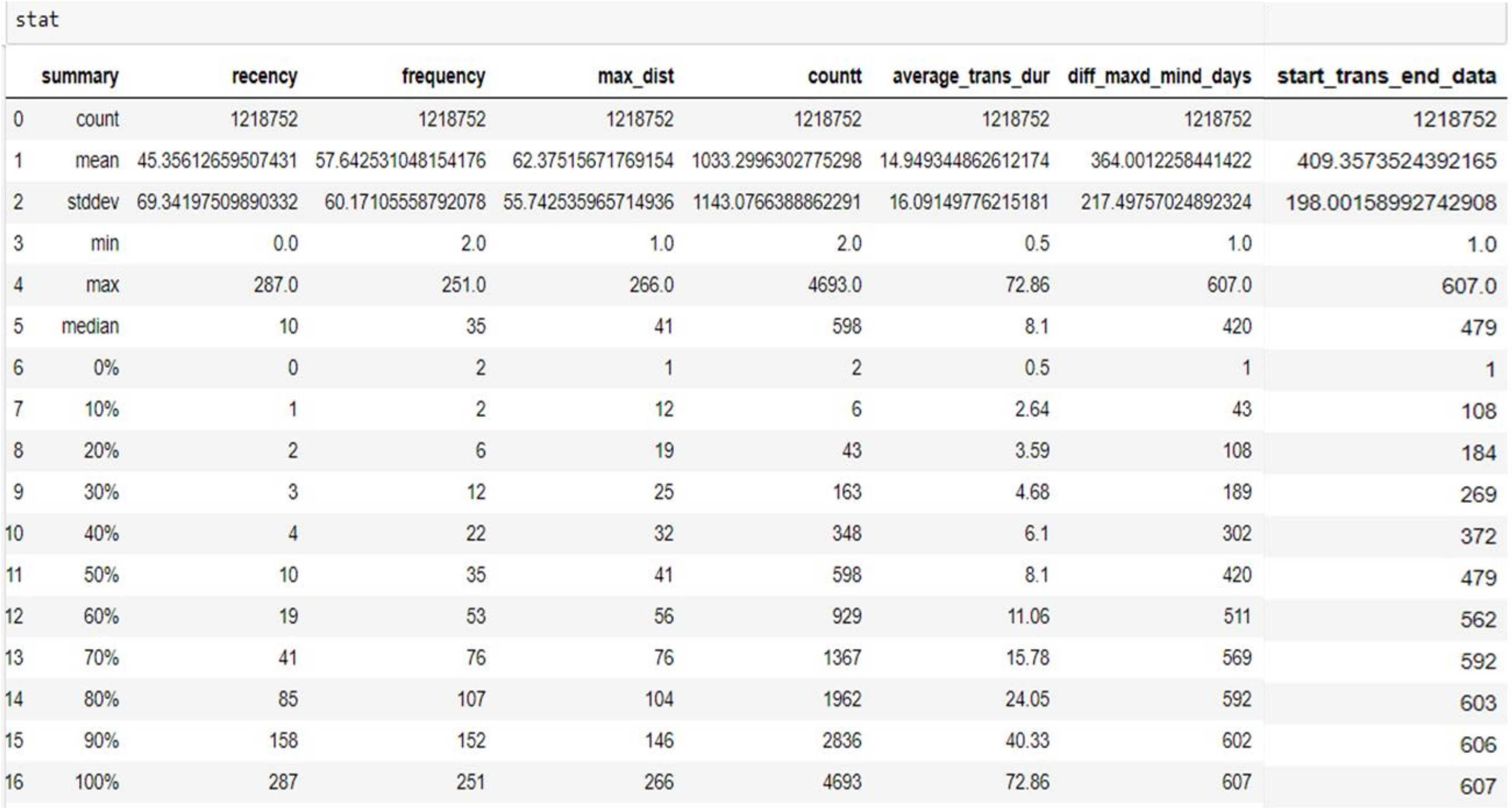
Figure 36: Number of users without outliers

After reading the csv file in Pyspark, we computed some statistic metrics for our new filtered data. In this point, we have to note that, because of the large size of data, it was vital to divide the dataframe to parts and then apply the function for the statistics on each of these parts, because the kernel was shut down every time I was trying to run the function on the whole dataframe. After joining the individual parts, we finally got the statistic metrics for our dataset. Below are the utilized function (figure 37) and then the table (figure 38) with the statistic values of our data set:



```
#define a function that computes summary statisstics
def describe_stats(dataframe, columns, deciles=False):

    if deciles:
        percentiles = np.array(range(0, 110, 10)) #takes all the percentages from 0% -100%  with step 10
    else:
        percentiles = [25, 50, 75]

    percs = np.transpose([np.percentile(dataframe.select(x).collect(), percentiles) for x in columns])
    percs = pd.DataFrame(percs, columns=columns)  #create a dataframe with the percentages
    percs['summary'] = [str(p) + '%' for p in percentiles]  #create a new column named "summary"

    spark_describe = dataframe.describe().toPandas()  #convert the output of the command 'decribe' in panads data frame

    #for median
    df_pd=dataframe.toPandas()
    mc=df_pd.loc[:,"countt"].median()
    mr=df_pd.loc[:,"recency"].median()
    mf=df_pd.loc[:,"frequency"].median()
    md=df_pd.loc[:,"max_dist"].median()
    ma=df_pd.loc[:,"average_trans_dur"].median()
    mdm=df_pd.loc[:,"diff_maxd_mind_days"].median()
    mse=df_pd.loc[:,"start_trans_end_data"].median()


    mednew= [{'countt':mc,'recency':mr,'frequency':mf,'max_dist':md,'average_trans_dur':ma,
              'diff_maxd_mind_days':mdm,'start_trans_end_data':mse}]
    median_df=pd.DataFrame(mednew, index =['median'])
    #concat the basic statistic measures from command describe and the percentiles
    new_df = pd.concat([spark_describe,median_df,percs],ignore_index=True)
    new_df = new_df.round(2)
    new_df=new_df[['summary'] + columns]
    new_df.loc[5,'summary'] = 'median'

    return new_df
```

Figure 37: Function for statistics



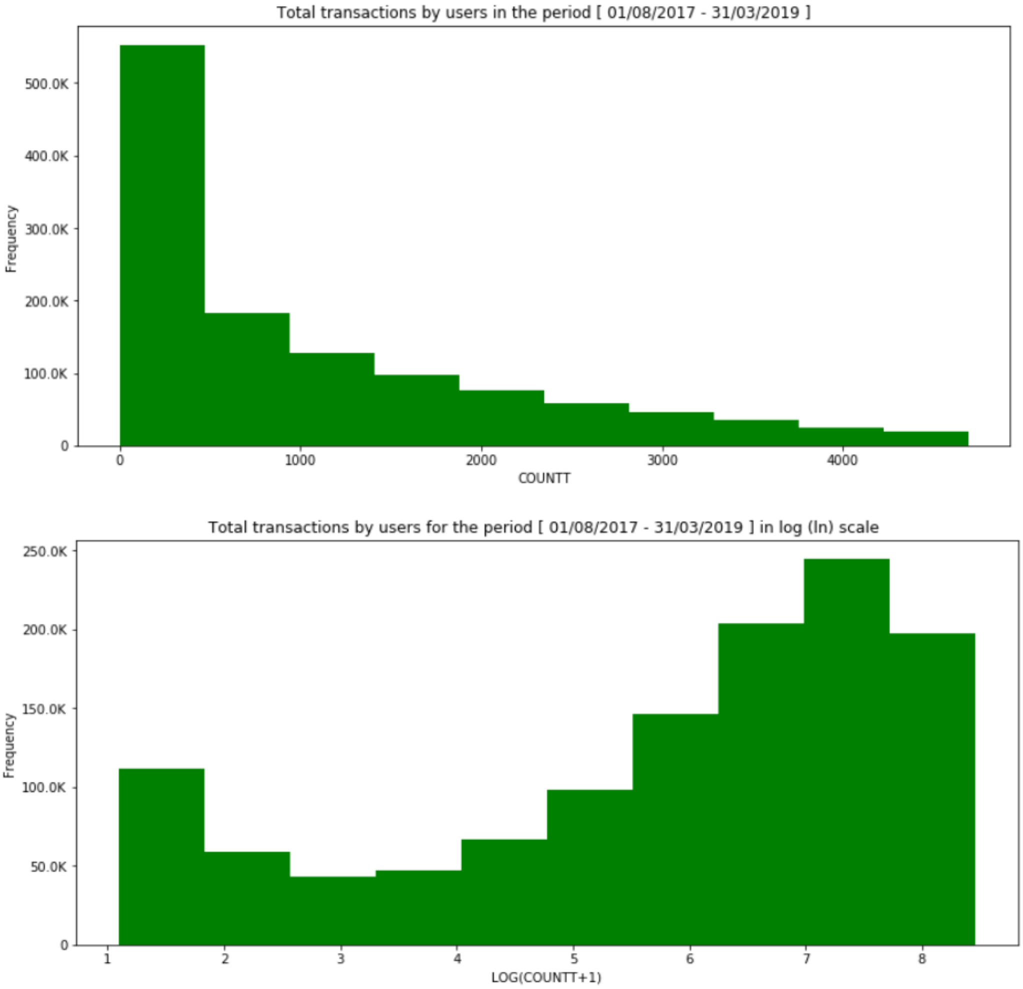
stat

	summary	recency	frequency	max_dist	countt	average_trans_dur	diff_maxd_mind_days	start_trans_end_data
0	count	1218752	1218752	1218752	1218752	1218752	1218752	1218752
1	mean	45.35612659507431	57.642531048154176	62.37515671769154	1033.2996302775298	14.949344862612174	364.0012258441422	409.3573524392165
2	stddev	69.34197509890332	60.17105558792078	55.742535965714936	1143.0766388862291	16.09149776215181	217.49757024892324	198.00158992742908
3	min	0.0	2.0	1.0	2.0	0.5	1.0	1.0
4	max	287.0	251.0	266.0	4693.0	72.86	607.0	607.0
5	median	10	35	41	598	8.1	420	479
6	0%	0	2	1	2	0.5	1	1
7	10%	1	2	12	6	2.64	43	108
8	20%	2	6	19	43	3.59	108	184
9	30%	3	12	25	163	4.68	189	269
10	40%	4	22	32	348	6.1	302	372
11	50%	10	35	41	598	8.1	420	479
12	60%	19	53	56	929	11.06	511	562
13	70%	41	76	76	1367	15.78	569	592
14	80%	85	107	104	1962	24.05	592	603
15	90%	158	152	146	2836	40.33	602	606
16	100%	287	251	266	4693	72.86	607	607

Figure 38: Table for Statistics

4.2.3.2 Plots

After computing some statistics for our dataset the next step was to make some plots for our numeric variables. Due to the fact that we had case in which one or a few points were much larger than the bulk of the data, it was necessary to take the log of the columns and especially the log (a+1) because there were values which were equal to zero (0) and the log (0) is not specified. So, we created the spark dataframe “freq_equp2_log” and we plot the log (variables +1). These plots are illustrated in the below figures (Each time, together with the plot with the original values is also the plot for the same variable but in the log scale so that they can be compared :


Total transactions by users in the period [01/08/2017 - 31/03/2019]

Total transactions by users for the period [01/08/2017 - 31/03/2019] in log (ln) scale

Figure 39: Histograms for countt

From the figure 39 (first plot) we could observe that the majority of the users’ transactions lies in the range of [1,500].

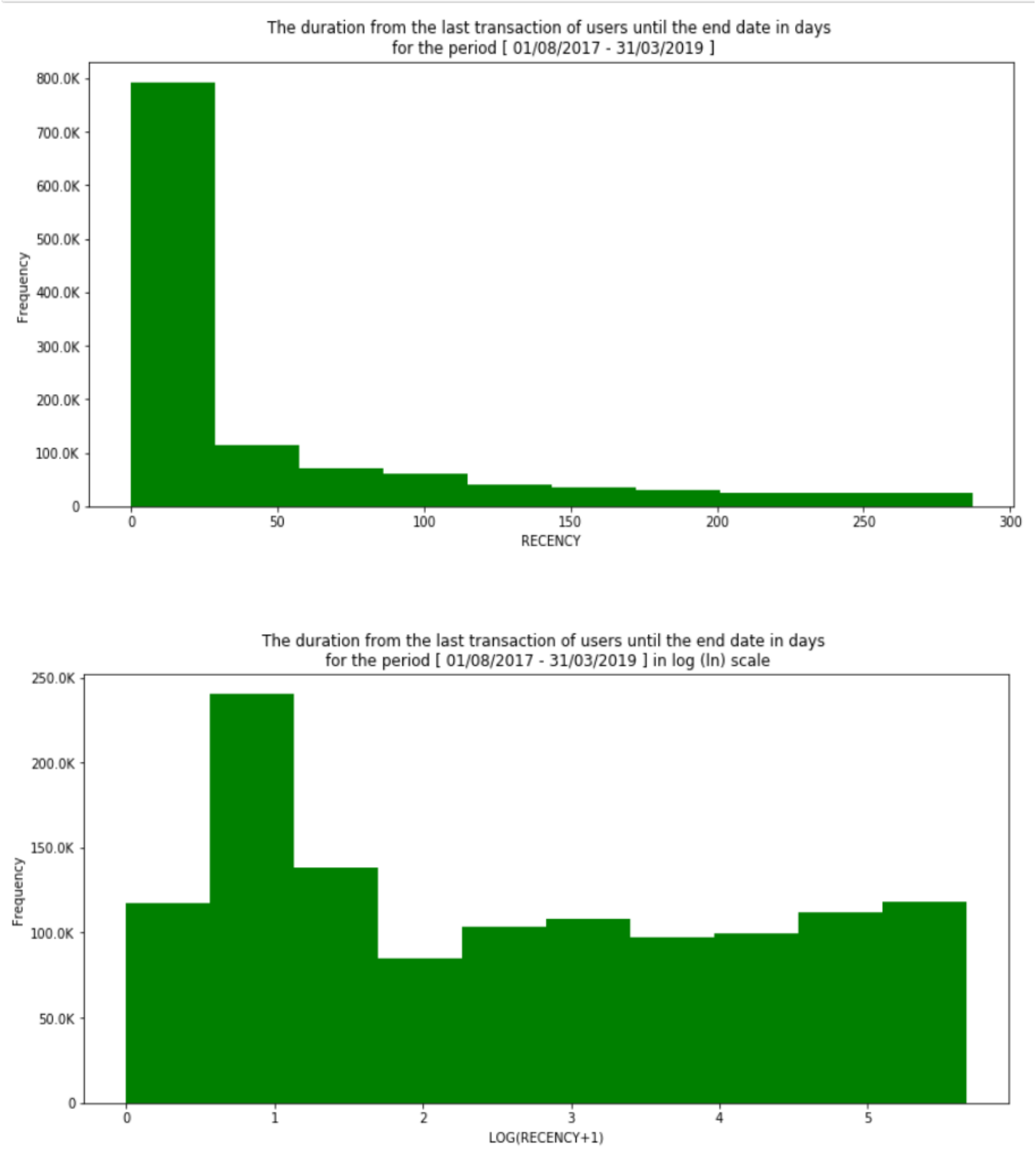


Figure 40: Histograms for recency

From the figure 40 (first graph) we could observe that the duration from the last transaction of users until 31/03/2019 mainly varies between values 0 - 25 days.

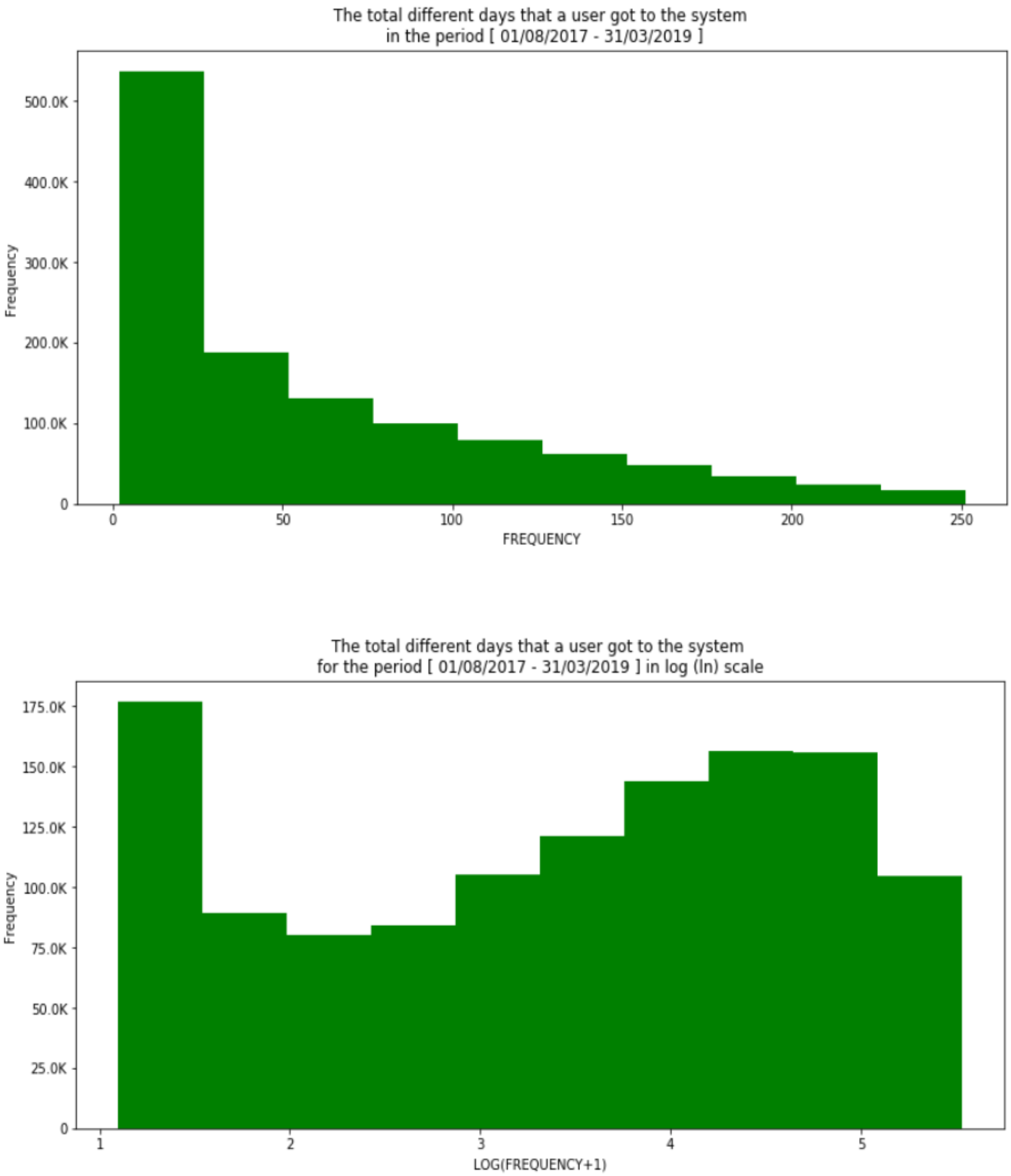


Figure 41: Histograms for frequency

From the figure 41 (first plot) we could observe that the majority of the users connected to the NBG internet banking from one to 25 times.

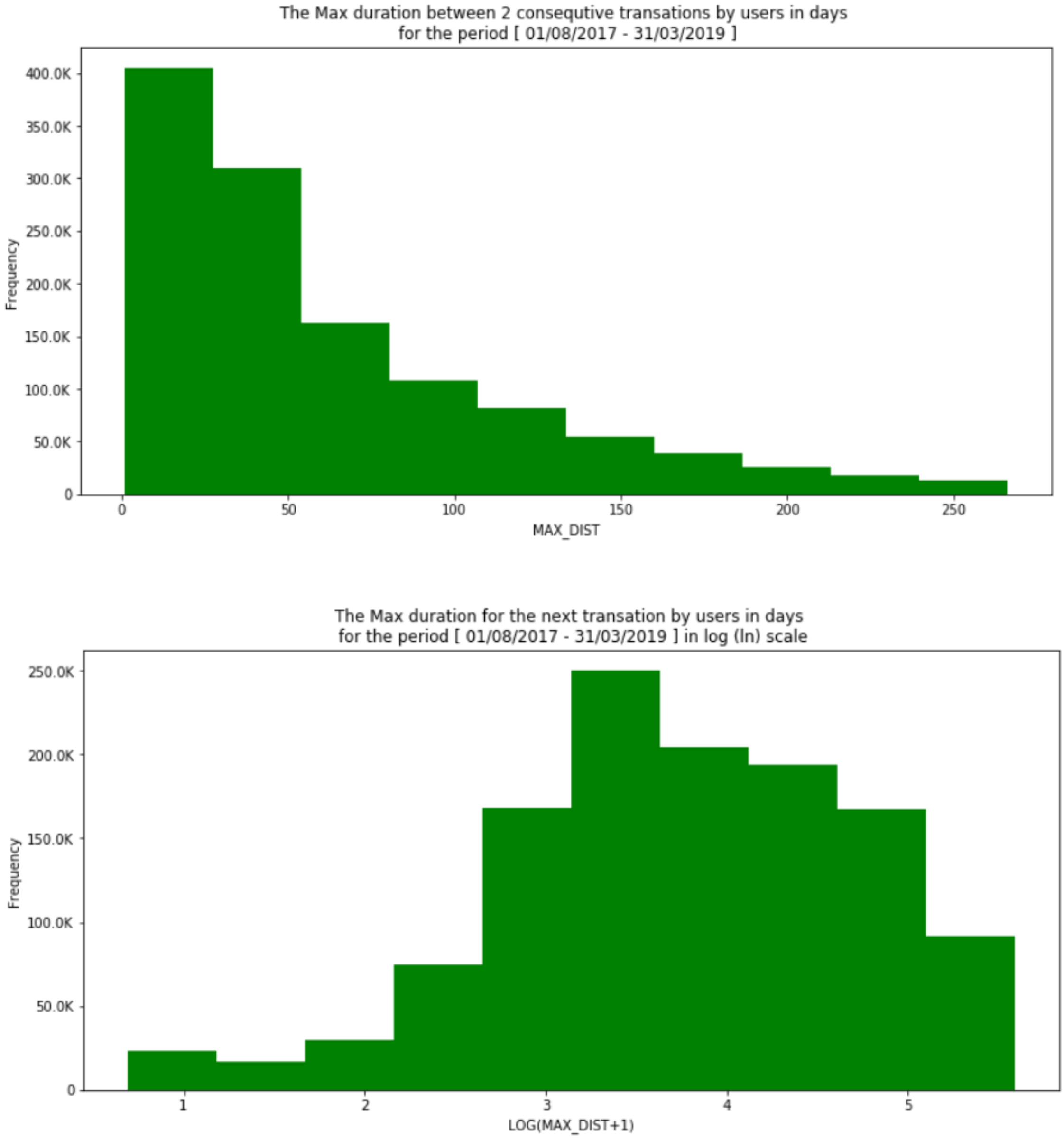


Figure 42: Histograms for max_dist

From the figure 42 (first graph) we could observe that after the 50 days there is a relatively uniform rate of reduction as far as the variable max_dist is concerned. In addition, the majority of the customers made its next consecutive transaction within 50 days maximum.

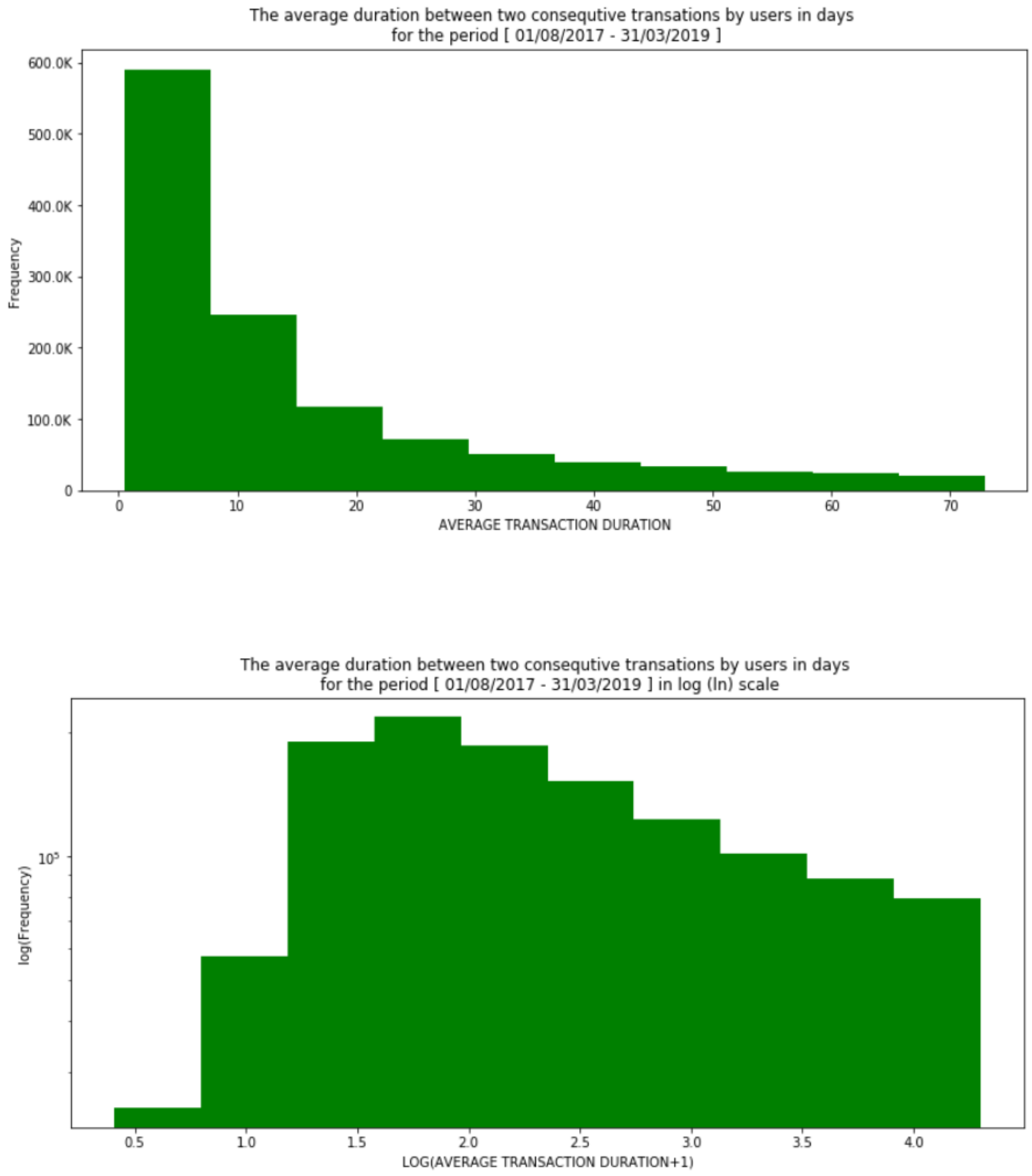


Figure 43: Histograms for average_trans_dur

Observing the first plot of the figure 43 we could say that the majority of the customers made its next consecutive transaction within 15 days in average and mainly within a week (7 days) in average.

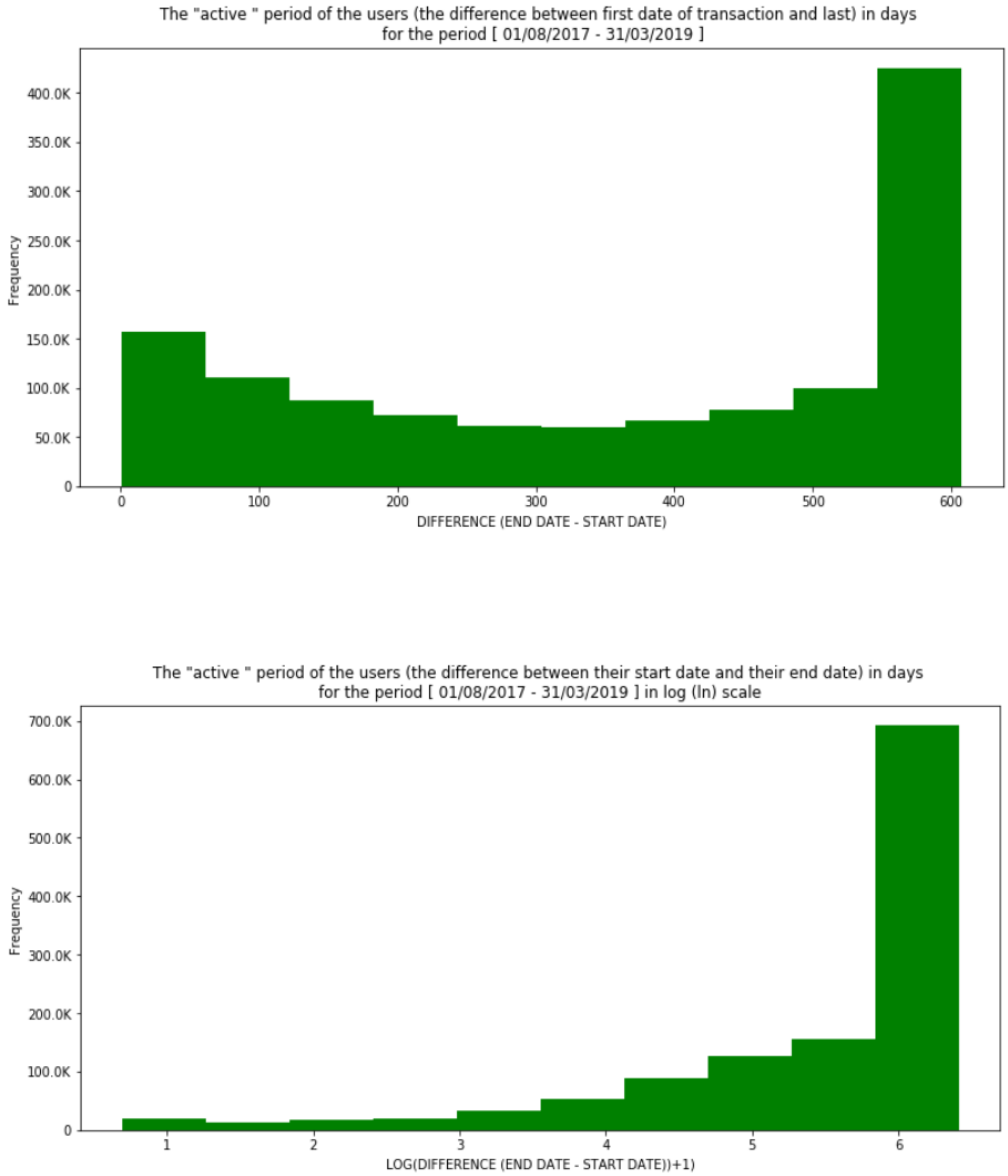


Figure 44: Histograms for diff_maxd_mind_days

As far as the diff_maxd_mind_days variable is concerned, we could say that there is a reverse path compared to the previous plots. Now, the majority is gathered on the right part of the graph. From the first plot we could say that the “active” period of users which is stated as the duration between the first and last transaction of each user is mainly between 550 to 600 days.

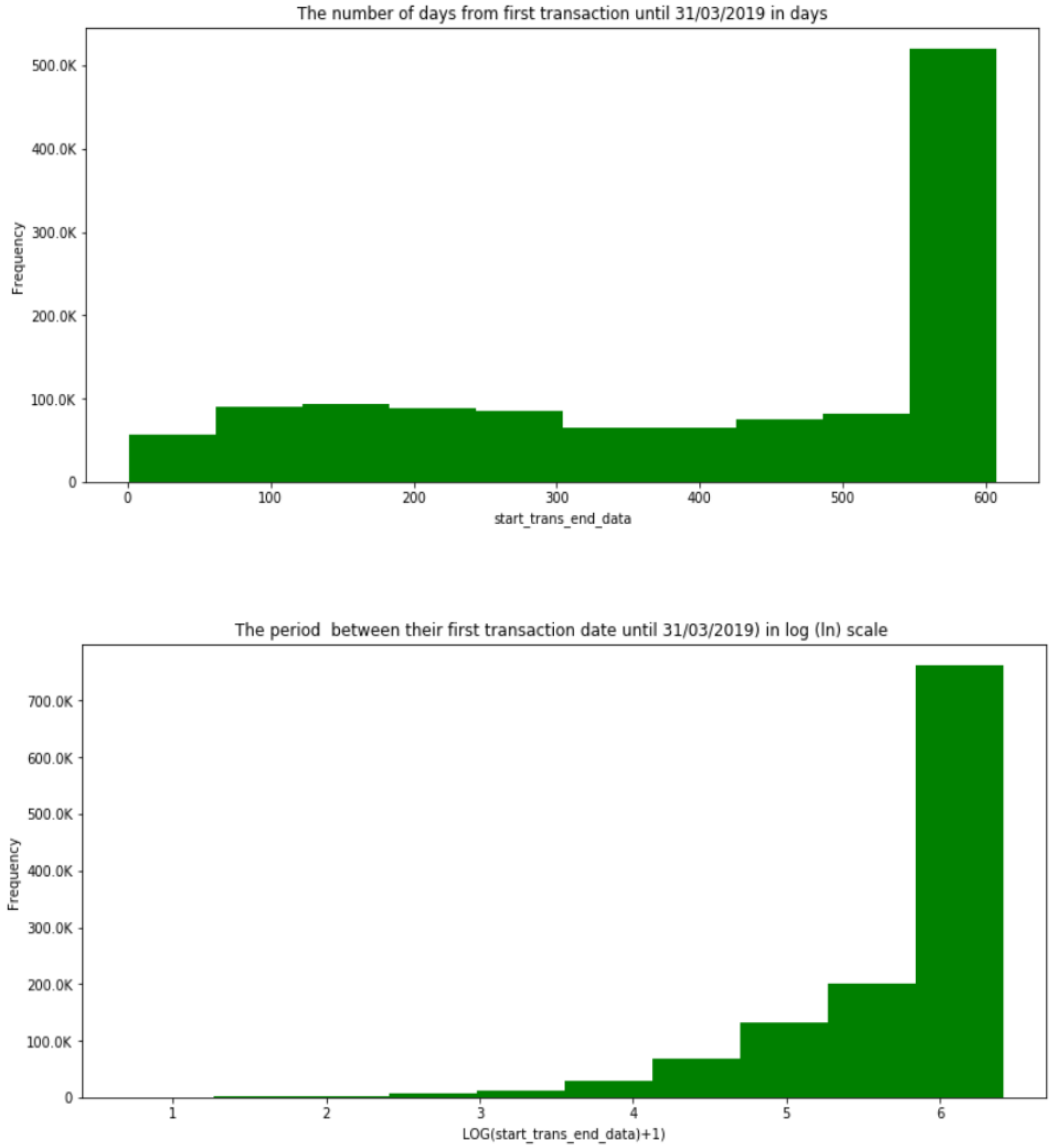
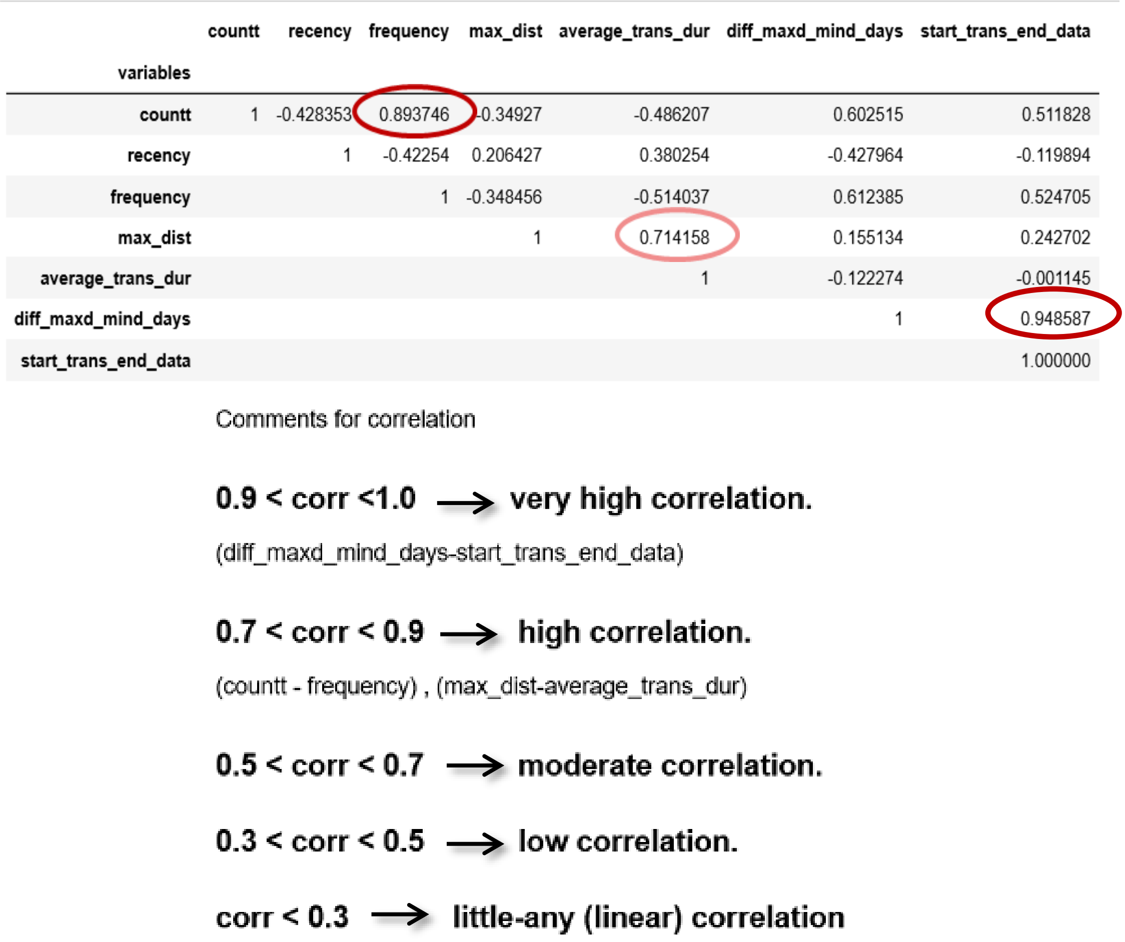


Figure 45: Histograms for start_trans_end_data

Finally, as far as the variable ‘start_trans_end_data’ the majority of the data is gathered in the right side of the graph. Observing the first plot of the figure 45 we could say that the largest amount of the clients made its first transaction before 550 - 600 days from the 31/03/2019. This means that the majority of people we work with is clients of NBG over a year and a half.

4.2.3.3 Correlation

After creating the plots, our next move was the clustering phase. One significant requirement before applying K-means algorithm is removing the variables that are very high correlated (use of spearman correlation - variables are continuous) because these variables don't provide any additional independent information and the algorithm will give more weight in computing the distance between two points. For this reason, it is recommended for the analysts to eliminate one of the two variables from their analysis. The variable to be retained in the analysis should be usually selected based on its practical usefulness. As a result, a correlation matrix should had been constructed, which was done at windows jupyter notebook. Following is a table with the correlation values between variables – features (figure 46) and then the correlation matrix of them (figure 47).



	countt	recency	frequency	max_dist	average_trans_dur	diff_maxd_mind_days	start_trans_end_data
variables							
countt	1	-0.428353	0.893746	-0.34927	-0.486207	0.602515	0.511828
recency		1	-0.42254	0.206427	0.380254	-0.427964	-0.119894
frequency			1	-0.348456	-0.514037	0.612385	0.524705
max_dist				1	0.714158	0.155134	0.242702
average_trans_dur					1	-0.122274	-0.001145
diff_maxd_mind_days						1	0.948587
start_trans_end_data							1.000000

Comments for correlation

0.9 < corr <1.0 → very high correlation.

(diff_maxd_mind_days-start_trans_end_data)

0.7 < corr < 0.9 → high correlation.

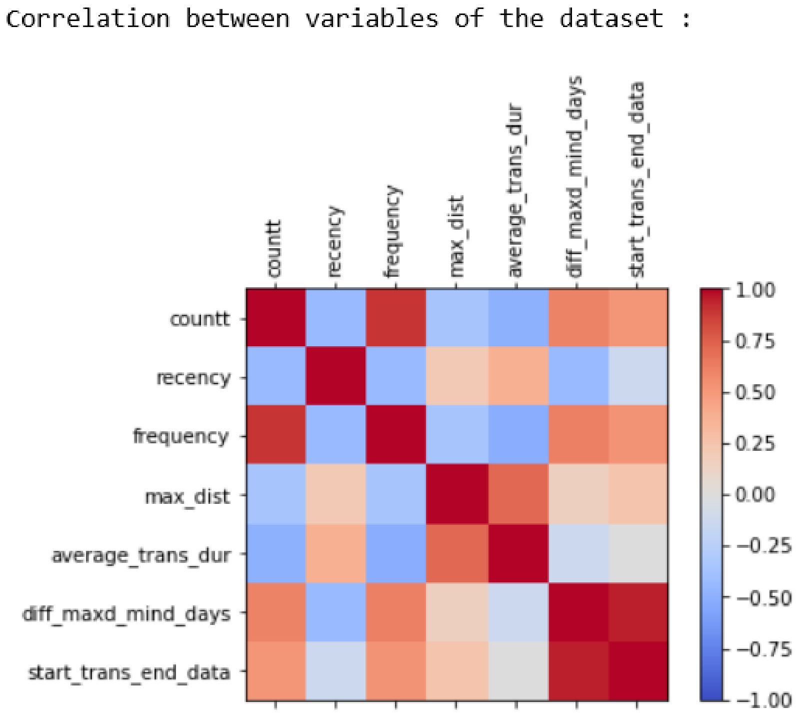
(countt - frequency) , (max_dist-average_trans_dur)

0.5 < corr < 0.7 → moderate correlation.

0.3 < corr < 0.5 → low correlation.

corr < 0.3 → little-any (linear) correlation

Figure 46: Correlation table between the variables



```
Correlation between variables of the dataset :
```

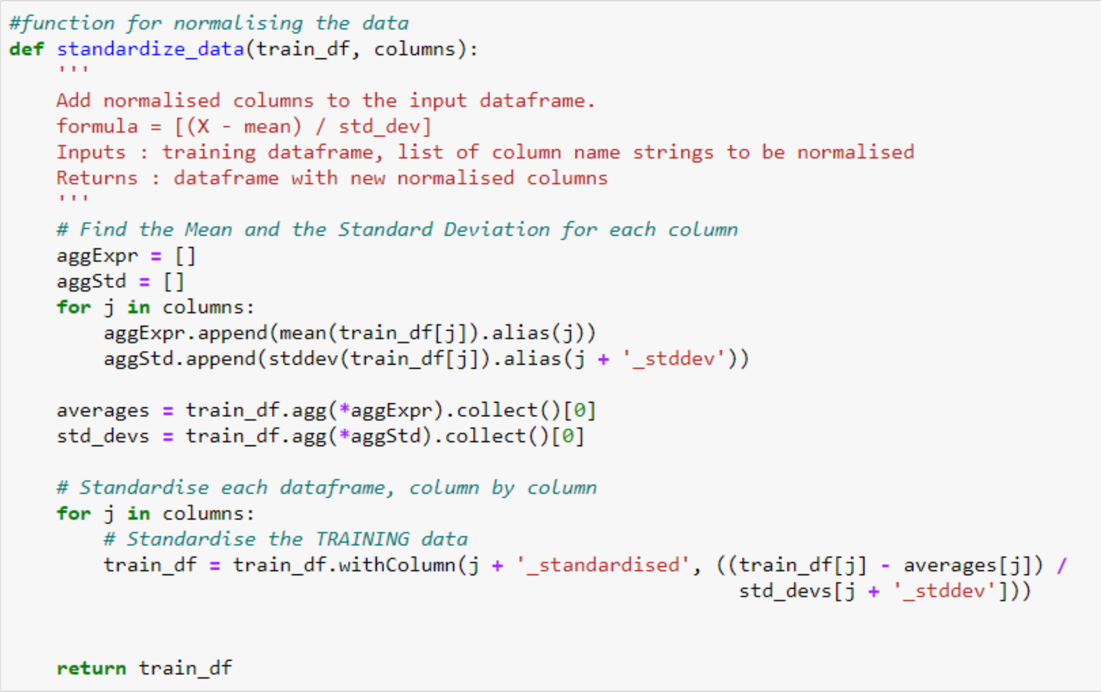
Figure 47: Correlation matrix

In the above correlation matrix (figure 47) the variables with bright red color are positively highly linearly correlated whereas the variables with bright blue color are negatively highly linearly correlated. In agreement with the previous correlated table, from correlation matrix we could see that there is a very strong positive correlation between the variables start_trans_end_data and diff_maxd_mind_days, between the countt and frequency and a mediocre strong correlation between the variables max_dist and average_trans_dur.

Diff_maxd_mind_days was very highly correlated with start_trans_end_data (corr = 0,95). But, due to the fact that start_trans_end_data was more important than the diff_maxd_mind_days for our project, we dropped the second one for all clustering phases.

4.2.3.4 Clustering

For the clustering task there were executed four (4) approaches so as to run the K-means algorithm and decide which of them satisfies best the distribution of our data. The library that was utilized for importing and running K-means was the “pyspark.ml”. Before that, as we mentioned in chapter 3 (3.5 Data mining (clustering)), the features should be comparable so as the K-means be as impartial as possible and give as good a result as possible. For this reason, it was vital the role of standardization of the variables – features. Using the bellow function (figure 48) we standardized the chosen - candidate features each time in each approach.



```
#function for normalising the data
def standardize_data(train_df, columns):
    '''
    Add normalised columns to the input dataframe.
    formula = [(X - mean) / std_dev]
    Inputs : training dataframe, list of column name strings to be normalised
    Returns : dataframe with new normalised columns
    '''
    # Find the Mean and the Standard Deviation for each column
    aggExpr = []
    aggStd = []
    for j in columns:
        aggExpr.append(mean(train_df[j]).alias(j))
        aggStd.append(stddev(train_df[j]).alias(j + '_stddev'))

    averages = train_df.agg(*aggExpr).collect()[0]
    std_devs = train_df.agg(*aggStd).collect()[0]

    # Standardise each dataframe, column by column
    for j in columns:
        # Standardise the TRAINING data
        train_df = train_df.withColumn(j + '_standardised', ((train_df[j] - averages[j]) /
                                                             std_devs[j + '_stddev']))


    return train_df
```

Figure 48: Function for standardizing spark features

Afterwards, due to the fact that Spark ML requires the input features to be gathered in a single column of the dataframe, usually named features, (StackOverflow, 2017) we had to store all candidate features as an array of floats, and store this array as a column called "features" in each time. Since we did no longer need the original columns we filtered them out with a select statement choosing the columns we are interested in. The method that used for assembling the data each time was the “VectorAssembler” from pyspark.ml.feature library.

Now, let us lay the four utilizing approaches out. The first two approaches used the log data, whereas the last two ones used the initial data. It should be mentioned that the K-means starts its operation initializing randomly centroids so, it may-will converge to a different point each time. As a result, for each of the four approaches we ran K means many times (Indicatively 3 will be presented each time), changing the number in the argument “seed()” while fitting our model.

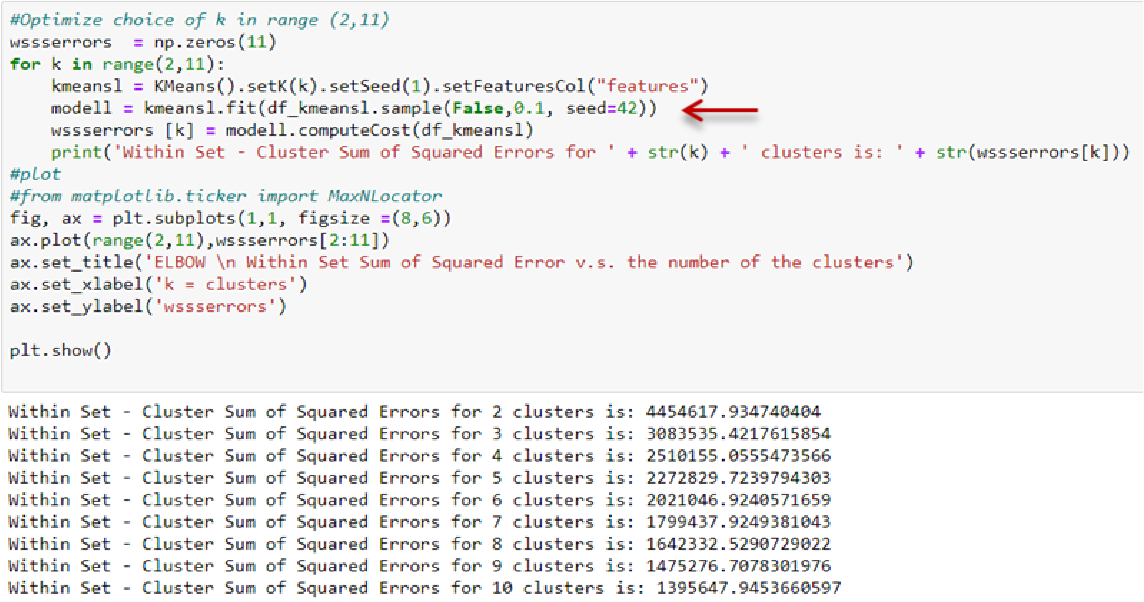
4.2.3.4.1 1st approach

(log features – no diff_maxd_mind_days)

In the first approach the following log features were used:

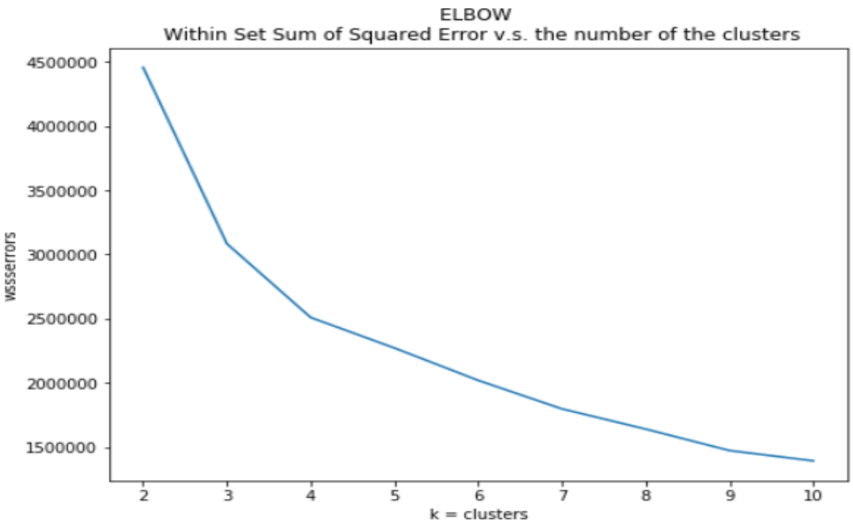
'ln(countt+1)', 'ln(recency+1)', 'ln(frequency+1)', 'ln(max_dist+1)', 'ln(average_trans_dur+1)' and 'ln(start_trans_end_data+1)'.

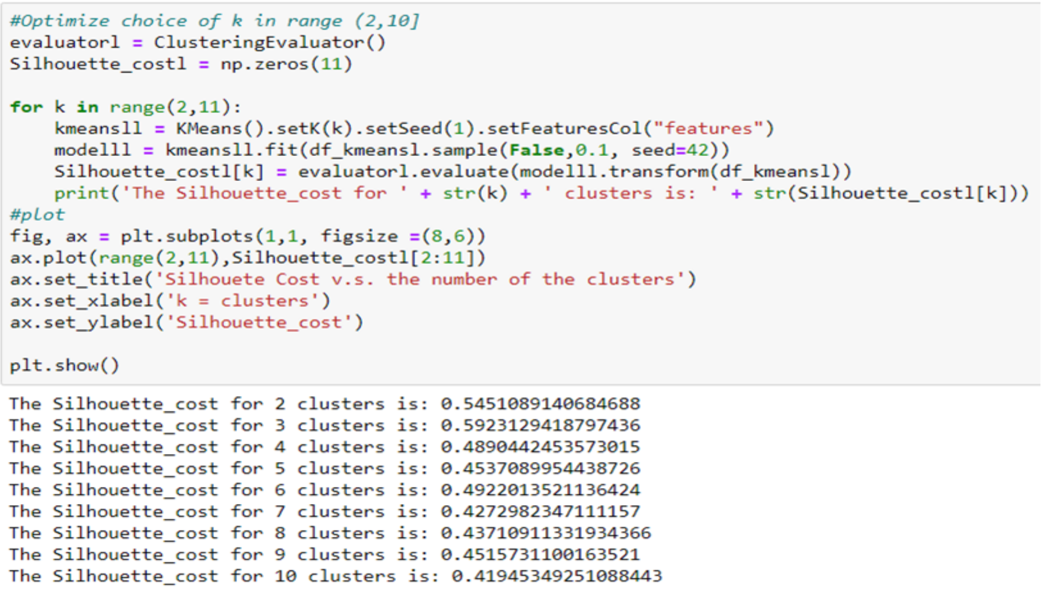
Applying the elbow method and silhouette analysis we had the following outputs:



```
#Optimize choice of k in range (2,11)
wssserrors  = np.zeros(11)
for k in range(2,11):
    kmeansl = KMeans().setK(k).setSeed(1).setFeaturesCol("features")
    modell = kmeansl.fit(df_kmeansl.sample(False,0.1, seed=42))
    wssserrors [k] = modell.computeCost(df_kmeansl)
    print('Within Set - Cluster Sum of Squared Errors for ' + str(k) + ' clusters is: ' + str(wssserrors[k]))
#plot
#from matplotlib.ticker import MaxNLocator
fig, ax = plt.subplots(1,1, figsize =(8,6))
ax.plot(range(2,11),wssserrors[2:11])
ax.set_title('ELBOW \n Within Set Sum of Squared Error v.s. the number of the clusters')
ax.set_xlabel('k = clusters')
ax.set_ylabel('wssserrors')

plt.show()
```

```
Within Set - Cluster Sum of Squared Errors for 2 clusters is: 4454617.934740404
Within Set - Cluster Sum of Squared Errors for 3 clusters is: 3083535.4217615854
Within Set - Cluster Sum of Squared Errors for 4 clusters is: 2510155.0555473566
Within Set - Cluster Sum of Squared Errors for 5 clusters is: 2272829.7239794303
Within Set - Cluster Sum of Squared Errors for 6 clusters is: 2021046.9240571659
Within Set - Cluster Sum of Squared Errors for 7 clusters is: 1799437.9249381043
Within Set - Cluster Sum of Squared Errors for 8 clusters is: 1642332.5290729022
Within Set - Cluster Sum of Squared Errors for 9 clusters is: 1475276.7078301976
Within Set - Cluster Sum of Squared Errors for 10 clusters is: 1395647.9453660597
```





```
#Optimize choice of k in range (2,10]
evaluatorl = ClusteringEvaluator()
Silhouette_costl = np.zeros(11)

for k in range(2,11):
    kmeansll = KMeans().setK(k).setSeed(1).setFeaturesCol("features")
    modelll = kmeansll.fit(df_kmeansl.sample(False,0.1, seed=42))
    Silhouette_costl[k] = evaluatorl.evaluate(modelll.transform(df_kmeansl))
    print('The Silhouette_cost for ' + str(k) + ' clusters is: ' + str(Silhouette_costl[k]))
#plot
fig, ax = plt.subplots(1,1, figsize =(8,6))
ax.plot(range(2,11),Silhouette_costl[2:11])
ax.set_title('Silhouete Cost v.s. the number of the clusters')
ax.set_xlabel('k = clusters')
ax.set_ylabel('Silhouette_cost')

plt.show()
```

```
The Silhouette_cost for 2 clusters is: 0.5451089140684688
The Silhouette_cost for 3 clusters is: 0.5923129418797436
The Silhouette_cost for 4 clusters is: 0.4890442453573015
The Silhouette_cost for 5 clusters is: 0.4537089954438726
The Silhouette_cost for 6 clusters is: 0.4922013521136424
The Silhouette_cost for 7 clusters is: 0.4272982347111157
The Silhouette_cost for 8 clusters is: 0.43710911331934366
The Silhouette_cost for 9 clusters is: 0.4515731100163521
The Silhouette_cost for 10 clusters is: 0.41945349251088443
```

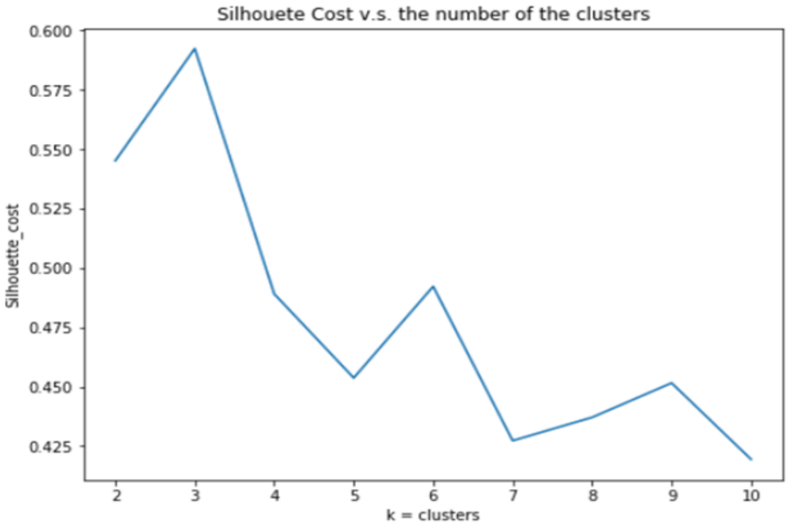
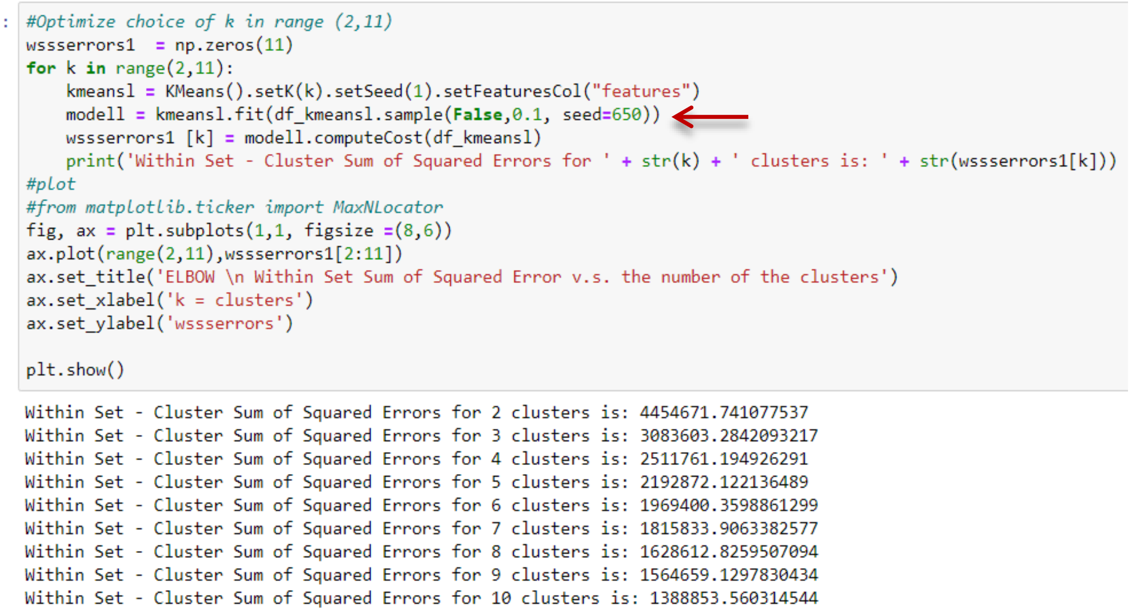


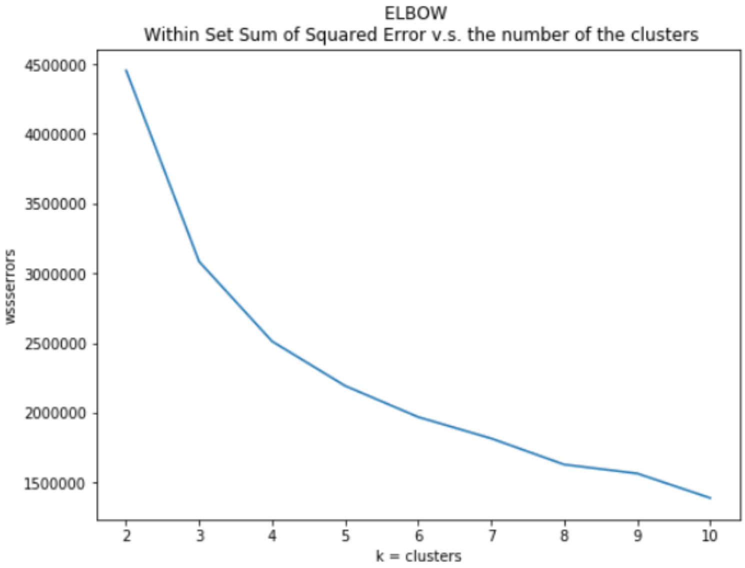
Figure 49: Elbow and Silhouette for seed =42 approach 1 log data

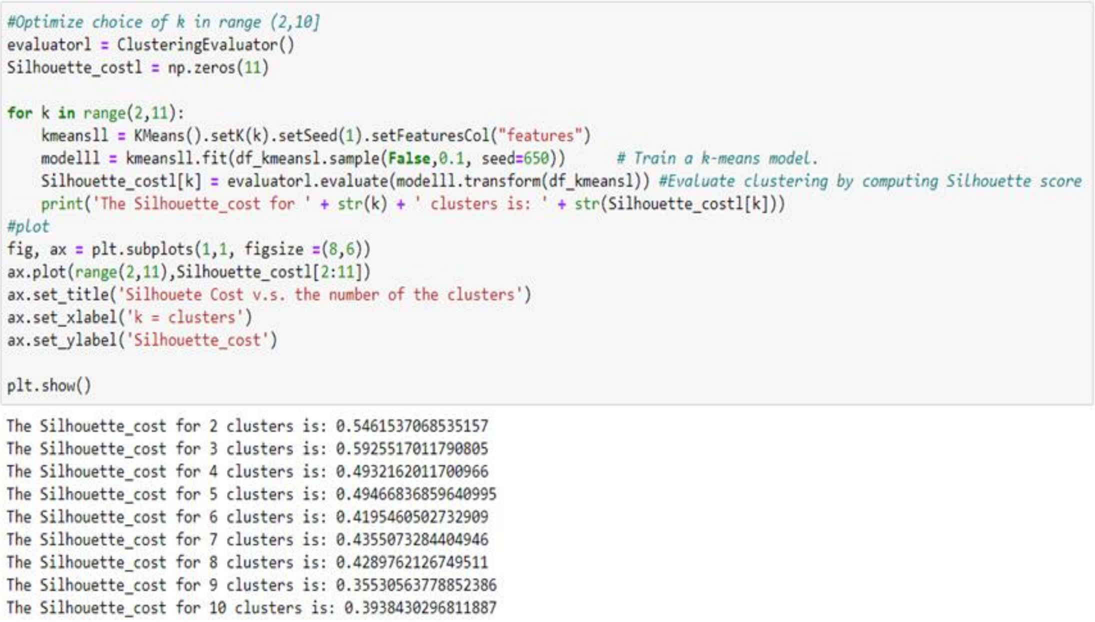
From the figure 49 (seed = 42) we could say that from the elbow the optimal number of clusters seems to be k = 3 or k = 4. From the silhouette analysis, the optimal number of clusters seems to be k = 3.



```
#Optimize choice of k in range (2,11)
wssserrors1  = np.zeros(11)
for k in range(2,11):
    kmeansl = KMeans().setK(k).setSeed(1).setFeaturesCol("features")
    modell = kmeansl.fit(df_kmeansl.sample(False,0.1, seed=650))
    wssserrors1 [k] = modell.computeCost(df_kmeansl)
    print('Within Set - Cluster Sum of Squared Errors for ' + str(k) + ' clusters is: ' + str(wssserrors1[k]))
#plot
#from matplotlib.ticker import MaxNLocator
fig, ax = plt.subplots(1,1, figsize =(8,6))
ax.plot(range(2,11),wssserrors1[2:11])
ax.set_title('ELBOW \n Within Set Sum of Squared Error v.s. the number of the clusters')
ax.set_xlabel('k = clusters')
ax.set_ylabel('wssserrors')

plt.show()
```

```
Within Set - Cluster Sum of Squared Errors for 2 clusters is: 4454671.741077537
Within Set - Cluster Sum of Squared Errors for 3 clusters is: 3083603.2842093217
Within Set - Cluster Sum of Squared Errors for 4 clusters is: 2511761.194926291
Within Set - Cluster Sum of Squared Errors for 5 clusters is: 2192872.122136489
Within Set - Cluster Sum of Squared Errors for 6 clusters is: 1969400.3598861299
Within Set - Cluster Sum of Squared Errors for 7 clusters is: 1815833.9063382577
Within Set - Cluster Sum of Squared Errors for 8 clusters is: 1628612.8259507094
Within Set - Cluster Sum of Squared Errors for 9 clusters is: 1564659.1297830434
Within Set - Cluster Sum of Squared Errors for 10 clusters is: 1388853.560314544
```





```
#Optimize choice of k in range (2,10]
evaluatorl = ClusteringEvaluator()
Silhouette_costl = np.zeros(11)

for k in range(2,11):
    kmeansll = KMeans().setK(k).setSeed(1).setFeaturesCol("features")
    modelll = kmeansll.fit(df_kmeansl.sample(False,0.1, seed=650))      # Train a k-means model.
    Silhouette_costl[k] = evaluatorl.evaluate(modelll.transform(df_kmeansl)) #Evaluate clustering by computing Silhouette score
    print('The Silhouette_cost for ' + str(k) + ' clusters is: ' + str(Silhouette_costl[k]))
#plot
fig, ax = plt.subplots(1,1, figsize =(8,6))
ax.plot(range(2,11),Silhouette_costl[2:11])
ax.set_title('Silhouete Cost v.s. the number of the clusters')
ax.set_xlabel('k = clusters')
ax.set_ylabel('Silhouette_cost')

plt.show()
```

```
The Silhouette_cost for 2 clusters is: 0.5461537068535157
The Silhouette_cost for 3 clusters is: 0.5925517011790805
The Silhouette_cost for 4 clusters is: 0.4932162011700966
The Silhouette_cost for 5 clusters is: 0.49466836859640995
The Silhouette_cost for 6 clusters is: 0.4195460502732909
The Silhouette_cost for 7 clusters is: 0.4355073284404946
The Silhouette_cost for 8 clusters is: 0.4289762126749511
The Silhouette_cost for 9 clusters is: 0.35530563778852386
The Silhouette_cost for 10 clusters is: 0.3938430296811887
```

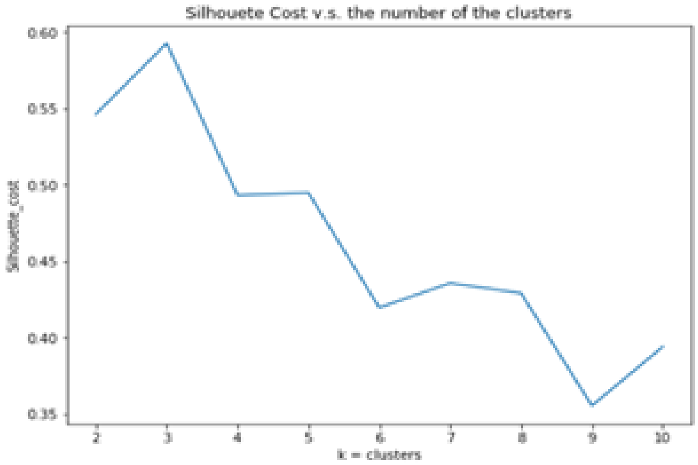
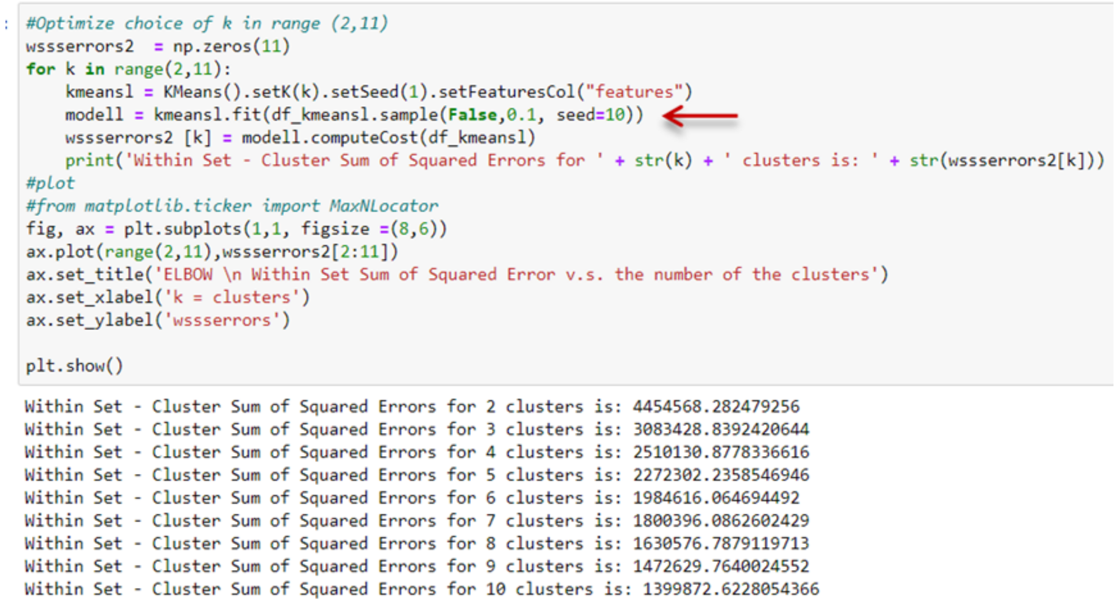


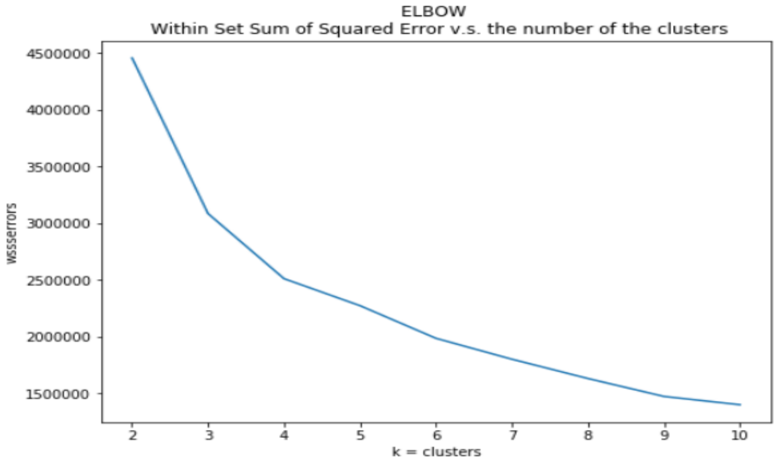
Figure 50: Elbow and Silhouette for seed = 650 approach 1 log data

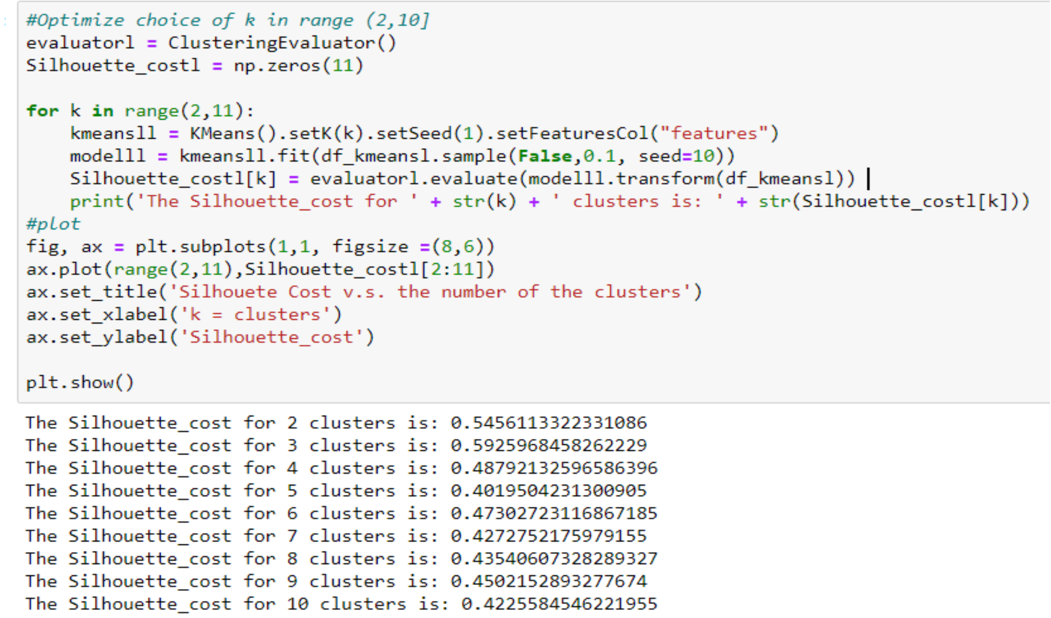
From the figure 50 (seed = 650) we could say that from the elbow the optimal number of clusters seems to be k = 3 or k = 4. From the silhouette analysis, the optimal number of clusters seems to be k = 3.



```
#Optimize choice of k in range (2,11)
wssserrors2  = np.zeros(11)
for k in range(2,11):
    kmeansl = KMeans().setK(k).setSeed(1).setFeaturesCol("features")
    modell = kmeansl.fit(df_kmeansl.sample(False,0.1, seed=10))
    wssserrors2 [k] = modell.computeCost(df_kmeansl)
    print('Within Set - Cluster Sum of Squared Errors for ' + str(k) + ' clusters is: ' + str(wssserrors2[k]))
#plot
#from matplotlib.ticker import MaxNLocator
fig, ax = plt.subplots(1,1, figsize =(8,6))
ax.plot(range(2,11),wssserrors2[2:11])
ax.set_title('ELBOW \n Within Set Sum of Squared Error v.s. the number of the clusters')
ax.set_xlabel('k = clusters')
ax.set_ylabel('wssserrors')

plt.show()
```

```
Within Set - Cluster Sum of Squared Errors for 2 clusters is: 4454568.282479256
Within Set - Cluster Sum of Squared Errors for 3 clusters is: 3083428.8392420644
Within Set - Cluster Sum of Squared Errors for 4 clusters is: 2510130.8778336616
Within Set - Cluster Sum of Squared Errors for 5 clusters is: 2272302.2358546946
Within Set - Cluster Sum of Squared Errors for 6 clusters is: 1984616.064694492
Within Set - Cluster Sum of Squared Errors for 7 clusters is: 1800396.0862602429
Within Set - Cluster Sum of Squared Errors for 8 clusters is: 1630576.7879119713
Within Set - Cluster Sum of Squared Errors for 9 clusters is: 1472629.7640024552
Within Set - Cluster Sum of Squared Errors for 10 clusters is: 1399872.6228054366
```





```
#Optimize choice of k in range (2,10]
evaluatorl = ClusteringEvaluator()
Silhouette_costl = np.zeros(11)

for k in range(2,11):
    kmeansll = KMeans().setK(k).setSeed(1).setFeaturesCol("features")
    modelll = kmeansll.fit(df_kmeansl.sample(False,0.1, seed=10))
    Silhouette_costl[k] = evaluatorl.evaluate(modelll.transform(df_kmeansl))
    print('The Silhouette_cost for ' + str(k) + ' clusters is: ' + str(Silhouette_costl[k]))
#plot
fig, ax = plt.subplots(1,1, figsize =(8,6))
ax.plot(range(2,11),Silhouette_costl[2:11])
ax.set_title('Silhouete Cost v.s. the number of the clusters')
ax.set_xlabel('k = clusters')
ax.set_ylabel('Silhouette_cost')

plt.show()
```

```
The Silhouette_cost for 2 clusters is: 0.5456113322331086
The Silhouette_cost for 3 clusters is: 0.5925968458262229
The Silhouette_cost for 4 clusters is: 0.48792132596586396
The Silhouette_cost for 5 clusters is: 0.4019504231300905
The Silhouette_cost for 6 clusters is: 0.47302723116867185
The Silhouette_cost for 7 clusters is: 0.4272752175979155
The Silhouette_cost for 8 clusters is: 0.43540607328289327
The Silhouette_cost for 9 clusters is: 0.4502152893277674
The Silhouette_cost for 10 clusters is: 0.4225584546221955
```

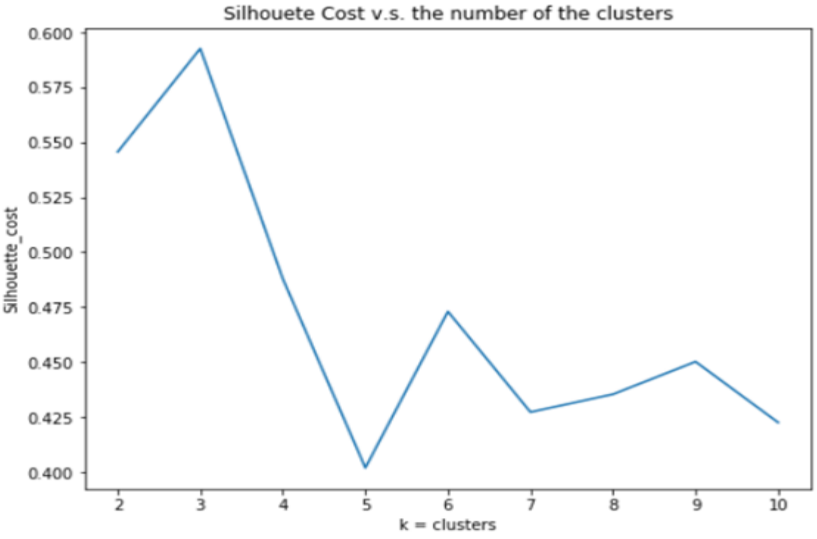


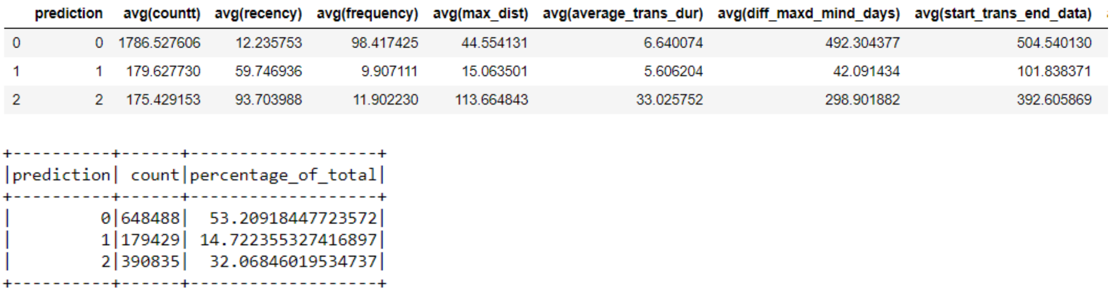
Figure 51: Elbow and Silhouette for seed = 10 approach 1 log data

From the figure 51 (seed = 10) we could say that from the elbow the optimal number of clusters seems to be k = 3 or k = 4. From the silhouette analysis, the optimal number of clusters seems to be k = 3

Considering all the above tests from Elbow and Silhouette we would say that k=3 is the optimal number of clusters. The Elbow method proposed and the value k=4 beside the k=3.

- K=3

As far as the k = 3, the best value for silhouette score achieved with seed = 10. So, we applied K-means with this specific seed and k=3. We computed the mean of all features for each prediction – type of cluster. The results were the following (figure 52):



	prediction	avg(countt)	avg(recency)	avg(frequency)	avg(max_dist)	avg(average_trans_dur)	avg(diff_maxd_mind_days)	avg(start_trans_end_data)
0	0	1786.527606	12.235753	98.417425	44.554131	6.640074	492.304377	504.540130
1	1	179.627730	59.746936	9.907111	15.063501	5.606204	42.091434	101.838371
2	2	175.429153	93.703988	11.902230	113.664843	33.025752	298.901882	392.605869

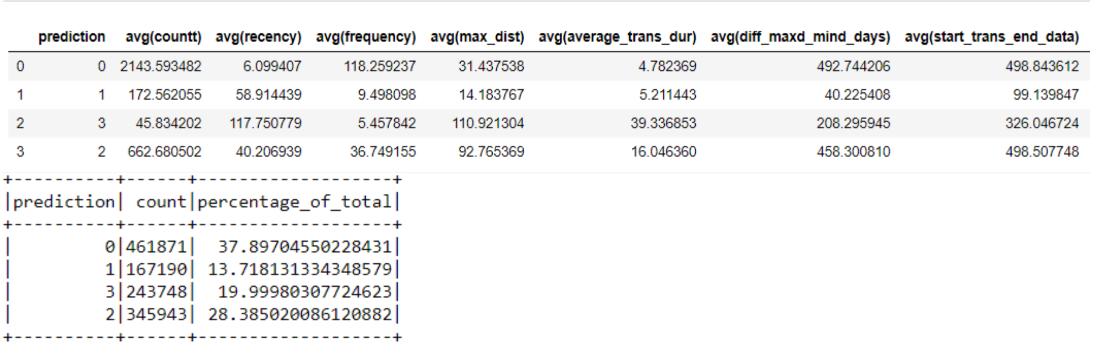
```
+----------+------+------------------+
|prediction| count|percentage_of_total|
+----------+------+------------------+
|         0|648488|  53.20918447723572|
|         1|179429| 14.722355327416897|
|         2|390835|  32.06846019534737|
+----------+------+------------------+
```

Figure 52: Descriptive elements in approach 1 and k =3

Observing the tables in figure 53 we could say that it seems that there are well separated clusters.

- K=4

For k=4 we implemented K-means with seed = 10 (it offered the smallest WSSE according to Elbow method) and although there were good enough separated clusters again, we did not accepted it due to the fact that silhouette score did not agree with it. The relevant results are the following (figure 53):



	prediction	avg(countt)	avg(recency)	avg(frequency)	avg(max_dist)	avg(average_trans_dur)	avg(diff_maxd_mind_days)	avg(start_trans_end_data)
0	0	2143.593482	6.099407	118.259237	31.437538	4.782369	492.744206	498.843612
1	1	172.562055	58.914439	9.498098	14.183767	5.211443	40.225408	99.139847
2	3	45.834202	117.750779	5.457842	110.921304	39.336853	208.295945	326.046724
3	2	662.680502	40.206939	36.749155	92.765369	16.046360	458.300810	498.507748

```
+----------+------+-------------------+
|prediction| count|percentage_of_total|
+----------+------+-------------------+
|         0|461871|  37.89704550228431|
|         1|167190| 13.718131334348579|
|         3|243748|  19.99980307724623|
|         2|345943| 28.385020086120882|
+----------+------+-------------------+
```

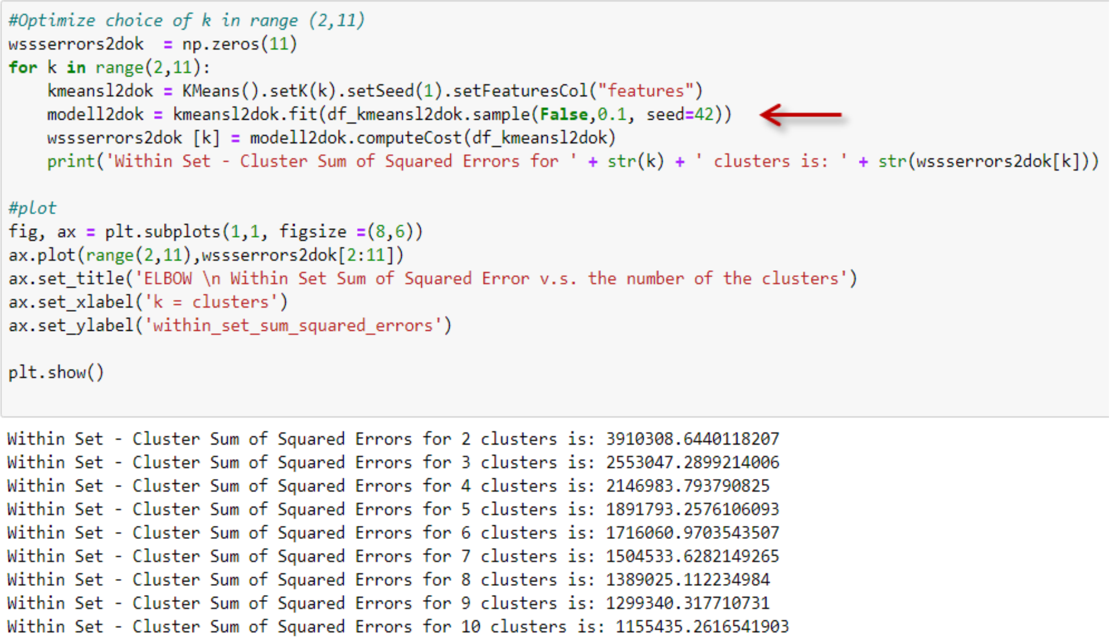
Figure 53: Descriptive elements in approach 1 and k =4

4.2.3.4.2 2nd approach

log features – no diff_maxd_mind_days and countt

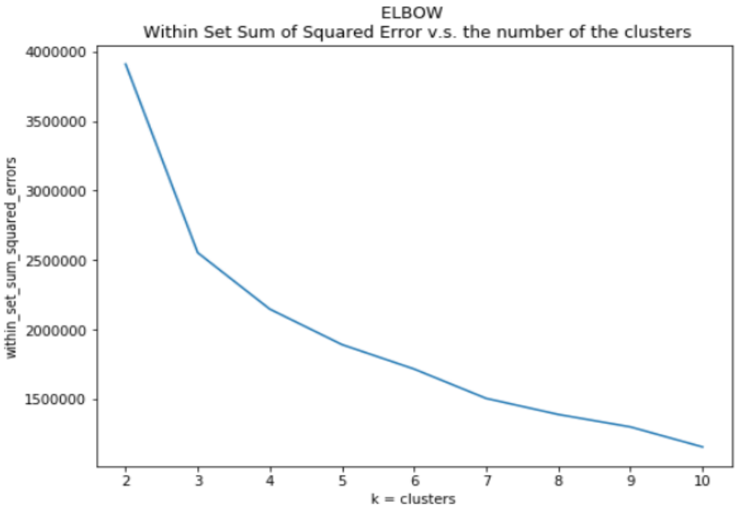
Pearson corr (countt, frequency) = 0.89 = high correlation => drop countt because frequency is more important .

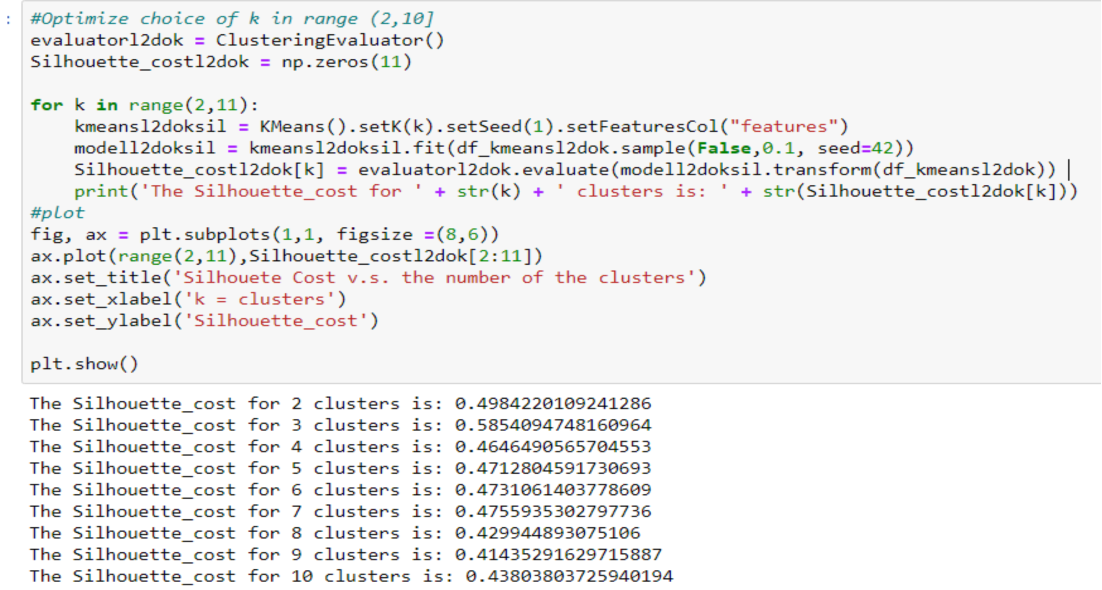
In the second approach the following log features were used: 'ln(recency+1)', 'ln(frequency+1)', 'ln(max_dist+1)', 'ln(average_trans_dur+1)' and 'ln(start_trans_end_data+1)'. Applying the elbow method and silhouette analysis we had the following outputs:



```
#Optimize choice of k in range (2,11)
wssserrors2dok  = np.zeros(11)
for k in range(2,11):
    kmeansl2dok = KMeans().setK(k).setSeed(1).setFeaturesCol("features")
    modell2dok = kmeansl2dok.fit(df_kmeansl2dok.sample(False,0.1, seed=42))
    wssserrors2dok [k] = modell2dok.computeCost(df_kmeansl2dok)
    print('Within Set - Cluster Sum of Squared Errors for ' + str(k) + ' clusters is: ' + str(wssserrors2dok[k]))

#plot
fig, ax = plt.subplots(1,1, figsize =(8,6))
ax.plot(range(2,11),wssserrors2dok[2:11])
ax.set_title('ELBOW \n Within Set Sum of Squared Error v.s. the number of the clusters')
ax.set_xlabel('k = clusters')
ax.set_ylabel('within_set_sum_squared_errors')

plt.show()
```

```
Within Set - Cluster Sum of Squared Errors for 2 clusters is: 3910308.6440118207
Within Set - Cluster Sum of Squared Errors for 3 clusters is: 2553047.2899214006
Within Set - Cluster Sum of Squared Errors for 4 clusters is: 2146983.793790825
Within Set - Cluster Sum of Squared Errors for 5 clusters is: 1891793.2576106093
Within Set - Cluster Sum of Squared Errors for 6 clusters is: 1716060.9703543507
Within Set - Cluster Sum of Squared Errors for 7 clusters is: 1504533.6282149265
Within Set - Cluster Sum of Squared Errors for 8 clusters is: 1389025.112234984
Within Set - Cluster Sum of Squared Errors for 9 clusters is: 1299340.317710731
Within Set - Cluster Sum of Squared Errors for 10 clusters is: 1155435.2616541903
```





```
#Optimize choice of k in range (2,10]
evaluatorl2dok = ClusteringEvaluator()
Silhouette_costl2dok = np.zeros(11)

for k in range(2,11):
    kmeansl2doksil = KMeans().setK(k).setSeed(1).setFeaturesCol("features")
    modell2doksil = kmeansl2doksil.fit(df_kmeansl2dok.sample(False,0.1, seed=42))
    Silhouette_costl2dok[k] = evaluatorl2dok.evaluate(modell2doksil.transform(df_kmeansl2dok))
    print('The Silhouette_cost for ' + str(k) + ' clusters is: ' + str(Silhouette_costl2dok[k]))
#plot
fig, ax = plt.subplots(1,1, figsize =(8,6))
ax.plot(range(2,11),Silhouette_costl2dok[2:11])
ax.set_title('Silhouete Cost v.s. the number of the clusters')
ax.set_xlabel('k = clusters')
ax.set_ylabel('Silhouette_cost')

plt.show()
```

```
The Silhouette_cost for 2 clusters is: 0.4984220109241286
The Silhouette_cost for 3 clusters is: 0.5854094748160964
The Silhouette_cost for 4 clusters is: 0.4646490565704553
The Silhouette_cost for 5 clusters is: 0.4712804591730693
The Silhouette_cost for 6 clusters is: 0.4731061403778609
The Silhouette_cost for 7 clusters is: 0.4755935302797736
The Silhouette_cost for 8 clusters is: 0.429944893075106
The Silhouette_cost for 9 clusters is: 0.41435291629715887
The Silhouette_cost for 10 clusters is: 0.43803803725940194
```

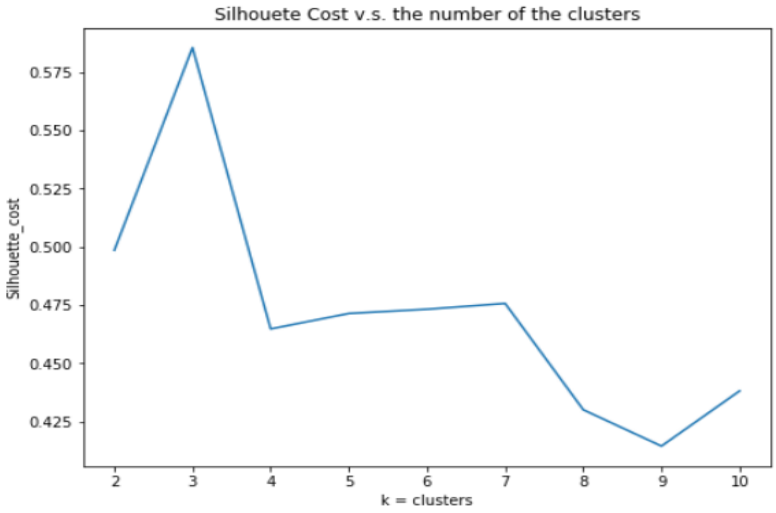
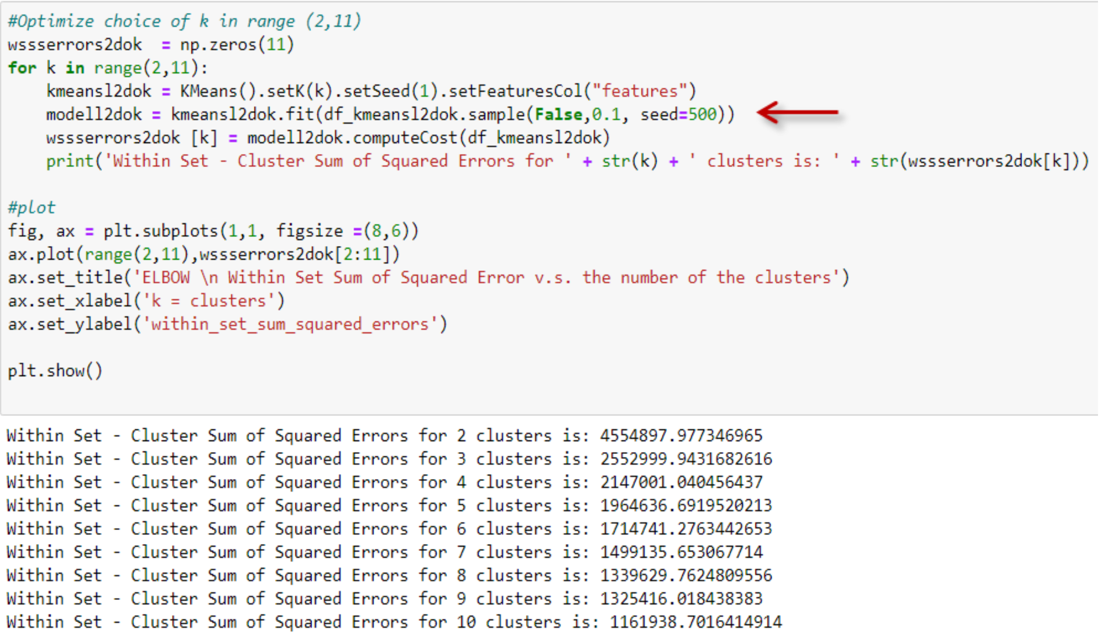


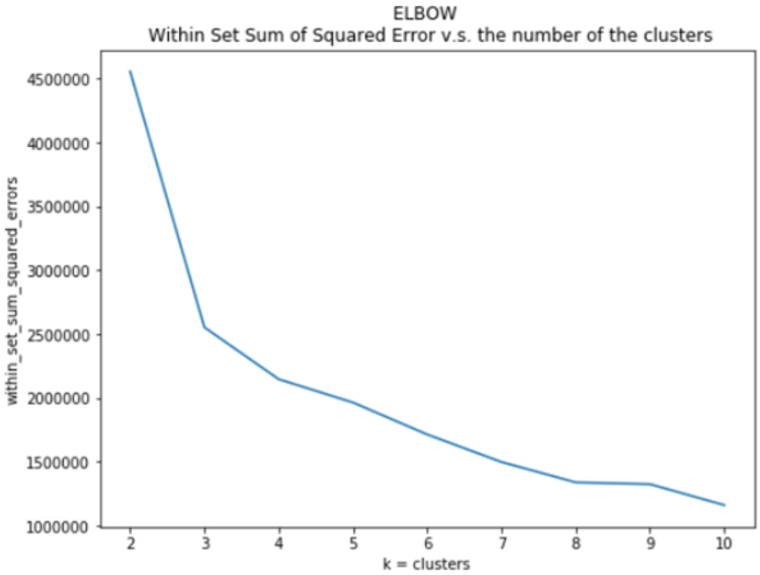
Figure 54: Elbow and Silhouette for seed =42 approach 2 log data

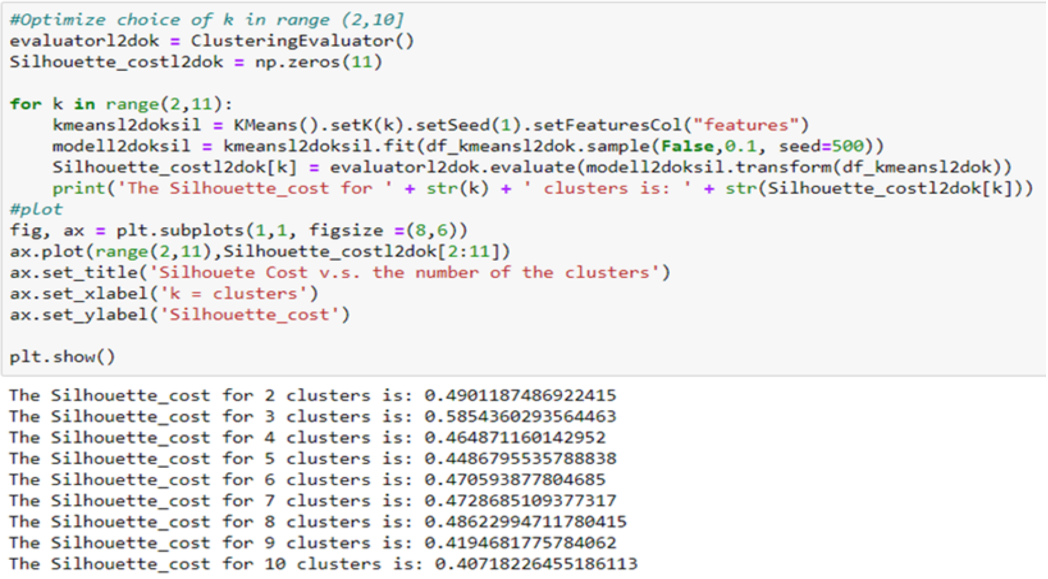
From the figure 54 (seed = 42) we could say that from the elbow the optimal number of clusters seems to be k = 3. From the silhouette analysis, the optimal number of clusters seems to be k = 3.



```
#Optimize choice of k in range (2,11)
wssserrors2dok  = np.zeros(11)
for k in range(2,11):
    kmeansl2dok = KMeans().setK(k).setSeed(1).setFeaturesCol("features")
    modell2dok = kmeansl2dok.fit(df_kmeansl2dok.sample(False,0.1, seed=500))
    wssserrors2dok [k] = modell2dok.computeCost(df_kmeansl2dok)
    print('Within Set - Cluster Sum of Squared Errors for ' + str(k) + ' clusters is: ' + str(wssserrors2dok[k]))

#plot
fig, ax = plt.subplots(1,1, figsize =(8,6))
ax.plot(range(2,11),wssserrors2dok[2:11])
ax.set_title('ELBOW \n Within Set Sum of Squared Error v.s. the number of the clusters')
ax.set_xlabel('k = clusters')
ax.set_ylabel('within_set_sum_squared_errors')

plt.show()
```

```
Within Set - Cluster Sum of Squared Errors for 2 clusters is: 4554897.977346965
Within Set - Cluster Sum of Squared Errors for 3 clusters is: 2552999.9431682616
Within Set - Cluster Sum of Squared Errors for 4 clusters is: 2147001.040456437
Within Set - Cluster Sum of Squared Errors for 5 clusters is: 1964636.6919520213
Within Set - Cluster Sum of Squared Errors for 6 clusters is: 1714741.2763442653
Within Set - Cluster Sum of Squared Errors for 7 clusters is: 1499135.653067714
Within Set - Cluster Sum of Squared Errors for 8 clusters is: 1339629.7624809556
Within Set - Cluster Sum of Squared Errors for 9 clusters is: 1325416.018438383
Within Set - Cluster Sum of Squared Errors for 10 clusters is: 1161938.7016414914
```





```
#Optimize choice of k in range (2,10]
evaluatorl2dok = ClusteringEvaluator()
Silhouette_costl2dok = np.zeros(11)

for k in range(2,11):
    kmeansl2doksil = KMeans().setK(k).setSeed(1).setFeaturesCol("features")
    modell2doksil = kmeansl2doksil.fit(df_kmeansl2dok.sample(False,0.1, seed=500))
    Silhouette_costl2dok[k] = evaluatorl2dok.evaluate(modell2doksil.transform(df_kmeansl2dok))
    print('The Silhouette_cost for ' + str(k) + ' clusters is: ' + str(Silhouette_costl2dok[k]))
#plot
fig, ax = plt.subplots(1,1, figsize =(8,6))
ax.plot(range(2,11),Silhouette_costl2dok[2:11])
ax.set_title('Silhouete Cost v.s. the number of the clusters')
ax.set_xlabel('k = clusters')
ax.set_ylabel('Silhouette_cost')

plt.show()
```

```
The Silhouette_cost for 2 clusters is: 0.4901187486922415
The Silhouette_cost for 3 clusters is: 0.5854360293564463
The Silhouette_cost for 4 clusters is: 0.464871160142952
The Silhouette_cost for 5 clusters is: 0.4486795535788838
The Silhouette_cost for 6 clusters is: 0.470593877804685
The Silhouette_cost for 7 clusters is: 0.4728685109377317
The Silhouette_cost for 8 clusters is: 0.48622994711780415
The Silhouette_cost for 9 clusters is: 0.4194681775784062
The Silhouette_cost for 10 clusters is: 0.40718226455186113
```

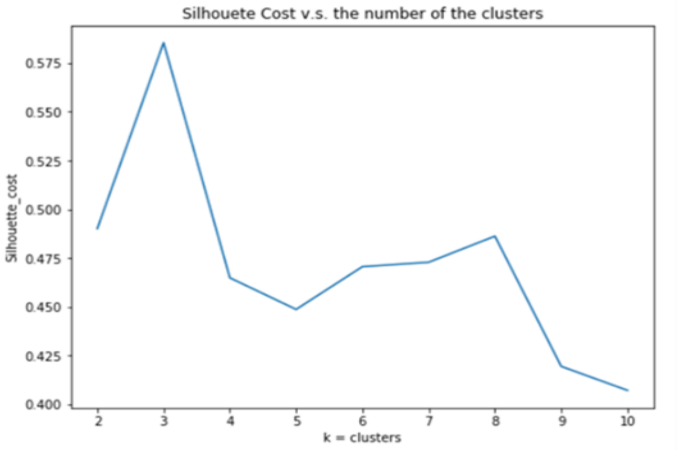
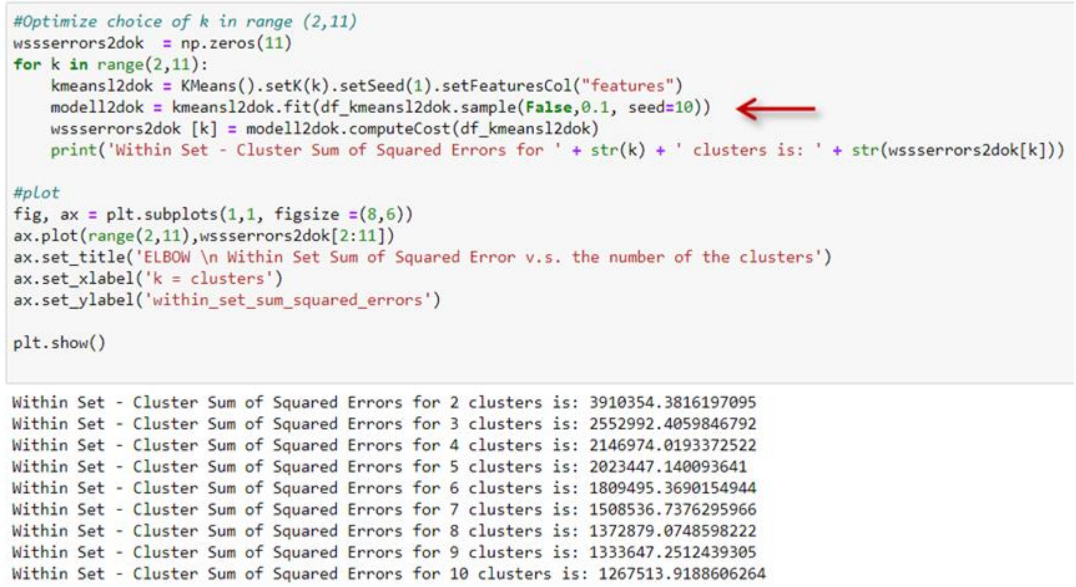


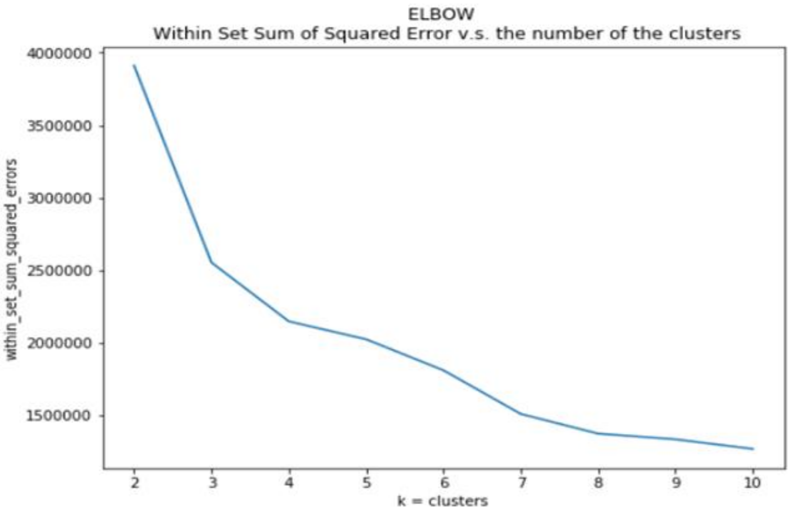
Figure 55: Elbow and Silhouette for seed =500 approach 2 log data

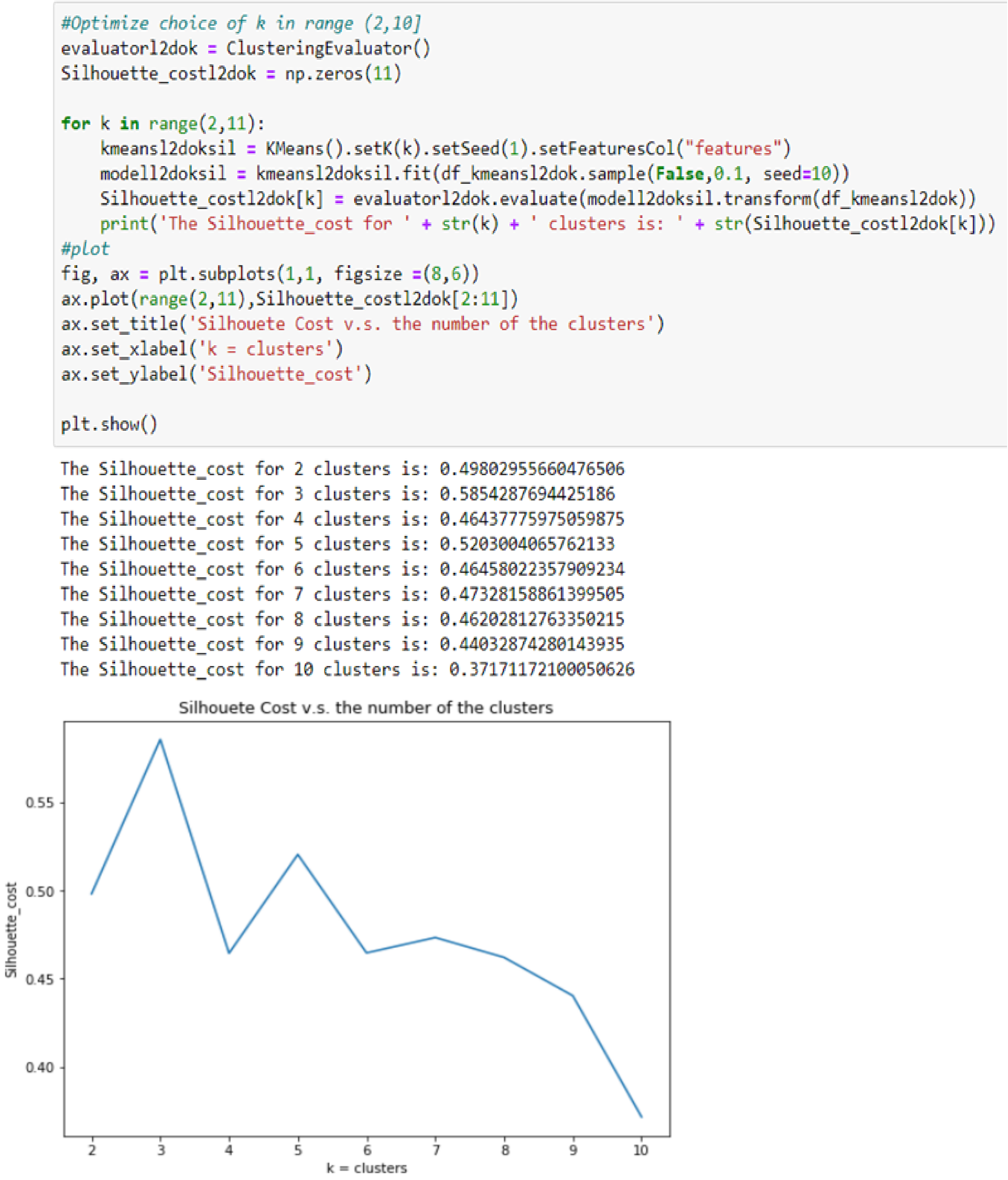
From the figure 55 (seed = 500) we could say that from the elbow the optimal number of clusters seems to be k = 3 or k = 4. From the silhouette analysis, the optimal number of clusters seems to be k = 3.



```
#Optimize choice of k in range (2,11)
wssserrors2dok  = np.zeros(11)
for k in range(2,11):
    kmeansl2dok = KMeans().setK(k).setSeed(1).setFeaturesCol("features")
    modell2dok = kmeansl2dok.fit(df_kmeansl2dok.sample(False,0.1, seed=10))
    wssserrors2dok [k] = modell2dok.computeCost(df_kmeansl2dok)
    print('Within Set - Cluster Sum of Squared Errors for ' + str(k) + ' clusters is: ' + str(wssserrors2dok[k]))

#plot
fig, ax = plt.subplots(1,1, figsize =(8,6))
ax.plot(range(2,11),wssserrors2dok[2:11])
ax.set_title('ELBOW \n Within Set Sum of Squared Error v.s. the number of the clusters')
ax.set_xlabel('k = clusters')
ax.set_ylabel('within_set_sum_squared_errors')

plt.show()
```

```
Within Set - Cluster Sum of Squared Errors for 2 clusters is: 3910354.3816197095
Within Set - Cluster Sum of Squared Errors for 3 clusters is: 2552992.4059846792
Within Set - Cluster Sum of Squared Errors for 4 clusters is: 2146974.0193372522
Within Set - Cluster Sum of Squared Errors for 5 clusters is: 2023447.140093641
Within Set - Cluster Sum of Squared Errors for 6 clusters is: 1809495.3690154944
Within Set - Cluster Sum of Squared Errors for 7 clusters is: 1508536.7376295966
Within Set - Cluster Sum of Squared Errors for 8 clusters is: 1372879.0748598222
Within Set - Cluster Sum of Squared Errors for 9 clusters is: 1333647.2512439305
Within Set - Cluster Sum of Squared Errors for 10 clusters is: 1267513.9188606264
```





```
#Optimize choice of k in range (2,10]
evaluatorl2dok = ClusteringEvaluator()
Silhouette_costl2dok = np.zeros(11)

for k in range(2,11):
    kmeansl2doksil = KMeans().setK(k).setSeed(1).setFeaturesCol("features")
    modell2doksil = kmeansl2doksil.fit(df_kmeansl2dok.sample(False,0.1, seed=10))
    Silhouette_costl2dok[k] = evaluatorl2dok.evaluate(modell2doksil.transform(df_kmeansl2dok))
    print('The Silhouette_cost for ' + str(k) + ' clusters is: ' + str(Silhouette_costl2dok[k]))
#plot
fig, ax = plt.subplots(1,1, figsize =(8,6))
ax.plot(range(2,11),Silhouette_costl2dok[2:11])
ax.set_title('Silhouete Cost v.s. the number of the clusters')
ax.set_xlabel('k = clusters')
ax.set_ylabel('Silhouette_cost')

plt.show()
```

```
The Silhouette_cost for 2 clusters is: 0.49802955660476506
The Silhouette_cost for 3 clusters is: 0.5854287694425186
The Silhouette_cost for 4 clusters is: 0.46437775975059875
The Silhouette_cost for 5 clusters is: 0.5203004065762133
The Silhouette_cost for 6 clusters is: 0.46458022357909234
The Silhouette_cost for 7 clusters is: 0.47328158861399505
The Silhouette_cost for 8 clusters is: 0.46202812763350215
The Silhouette_cost for 9 clusters is: 0.44032874280143935
The Silhouette_cost for 10 clusters is: 0.37171172100050626
```

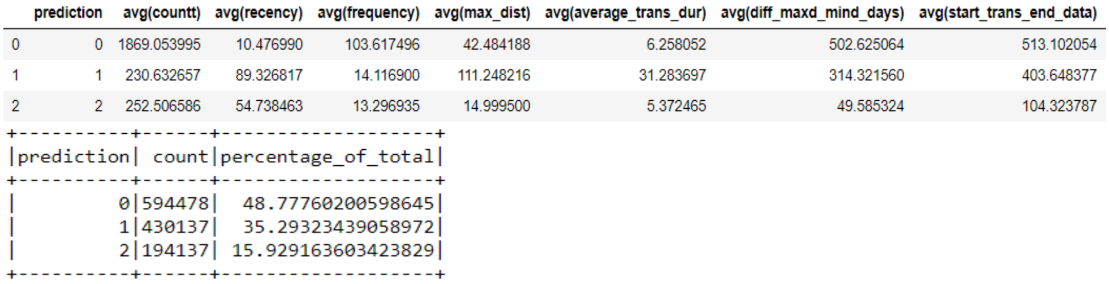
Figure 56: Elbow and Silhouette for seed =10 approach 2 log data

From the figure 56 (seed = 10) we could say that from the elbow the optimal number of clusters seems to be k = 3 or k = 4. From the silhouette analysis, the optimal number of clusters seems to be k = 3.

Considering all the above tests from Elbow and Silhouette we would say that k=3 is the common optimal number of clusters. The Elbow method proposed beside the k = 3 and the value k=4 as a candidate optimal number of clusters.

- K=3

As far as the k = 3, the best value for silhouette score achieved with seed = 500. So, we applied K-means with this specific seed and k=3 and we computed the mean of all features for each prediction – type of cluster. The results were the following (figure 57):



	prediction	avg(countt)	avg(recency)	avg(frequency)	avg(max_dist)	avg(average_trans_dur)	avg(diff_maxd_mind_days)	avg(start_trans_end_data)
0	0	1869.053995	10.476990	103.617496	42.484188	6.258052	502.625064	513.102054
1	1	230.632657	89.326817	14.116900	111.248216	31.283697	314.321560	403.648377
2	2	252.506586	54.738463	13.296935	14.999500	5.372465	49.585324	104.323787

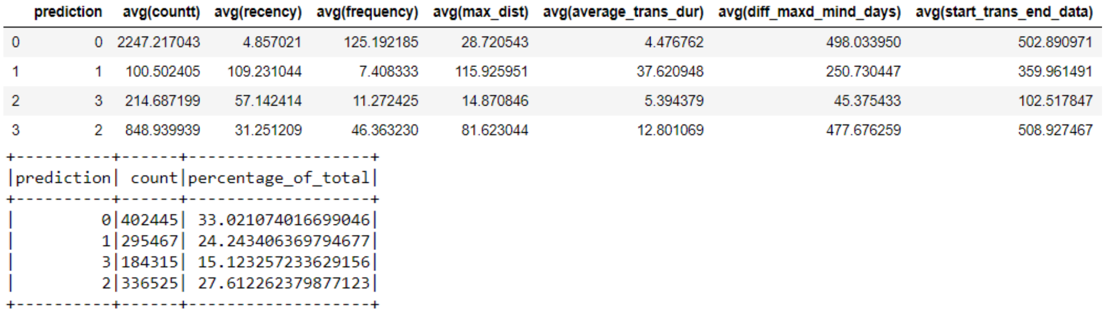
```
+----------+------+------------------+
|prediction| count|percentage_of_total|
+----------+------+------------------+
|         0|594478|  48.7776020059864 5|
|         1|430137|  35.29323439058972|
|         2|194137| 15.92916360342382 9|
+----------+------+------------------+
```

Figure 57: Descriptive elements in approach 2 and k =3

Observing the tables in figure 57 we could say that it seems there are well separated clusters.

- K=4

For k=4 we implemented K-means with seed = 10 (it offered the smallest WSSE according to Elbow method) but although there were good enough separated clusters again, we did not accepted it due to the fact that silhouette score did not agree with it. The relevant results are the following (figure 58):



	prediction	avg(countt)	avg(recency)	avg(frequency)	avg(max_dist)	avg(average_trans_dur)	avg(diff_maxd_mind_days)	avg(start_trans_end_data)
0	0	2247.217043	4.857021	125.192185	28.720543	4.476762	498.033950	502.890971
1	1	100.502405	109.231044	7.408333	115.925951	37.620948	250.730447	359.961491
2	3	214.687199	57.142414	11.272425	14.870846	5.394379	45.375433	102.517847
3	2	848.939939	31.251209	46.363230	81.623044	12.801069	477.676259	508.927467

```
+----------+------+------------------+
|prediction| count|percentage_of_total|
+----------+------+------------------+
|         0|402445| 33.021074016699046|
|         1|295467| 24.243406369794677|
|         3|184315| 15.123257233629156|
|         2|336525| 27.612262379877123|
+----------+------+------------------+
```

Figure 58: Descriptive elements in approach 2 and k =4

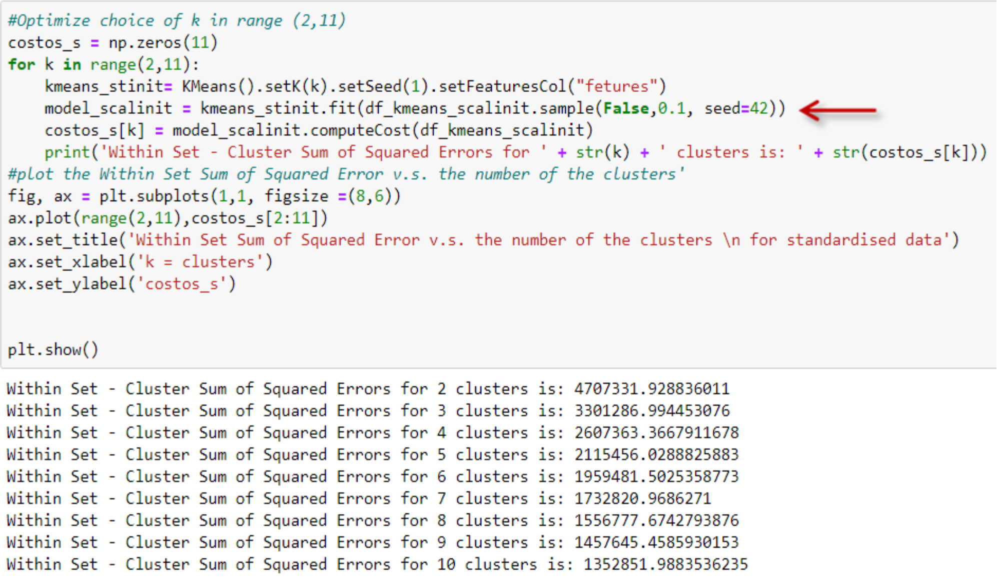
4.2.3.4.3 3rd approach

initial features – no diff_maxd_mind_days

In the third approach the following features were used:

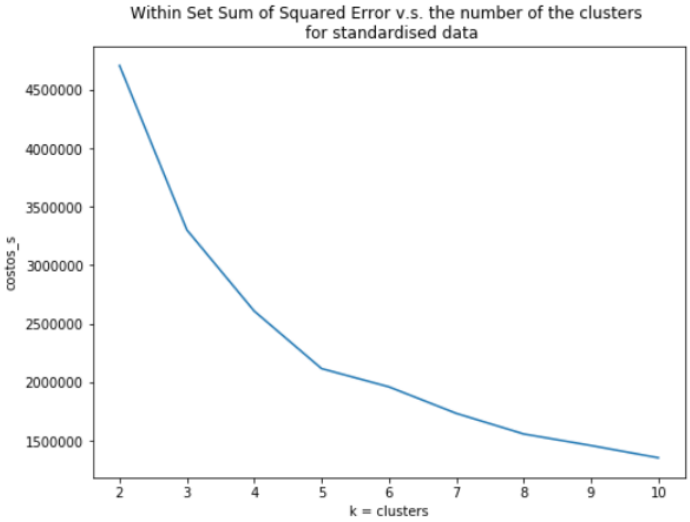
'countt', 'recency' ,'frequency', 'max_dist', 'average_trans_dur', 'start_trans_end_data'

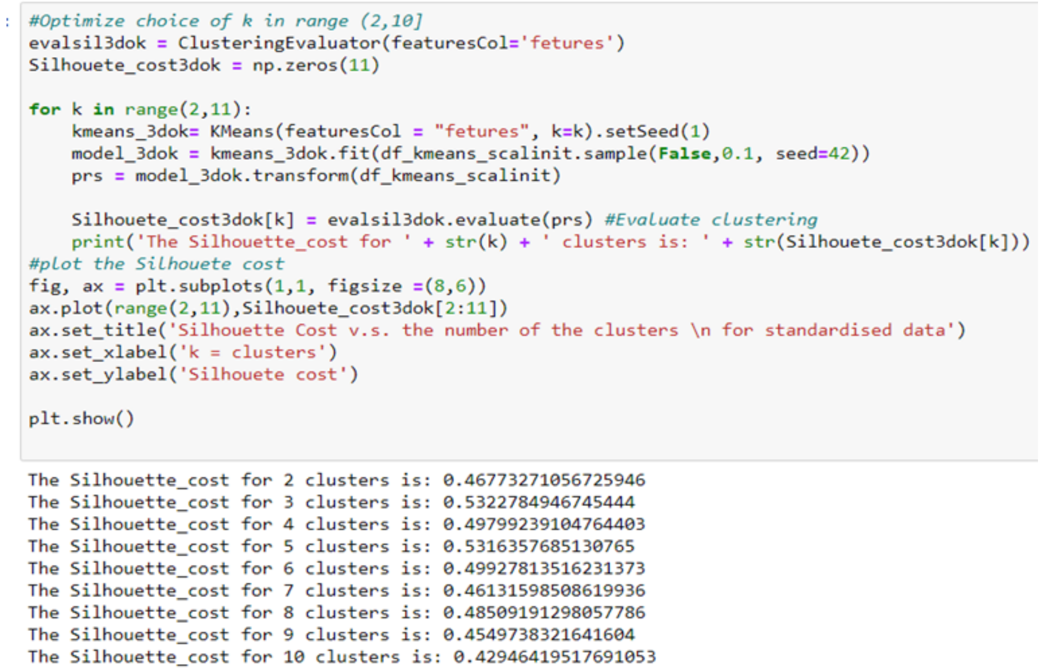
Applying the elbow method and silhouette analysis we had the following outputs:



```
#Optimize choice of k in range (2,11)
costos_s = np.zeros(11)
for k in range(2,11):
    kmeans_stinit= KMeans().setK(k).setSeed(1).setFeaturesCol("fetures")
    model_scalinit = kmeans_stinit.fit(df_kmeans_scalinit.sample(False,0.1, seed=42))
    costos_s[k] = model_scalinit.computeCost(df_kmeans_scalinit)
    print('Within Set - Cluster Sum of Squared Errors for ' + str(k) + ' clusters is: ' + str(costos_s[k]))
#plot the Within Set Sum of Squared Error v.s. the number of the clusters'
fig, ax = plt.subplots(1,1, figsize =(8,6))
ax.plot(range(2,11),costos_s[2:11])
ax.set_title('Within Set Sum of Squared Error v.s. the number of the clusters \n for standardised data')
ax.set_xlabel('k = clusters')
ax.set_ylabel('costos_s')


plt.show()
```

```
Within Set - Cluster Sum of Squared Errors for 2 clusters is: 4707331.928836011
Within Set - Cluster Sum of Squared Errors for 3 clusters is: 3301286.994453076
Within Set - Cluster Sum of Squared Errors for 4 clusters is: 2607363.3667911678
Within Set - Cluster Sum of Squared Errors for 5 clusters is: 2115456.0288825883
Within Set - Cluster Sum of Squared Errors for 6 clusters is: 1959481.5025358773
Within Set - Cluster Sum of Squared Errors for 7 clusters is: 1732820.9686271
Within Set - Cluster Sum of Squared Errors for 8 clusters is: 1556777.6742793876
Within Set - Cluster Sum of Squared Errors for 9 clusters is: 1457645.4585930153
Within Set - Cluster Sum of Squared Errors for 10 clusters is: 1352851.9883536235
```





```
#Optimize choice of k in range (2,10]
evalsil3dok = ClusteringEvaluator(featuresCol='fetures')
Silhouete_cost3dok = np.zeros(11)

for k in range(2,11):
    kmeans_3dok= KMeans(featuresCol = "fetures", k=k).setSeed(1)
    model_3dok = kmeans_3dok.fit(df_kmeans_scalinit.sample(False,0.1, seed=42))
    prs = model_3dok.transform(df_kmeans_scalinit)

    Silhouete_cost3dok[k] = evalsil3dok.evaluate(prs) #Evaluate clustering
    print('The Silhouette_cost for ' + str(k) + ' clusters is: ' + str(Silhouete_cost3dok[k]))
#plot the Silhouete cost
fig, ax = plt.subplots(1,1, figsize =(8,6))
ax.plot(range(2,11),Silhouete_cost3dok[2:11])
ax.set_title('Silhouette Cost v.s. the number of the clusters \n for standardised data')
ax.set_xlabel('k = clusters')
ax.set_ylabel('Silhouete cost')

plt.show()
```

```
The Silhouette_cost for 2 clusters is: 0.46773271056725946
The Silhouette_cost for 3 clusters is: 0.5322784946745444
The Silhouette_cost for 4 clusters is: 0.49799239104764403
The Silhouette_cost for 5 clusters is: 0.5316357685130765
The Silhouette_cost for 6 clusters is: 0.49927813516231373
The Silhouette_cost for 7 clusters is: 0.46131598508619936
The Silhouette_cost for 8 clusters is: 0.48509191298057786
The Silhouette_cost for 9 clusters is: 0.4549738321641604
The Silhouette_cost for 10 clusters is: 0.42946419517691053
```

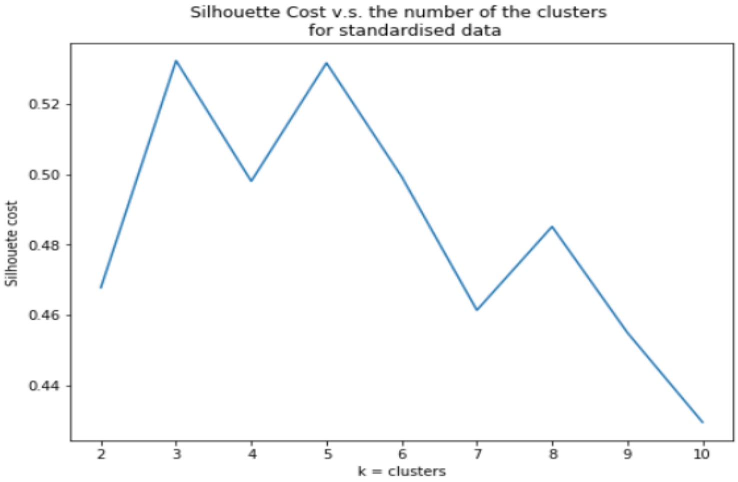
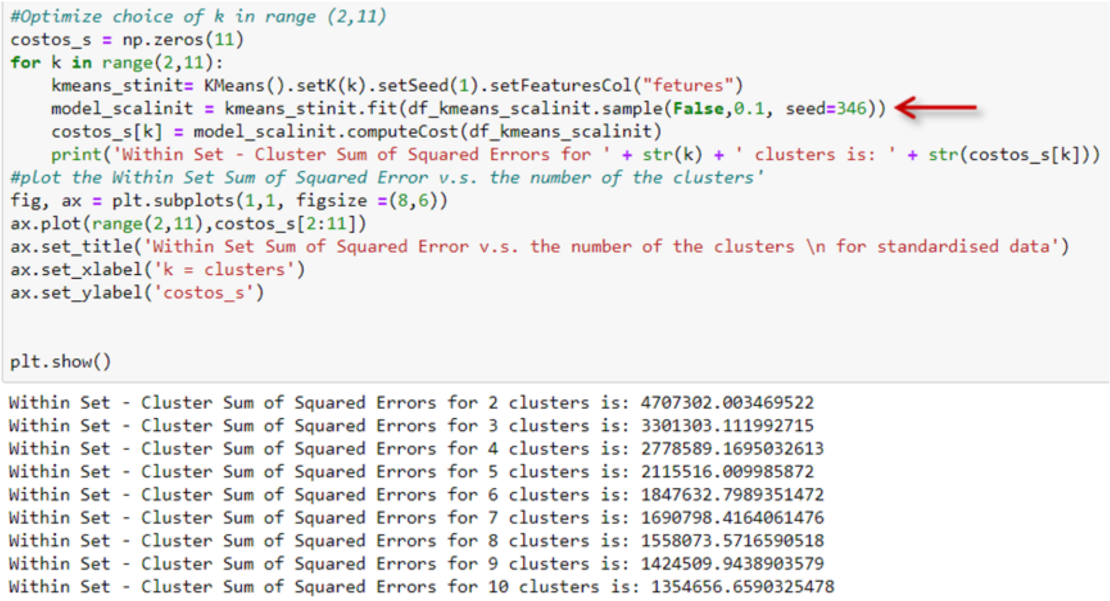


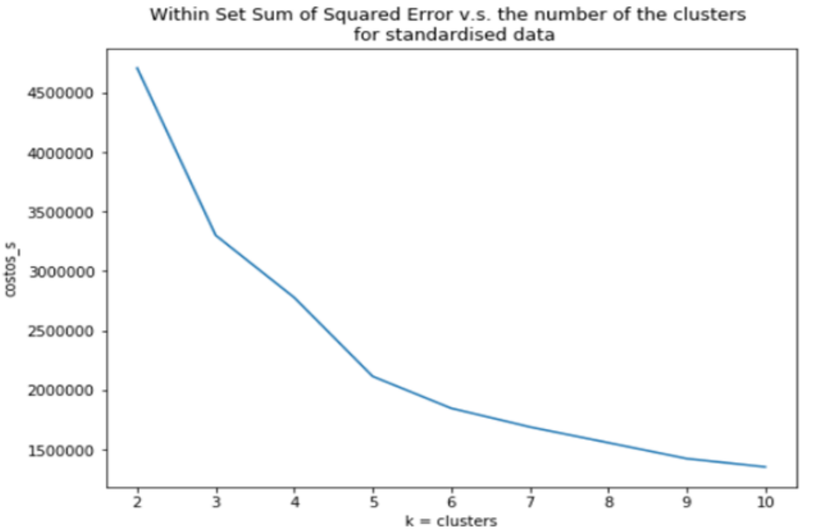
Figure 59: Elbow and Silhouette for seed =42 approach 3 initial data

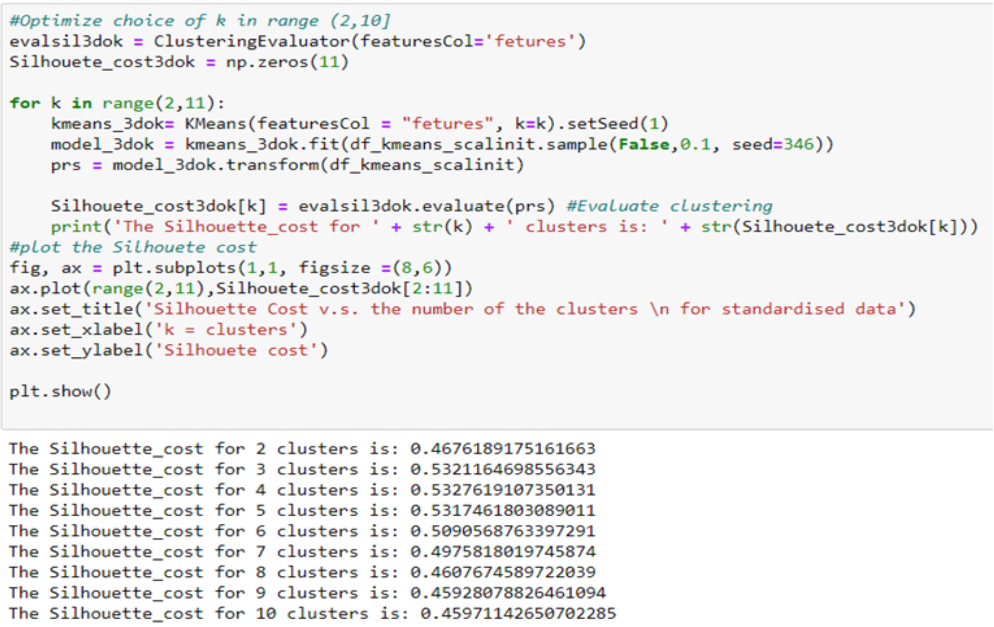
From the figure 59 (seed = 42) we could say that from the elbow method the optimal number of clusters seems to be k = 3 or k=4 or k = 5. From the silhouette analysis, the optimal number of clusters seems to be k = 3 or k = 5.



```
#Optimize choice of k in range (2,11)
costos_s = np.zeros(11)
for k in range(2,11):
    kmeans_stinit= KMeans().setK(k).setSeed(1).setFeaturesCol("fetures")
    model_scalinit = kmeans_stinit.fit(df_kmeans_scalinit.sample(False,0.1, seed=346))
    costos_s[k] = model_scalinit.computeCost(df_kmeans_scalinit)
    print('Within Set - Cluster Sum of Squared Errors for ' + str(k) + ' clusters is: ' + str(costos_s[k]))
#plot the Within Set Sum of Squared Error v.s. the number of the clusters'
fig, ax = plt.subplots(1,1, figsize =(8,6))
ax.plot(range(2,11),costos_s[2:11])
ax.set_title('Within Set Sum of Squared Error v.s. the number of the clusters \n for standardised data')
ax.set_xlabel('k = clusters')
ax.set_ylabel('costos_s')


plt.show()
```

```
Within Set - Cluster Sum of Squared Errors for 2 clusters is: 4707302.003469522
Within Set - Cluster Sum of Squared Errors for 3 clusters is: 3301303.111992715
Within Set - Cluster Sum of Squared Errors for 4 clusters is: 2778589.1695032613
Within Set - Cluster Sum of Squared Errors for 5 clusters is: 2115516.009985872
Within Set - Cluster Sum of Squared Errors for 6 clusters is: 1847632.7989351472
Within Set - Cluster Sum of Squared Errors for 7 clusters is: 1690798.4164061476
Within Set - Cluster Sum of Squared Errors for 8 clusters is: 1558073.5716590518
Within Set - Cluster Sum of Squared Errors for 9 clusters is: 1424509.9438903579
Within Set - Cluster Sum of Squared Errors for 10 clusters is: 1354656.6590325478
```





```
#Optimize choice of k in range (2,10]
evalsil3dok = ClusteringEvaluator(featuresCol='fetures')
Silhouete_cost3dok = np.zeros(11)

for k in range(2,11):
    kmeans_3dok= KMeans(featuresCol = "fetures", k=k).setSeed(1)
    model_3dok = kmeans_3dok.fit(df_kmeans_scalinit.sample(False,0.1, seed=346))
    prs = model_3dok.transform(df_kmeans_scalinit)

    Silhouete_cost3dok[k] = evalsil3dok.evaluate(prs) #Evaluate clustering
    print('The Silhouette_cost for ' + str(k) + ' clusters is: ' + str(Silhouete_cost3dok[k]))
#plot the Silhouete cost
fig, ax = plt.subplots(1,1, figsize =(8,6))
ax.plot(range(2,11),Silhouete_cost3dok[2:11])
ax.set_title('Silhouette Cost v.s. the number of the clusters \n for standardised data')
ax.set_xlabel('k = clusters')
ax.set_ylabel('Silhouete cost')

plt.show()
```

```
The Silhouette_cost for 2 clusters is: 0.4676189175161663
The Silhouette_cost for 3 clusters is: 0.5321164698556343
The Silhouette_cost for 4 clusters is: 0.5327619107350131
The Silhouette_cost for 5 clusters is: 0.5317461803089011
The Silhouette_cost for 6 clusters is: 0.5090568763397291
The Silhouette_cost for 7 clusters is: 0.4975818019745874
The Silhouette_cost for 8 clusters is: 0.4607674589722039
The Silhouette_cost for 9 clusters is: 0.45928078826461094
The Silhouette_cost for 10 clusters is: 0.45971142650702285
```

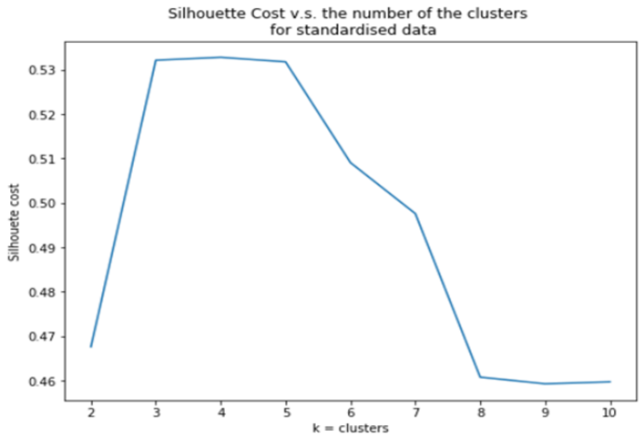
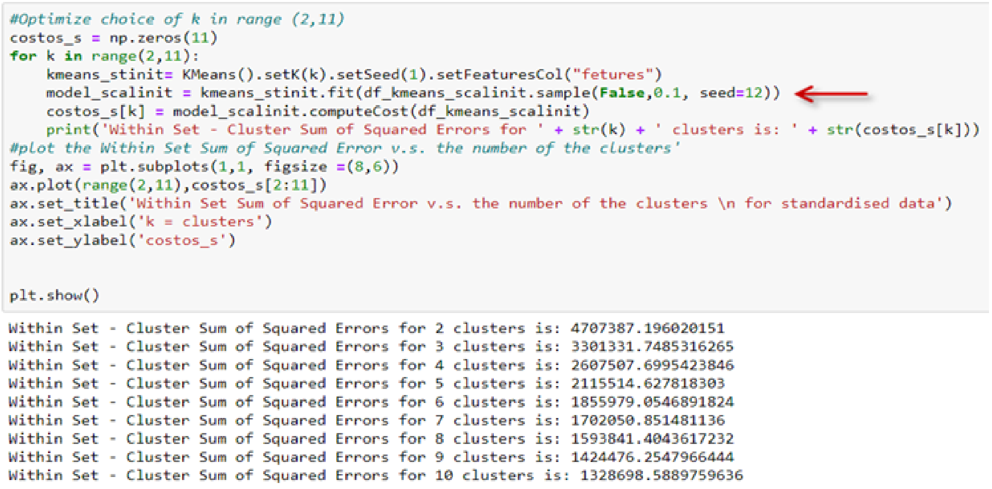


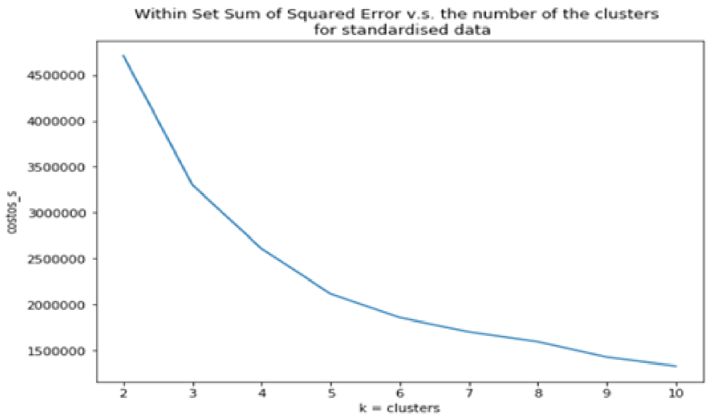
Figure 60: Elbow and Silhouette for seed =346 approach 3 initial data

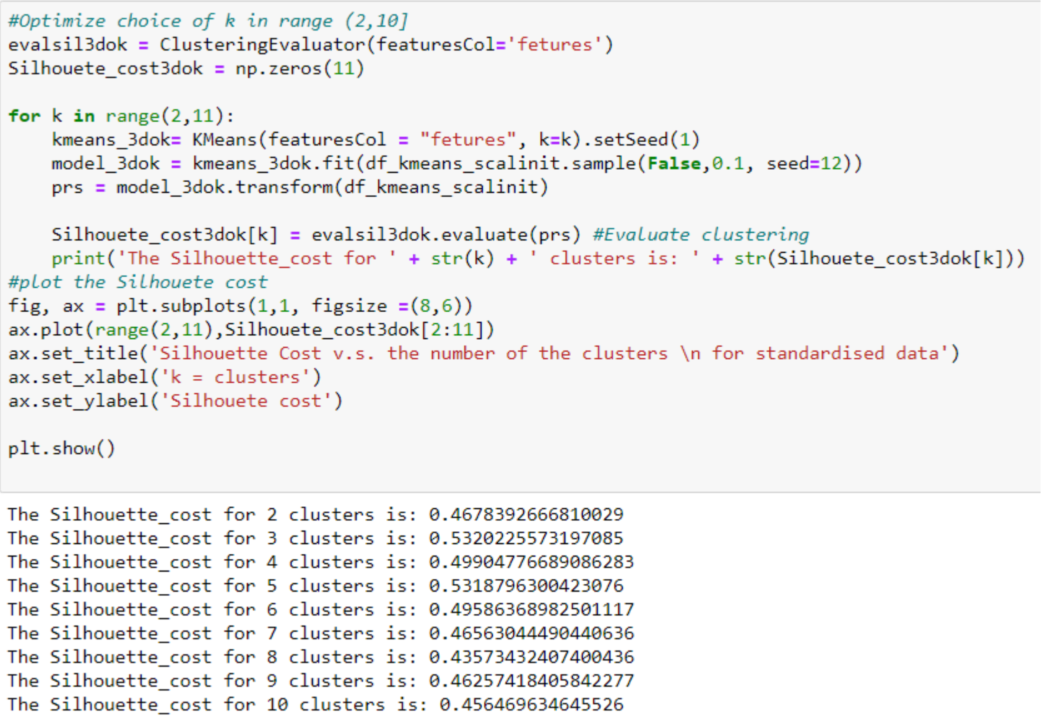
From the figure 60 (seed = 346) we could say that from the elbow method the optimal number of clusters seems to be k = 3 or k = 5. From the silhouette analysis, the optimal number of clusters seems to be k = 3 or κ=4 or k = 5.



```
#Optimize choice of k in range (2,11)
costos_s = np.zeros(11)
for k in range(2,11):
    kmeans_stinit= KMeans().setK(k).setSeed(1).setFeaturesCol("fetures")
    model_scalinit = kmeans_stinit.fit(df_kmeans_scalinit.sample(False,0.1, seed=12))
    costos_s[k] = model_scalinit.computeCost(df_kmeans_scalinit)
    print('Within Set - Cluster Sum of Squared Errors for ' + str(k) + ' clusters is: ' + str(costos_s[k]))
#plot the Within Set Sum of Squared Error v.s. the number of the clusters'
fig, ax = plt.subplots(1,1, figsize =(8,6))
ax.plot(range(2,11),costos_s[2:11])
ax.set_title('Within Set Sum of Squared Error v.s. the number of the clusters \n for standardised data')
ax.set_xlabel('k = clusters')
ax.set_ylabel('costos_s')


plt.show()
```

```
Within Set - Cluster Sum of Squared Errors for 2 clusters is: 4707387.196020151
Within Set - Cluster Sum of Squared Errors for 3 clusters is: 3301331.7485316265
Within Set - Cluster Sum of Squared Errors for 4 clusters is: 2607507.6995423846
Within Set - Cluster Sum of Squared Errors for 5 clusters is: 2115514.627818303
Within Set - Cluster Sum of Squared Errors for 6 clusters is: 1855979.0546891824
Within Set - Cluster Sum of Squared Errors for 7 clusters is: 1702050.851481136
Within Set - Cluster Sum of Squared Errors for 8 clusters is: 1593841.4043617232
Within Set - Cluster Sum of Squared Errors for 9 clusters is: 1424476.2547966444
Within Set - Cluster Sum of Squared Errors for 10 clusters is: 1328698.5889759636
```





```
#Optimize choice of k in range (2,10]
evalsil3dok = ClusteringEvaluator(featuresCol='fetures')
Silhouete_cost3dok = np.zeros(11)

for k in range(2,11):
    kmeans_3dok= KMeans(featuresCol = "fetures", k=k).setSeed(1)
    model_3dok = kmeans_3dok.fit(df_kmeans_scalinit.sample(False,0.1, seed=12))
    prs = model_3dok.transform(df_kmeans_scalinit)

    Silhouete_cost3dok[k] = evalsil3dok.evaluate(prs) #Evaluate clustering
    print('The Silhouette_cost for ' + str(k) + ' clusters is: ' + str(Silhouete_cost3dok[k]))
#plot the Silhouete cost
fig, ax = plt.subplots(1,1, figsize =(8,6))
ax.plot(range(2,11),Silhouete_cost3dok[2:11])
ax.set_title('Silhouette Cost v.s. the number of the clusters \n for standardised data')
ax.set_xlabel('k = clusters')
ax.set_ylabel('Silhouete cost')

plt.show()
```

```
The Silhouette_cost for 2 clusters is: 0.4678392666810029
The Silhouette_cost for 3 clusters is: 0.5320225573197085
The Silhouette_cost for 4 clusters is: 0.49904776689086283
The Silhouette_cost for 5 clusters is: 0.5318796300423076
The Silhouette_cost for 6 clusters is: 0.49586368982501117
The Silhouette_cost for 7 clusters is: 0.46563044490440636
The Silhouette_cost for 8 clusters is: 0.43573432407400436
The Silhouette_cost for 9 clusters is: 0.46257418405842277
The Silhouette_cost for 10 clusters is: 0.456469634645526
```

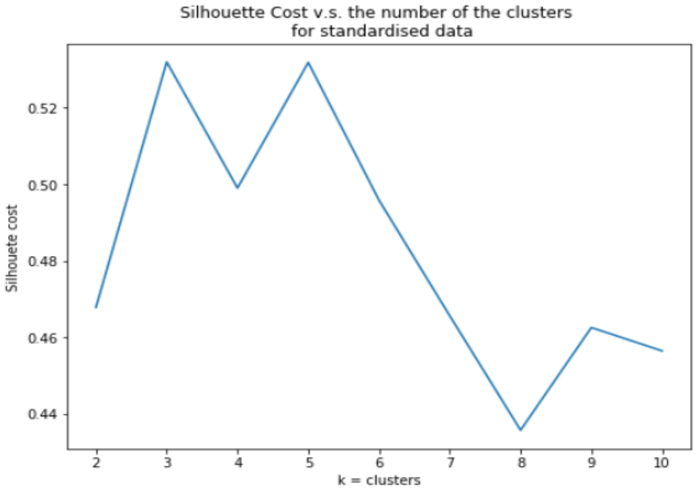


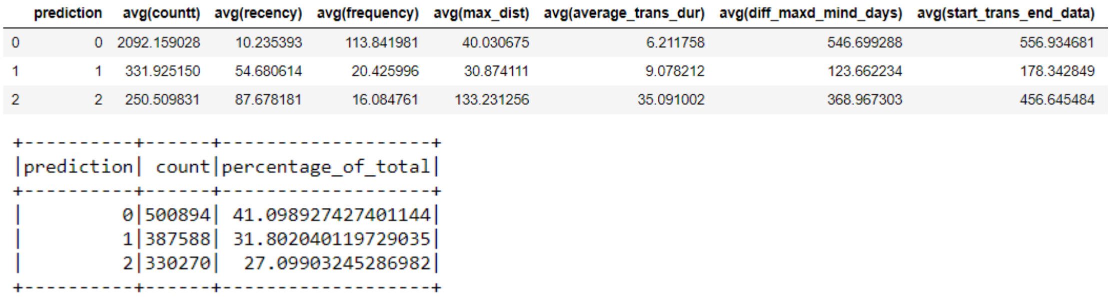
Figure 61: Elbow and Silhouette for seed = 12 approach 3 initial data

From the figure 61 (seed = 12) we could say that from the elbow method the optimal number of clusters seems to be k = 3 or k=4 or k = 5. From the silhouette analysis, the optimal number of clusters seems to be k = 3 or k = 5.

Considering all the above tests from Elbow and Silhouette we would say that k=3 or k = 4 or k = 5 as candidate optimal number of clusters. Something that surprised us is that for seed = 346 appeared the value k = 4, something that did not happen in the other two values.

- K=3

As far as the k = 3, the best value for silhouette score achieved with seed = 42. So, we applied K-means with this specific seed and k=3 and we computed the mean of all features for each prediction – type of cluster. The results were the following (figure 62):



	prediction	avg(countt)	avg(recency)	avg(frequency)	avg(max_dist)	avg(average_trans_dur)	avg(diff_maxd_mind_days)	avg(start_trans_end_data)
0	0	2092.159028	10.235393	113.841981	40.030675	6.211758	546.699288	556.934681
1	1	331.925150	54.680614	20.425996	30.874111	9.078212	123.662234	178.342849
2	2	250.509831	87.678181	16.084761	133.231256	35.091002	368.967303	456.645484

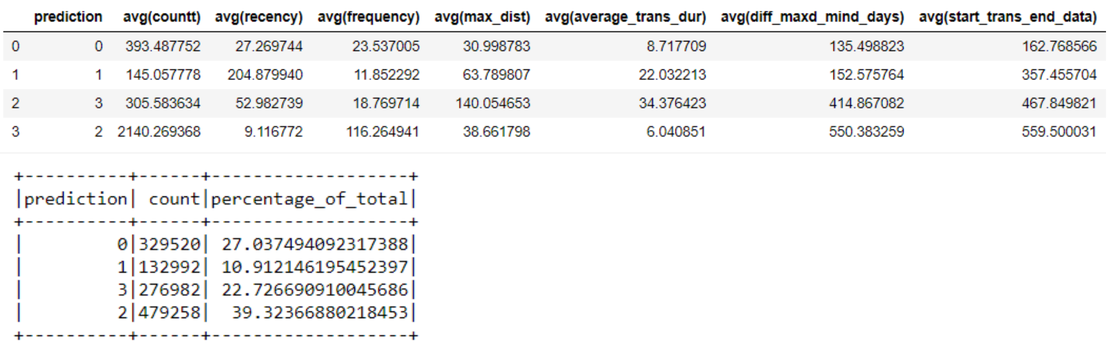
```
+----------+------+------------------+
|prediction| count|percentage_of_total|
+----------+------+------------------+
|         0|500894| 41.098927427401144|
|         1|387588| 31.802040119729035|
|         2|330270|  27.09903245286982|
+----------+------+------------------+
```

Figure 62: Descriptive elements in approach 3 and k =3

Observing the tables in figure 62 we could say that it seems there are well separated clusters.

- K=4

For k=4 we implemented K-means with seed = 346 but although there were good enough separated clusters again, we did not accepted it due to the fact that elbow method did not agree with it. The relevant results are the following (figure 63):



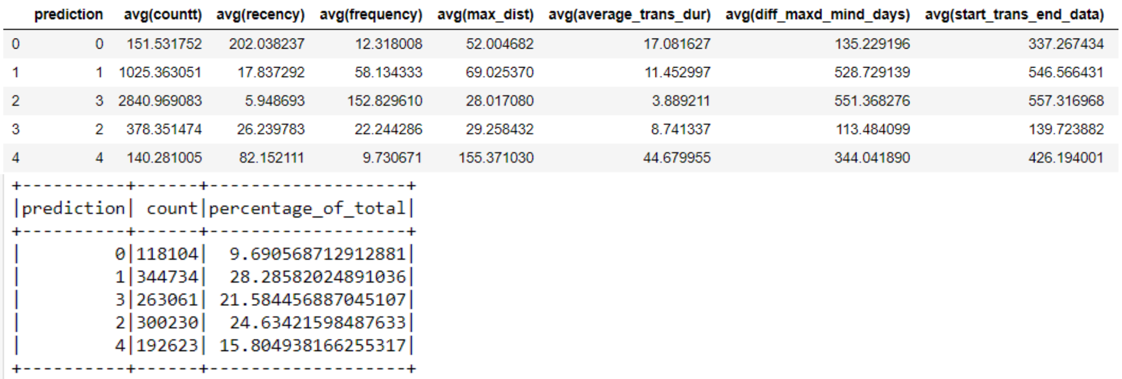
	prediction	avg(countt)	avg(recency)	avg(frequency)	avg(max_dist)	avg(average_trans_dur)	avg(diff_maxd_mind_days)	avg(start_trans_end_data)
0	0	393.487752	27.269744	23.537005	30.998783	8.717709	135.498823	162.768566
1	1	145.057778	204.879940	11.852292	63.789807	22.032213	152.575764	357.455704
2	3	305.583634	52.982739	18.769714	140.054653	34.376423	414.867082	467.849821
3	2	2140.269368	9.116772	116.264941	38.661798	6.040851	550.383259	559.500031

```
+----------+------+------------------+
|prediction| count|percentage_of_total|
+----------+------+------------------+
|         0|329520| 27.037494092317388|
|         1|132992| 10.912146195452397|
|         3|276982| 22.726690910045686|
|         2|479258|  39.32366880218453|
+----------+------+------------------+
```

Figure 63: Descriptive elements in approach 3 and k =4

- K=5

As far as the k = 5, the best value for silhouette score achieved with seed = 12. So, we applied K-means with this specific seed and k=5 and we computed the mean of all features for each prediction – type of cluster. The results were the following (figure 64):



	prediction	avg(countt)	avg(recency)	avg(frequency)	avg(max_dist)	avg(average_trans_dur)	avg(diff_maxd_mind_days)	avg(start_trans_end_data)
0	0	151.531752	202.038237	12.318008	52.004682	17.081627	135.229196	337.267434
1	1	1025.363051	17.837292	58.134333	69.025370	11.452997	528.729139	546.566431
2	3	2840.969083	5.948693	152.829610	28.017080	3.889211	551.368276	557.316968
3	2	378.351474	26.239783	22.244286	29.258432	8.741337	113.484099	139.723882
4	4	140.281005	82.152111	9.730671	155.371030	44.679955	344.041890	426.194001

```
+----------+------+------------------+
|prediction| count|percentage_of_total|
+----------+------+------------------+
|         0|118104|  9.690568712912881|
|         1|344734|  28.28582024891036|
|         3|263061| 21.584456887045107|
|         2|300230|  24.63421598487633|
|         4|192623| 15.804938166255317|
+----------+------+------------------+
```

Figure 64: Descriptive elements in approach 3 and k =5

Observing the tables in figure 64 we could say that it seems there are no well separated clusters.

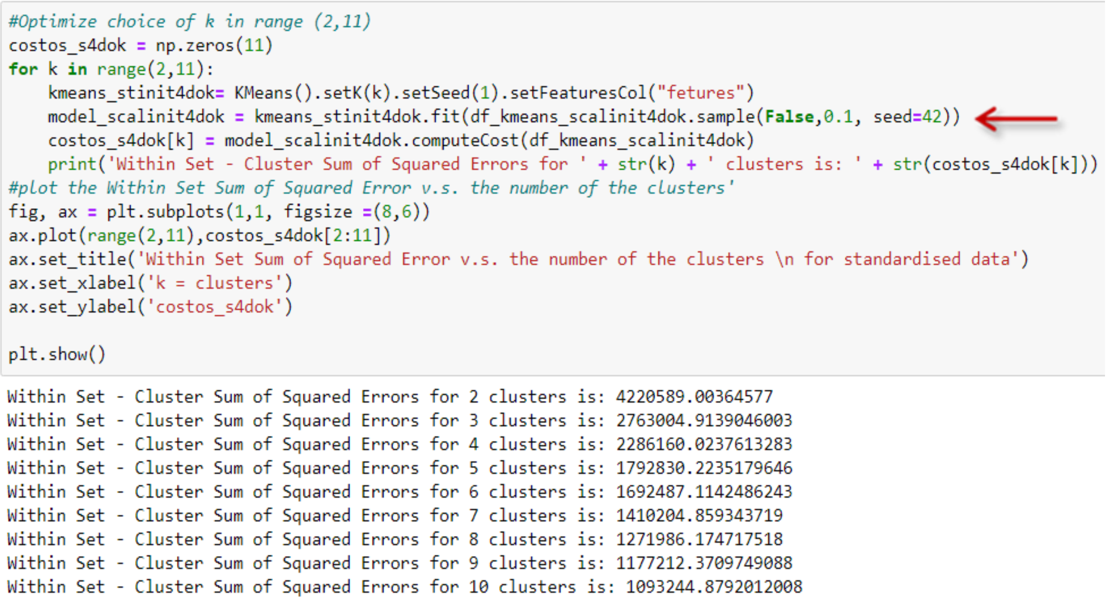
4.2.3.4.4 4th approach

initial features – no diff_maxd_mind_days and countt

In the fourth approach the following features were used:

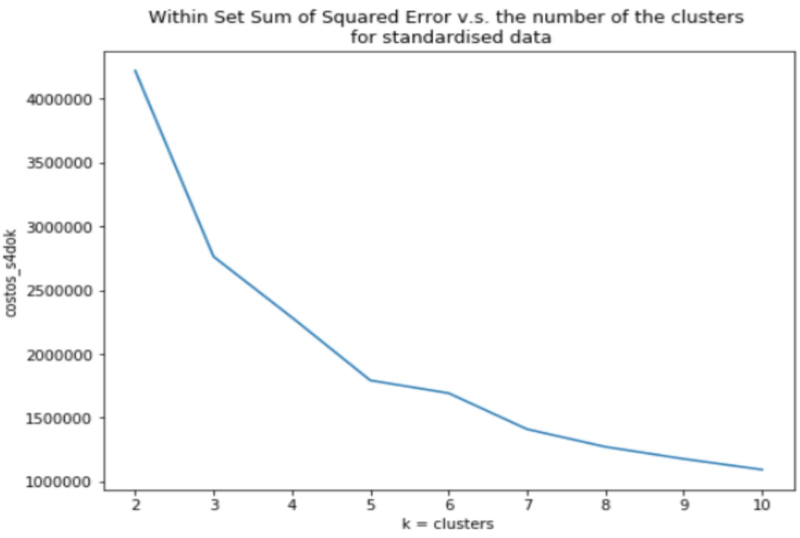
'recency' ,'frequency', 'max_dist', 'average_trans_dur', 'start_trans_end_data'

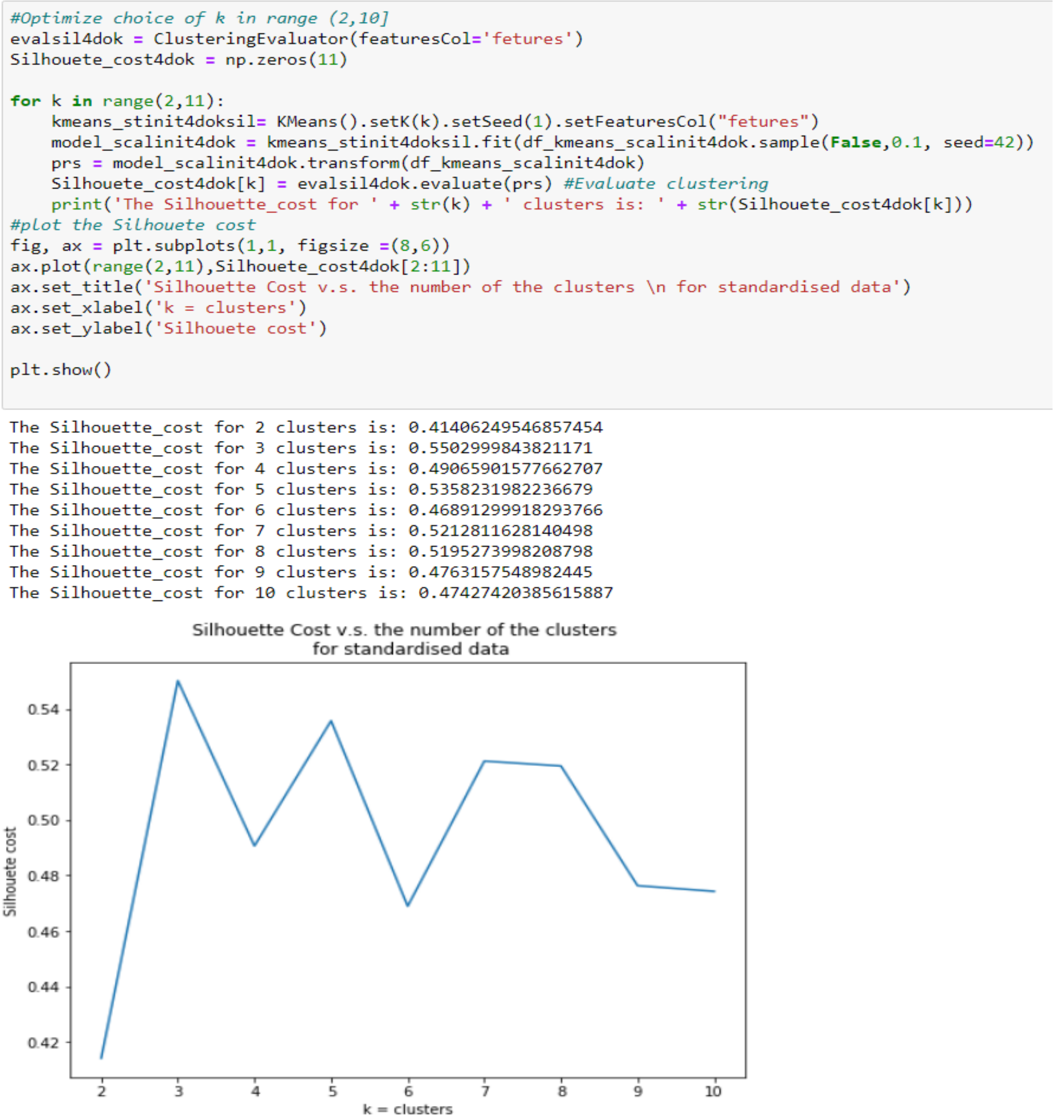
Applying the elbow method and silhouette analysis we had the following outputs:



```
#Optimize choice of k in range (2,11)
costos_s4dok = np.zeros(11)
for k in range(2,11):
    kmeans_stinit4dok= KMeans().setK(k).setSeed(1).setFeaturesCol("fetures")
    model_scalinit4dok = kmeans_stinit4dok.fit(df_kmeans_scalinit4dok.sample(False,0.1, seed=42))
    costos_s4dok[k] = model_scalinit4dok.computeCost(df_kmeans_scalinit4dok)
    print('Within Set - Cluster Sum of Squared Errors for ' + str(k) + ' clusters is: ' + str(costos_s4dok[k]))
#plot the Within Set Sum of Squared Error v.s. the number of the clusters'
fig, ax = plt.subplots(1,1, figsize =(8,6))
ax.plot(range(2,11),costos_s4dok[2:11])
ax.set_title('Within Set Sum of Squared Error v.s. the number of the clusters \n for standardised data')
ax.set_xlabel('k = clusters')
ax.set_ylabel('costos_s4dok')

plt.show()
```

```
Within Set - Cluster Sum of Squared Errors for 2 clusters is: 4220589.00364577
Within Set - Cluster Sum of Squared Errors for 3 clusters is: 2763004.9139046003
Within Set - Cluster Sum of Squared Errors for 4 clusters is: 2286160.0237613283
Within Set - Cluster Sum of Squared Errors for 5 clusters is: 1792830.2235179646
Within Set - Cluster Sum of Squared Errors for 6 clusters is: 1692487.1142486243
Within Set - Cluster Sum of Squared Errors for 7 clusters is: 1410204.859343719
Within Set - Cluster Sum of Squared Errors for 8 clusters is: 1271986.174717518
Within Set - Cluster Sum of Squared Errors for 9 clusters is: 1177212.3709749088
Within Set - Cluster Sum of Squared Errors for 10 clusters is: 1093244.8792012008
```



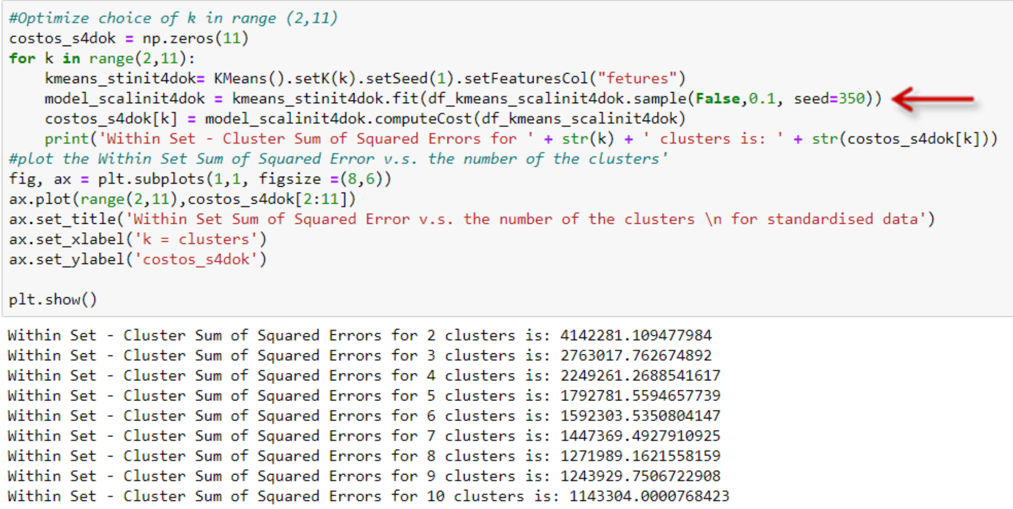


```
#Optimize choice of k in range (2,10]
evalsil4dok = ClusteringEvaluator(featuresCol='fetures')
Silhouete_cost4dok = np.zeros(11)

for k in range(2,11):
    kmeans_stinit4doksil= KMeans().setK(k).setSeed(1).setFeaturesCol("fetures")
    model_scalinit4dok = kmeans_stinit4doksil.fit(df_kmeans_scalinit4dok.sample(False,0.1, seed=42))
    prs = model_scalinit4dok.transform(df_kmeans_scalinit4dok)
    Silhouete_cost4dok[k] = evalsil4dok.evaluate(prs) #Evaluate clustering
    print('The Silhouette_cost for ' + str(k) + ' clusters is: ' + str(Silhouete_cost4dok[k]))
#plot the Silhouete cost
fig, ax = plt.subplots(1,1, figsize =(8,6))
ax.plot(range(2,11),Silhouete_cost4dok[2:11])
ax.set_title('Silhouette Cost v.s. the number of the clusters \n for standardised data')
ax.set_xlabel('k = clusters')
ax.set_ylabel('Silhouete cost')

plt.show()
```

```
The Silhouette_cost for 2 clusters is: 0.41406249546857454
The Silhouette_cost for 3 clusters is: 0.5502999843821171
The Silhouette_cost for 4 clusters is: 0.49065901577662707
The Silhouette_cost for 5 clusters is: 0.5358231982236679
The Silhouette_cost for 6 clusters is: 0.46891299918293766
The Silhouette_cost for 7 clusters is: 0.5212811628140498
The Silhouette_cost for 8 clusters is: 0.5195273998208798
The Silhouette_cost for 9 clusters is: 0.4763157548982445
The Silhouette_cost for 10 clusters is: 0.47427420385615887
```

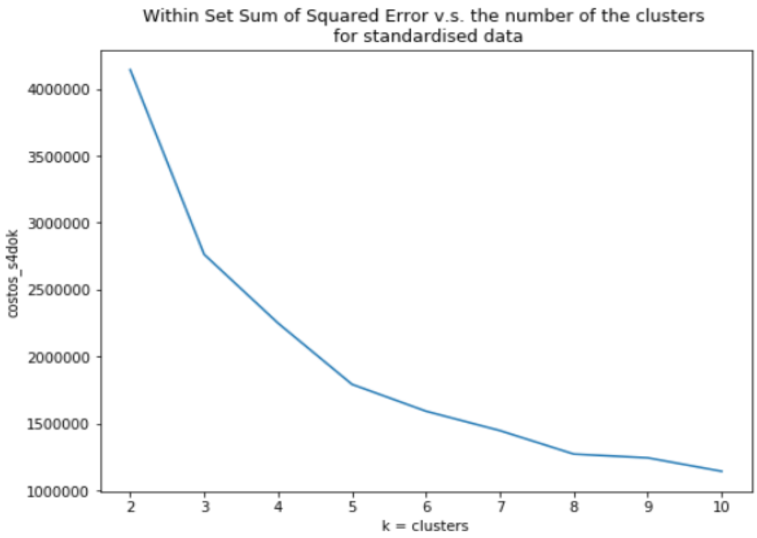
Figure 65: Elbow and Silhouette for seed = 42 approach 4 initial data

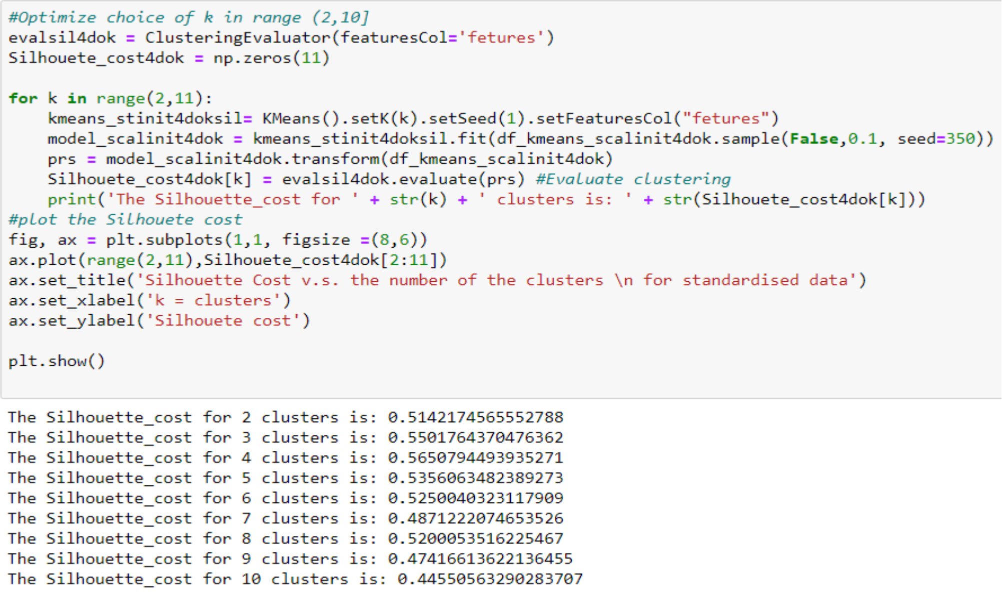
From the figure 65 (seed = 42) we could say that from the elbow method the optimal number of clusters seems to be k = 3 or k = 5. From the silhouette analysis, the optimal number of clusters seems to be k = 3 or k = 5.



```
#Optimize choice of k in range (2,11)
costos_s4dok = np.zeros(11)
for k in range(2,11):
    kmeans_stinit4dok= KMeans().setK(k).setSeed(1).setFeaturesCol("fetures")
    model_scalinit4dok = kmeans_stinit4dok.fit(df_kmeans_scalinit4dok.sample(False,0.1, seed=350))
    costos_s4dok[k] = model_scalinit4dok.computeCost(df_kmeans_scalinit4dok)
    print('Within Set - Cluster Sum of Squared Errors for ' + str(k) + ' clusters is: ' + str(costos_s4dok[k]))
#plot the Within Set Sum of Squared Error v.s. the number of the clusters'
fig, ax = plt.subplots(1,1, figsize =(8,6))
ax.plot(range(2,11),costos_s4dok[2:11])
ax.set_title('Within Set Sum of Squared Error v.s. the number of the clusters \n for standardised data')
ax.set_xlabel('k = clusters')
ax.set_ylabel('costos_s4dok')

plt.show()
```

```
Within Set - Cluster Sum of Squared Errors for 2 clusters is: 4142281.109477984
Within Set - Cluster Sum of Squared Errors for 3 clusters is: 2763017.762674892
Within Set - Cluster Sum of Squared Errors for 4 clusters is: 2249261.2688541617
Within Set - Cluster Sum of Squared Errors for 5 clusters is: 1792781.5594657739
Within Set - Cluster Sum of Squared Errors for 6 clusters is: 1592303.5350804147
Within Set - Cluster Sum of Squared Errors for 7 clusters is: 1447369.4927910925
Within Set - Cluster Sum of Squared Errors for 8 clusters is: 1271989.1621558159
Within Set - Cluster Sum of Squared Errors for 9 clusters is: 1243929.7506722908
Within Set - Cluster Sum of Squared Errors for 10 clusters is: 1143304.0000768423
```





```
#Optimize choice of k in range (2,10]
evalsil4dok = ClusteringEvaluator(featuresCol='fetures')
Silhouete_cost4dok = np.zeros(11)

for k in range(2,11):
    kmeans_stinit4doksil= KMeans().setK(k).setSeed(1).setFeaturesCol("fetures")
    model_scalinit4dok = kmeans_stinit4doksil.fit(df_kmeans_scalinit4dok.sample(False,0.1, seed=350))
    prs = model_scalinit4dok.transform(df_kmeans_scalinit4dok)
    Silhouete_cost4dok[k] = evalsil4dok.evaluate(prs) #Evaluate clustering
    print('The Silhouette_cost for ' + str(k) + ' clusters is: ' + str(Silhouete_cost4dok[k]))
#plot the Silhouete cost
fig, ax = plt.subplots(1,1, figsize =(8,6))
ax.plot(range(2,11),Silhouete_cost4dok[2:11])
ax.set_title('Silhouette Cost v.s. the number of the clusters \n for standardised data')
ax.set_xlabel('k = clusters')
ax.set_ylabel('Silhouete cost')

plt.show()
```

```
The Silhouette_cost for 2 clusters is: 0.5142174565552788
The Silhouette_cost for 3 clusters is: 0.5501764370476362
The Silhouette_cost for 4 clusters is: 0.5650794493935271
The Silhouette_cost for 5 clusters is: 0.5356063482389273
The Silhouette_cost for 6 clusters is: 0.5250040323117909
The Silhouette_cost for 7 clusters is: 0.4871222074653526
The Silhouette_cost for 8 clusters is: 0.5200053516225467
The Silhouette_cost for 9 clusters is: 0.47416613622136455
The Silhouette_cost for 10 clusters is: 0.44550563290283707
```

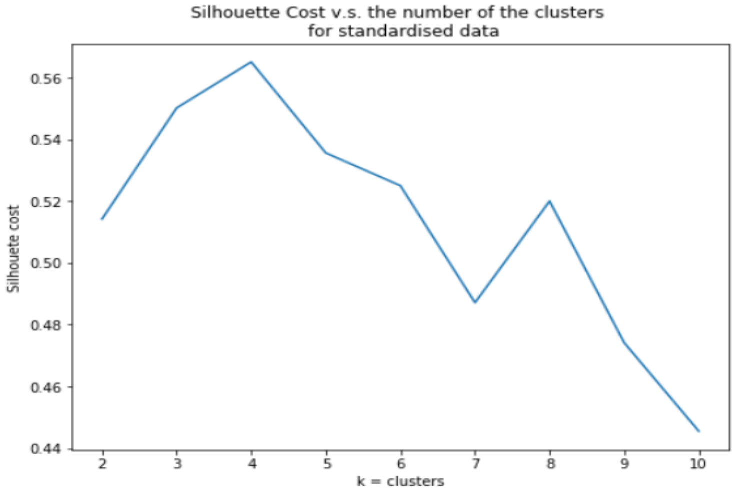
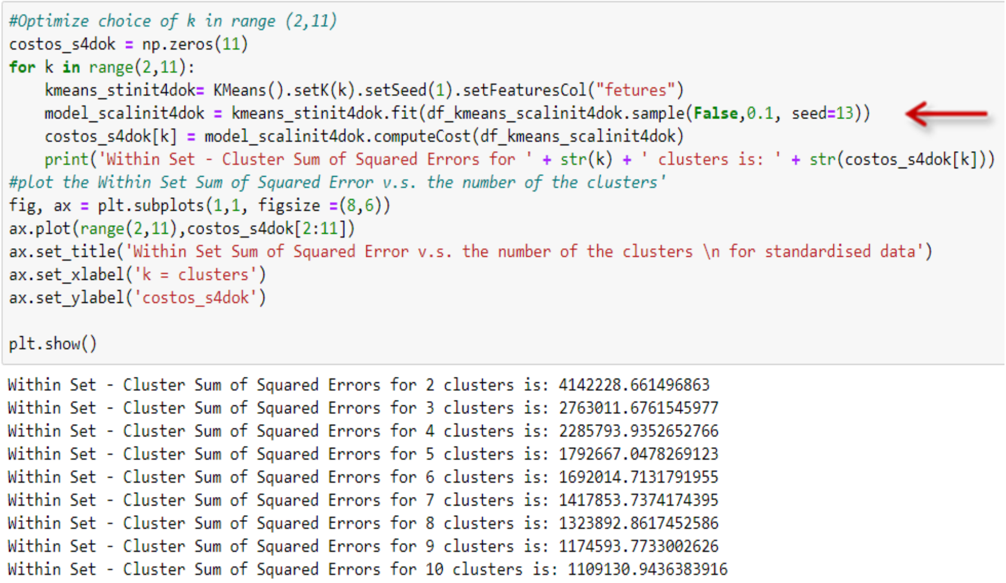


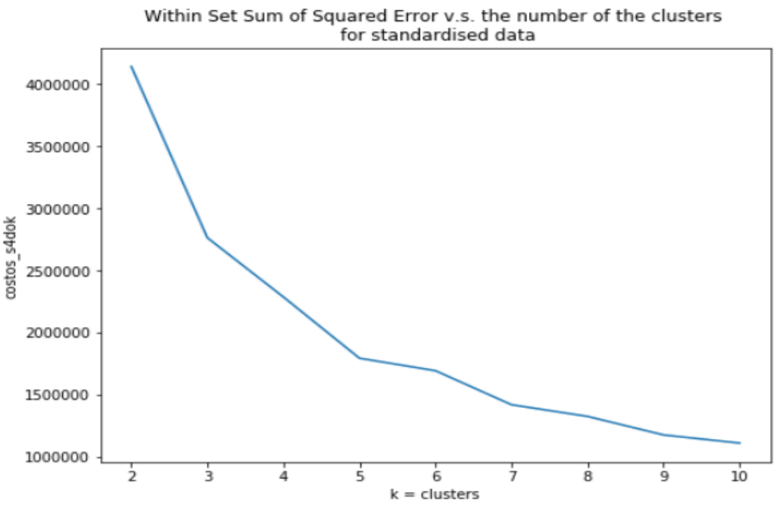
Figure 66: Elbow and Silhouette for seed = 350 approach 4 initial data

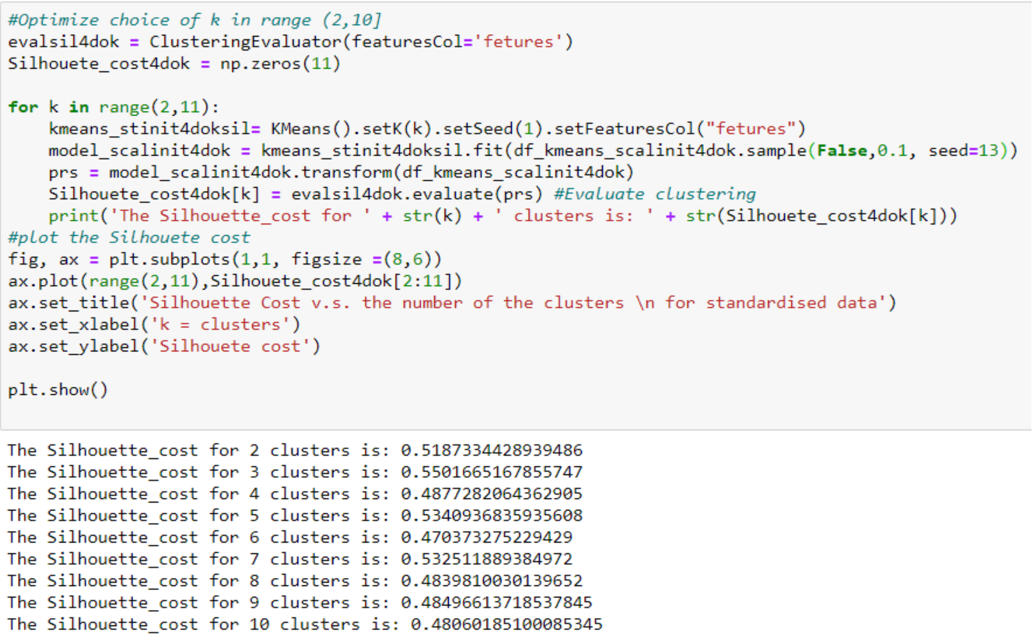
From the figure 66 (seed = 350) we could say that from the elbow method the optimal number of clusters seems to be k = 3 or k = 5. From the silhouette analysis, the optimal number of clusters seems to be k=3 or k = 4.



```
#Optimize choice of k in range (2,11)
costos_s4dok = np.zeros(11)
for k in range(2,11):
    kmeans_stinit4dok= KMeans().setK(k).setSeed(1).setFeaturesCol("fetures")
    model_scalinit4dok = kmeans_stinit4dok.fit(df_kmeans_scalinit4dok.sample(False,0.1, seed=13))
    costos_s4dok[k] = model_scalinit4dok.computeCost(df_kmeans_scalinit4dok)
    print('Within Set - Cluster Sum of Squared Errors for ' + str(k) + ' clusters is: ' + str(costos_s4dok[k]))
#plot the Within Set Sum of Squared Error v.s. the number of the clusters'
fig, ax = plt.subplots(1,1, figsize =(8,6))
ax.plot(range(2,11),costos_s4dok[2:11])
ax.set_title('Within Set Sum of Squared Error v.s. the number of the clusters \n for standardised data')
ax.set_xlabel('k = clusters')
ax.set_ylabel('costos_s4dok')

plt.show()
```

```
Within Set - Cluster Sum of Squared Errors for 2 clusters is: 4142228.661496863
Within Set - Cluster Sum of Squared Errors for 3 clusters is: 2763011.6761545977
Within Set - Cluster Sum of Squared Errors for 4 clusters is: 2285793.9352652766
Within Set - Cluster Sum of Squared Errors for 5 clusters is: 1792667.0478269123
Within Set - Cluster Sum of Squared Errors for 6 clusters is: 1692014.7131791955
Within Set - Cluster Sum of Squared Errors for 7 clusters is: 1417853.7374174395
Within Set - Cluster Sum of Squared Errors for 8 clusters is: 1323892.8617452586
Within Set - Cluster Sum of Squared Errors for 9 clusters is: 1174593.7733002626
Within Set - Cluster Sum of Squared Errors for 10 clusters is: 1109130.9436383916
```





```
#Optimize choice of k in range (2,10]
evalsil4dok = ClusteringEvaluator(featuresCol='fetures')
Silhouete_cost4dok = np.zeros(11)

for k in range(2,11):
    kmeans_stinit4doksil= KMeans().setK(k).setSeed(1).setFeaturesCol("fetures")
    model_scalinit4dok = kmeans_stinit4doksil.fit(df_kmeans_scalinit4dok.sample(False,0.1, seed=13))
    prs = model_scalinit4dok.transform(df_kmeans_scalinit4dok)
    Silhouete_cost4dok[k] = evalsil4dok.evaluate(prs) #Evaluate clustering
    print('The Silhouete_cost for ' + str(k) + ' clusters is: ' + str(Silhouete_cost4dok[k]))
#plot the Silhouete cost
fig, ax = plt.subplots(1,1, figsize =(8,6))
ax.plot(range(2,11),Silhouete_cost4dok[2:11])
ax.set_title('Silhouette Cost v.s. the number of the clusters \n for standardised data')
ax.set_xlabel('k = clusters')
ax.set_ylabel('Silhouete cost')

plt.show()
```

```
The Silhouete_cost for 2 clusters is: 0.5187334428939486
The Silhouete_cost for 3 clusters is: 0.5501665167855747
The Silhouete_cost for 4 clusters is: 0.4877282064362905
The Silhouete_cost for 5 clusters is: 0.5340936835935608
The Silhouete_cost for 6 clusters is: 0.470373275229429
The Silhouete_cost for 7 clusters is: 0.532511889384972
The Silhouete_cost for 8 clusters is: 0.4839810030139652
The Silhouete_cost for 9 clusters is: 0.48496613718537845
The Silhouete_cost for 10 clusters is: 0.48060185100085345
```

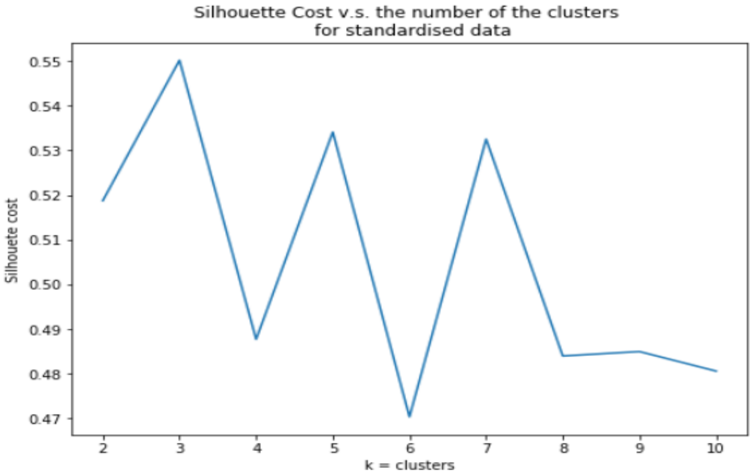


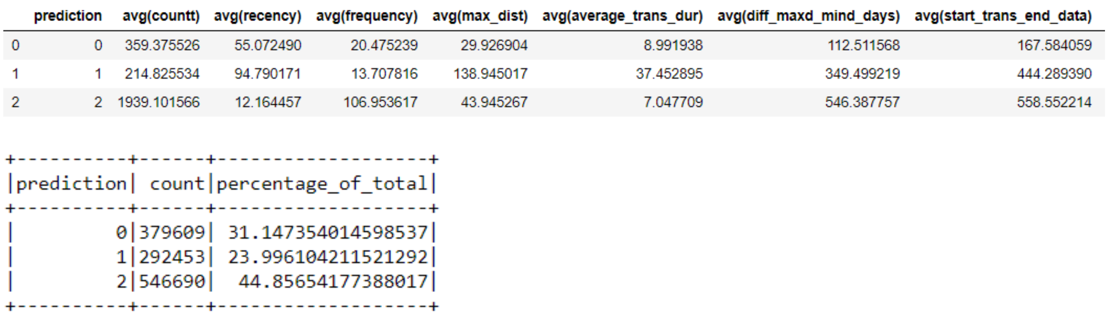
Figure 67: Elbow and Silhouette for seed = 13 approach 4 initial data

From the figure 67 (seed = 13) we could say that from the elbow method the optimal number of clusters seems to be k = 3 or k = 5. From the silhouette analysis, the optimal number of clusters seems to be k=3 or k = 5.

Considering all the above tests from Elbow and Silhouette we would say that k=3 or k = 5 as candidate optimal number of clusters. Something that surprised us again is that for seed = 350 appeared the value k = 4, something that did not happen in the other two values.

- K=3

As far as the k = 3, the best value for silhouette score achieved with seed = 42. So, we applied K-means with this specific seed and k=3 and we computed the mean of all features for each prediction – type of cluster. The results were the following (figure 68):



	prediction	avg(countt)	avg(recency)	avg(frequency)	avg(max_dist)	avg(average_trans_dur)	avg(diff_maxd_mind_days)	avg(start_trans_end_data)
0	0	359.375526	55.072490	20.475239	29.926904	8.991938	112.511568	167.584059
1	1	214.825534	94.790171	13.707816	138.945017	37.452895	349.499219	444.289390
2	2	1939.101566	12.164457	106.953617	43.945267	7.047709	546.387757	558.552214

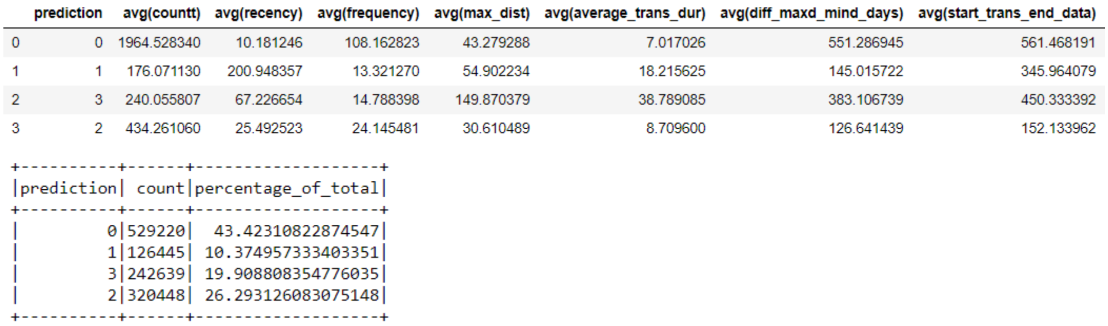
```
+----------+------+------------------+
|prediction| count|percentage_of_total|
+----------+------+------------------+
|         0|379609| 31.147354014598537|
|         1|292453| 23.996104211521292|
|         2|546690|  44.85654177388017|
+----------+------+------------------+
```

Figure 68: Descriptive elements in approach 4 and k =3

Observing the tables in figure 68 we could say that it seems there are well separated clusters.

- K=4

For k=4 we implemented K-means with seed = 350 but although there were good enough separated clusters again, we did not accepted it due to the fact that elbow method did not agree with it. The relevant results are the following (figure 69):



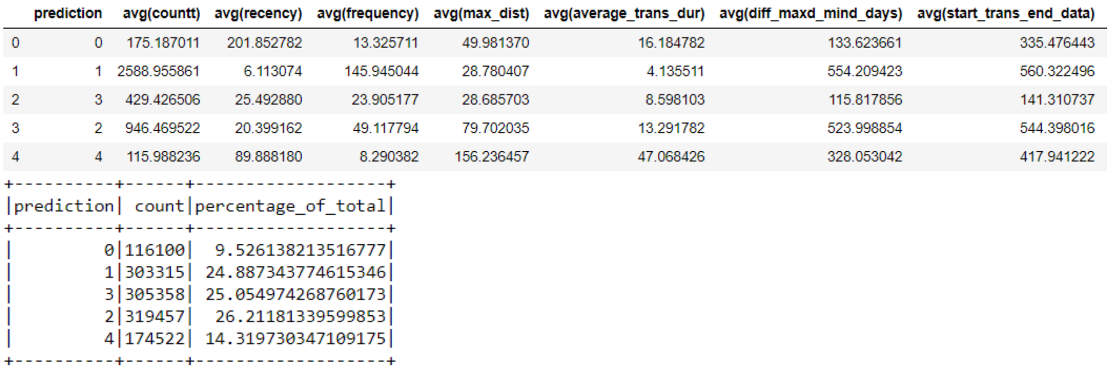
	prediction	avg(countt)	avg(recency)	avg(frequency)	avg(max_dist)	avg(average_trans_dur)	avg(diff_maxd_mind_days)	avg(start_trans_end_data)
0	0	1964.528340	10.181246	108.162823	43.279288	7.017026	551.286945	561.468191
1	1	176.071130	200.948357	13.321270	54.902234	18.215625	145.015722	345.964079
2	3	240.055807	67.226654	14.788398	149.870379	38.789085	383.106739	450.333392
3	2	434.261060	25.492523	24.145481	30.610489	8.709600	126.641439	152.133962

```
+----------+------+------------------+
|prediction| count|percentage_of_total|
+----------+------+------------------+
|         0|529220|  43.42310822874547|
|         1|126445| 10.374957333403351|
|         3|242639| 19.908808354776035|
|         2|320448| 26.293126083075148|
+----------+------+------------------+
```

Figure 69: Descriptive elements in approach 4 and k =4

- K=5

As far as the k = 5, the best value for silhouette score achieved with seed = 42. So, we applied K-means with this specific seed and k=5 and we computed the mean of all features for each prediction – type of cluster. The results were the following (figure 70):



	prediction	avg(countt)	avg(recency)	avg(frequency)	avg(max_dist)	avg(average_trans_dur)	avg(diff_maxd_mind_days)	avg(start_trans_end_data)
0	0	175.187011	201.852782	13.325711	49.981370	16.184782	133.623661	335.476443
1	1	2588.955861	6.113074	145.945044	28.780407	4.135511	554.209423	560.322496
2	3	429.426506	25.492880	23.905177	28.685703	8.598103	115.817856	141.310737
3	2	946.469522	20.399162	49.117794	79.702035	13.291782	523.998854	544.398016
4	4	115.988236	89.888180	8.290382	156.236457	47.068426	328.053042	417.941222

```
+----------+------+------------------+
|prediction| count|percentage_of_total|
+----------+------+------------------+
|         0|116100|  9.526138213516777|
|         1|303315| 24.887343774615346|
|         3|305358| 25.054974268760173|
|         2|319457|  26.21181339599853|
|         4|174522| 14.319730347109175|
+----------+------+------------------+
```

Figure 70: Descriptive elements in approach 4 and k =5

Observing the tables in figure 70 we could say that it seems there are no well separated clusters.

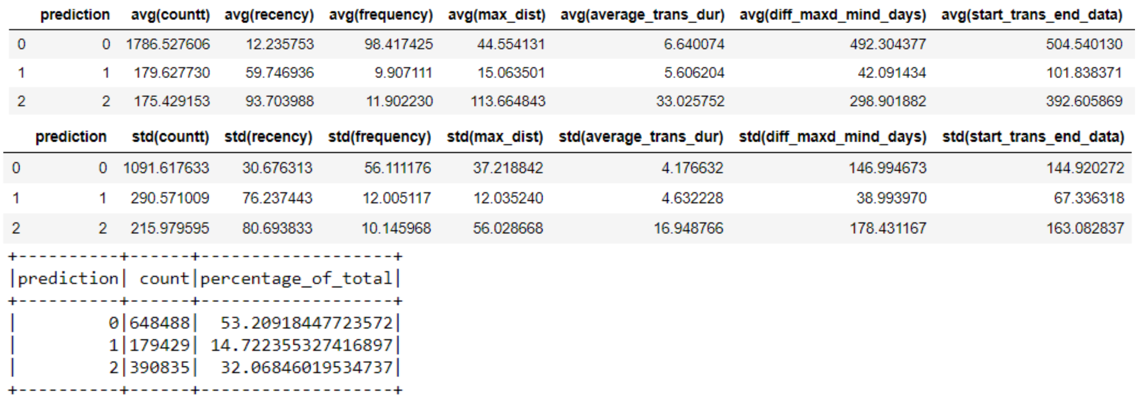
4.2.3.5 Conclusions

- From all the above tests, the conclusion is that for any number other than 3 the clustering has no "meaningful" clusters. As a result the optimal number of clusters is the k = 3. Even though k=3 is the optimal number of clusters, the decision which of all the approaches is the best "grouping" for the client base is difficult. In some variables, there is no an invisible separation of data into clusters because the values are very close to each other among the clusters.

- So as to be able to decide which approach satisfies “best” the distribution of our features it was needed to calculate beside the mean of all features for each cluster and the standard deviation of them. In that way, we could see how much stable the mean values are, i.e. how close to the mean of the data set, on average, the data instances are.

So, we have the following:

1st approach k=3

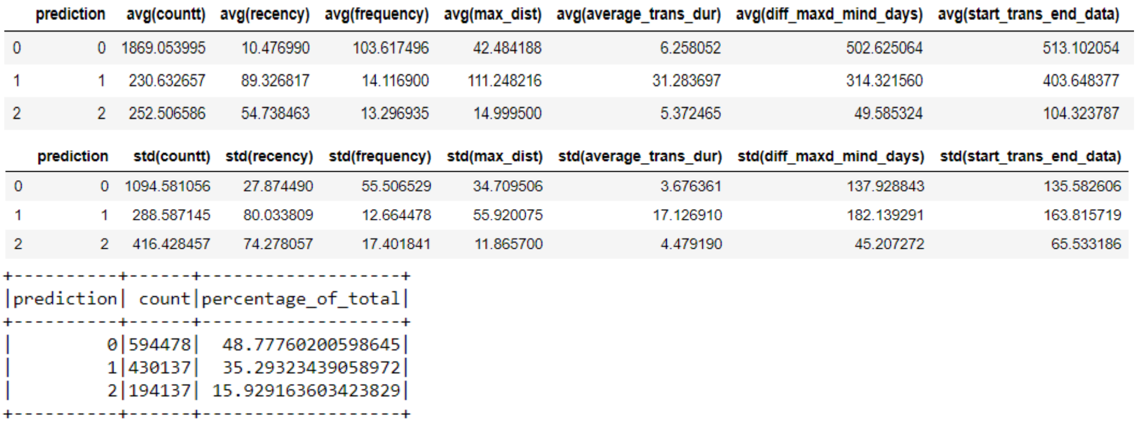


	prediction	avg(countt)	avg(recency)	avg(frequency)	avg(max_dist)	avg(average_trans_dur)	avg(diff_maxd_mind_days)	avg(start_trans_end_data)
0	0	1786.527606	12.235753	98.417425	44.554131	6.640074	492.304377	504.540130
1	1	179.627730	59.746936	9.907111	15.063501	5.606204	42.091434	101.838371
2	2	175.429153	93.703988	11.902230	113.664843	33.025752	298.901882	392.605869

	prediction	std(countt)	std(recency)	std(frequency)	std(max_dist)	std(average_trans_dur)	std(diff_maxd_mind_days)	std(start_trans_end_data)
0	0	1091.617633	30.676313	56.111176	37.218842	4.176632	146.994673	144.920272
1	1	290.571009	76.237443	12.005117	12.035240	4.632228	38.993970	67.336318
2	2	215.979595	80.693833	10.145968	56.028668	16.948766	178.431167	163.082837

```
+----------+------+------------------+
|prediction| count|percentage_of_total|
+----------+------+------------------+
|         0|648488|  53.20918447723572|
|         1|179429| 14.722355327416897|
|         2|390835|  32.06846019534737|
+----------+------+------------------+
```

2nd approach k=3

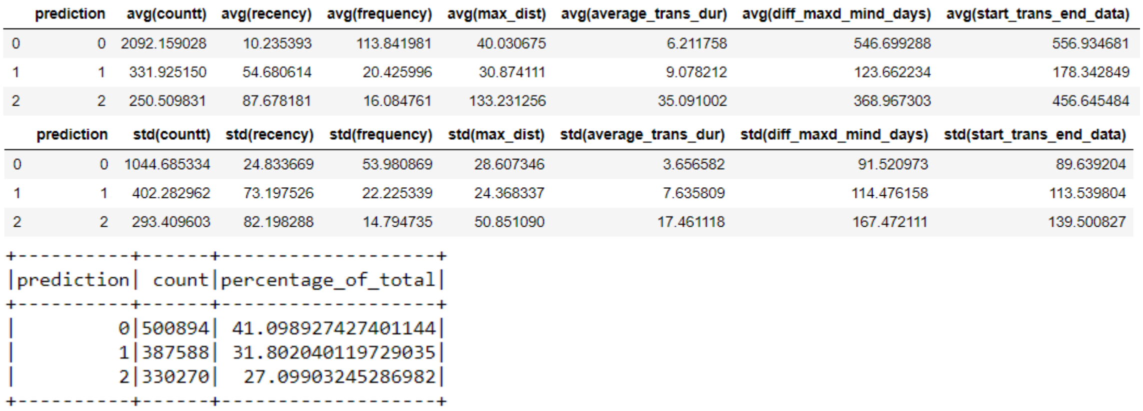


	prediction	avg(countt)	avg(recency)	avg(frequency)	avg(max_dist)	avg(average_trans_dur)	avg(diff_maxd_mind_days)	avg(start_trans_end_data)
0	0	1869.053995	10.476990	103.617496	42.484188	6.258052	502.625064	513.102054
1	1	230.632657	89.326817	14.116900	111.248216	31.283697	314.321560	403.648377
2	2	252.506586	54.738463	13.296935	14.999500	5.372465	49.585324	104.323787

	prediction	std(countt)	std(recency)	std(frequency)	std(max_dist)	std(average_trans_dur)	std(diff_maxd_mind_days)	std(start_trans_end_data)
0	0	1094.581056	27.874490	55.506529	34.709506	3.676361	137.928843	135.582606
1	1	288.587145	80.033809	12.664478	55.920075	17.126910	182.139291	163.815719
2	2	416.428457	74.278057	17.401841	11.865700	4.479190	45.207272	65.533186

```
+----------+------+------------------+
|prediction| count|percentage_of_total|
+----------+------+------------------+
|         0|594478|  48.77760200598645|
|         1|430137|  35.29323439058972|
|         2|194137| 15.929163603423829|
+----------+------+------------------+
```

3rd approach   k=3

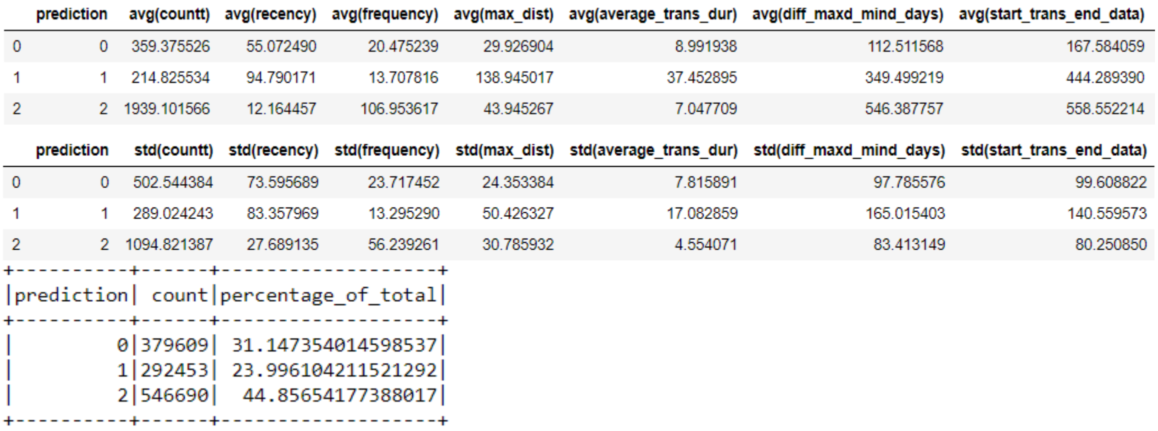


	prediction	avg(countt)	avg(recency)	avg(frequency)	avg(max_dist)	avg(average_trans_dur)	avg(diff_maxd_mind_days)	avg(start_trans_end_data)
0	0	2092.159028	10.235393	113.841981	40.030675	6.211758	546.699288	556.934681
1	1	331.925150	54.680614	20.425996	30.874111	9.078212	123.662234	178.342849
2	2	250.509831	87.678181	16.084761	133.231256	35.091002	368.967303	456.645484

	prediction	std(countt)	std(recency)	std(frequency)	std(max_dist)	std(average_trans_dur)	std(diff_maxd_mind_days)	std(start_trans_end_data)
0	0	1044.685334	24.833669	53.980869	28.607346	3.656582	91.520973	89.639204
1	1	402.282962	73.197526	22.225339	24.368337	7.635809	114.476158	113.539804
2	2	293.409603	82.198288	14.794735	50.851090	17.461118	167.472111	139.500827

```
+----------+------+------------------+
|prediction| count|percentage_of_total|
+----------+------+------------------+
|         0|500894| 41.098927427401144|
|         1|387588| 31.802040119729035|
|         2|330270|  27.09903245286982|
+----------+------+------------------+
```

4th approach   k=3



	prediction	avg(countt)	avg(recency)	avg(frequency)	avg(max_dist)	avg(average_trans_dur)	avg(diff_maxd_mind_days)	avg(start_trans_end_data)
0	0	359.375526	55.072490	20.475239	29.926904	8.991938	112.511568	167.584059
1	1	214.825534	94.790171	13.707816	138.945017	37.452895	349.499219	444.289390
2	2	1939.101566	12.164457	106.953617	43.945267	7.047709	546.387757	558.552214

	prediction	std(countt)	std(recency)	std(frequency)	std(max_dist)	std(average_trans_dur)	std(diff_maxd_mind_days)	std(start_trans_end_data)
0	0	502.544384	73.595689	23.717452	24.353384	7.815891	97.785576	99.608822
1	1	289.024243	83.357969	13.295290	50.426327	17.082859	165.015403	140.559573
2	2	1094.821387	27.689135	56.239261	30.785932	4.554071	83.413149	80.250850

```
+----------+------+------------------+
|prediction| count|percentage_of_total|
+----------+------+------------------+
|         0|379609|  31.147354014598537|
|         1|292453|  23.996104211521292|
|         2|546690|   44.85654177388017|
+----------+------+------------------+
```

Figure 71: Summary of all approaches in clustering phase

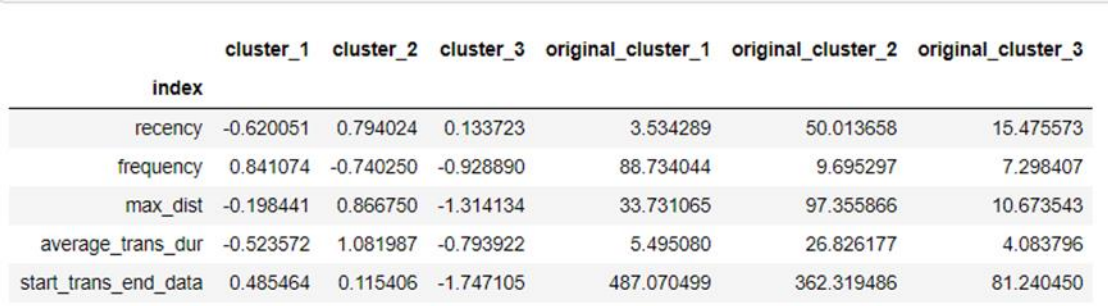
As we mentioned in our research strategy, we are interested for the variable max_dist, so it is the first variable that we looked if its values were good enough so as to continue with the procedure of extracting the labels.

Observing the tables of figure 71 we saw that in the third and fourth approach the standard deviation of the variable max_dist has higher values than the respective one in the first two ones. This means that our data instances are farther away from the mean, on average. As result the optimal approach is hidden in the first two ones which concern the log data. An addition element that we took to consideration so as to reject the approaches 3 and 4 is that in the latter, for each cluster the result from the sum of avg (max_dist) + std (max_dist) overlaps the avg (max_dist) of someone else's cluster. More specific, in third approach, for the cluster 1 we have
30,87 + 24,37 = 55,24 which overlaps the 40, 03 = avg(max_dist) of cluster 0 and in the fourth approach for the cluster 0 we have 29,93 + 24,35 = 54,28 which overlaps the 43,95 = avg(max_dist) of cluster 2.

Moreover, between the approaches 1 and 2, the approach 2 has lower values as far as the standard deviation of max_dist is concerned in comparison with the approach 1.

Conclusion: The “best” result is given by the approach 2.

For the approach 2 in which we have log data and we have dropped the variables diff_maxd_mind_days and countt because of the high correlation the centroids for the clusters are the following (figure 72):



index	cluster_1	cluster_2	cluster_3	original_cluster_1	original_cluster_2	original_cluster_3
recency	-0.620051	0.794024	0.133723	3.534289	50.013658	15.475573
frequency	0.841074	-0.740250	-0.928890	88.734044	9.695297	7.298407
max_dist	-0.198441	0.866750	-1.314134	33.731065	97.355866	10.673543
average_trans_dur	-0.523572	1.081987	-0.793922	5.495080	26.826177	4.083796
start_trans_end_data	0.485464	0.115406	-1.747105	487.070499	362.319486	81.240450

Figure 72: Centroids of clusters (approach 2, k=3)

As someone could see in the figure 72, there are 6 columns. The columns with the prefix “original” (the last three) represent the original values of the centroids whereas the rest represent the “encoding” centroids. Due to the fact that we had standardized and log our data we had to transform (inverse standardization, inverse log (= exp)) the values so as to take the original ones. The formula for the inverse of standardization is the following:

originate_value = standardised_value * std (column) + mean (column)

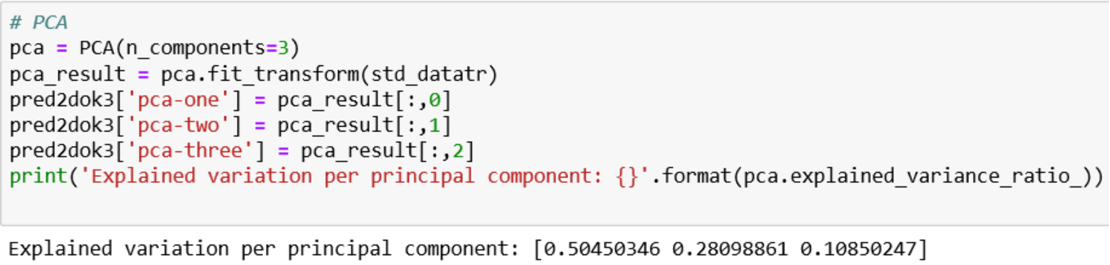
Note: There is a misleading question about the feature importance and the centroids in the world of data science. Some analysts believe that the higher the value of a feature in the centroids, the more influence on the cluster or similar. For example, observing someone the figure 72 and more specific the value in the variable “start_trans_end_data” in the column “original_cluster_3” being the largest value from all the rest values in this column, they could assume that this feature has the greatest influence in the cluster 3. This belief is clearly incorrect because it is just the location of the cluster centroid in the corresponding dimension. The range of the value does not have anything to do with the importance or influence of this attribute for the cluster (with the exception of tf-idf like features like in text clustering etc.). (RapidMiner, 2009)

4.2.3.6 Visualization of the best clustering

After finding the “best” clustering result, the next move was to visualize it. The visualization phase was implemented on windows jupyter notebook and with the use of PCA. The exported csv file with name “pd_pred2dok3fromubuntu.csv” renamed to “pred2dok3” on jupyter notebook.

It was important for us to scale - standardize the data (with the use of “StandardScaler()”of sklearn) before the plotting because of the PCA‘s sensitivity to extreme values.

Executing the PCA transformation we got the following values for the percentage of explained variation per principal component (figure 73):



```
# PCA
pca = PCA(n_components=3)
pca_result = pca.fit_transform(std_datatr)
pred2dok3['pca-one'] = pca_result[:,0]
pred2dok3['pca-two'] = pca_result[:,1]
pred2dok3['pca-three'] = pca_result[:,2]
print('Explained variation per principal component: {}'.format(pca.explained_variance_ratio_))
```

```
Explained variation per principal component: [0.50450346 0.28098861 0.10850247]
```

Figure 73: Explained variation per principal component

From the figure 73 it seems that the first component gathering the most variation explains the 50,45% of the total variation of the data (the half of the data), the second the 28,1% and the third one the 10,885%.

Executing scatterplots for our data from seaborn and matplotlib libraries in two dimensions (2-D) and in three dimensions (3-D) respectively, we had the following results (figures 74 and figures 75):

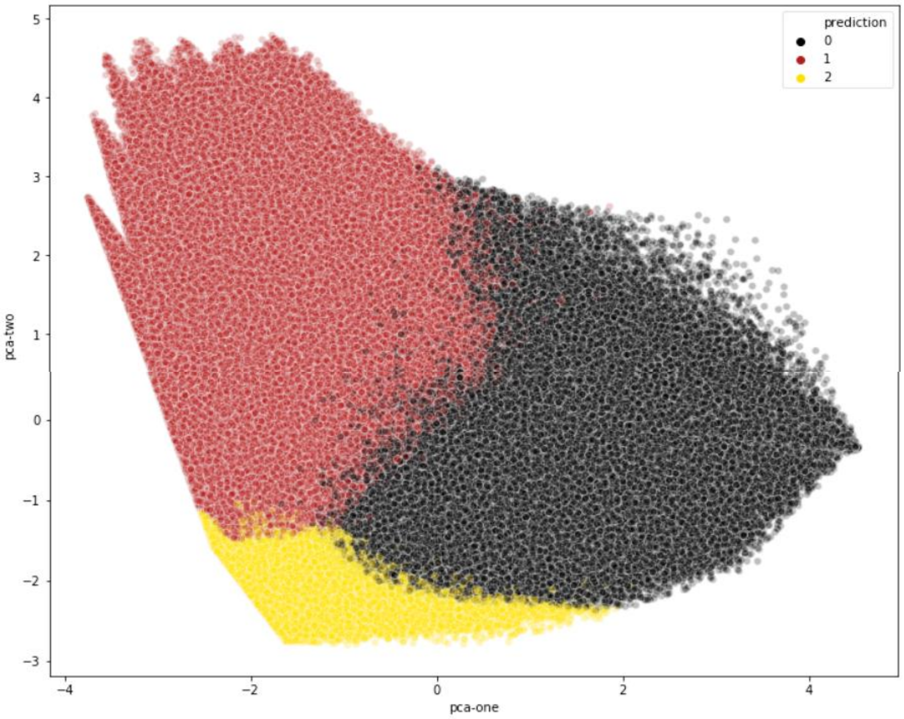


Figure 74: 2-D Visualization

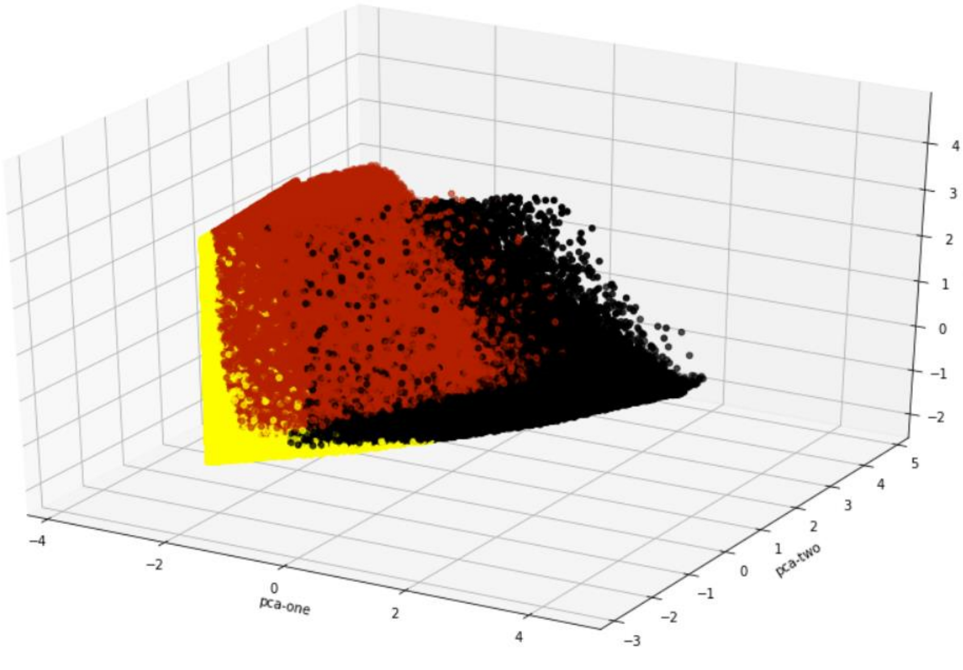
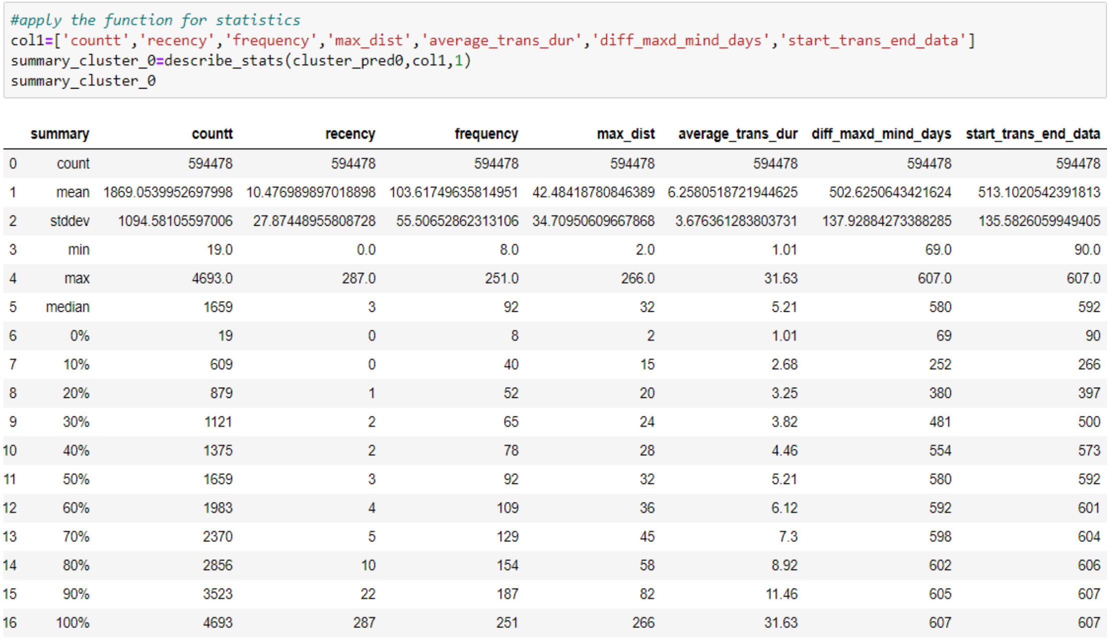


Figure 75: 3-D Visualization

4.2.3.7 Some statistics about the clusters with the "optimal" k

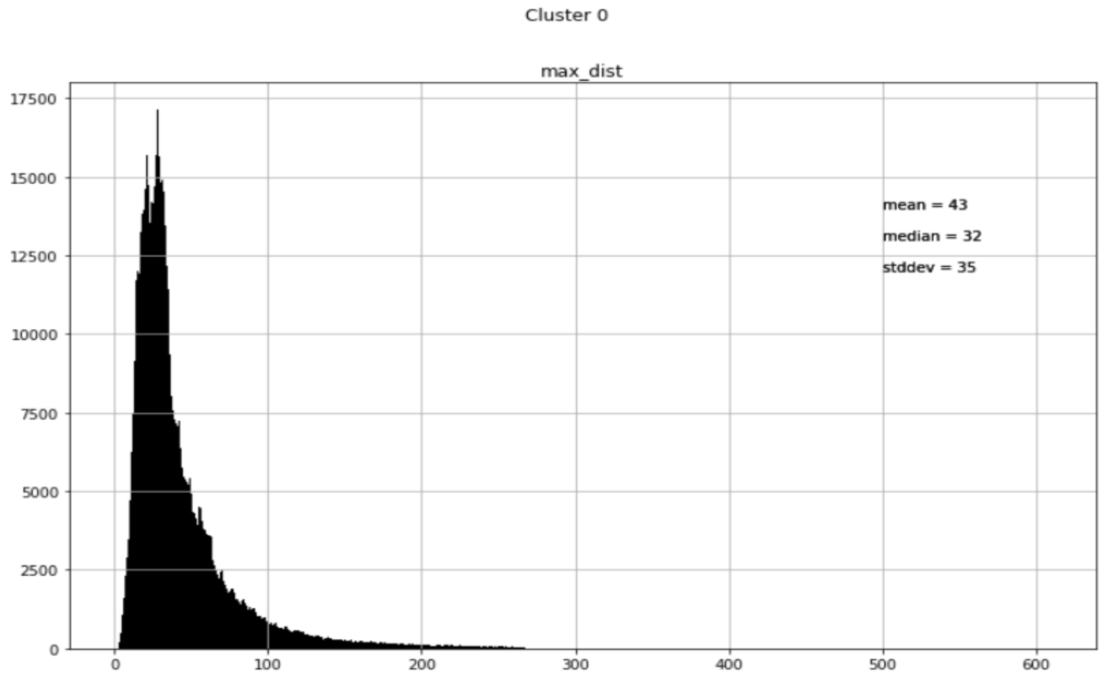
After the “best” clustering (approach 2) we computed some statistics for the features for each of the clusters and then we plotted the variable (max_dist) that we are interested in and then the rest of features as follows:

- Cluster 0 (48,78% of the total data)



```
#apply the function for statistics
col1=['countt','recency','frequency','max_dist','average_trans_dur','diff_maxd_mind_days','start_trans_end_data']
summary_cluster_0=describe_stats(cluster_pred0,col1,1)
summary_cluster_0
```

	summary	countt	recency	frequency	max_dist	average_trans_dur	diff_maxd_mind_days	start_trans_end_data
0	count	594478	594478	594478	594478	594478	594478	594478
1	mean	1869.0539952697998	10.476989897018898	103.61749635814951	42.48418780846389	6.2580518721944625	502.6250643421624	513.1020542391813
2	stddev	1094.58105597006	27.87448955808728	55.50652862313106	34.70950609667868	3.676361283803731	137.92884273388285	135.5826059949405
3	min	19.0	0.0	8.0	2.0	1.01	69.0	90.0
4	max	4693.0	287.0	251.0	266.0	31.63	607.0	607.0
5	median	1659	3	92	32	5.21	580	592
6	0%	19	0	8	2	1.01	69	90
7	10%	609	0	40	15	2.68	252	266
8	20%	879	1	52	20	3.25	380	397
9	30%	1121	2	65	24	3.82	481	500
10	40%	1375	2	78	28	4.46	554	573
11	50%	1659	3	92	32	5.21	580	592
12	60%	1983	4	109	36	6.12	592	601
13	70%	2370	5	129	45	7.3	598	604
14	80%	2856	10	154	58	8.92	602	606
15	90%	3523	22	187	82	11.46	605	607
16	100%	4693	287	251	266	31.63	607	607



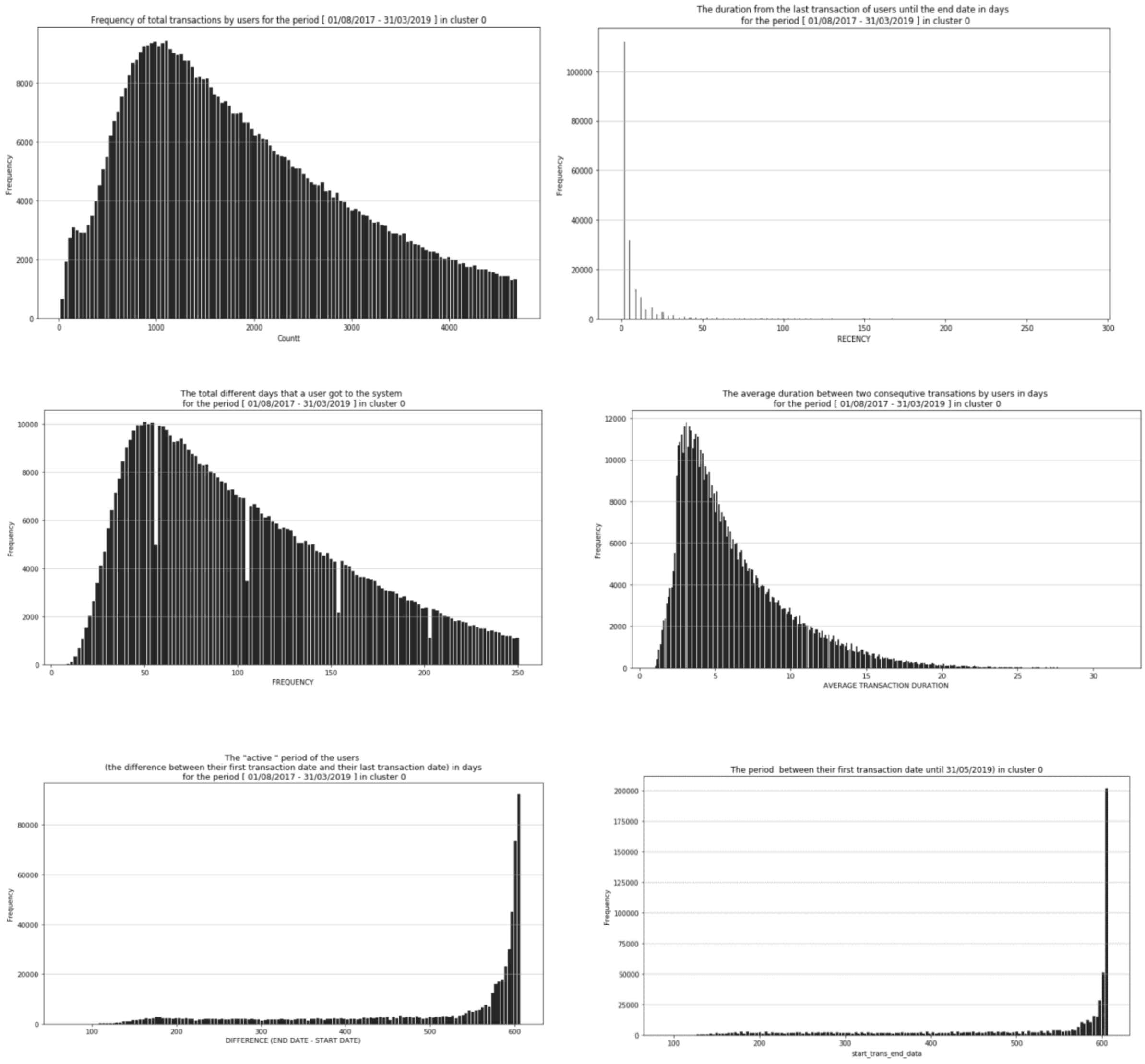
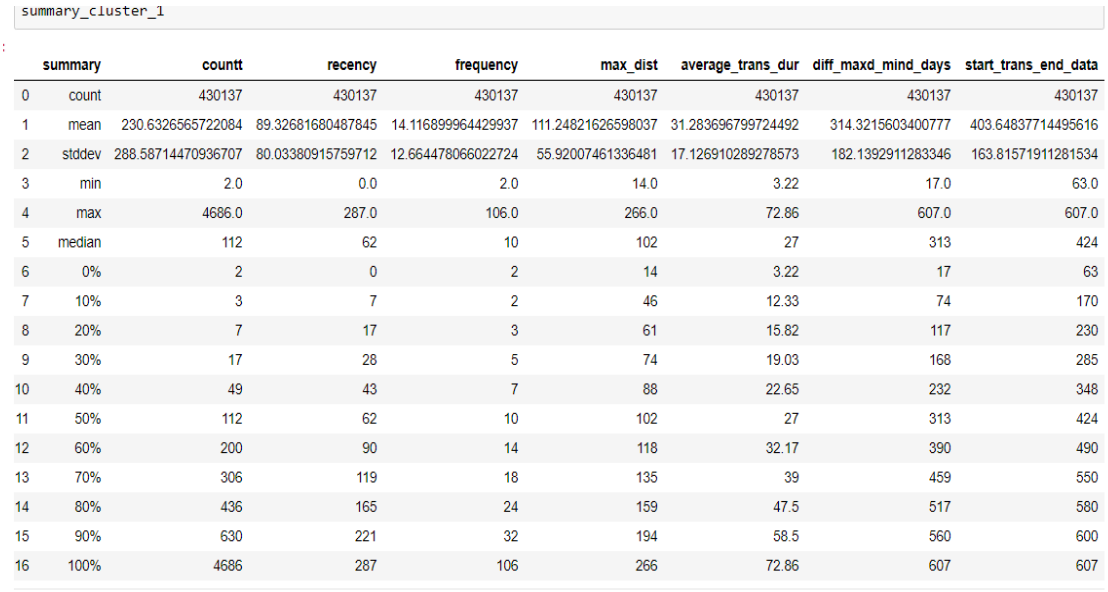


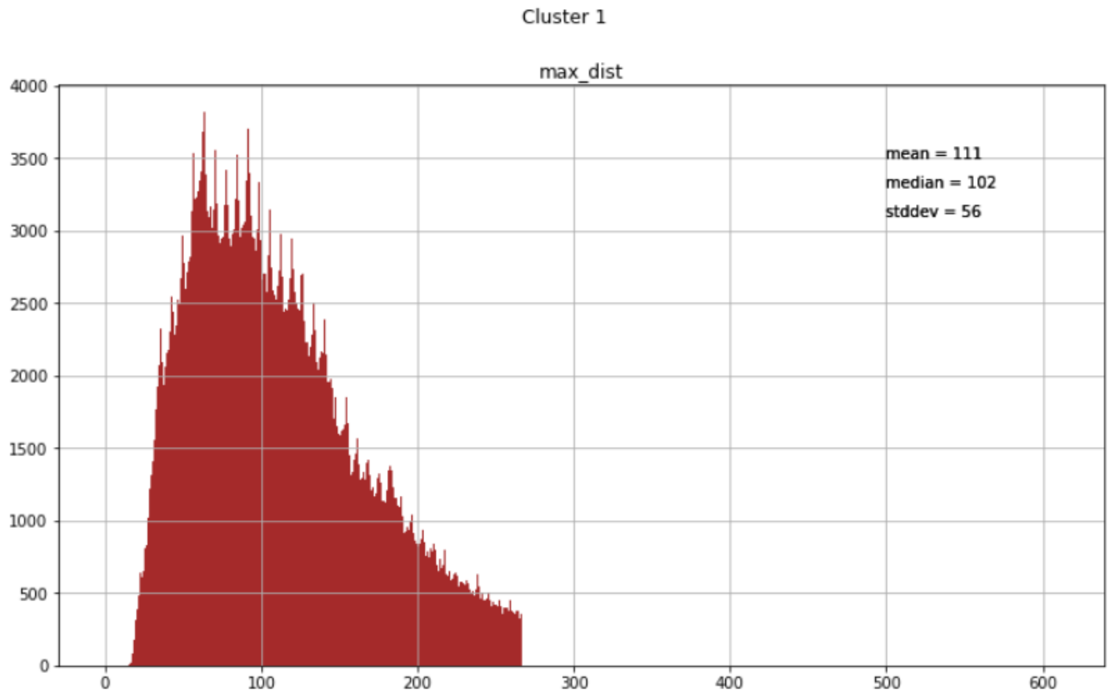
Figure 76: Cluster 0

- Cluster 1 (35,29% of the total data)



```
summary_cluster_1
```

	summary	countt	recency	frequency	max_dist	average_trans_dur	diff_maxd_mind_days	start_trans_end_data
0	count	430137	430137	430137	430137	430137	430137	430137
1	mean	230.6326565722084	89.32681680487845	14.116899964429937	111.24821626598037	31.283696799724492	314.3215603400777	403.64837714495616
2	stddev	288.58714470936707	80.03380915759712	12.664478066022724	55.92007461336481	17.126910289278573	182.1392911283346	163.81571911281534
3	min	2.0	0.0	2.0	14.0	3.22	17.0	63.0
4	max	4686.0	287.0	106.0	266.0	72.86	607.0	607.0
5	median	112	62	10	102	27	313	424
6	0%	2	0	2	14	3.22	17	63
7	10%	3	7	2	46	12.33	74	170
8	20%	7	17	3	61	15.82	117	230
9	30%	17	28	5	74	19.03	168	285
10	40%	49	43	7	88	22.65	232	348
11	50%	112	62	10	102	27	313	424
12	60%	200	90	14	118	32.17	390	490
13	70%	306	119	18	135	39	459	550
14	80%	436	165	24	159	47.5	517	580
15	90%	630	221	32	194	58.5	560	600
16	100%	4686	287	106	266	72.86	607	607



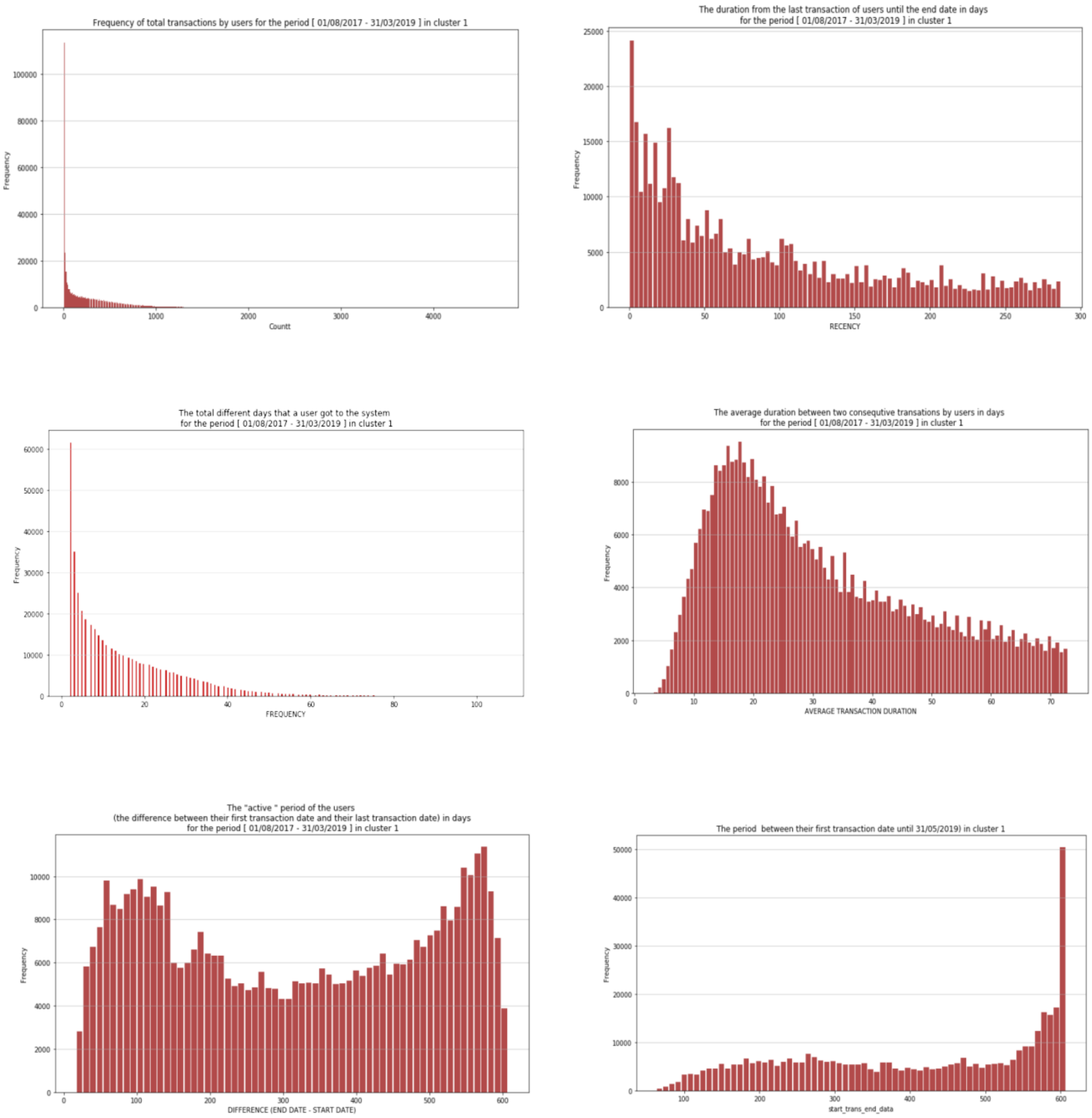
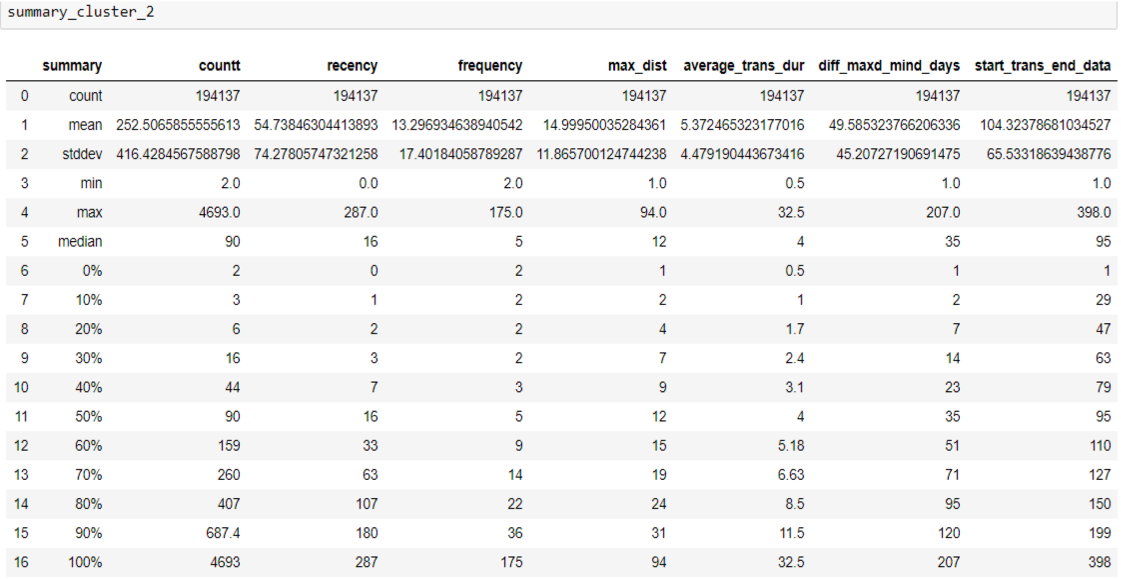


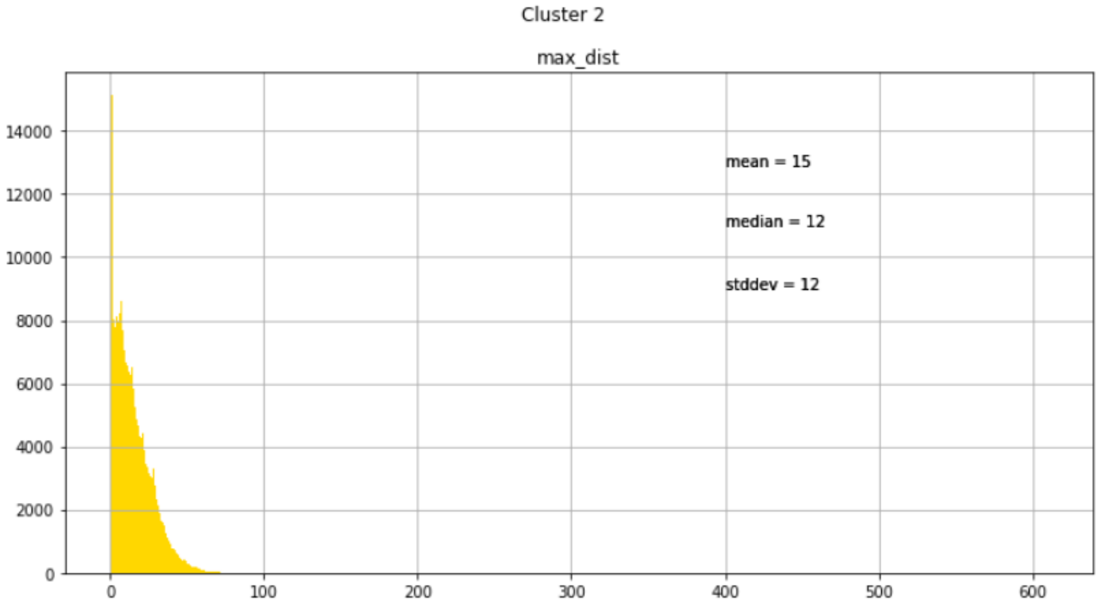
Figure 77: Cluster 1

- Cluster 2 (15,93% of the total data)



```
summary_cluster_2
```

	summary	countt	recency	frequency	max_dist	average_trans_dur	diff_maxd_mind_days	start_trans_end_data
0	count	194137	194137	194137	194137	194137	194137	194137
1	mean	252.5065855555613	54.73846304413893	13.296934638940542	14.99950035284361	5.372465323177016	49.585323766206336	104.32378681034527
2	stddev	416.4284567588798	74.27805747321258	17.40184058789287	11.865700124744238	4.479190443673416	45.20727190691475	65.53318639438776
3	min	2.0	0.0	2.0	1.0	0.5	1.0	1.0
4	max	4693.0	287.0	175.0	94.0	32.5	207.0	398.0
5	median	90	16	5	12	4	35	95
6	0%	2	0	2	1	0.5	1	1
7	10%	3	1	2	2	1	2	29
8	20%	6	2	2	4	1.7	7	47
9	30%	16	3	2	7	2.4	14	63
10	40%	44	7	3	9	3.1	23	79
11	50%	90	16	5	12	4	35	95
12	60%	159	33	9	15	5.18	51	110
13	70%	260	63	14	19	6.63	71	127
14	80%	407	107	22	24	8.5	95	150
15	90%	687.4	180	36	31	11.5	120	199
16	100%	4693	287	175	94	32.5	207	398



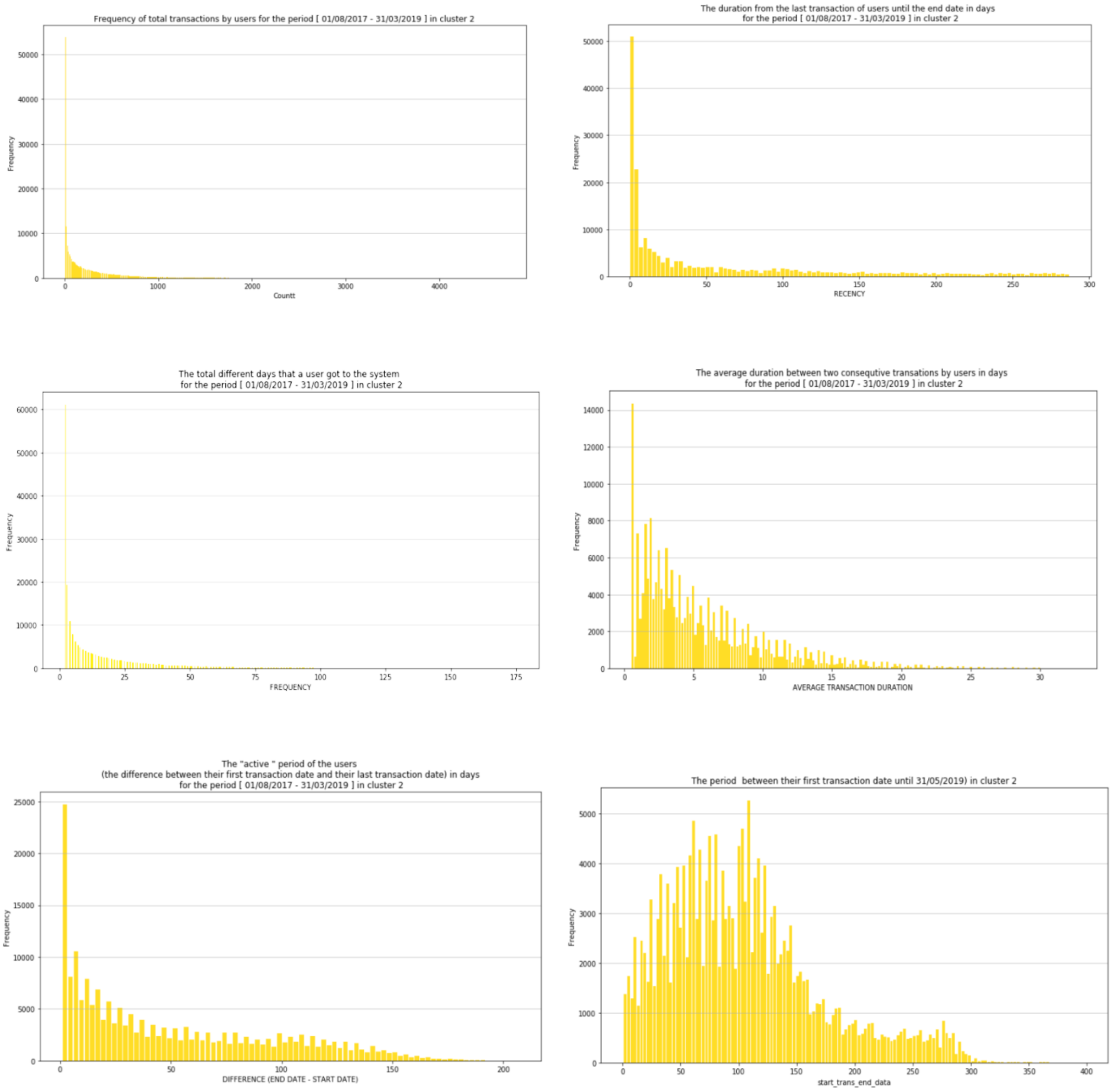


Figure 78: Cluster 2

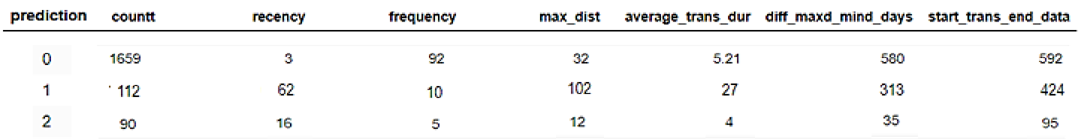
4.2.3.7 Comments and interpretation

Mean value (or average) and median are both statistical terms that play a somewhat similar role in understanding the central tendency of a set of statistical scores. Despite the fact that average has traditionally been more popular measure of a mid-point in a sample than the median, it has the drawback of not being a robust tool. Mean is largely affected by any single value being too high or too low compared to the rest of the sample (outliers). On the contrast, the median is much more robust and sensible. This is the reason in case of having skewness in our data, the median value is more representative than the mean value. The mean is used for normal distributions. (Diffen, 2019)

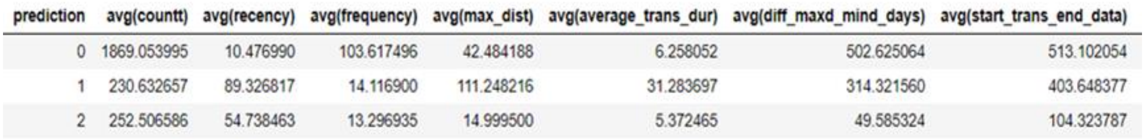
Observing the figures 76, 77 and 78 and more specific the plots, we could say that the distribution of the variable max_dist is characterized by skewness. . Consequently, as far as the variable max_dist is concerned, we took as more representative value for the procedure of extracting the labels, the median of each cluster. We had skew data in the rest of plots too.

To sum up, we have the following descriptive values for the clusters (figure 79):

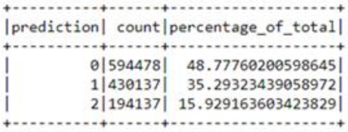
Median



prediction	countt	recency	frequency	max_dist	average_trans_dur	diff_maxd_mind_days	start_trans_end_data
0	1659	3	92	32	5.21	580	592
1	112	62	10	102	27	313	424
2	90	16	5	12	4	35	95



prediction	avg(countt)	avg(recency)	avg(frequency)	avg(max_dist)	avg(average_trans_dur)	avg(diff_maxd_mind_days)	avg(start_trans_end_data)
0	1869.053995	10.476990	103.617496	42.484188	6.258052	502.625064	513.102054
1	230.632657	89.326817	14.116900	111.248216	31.283697	314.321560	403.648377
2	252.506586	54.738463	13.296935	14.999500	5.372465	49.585324	104.323787



```
+----------+------+------------------+
|prediction| count|percentage_of_total|
+----------+------+------------------+
|         0|594478|  48.77760200598645|
|         1|430137|  35.29323439058972|
|         2|194137| 15.929163603423829|
+----------+------+------------------+
```

Figure 79: Median and average values for each cluster

Due to the fact that our data was masked, we did not know which username matches with businesses and which username matches with clients in order to give a good enough interpretation about the members of each category. In addition, we had no information about the money. However, we could make some assumptions. For the understanding of the clusters we based mainly on the variables max distance between two consecutive transactions.

In the cluster 2 we have median (max_dist) = 12. In this cluster would probably belong the following members:

- Independent businesspeople with their suppliers' transactions so as to keep their business running smoothly (for example an owner of a print shop should buy stationery)
- Salaried employees, retirees and civil servants that receive their salary monthly and get to the internet banking every fifteen days so as to meet personal needs.
- Construction workers and dockers that receive a daily or a weekly salary
- Accountants that work on behalf of independent businesspeople in order to arrange the third party payments.

In the cluster 0 we have median (max_dist) = 32. In this cluster would probably belong the following members:

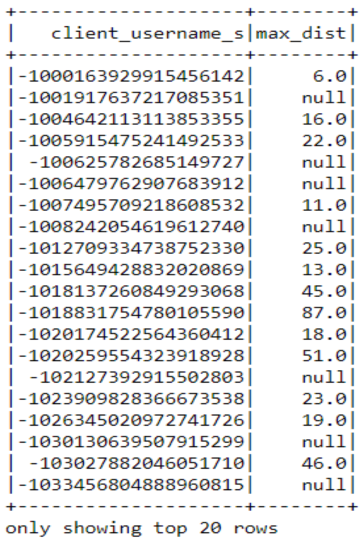
- House owners that receive the payment from their tenants every month
- Salaried employees, retirees and civil servants that get to the internet banking so as to check their salary monthly - account
- Real estate agents, insurance agents, hawkers (door-to-door sellers) and in general individuals that make money based on percentages on sales
- Individuals in a family that work but they do not need to use their account for family payments
- Cooks, waiters and receptionists that work in "season" in hotels and the latter provides them with accommodation and food in their workplace

In the cluster 1 we have median (max_dist) = 102. In this cluster would probably belong the following members:

- Companies of European specifications that receive and spend money from European Union for educational reasons and progress.
- Sailors making long trips that receive their salary at the end of their trips
- Not very active users
- Government investment, funds for educational reasons and progress
- Seasonal farmers that cultivate and market specific products only for some seasons

4.2.4 Data for the last four months (Lead period)

In the lead period we studied again the behavior of the clients and we focused on the variable max_dist in particular. The lead period included the last four months. This included the data with the same features as before but with start date 01/04/2019 and end date 31/07/2019. We executed again the queries that we executed the previous times and we exported our csv file (“churn_results_6_merged.csv”). Again, we read the data and checked for missing values using the function ‘count_results” but we fortunately we did not face null values and this time. The formed pyspark dataframe called “last4months”. From this dataframe we chose (as it was reasonable) only the users with which we worked in the whole clustering phase (joining spark dataframes) and then we selected only the columns with the client usernames and with max_dist. A sample of the output of this procedure was the following (figure 80):

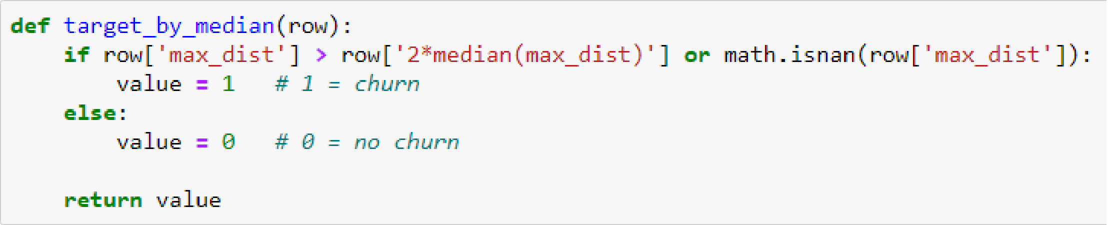


```
+--------------------+--------+
|   client_username_s|max_dist|
+--------------------+--------+
|-1000163929915456142|     6.0|
|-1001917637217085351|    null|
|-1004642113113853355|    16.0|
|-1005915475241492533|    22.0|
| -100625782685149727|    null|
|-1006479762907683912|    null|
|-1007495709218608532|    11.0|
|-1008242054619612740|    null|
|-1012709334738752330|    25.0|
|-1015649428832020869|    13.0|
|-1018137260849293068|    45.0|
|-1018831754780105590|    87.0|
|-1020174522564360412|    18.0|
|-1020259554323918928|    51.0|
| -102127392915502803|    null|
|-1023909828366673538|    23.0|
|-1026345020972741726|    19.0|
|-1030130639507915299|    null|
| -103027882046051710|    46.0|
|-1033456804888960815|    null|
+--------------------+--------+
only showing top 20 rows
```

Figure 80: max_dist for studied users in the period of the last four months

As someone could observe, there were null values in our (joined) dataframe. The null values represented the fact that these clients did not make any transaction in the last four months.

For the phase of extracting the labels (churn\ no churn) we needed to add a column (on the data frame which concerned each cluster with the double of the median value of max_dist of it ('2*median(max_dist)') and the column with the max_dist of each user within the last four months ('max_dist'). As already mentioned, the labels' extraction had to be in each cluster separately because of the different habits. The function that was used for the labels' extraction was the following (figure 81):



```python
def target_by_median(row):
    if row['max_dist'] > row['2*median(max_dist)'] or math.isnan(row['max_dist']):
        value = 1   # 1 = churn
    else:
        value = 0   # 0 = no churn

    return value
```

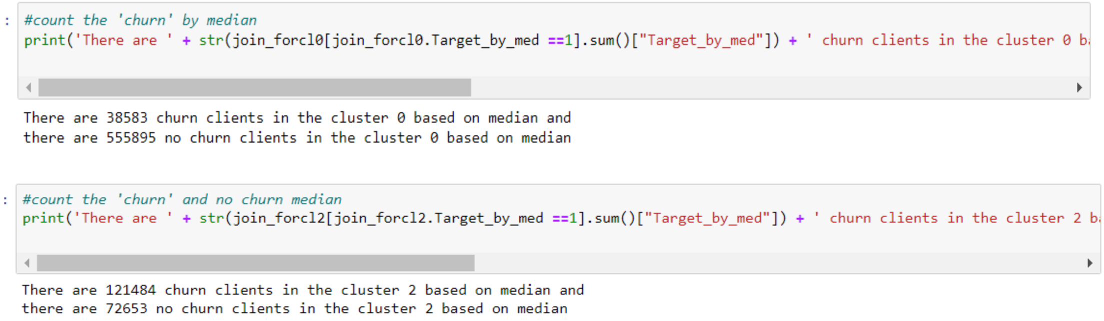
Figure 81: Function for extracting labels

This function gives as output the value 0 (no churner) for each user, if his max_dist value, as far as the last four month period is concerned, does not overlap the double value of the of cluster’s median max_dist which the user belongs to. On the contrast, the output is 1 (churner) if the user’s “new max_dist” value is larger than the latter. In other words, by natural interpretation, the user is characterized as churner if there is an anomaly in comparison with the cluster’s pattern in which the user belongs to.

While this function was applied successfully on the clusters 0 and 2, it could not be applied for the cluster 1 due to the large value of its median. Cluster 1 has median (max_dist) = 102 (days). This means that the double value of this one, i.e. its lead period is 204 days. Our window- lead period was 4 months = 121 days < 204 days. As a result, we could not infer something for this cluster.

The problem that we faced in that point was what to do with this cluster. The first option was not to deal with it at all because it was too early so as to decide which users belonged to it are churners or not churners. The second option was to characterize all users of it as churners on the grounds that they are bad customers. The third option was to characterize all users of it as no churners based on the fact that until 31/07/2019 which is the last date of our total data, these users are theoretically “active” clients and have not severed their relationship with the bank in any formal way. But, the last two options would offer an extra weight in one of two categories respectively and consequently we did not know if our predictive model in the classification phase would have a good perform owing to the bias of the model. From all the above we decided to drop the cluster 1.

Finally, after applying the function in figure 81 on the clusters 0 and 2 and then computing the number of churners and not churners in each cluster, we got the following results (figure 82) for each cluster and for the total (figure 83):



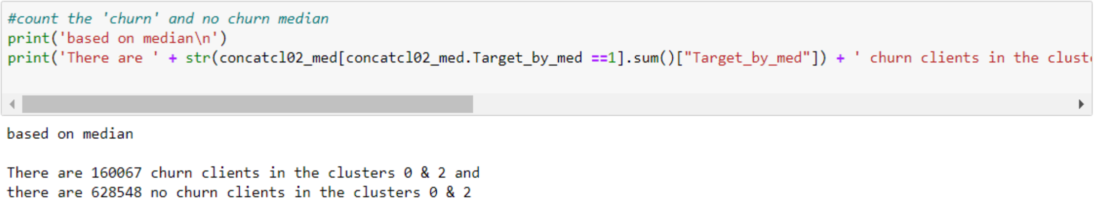
```
#count the 'churn' by median
print('There are ' + str(join_forcl0[join_forcl0.Target_by_med ==1].sum()["Target_by_med"]) + ' churn clients in the cluster 0 b
```

```
There are 38583 churn clients in the cluster 0 based on median and
there are 555895 no churn clients in the cluster 0 based on median
```

```
#count the 'churn' and no churn median
print('There are ' + str(join_forcl2[join_forcl2.Target_by_med ==1].sum()["Target_by_med"]) + ' churn clients in the cluster 2 b
```

```
There are 121484 churn clients in the cluster 2 based on median and
there are 72653 no churn clients in the cluster 2 based on median
```

Figure 82: Number of churners/not churners in each cluster



```
#count the 'churn' and no churn median
print('based on median\n')
print('There are ' + str(concatcl02_med[concatcl02_med.Target_by_med ==1].sum()["Target_by_med"]) + ' churn clients in the clust
```

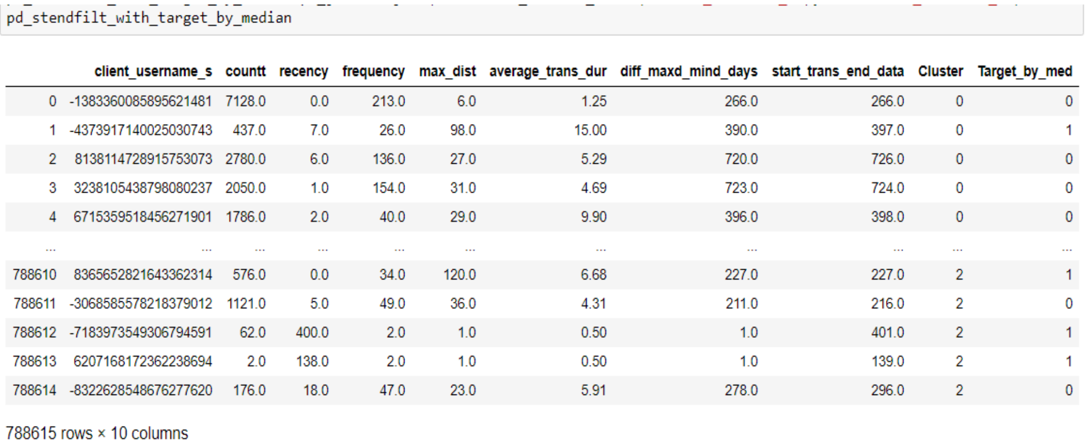
```
based on median

There are 160067 churn clients in the clusters 0 & 2 and
there are 628548 no churn clients in the clusters 0 & 2
```

Figure 83: Number of churners /not churners in total

Note: Something that we observed was that in cluster 2 which represent the “good” clients for the period from 1/04/2019 until 31/07/2019, the churners were more than the non-churners.

Having our disposal the desired labels, we concatenated them along with the cluster membership to our initial dataframe concerned the whole two year period (“stendfilt”) as we had mentioned in the page 46.

The final number of clients that we would work with from now on was 788.615. (We added the clients from clusters 0 and 2). Our total pandas dataframe called “pd_stendfilt_with_target_by_median” had the following form (figure 84):



```
pd_stendfilt_with_target_by_median
```

	client_username_s	countt	recency	frequency	max_dist	average_trans_dur	diff_maxd_mind_days	start_trans_end_data	Cluster	Target_by_med
0	-1383360085895621481	7128.0	0.0	213.0	6.0	1.25	266.0	266.0	0	0
1	-4373917140025030743	437.0	7.0	26.0	98.0	15.00	390.0	397.0	0	1
2	8138114728915753073	2780.0	6.0	136.0	27.0	5.29	720.0	726.0	0	0
3	3238105438798080237	2050.0	1.0	154.0	31.0	4.69	723.0	724.0	0	0
4	6715359518456271901	1786.0	2.0	40.0	29.0	9.90	396.0	398.0	0	0
...	...	...	...	...	...	...	...	...	...	...
788610	8365652821643362314	576.0	0.0	34.0	120.0	6.68	227.0	227.0	2	1
788611	-3068585578218379012	1121.0	5.0	49.0	36.0	4.31	211.0	216.0	2	0
788612	-7183973549306794591	62.0	400.0	2.0	1.0	0.50	1.0	401.0	2	1
788613	6207168172362238694	2.0	138.0	2.0	1.0	0.50	1.0	139.0	2	1
788614	-8322628548676277620	176.0	18.0	47.0	23.0	5.91	278.0	296.0	2	0

788615 rows × 10 columns

Figure 84: Total dataframe with labels

The final step before the classification task, was extracting the dataframes, the one concerning the whole two years and the other two concerning each cluster to csv files with names:

“data_for_classific_median_fromubuntu.csv”,

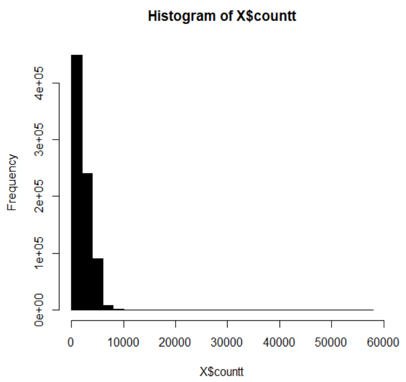
“data_for_classific_cl0median_fromubuntu.csv”,

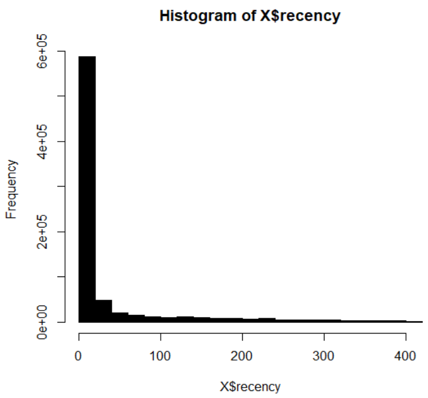
“data_for_classific_cl2median_fromubuntu.csv” respectively.

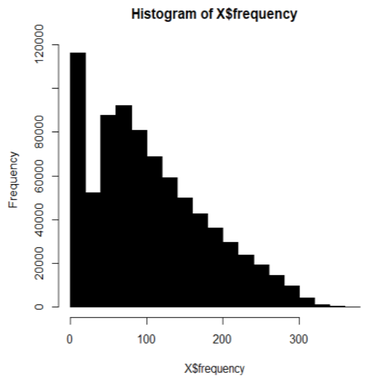
4.2.5 Whole data with labels

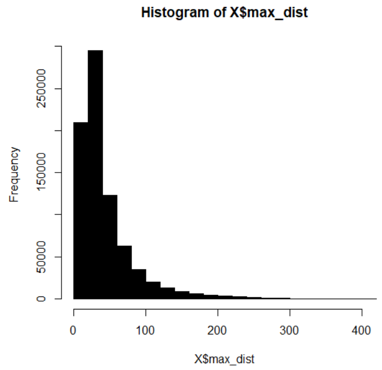
4.2.5.1 Summary and visualization

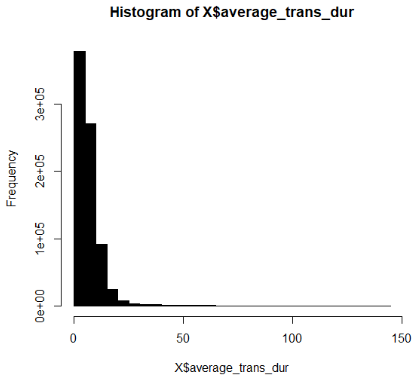
Reading the exported file for the whole two years from Ubuntu we displayed some histograms on R studio and some descriptive measures in anaconda jupyter notebook which are presented below (figure 85):

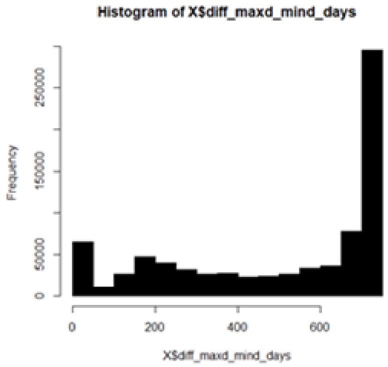

Histogram of X$countt
Frequency
0e+00
1e+05
2e+05
3e+05
4e+05
0
10000
20000
30000
40000
50000
60000
X$countt

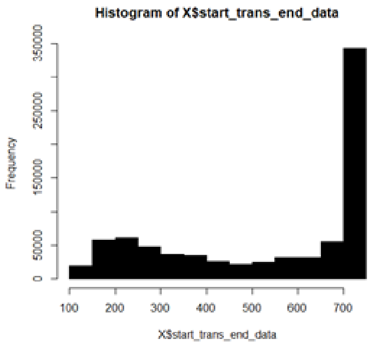

Histogram of X$recency
Frequency
0e+00
2e+05
4e+05
6e+05
0
100
200
300
400
X$recency

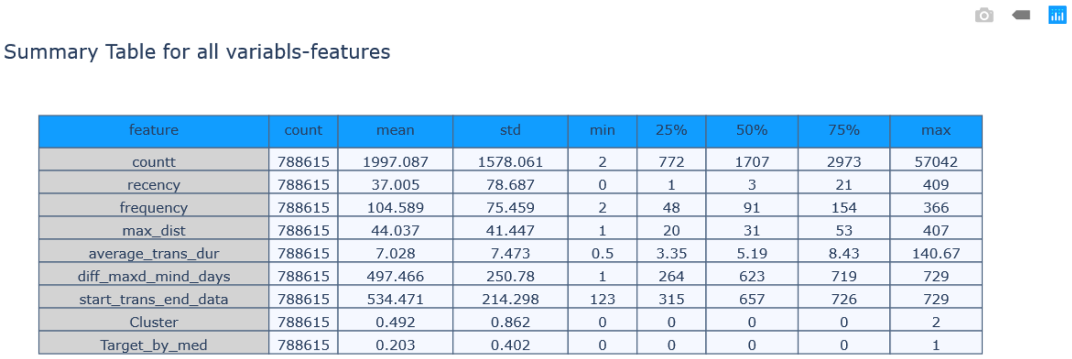

Histogram of X$frequency
Frequency
0
20000
40000
60000
80000
120000
0
100
200
300
X$frequency


Histogram of X$max_dist
Frequency
0
50000
150000
250000
0
100
200
300
400
X$max_dist


Histogram of X$average_trans_dur
Frequency
0e+00
1e+05
2e+05
3e+05
0
50
100
150
X$average_trans_dur


Histogram of X$diff_maxd_mind_days
Frequency
0
50000
150000
250000
0
200
400
600
X$diff_maxd_mind_days


Histogram of X$start_trans_end_data
Frequency
0
50000
150000
250000
350000
100
200
300
400
500
600
700
X$start_trans_end_data



Summary Table for all variabls-features

feature	count	mean	std	min	25%	50%	75%	max
countt	788615	1997.087	1578.061	2	772	1707	2973	57042
recency	788615	37.005	78.687	0	1	3	21	409
frequency	788615	104.589	75.459	2	48	91	154	366
max_dist	788615	44.037	41.447	1	20	31	53	407
average_trans_dur	788615	7.028	7.473	0.5	3.35	5.19	8.43	140.67
diff_maxd_mind_days	788615	497.466	250.78	1	264	623	719	729
start_trans_end_data	788615	534.471	214.298	123	315	657	726	729
Cluster	788615	0.492	0.862	0	0	0	0	2
Target_by_med	788615	0.203	0.402	0	0	0	0	1

Figure 85: Histograms/Statistics for the whole data with labels

Observing the histograms in figure 85 we could say that the main characteristic is the large skewness again. The data are gathered either on the left side of the graph or on the right.

In figures 86 and 87 we displayed two barplots about the percentage of clients who are churn/no churn. The first figure concerns each cluster separately wheras the second figure concerns the total client base:

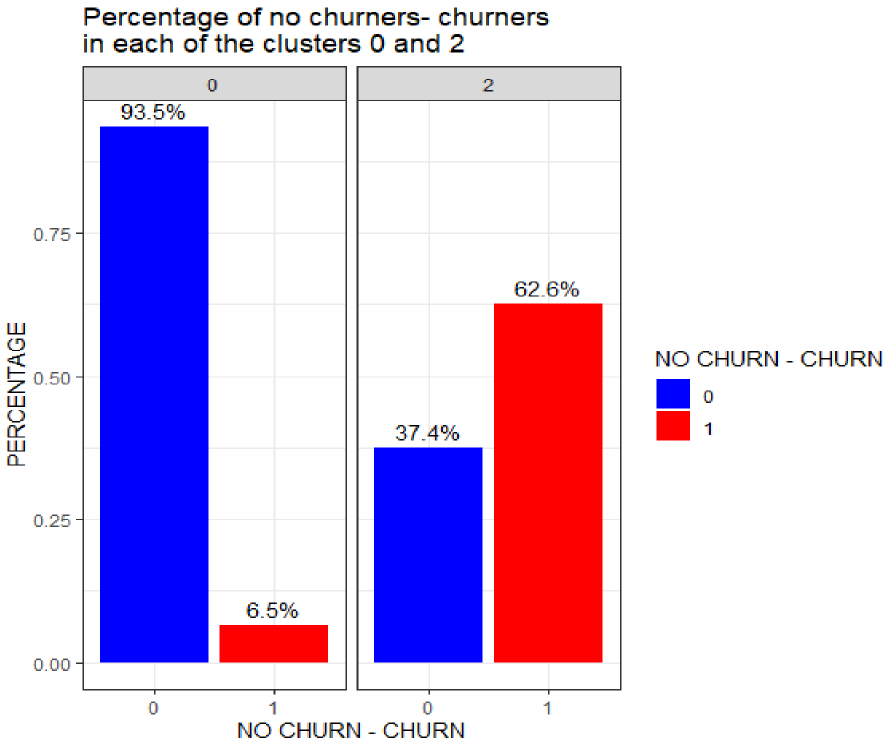


Figure 86: Percentage of churn/no churn in each cluster

From the barplot of figure 86 we see that the percentage of churners (red color) in cluster 0 is only 6,5% (small percentage) whereas in cluster 2 it surpasses the fifty percent of the total clients in cluster 2 as it is 62,6%, i.e. it exceeds the percentage of non-churners (blue color) (something that is absolutely unwanted). There is an absolute contradiction between the two clusters. Talking about absolute values, the churn clients in cluster 0 are 38.583 whereas the non-churners are 555.895 and in the cluster 2 the churners are 121.484 whereas the non-churners are 72.653.

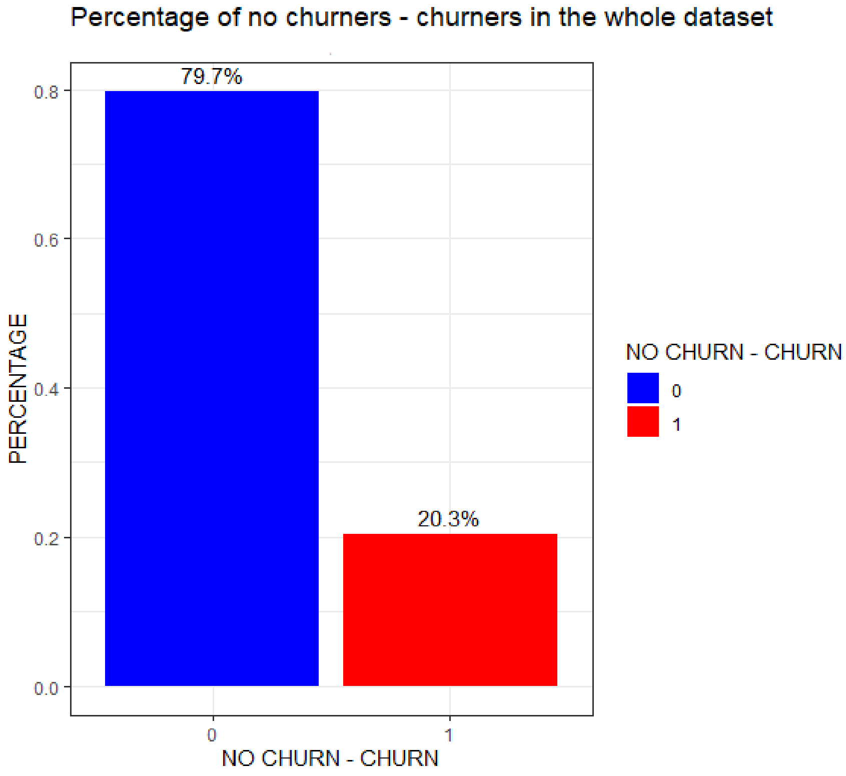


Figure 87: Percentage of churn/no churn in total

From the barplot of figure 87 we could see that the percentage of churners in the whole two years data is 20,3% (1/5 of the totality) whereas the respective one of non-churners is a 79,7%. Talking about absolute values, the churn clients are 160.067 whereas the non churn clients are 628.548 in total (figure 83).

In windows jupyter notebook we executed some interactive histograms (using the library plotly) for the whole dataset and for each cluster separately. The outputs are following:

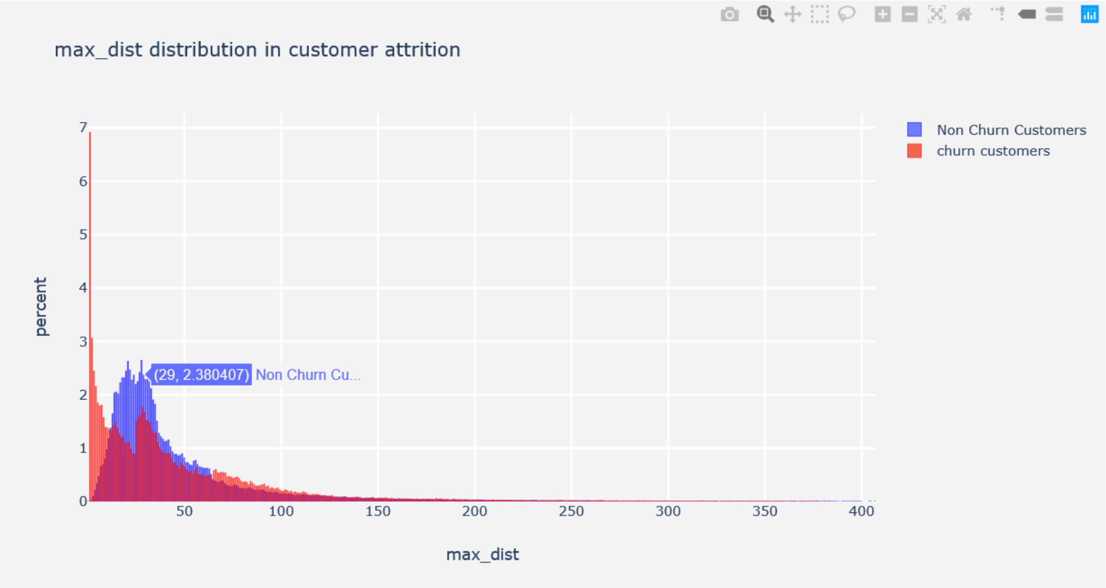


Figure 88: Interactive histogram for Max_dist in total with groups

The figure 88 is a screenshot of the respective interactive histogram for the variable max_dist for each group in the whole data. The histogram with the red color concerns the distribution of churners whereas the histogram with the blue color concerns the distribution of non-churners. The blue box is the point in which we placed the computer mouse in jupyter visual environment. The abscissa shows the absolute value of the max_dist and the ordinate shows the percentage of the respective group (no-churners if the color is blue and churners if the color is red) that have the specific value. In general, we could observe that in the range [12,63] of the feature max distance, the higher percentages belong to no churn clients. Something that interested us is that for the more recent time period between 1 to 10 days the percentage of churners is higher than the respective one of no churn clients. For this reason we executed the same plot for the same variable for each cluster separately.

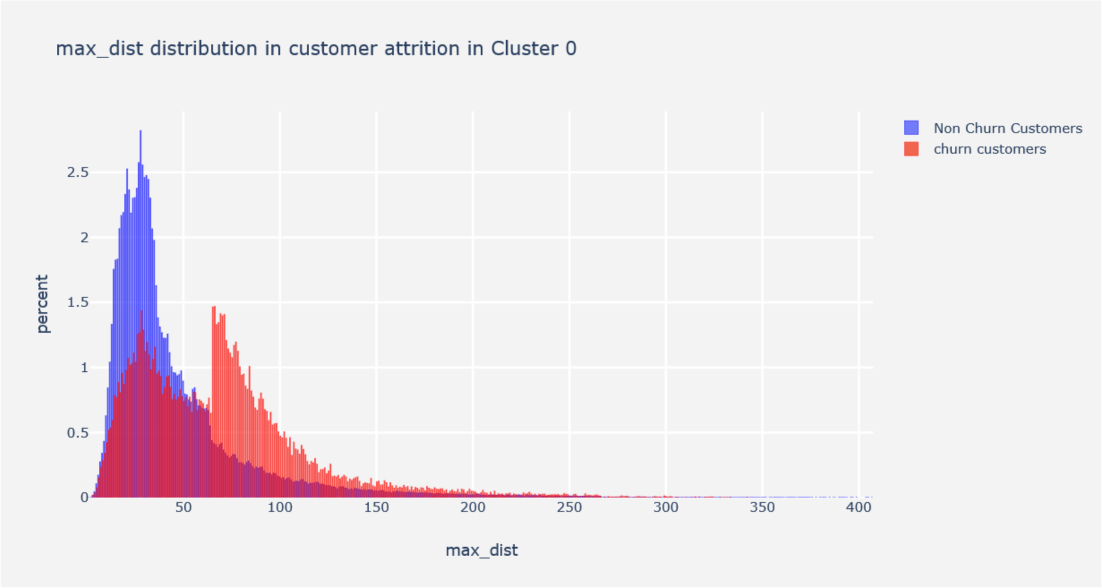


Figure 89: Histogram for max_dist in cluster 0 for each group

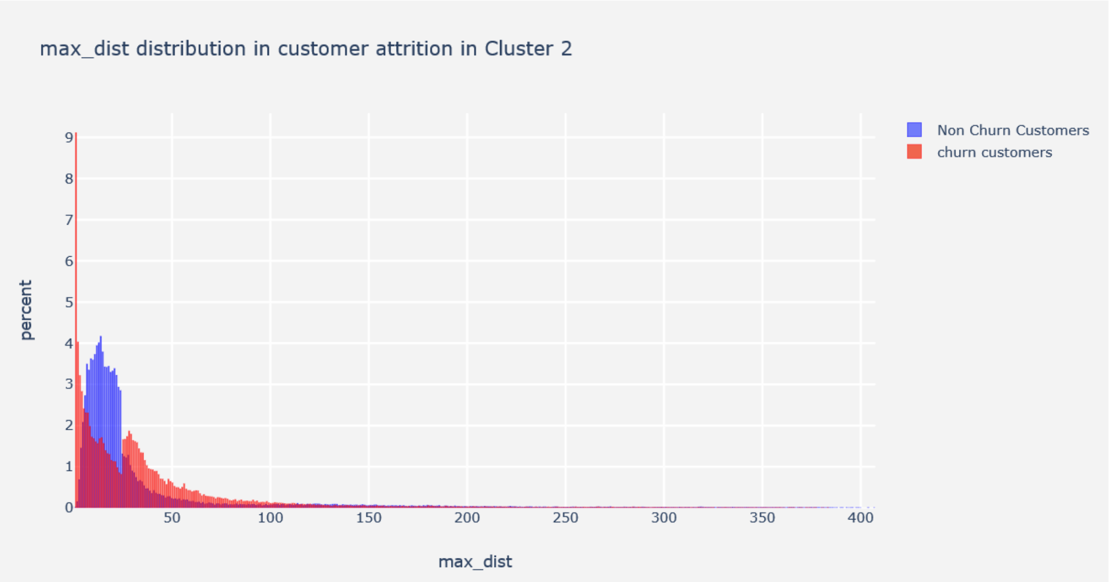


Figure 90: Histogram for max_dist in cluster 2 for each group

Observing the figure 90 we could justify the previous note. In addition, observing the figure 89, even though the no-churners in cluster 0 for the range [1,10] have larger percentage, the highest value is close to 1. As a result, combining the two clusters we Certainly understand the “paradox”.

4.2.5.2 Classification

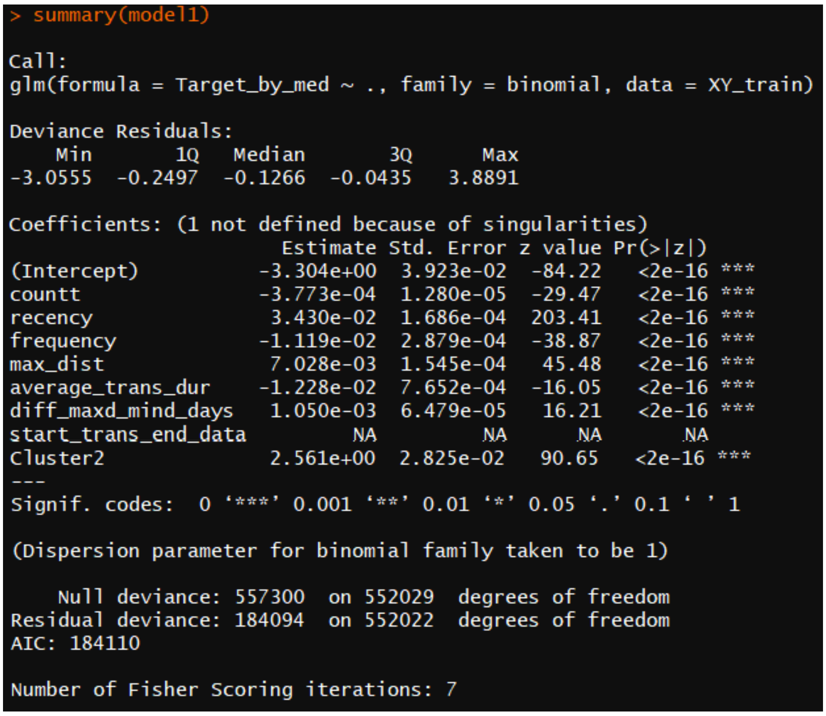
So as to execute the classification (on R-studio) we first split the data to training set ("XY_train"), development set ("XY_dev") and test set ("XY_test") (with their respective labels). We separated the target column ("Target_by_med") from each of the three extracted data sets creating the dataframes Y_train, Y_dev and Y_test respectively. Choosing the variables "countt", "recency", "frequency", "max_dist", "average_trans_dur", "diff_maxd_mind_days", "start_trans_end_data" and "Cluster" we were ready to run our first classifier the logistic regression. We executed 2 approaches for the logistic regression, the first with the initial data and the second with the respective log data.

4.2.5.2.1 Logistic regression

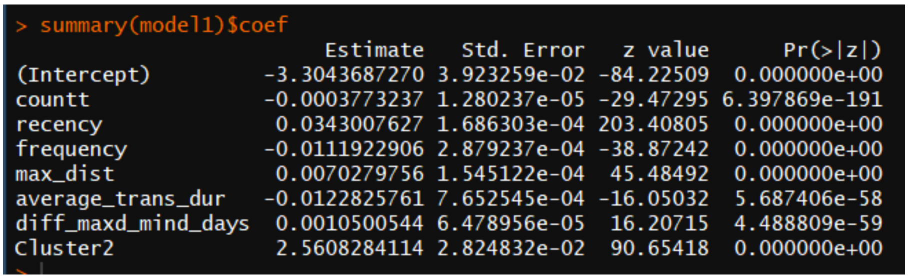
1st approach

a) full model

After fitting our training data to our model, displaying a summary of it and the command "summary(model1)$coef" for the coefficients of the model's variables we got the following outputs (figure 91):



```
> summary(model1)

Call:
glm(formula = Target_by_med ~ ., family = binomial, data = XY_train)

Deviance Residuals: 
    Min       1Q   Median       3Q      Max  
-3.0555  -0.2497  -0.1266  -0.0435   3.8891  

Coefficients: (1 not defined because of singularities)
                      Estimate Std. Error z value Pr(>|z|)    
(Intercept)         -3.304e+00  3.923e-02  -84.22   <2e-16 ***
countt              -3.773e-04  1.280e-05  -29.47   <2e-16 ***
recency              3.430e-02  1.686e-04  203.41   <2e-16 ***
frequency           -1.119e-02  2.879e-04  -38.87   <2e-16 ***
max_dist             7.028e-03  1.545e-04   45.48   <2e-16 ***
average_trans_dur   -1.228e-02  7.652e-04  -16.05   <2e-16 ***
diff_maxd_mind_days  1.050e-03  6.479e-05   16.21   <2e-16 ***
start_trans_end_data        NA         NA      NA       NA    
Cluster2             2.561e+00  2.825e-02   90.65   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 557300  on 552029  degrees of freedom
Residual deviance: 184094  on 552022  degrees of freedom
AIC: 184110

Number of Fisher Scoring iterations: 7
```



```
> summary(model1)$coef
                      Estimate   Std. Error   z value      Pr(>|z|)
(Intercept)        -3.3043687270 3.923259e-02 -84.22509  0.000000e+00
countt             -0.0003773237 1.280237e-05 -29.47295 6.397869e-191
recency             0.0343007627 1.686303e-04 203.40805  0.000000e+00
frequency          -0.0111922906 2.879237e-04 -38.87242  0.000000e+00
max_dist            0.0070279756 1.545122e-04  45.48492  0.000000e+00
average_trans_dur  -0.0122825761 7.652545e-04 -16.05032  5.687406e-58
diff_maxd_mind_days 0.0010500544 6.478956e-05  16.20715  4.488809e-59
Cluster2            2.5608284114 2.824832e-02  90.65418  0.000000e+00
```

Figure 91: Summary of the model logit

From the figure 91 in the first output we observed that the feature "start_trans_end_data" has "NA" in the columns instead of values. This means that there is a strong correlation between our independent variables. In the second image of the figure 91 we see that the model itself dropped the specific variable. In order to see with which variable the feature "start_trans_end_data" is highly correlated, we created an interactive correlation matrix (using plotly library) in jupyter and we found that this variable is very high correlated (corr = 0,95) with the variable diff_maxd_mind_days (figure 92):

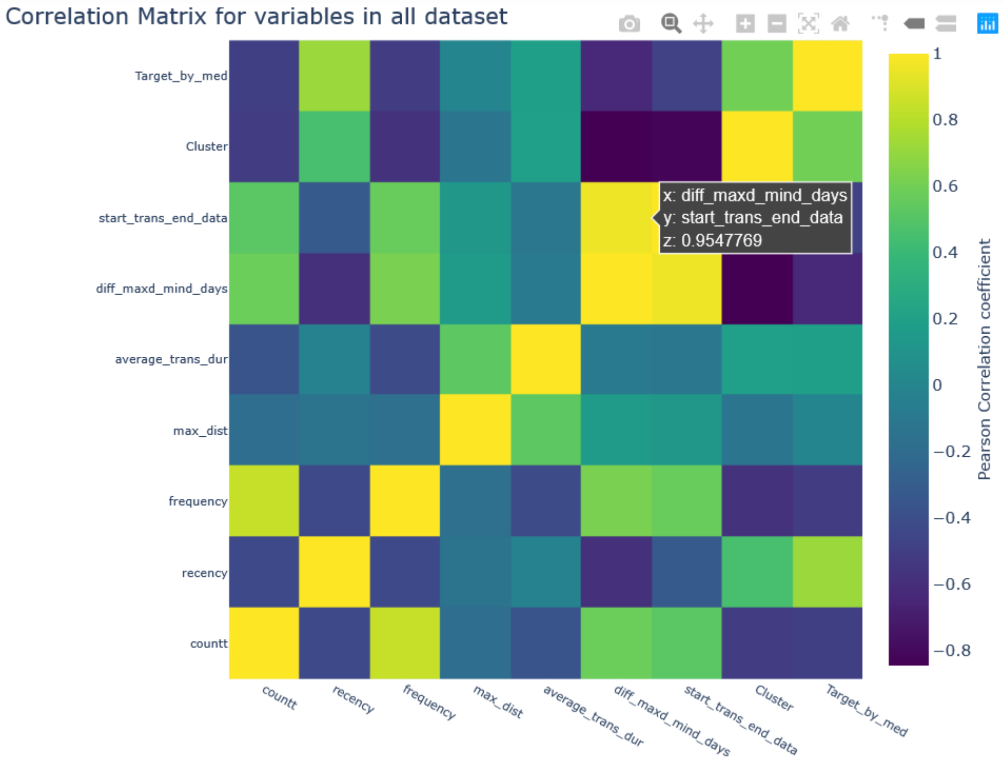
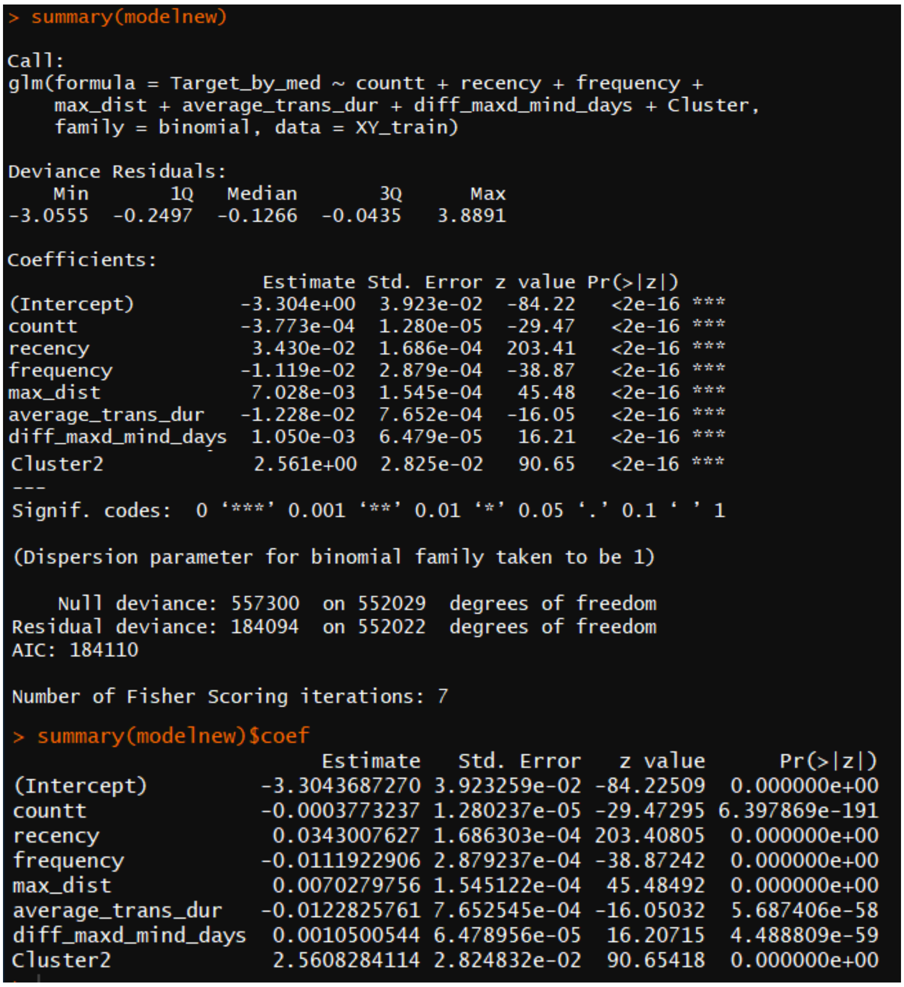


Figure 92: Correlation matrix (for the logit)

So, we fitted our model without this variable. The new outputs executing the same commands were the following (figure 93):

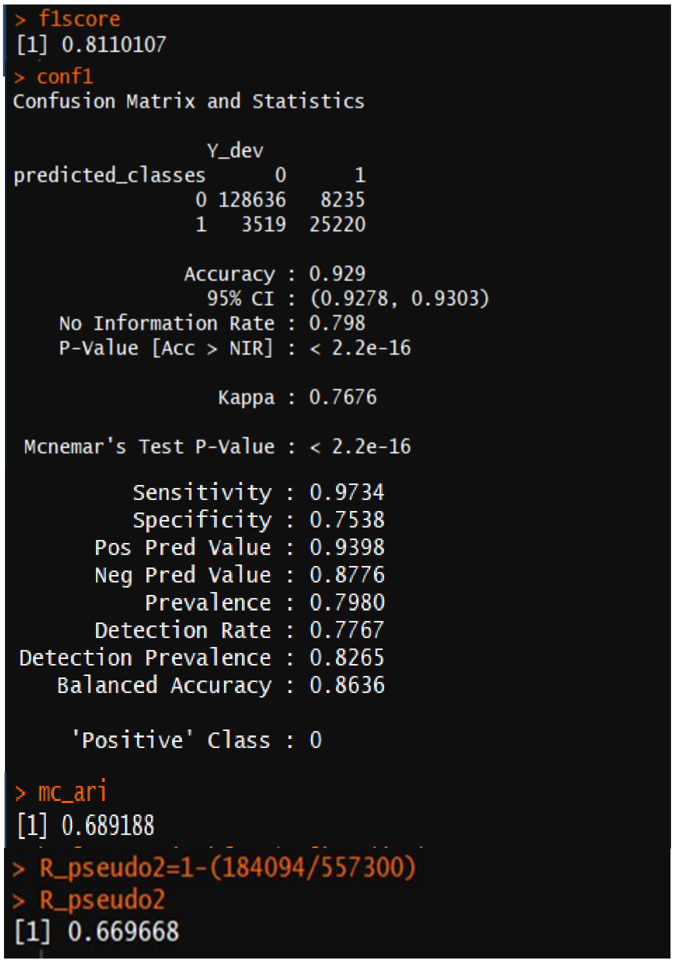


```
> summary(modelnew)

Call:
glm(formula = Target_by_med ~ countt + recency + frequency +
    max_dist + average_trans_dur + diff_maxd_mind_days + Cluster,
    family = binomial, data = XY_train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.0555  -0.2497  -0.1266  -0.0435   3.8891

Coefficients:
                     Estimate Std. Error z value Pr(>|z|)
(Intercept)        -3.304e+00  3.923e-02  -84.22   <2e-16 ***
countt             -3.773e-04  1.280e-05  -29.47   <2e-16 ***
recency             3.430e-02  1.686e-04  203.41   <2e-16 ***
frequency          -1.119e-02  2.879e-04  -38.87   <2e-16 ***
max_dist            7.028e-03  1.545e-04   45.48   <2e-16 ***
average_trans_dur  -1.228e-02  7.652e-04  -16.05   <2e-16 ***
diff_maxd_mind_days 1.050e-03  6.479e-05   16.21   <2e-16 ***
Cluster2            2.561e+00  2.825e-02   90.65   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 557300  on 552029  degrees of freedom
Residual deviance: 184094  on 552022  degrees of freedom
AIC: 184110

Number of Fisher Scoring iterations: 7

> summary(modelnew)$coef
                        Estimate   Std. Error   z value      Pr(>|z|)
(Intercept)         -3.3043687270 3.923259e-02 -84.22509  0.000000e+00
countt              -0.0003773237 1.280237e-05 -29.47295 6.397869e-191
recency              0.0343007627 1.686303e-04 203.40805  0.000000e+00
frequency           -0.0111922906 2.879237e-04 -38.87242  0.000000e+00
max_dist             0.0070279756 1.545122e-04  45.48492  0.000000e+00
average_trans_dur   -0.0122825761 7.652545e-04 -16.05032  5.687406e-58
diff_maxd_mind_days  0.0010500544 6.478956e-05  16.20715  4.488809e-59
Cluster2             2.5608284114 2.824832e-02  90.65418  0.000000e+00
```

Figure 93: Summary of the modelnew logit

From the figure 93, observing the estimated values for the coefficients of the input features and their standard errors (small values) we could say that the estimations are consistent enough. Also, we could say the variables recency, max_dist and diff_maxd_mind_days have positive influence the target variable (positive sign) whereas the rest have negative (negative sign). As far as the deviance is concerned, which is a measure of goodness of fit of a generalized linear model, we could say that there is a big reduction as from 557300 (with only intercept term) reduced to 184094 (by adding the features).

After fitting the model, we continued to prediction step on the development set. In order to examine the performance of our model we computed the f1 score, the confusion matrix and the ARI. The results-outputs were the following (figures 94):



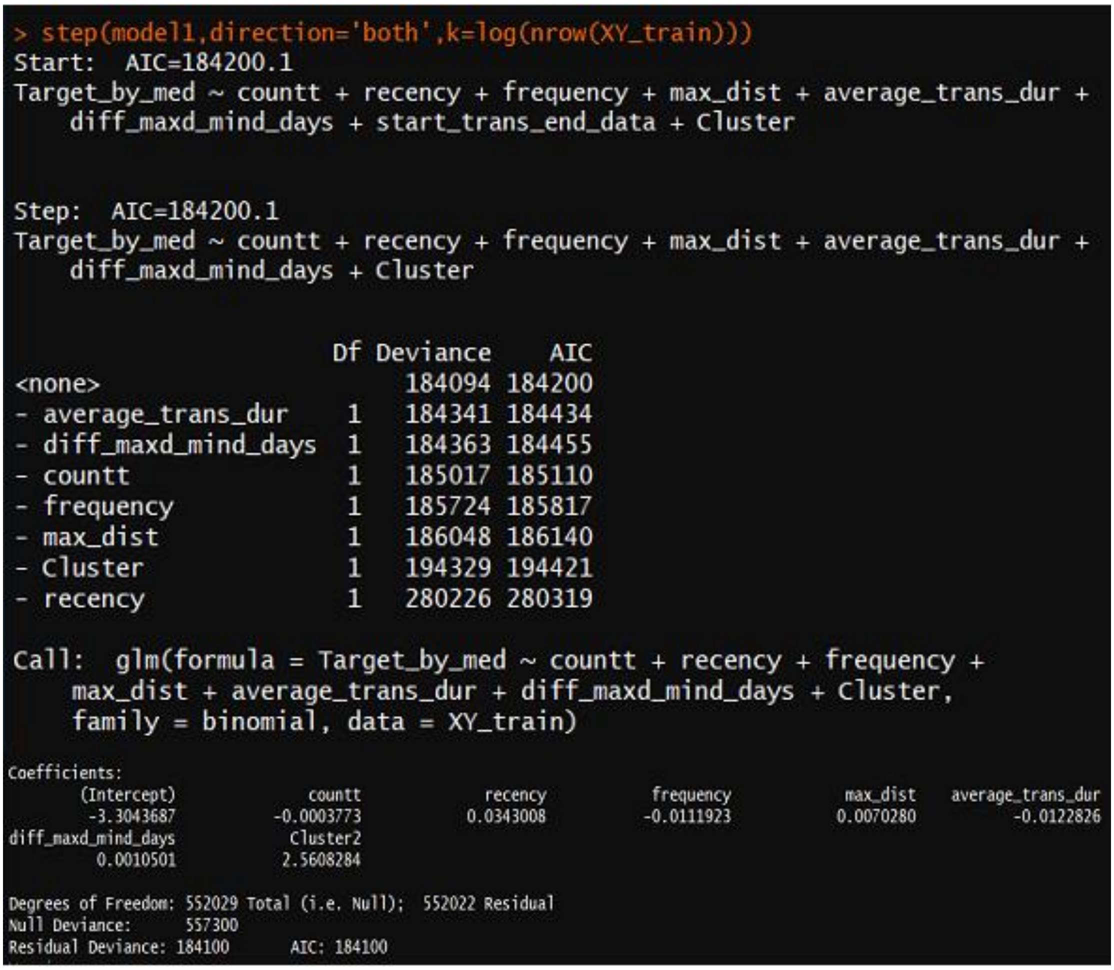
```
> f1score
[1] 0.8110107
> conf1
Confusion Matrix and Statistics

                 Y_dev
predicted_classes      0      1
                0 128636   8235
                1   3519  25220

               Accuracy : 0.929
                 95% CI : (0.9278, 0.9303)
    No Information Rate : 0.798
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7676

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9734
            Specificity : 0.7538
         Pos Pred Value : 0.9398
         Neg Pred Value : 0.8776
             Prevalence : 0.7980
         Detection Rate : 0.7767
   Detection Prevalence : 0.8265
      Balanced Accuracy : 0.8636

       'Positive' Class : 0

> mc_ari
[1] 0.689188
> R_pseudo2=1-(184094/557300)
> R_pseudo2
[1] 0.669668
```

Figure 94: performance of the modelnew logit

From the figure 94 we could say that the f1 score is 0.81, a good enough score for the performance of the model. From the confusion matrix we observe that our model predicted correctly 128.636 non churners and 25.220 churners. In addition, our model estimated incorrectly 11.754 customers as it characterized 3.519 non churners as churners and 8.235 churners as non-churners. The ARI score of the model is 0,689. ARI score has values in range [-1,1] and the value 1 is equivalent to perfect agreement. If 0.65 =< ARI < 0.80 then we say that we have moderate recovery. So, we could say that we have a mediocre agreement in our model. As far as the pseudo R squared is concerned, ideally we would like to have a value very close to 1 so as to say that our model has a very good predictive ability. Observing the output, we could say that the predictive ability of our model is very good.

b) feature selection

We executed feature selection using the stepwise regression defining as direction = both (forward and backward), but the results showed us that besides the variable start_trans_end_data” all the rest values are important for our model. As a result we went to the previous procedure as could someone see in the figure 95:



```
> step(model1,direction='both',k=log(nrow(XY_train)))
Start:  AIC=184200.1
Target_by_med ~ countt + recency + frequency + max_dist + average_trans_dur + 
    diff_maxd_mind_days + start_trans_end_data + Cluster


Step:  AIC=184200.1
Target_by_med ~ countt + recency + frequency + max_dist + average_trans_dur + 
    diff_maxd_mind_days + Cluster


                      Df Deviance    AIC
<none>                     184094 184200
- average_trans_dur    1   184341 184434
- diff_maxd_mind_days  1   184363 184455
- countt               1   185017 185110
- frequency            1   185724 185817
- max_dist             1   186048 186140
- Cluster              1   194329 194421
- recency              1   280226 280319

Call:  glm(formula = Target_by_med ~ countt + recency + frequency + 
    max_dist + average_trans_dur + diff_maxd_mind_days + Cluster, 
    family = binomial, data = XY_train)

Coefficients:
        (Intercept)               countt              recency            frequency             max_dist    average_trans_dur  
         -3.3043687           -0.0003773            0.0343008           -0.0111923            0.0070280           -0.0122826  
diff_maxd_mind_days             Cluster2  
          0.0010501            2.5608284  

Degrees of Freedom: 552029 Total (i.e. Null);  552022 Residual
Null Deviance:      557300 
Residual Deviance: 184100       AIC: 184100
```

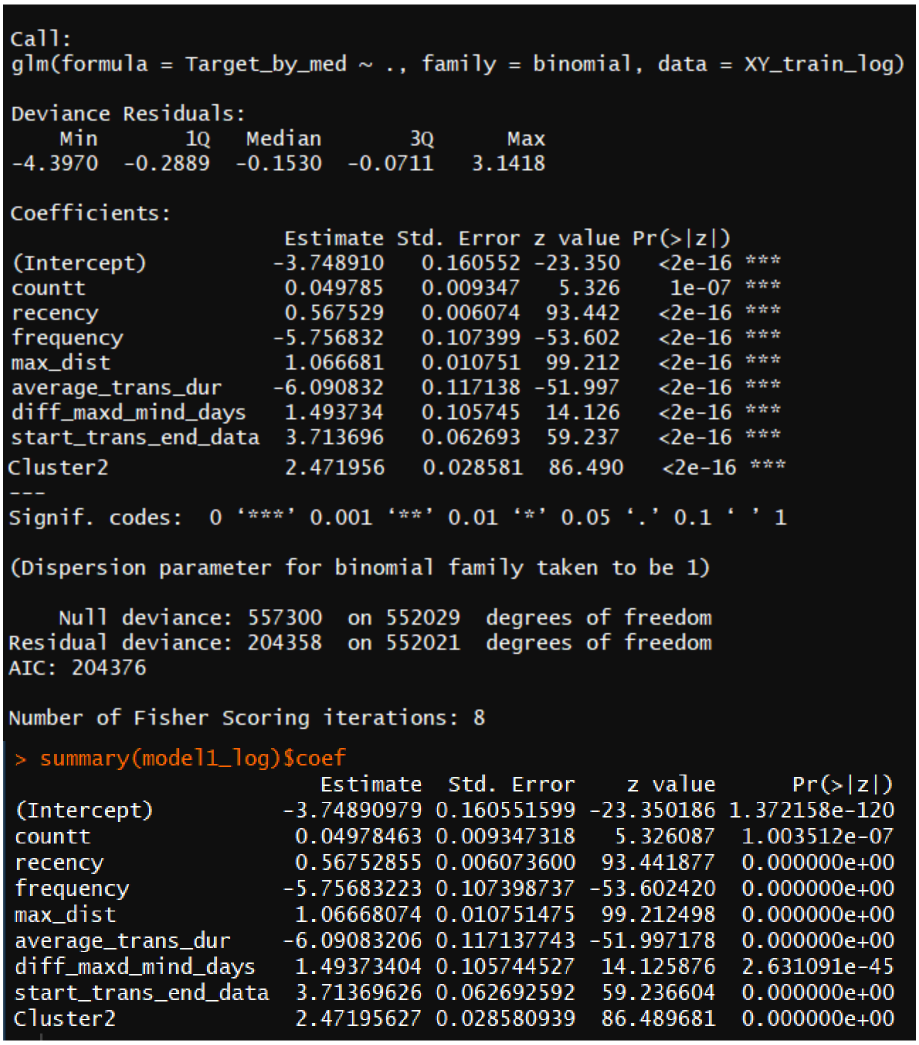
Figure 95: stepwise regression

From the figure 95 we see the AIC made no difference while dropping the variable “start_trans_end_data”.

2nd approach

a) full model

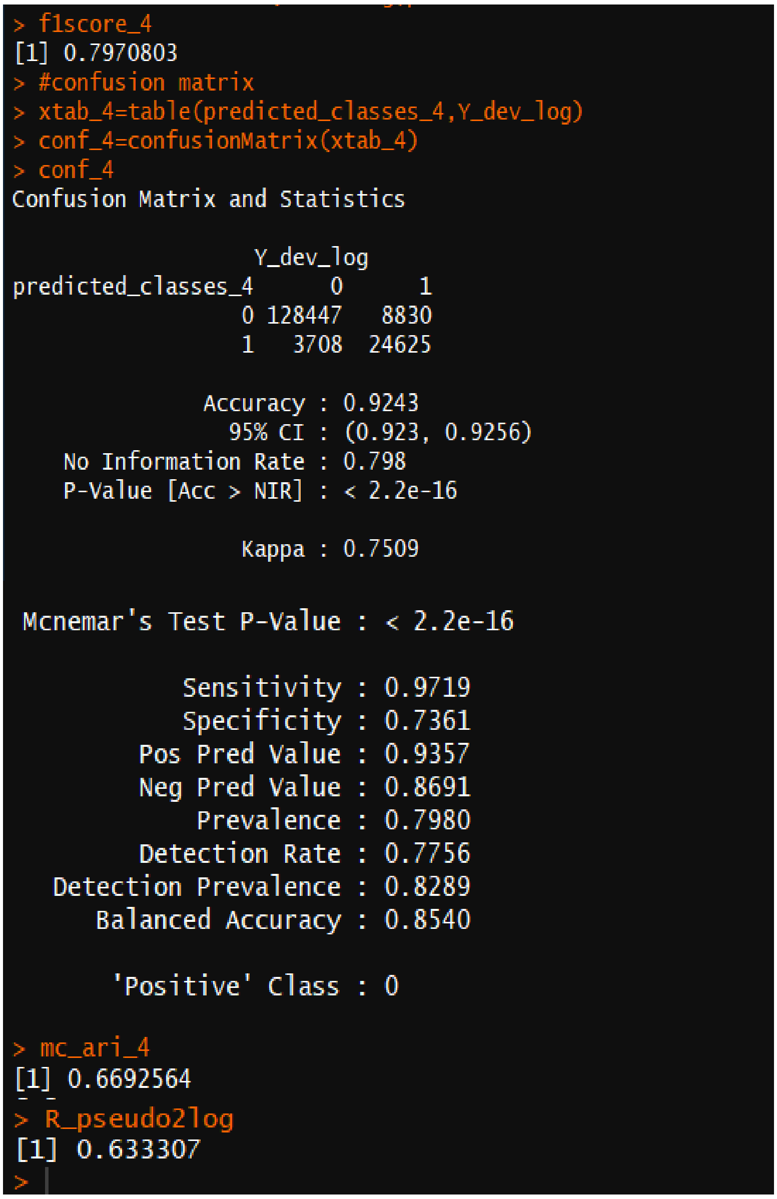
The second approach that was executed in logistic regression was the use of log data (in the whole data) so as the data become a little more normal. The trigger to use log data (and more specific the log(a+1) where a is feature) was the skewness of the data. Again, following the same procedure, the command “summary()” gave the following output (figure 96):



```
Call:
glm(formula = Target_by_med ~ ., family = binomial, data = XY_train_log)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-4.3970  -0.2889  -0.1530  -0.0711   3.1418

Coefficients:
                      Estimate Std. Error z value Pr(>|z|)
(Intercept)          -3.748910   0.160552 -23.350   <2e-16 ***
countt                0.049785   0.009347   5.326    1e-07 ***
recency               0.567529   0.006074  93.442   <2e-16 ***
frequency            -5.756832   0.107399 -53.602   <2e-16 ***
max_dist              1.066681   0.010751  99.212   <2e-16 ***
average_trans_dur    -6.090832   0.117138 -51.997   <2e-16 ***
diff_maxd_mind_days   1.493734   0.105745  14.126   <2e-16 ***
start_trans_end_data  3.713696   0.062693  59.237   <2e-16 ***
Cluster2              2.471956   0.028581  86.490   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 557300  on 552029  degrees of freedom
Residual deviance: 204358  on 552021  degrees of freedom
AIC: 204376

Number of Fisher Scoring iterations: 8

> summary(model1_log)$coef
                        Estimate  Std. Error    z value      Pr(>|z|)
(Intercept)          -3.74890979 0.160551599 -23.350186 1.372158e-120
countt                0.04978463 0.009347318   5.326087  1.003512e-07
recency               0.56752855 0.006073600  93.441877  0.000000e+00
frequency            -5.75683223 0.107398737 -53.602420  0.000000e+00
max_dist              1.06668074 0.010751475  99.212498  0.000000e+00
average_trans_dur    -6.09083206 0.117137743 -51.997178  0.000000e+00
diff_maxd_mind_days   1.49373404 0.105744527  14.125876  2.631091e-45
start_trans_end_data  3.71369626 0.062692592  59.236604  0.000000e+00
Cluster2              2.47195627 0.028580939  86.489681  0.000000e+00
```

Figure 96: Summary of model1_log

From the figure 96 we could observe that the standard errors have higher values in the case of log in comparison with the respective ones of no log (1^{st} approach.) This is implies that the estimated coefficients are not consistent enough and even a little modification could cause a huge change. As far as the evaluation of the performance is concerned, after fitting our model we got the following (figure 97):



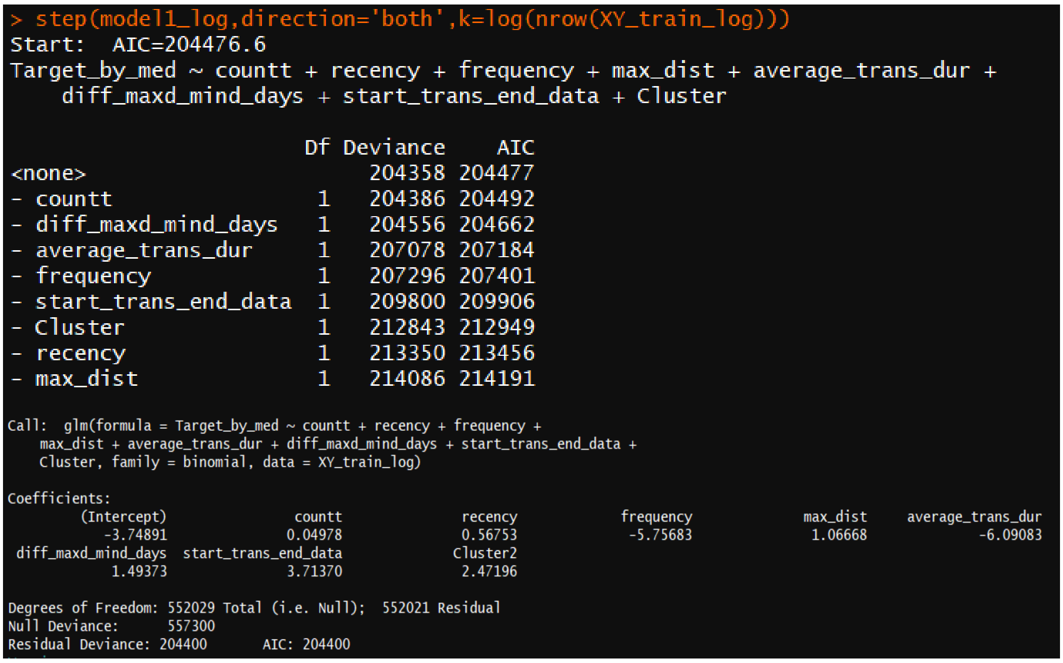
```
> f1score_4
[1] 0.7970803
> #confusion matrix
> xtab_4=table(predicted_classes_4,Y_dev_log)
> conf_4=confusionMatrix(xtab_4)
> conf_4
Confusion Matrix and Statistics

                   Y_dev_log
predicted_classes_4      0      1
                  0 128447   8830
                  1   3708  24625

               Accuracy : 0.9243
                 95% CI : (0.923, 0.9256)
    No Information Rate : 0.798
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7509

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9719
            Specificity : 0.7361
         Pos Pred Value : 0.9357
         Neg Pred Value : 0.8691
             Prevalence : 0.7980
         Detection Rate : 0.7756
   Detection Prevalence : 0.8289
      Balanced Accuracy : 0.8540

       'Positive' Class : 0

> mc_ari_4
[1] 0.6692564
> R_pseudo2log
[1] 0.633307
> 
```

Figure 97: Evaluation of logistic on log data

From the figure 97 we observe that the f1 score is 0,797 < 0.811 = f1 score in approach 1. As far as the confusion matrix is concerned, the model using log has predicted incorrectly more data than the model without log. Although the ARI lies in the same range so as the recovery be characterized as moderate [0.65, 0.8), the ARI score is lower than the ARI in approach 1. Although the pseudo R squared is lower than the respective one in the case without log we could say again that our model has very good predictive ability.

b) feature selection

After implementing stepwise regression we found that none variable was extracted (neither the start_trans_end_data). As a result, we were led to the case of full model as someone could see below in figure 98:

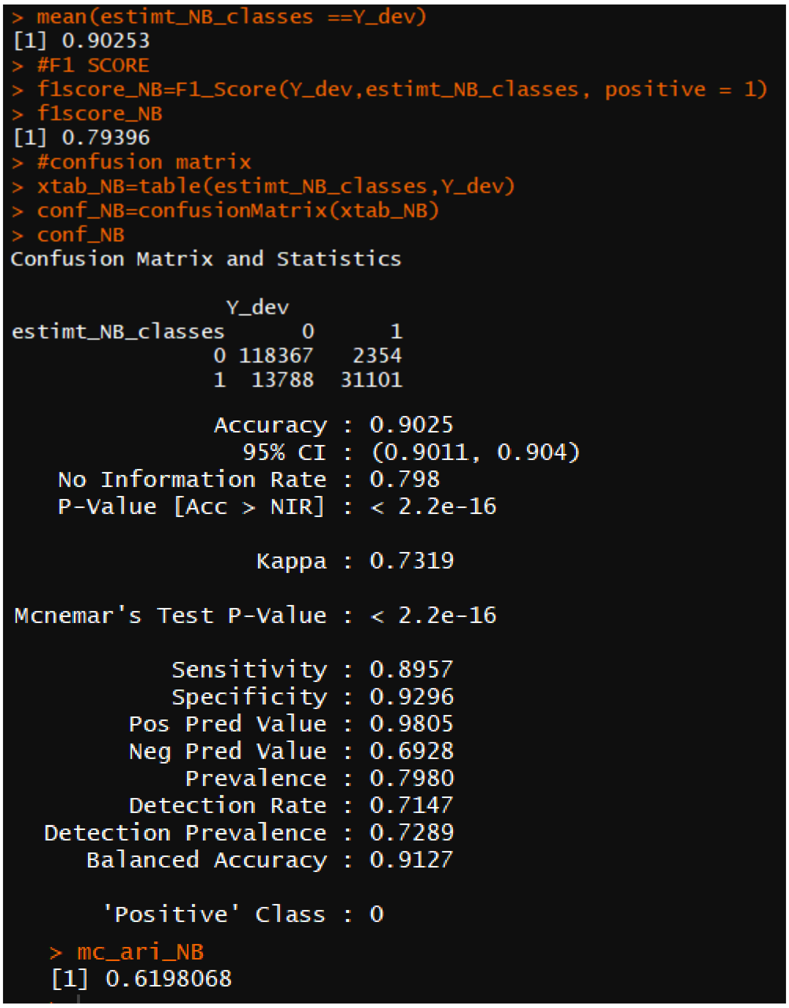


```
> step(model1_log,direction='both',k=log(nrow(XY_train_log)))
Start:  AIC=204476.6
Target_by_med ~ countt + recency + frequency + max_dist + average_trans_dur + 
    diff_maxd_mind_days + start_trans_end_data + Cluster

                       Df Deviance    AIC
<none>                      204358 204477
- countt                1   204386 204492
- diff_maxd_mind_days   1   204556 204662
- average_trans_dur     1   207078 207184
- frequency             1   207296 207401
- start_trans_end_data  1   209800 209906
- Cluster               1   212843 212949
- recency               1   213350 213456
- max_dist              1   214086 214191

Call:  glm(formula = Target_by_med ~ countt + recency + frequency + 
    max_dist + average_trans_dur + diff_maxd_mind_days + start_trans_end_data + 
    Cluster, family = binomial, data = XY_train_log)

Coefficients:
         (Intercept)                countt               recency             frequency              max_dist     average_trans_dur  
            -3.74891               0.04978               0.56753              -5.75683               1.06668              -6.09083  
 diff_maxd_mind_days  start_trans_end_data              Cluster2  
             1.49373               3.71370               2.47196  

Degrees of Freedom: 552029 Total (i.e. Null);  552021 Residual
Null Deviance:      557300 
Residual Deviance: 204400       AIC: 204400
```

Figure 98: stepwise on log (data+1)

4.2.5.2.2 Naïve Bayes

Using the command "naiveBayes()" we built a naïve Bayes model ("model_NB"). After fitting our model with the data, we continued to the predictions and then to the part of evaluating its performance. The outputs after executing the appropriate commands are the following (figure 99):



```
> mean(estimt_NB_classes ==Y_dev)
[1] 0.90253
> #F1 SCORE
> f1score_NB=F1_Score(Y_dev,estimt_NB_classes, positive = 1)
> f1score_NB
[1] 0.79396
> #confusion matrix
> xtab_NB=table(estimt_NB_classes,Y_dev)
> conf_NB=confusionMatrix(xtab_NB)
> conf_NB
Confusion Matrix and Statistics

                 Y_dev
estimt_NB_classes      0      1
                0 118367   2354
                1  13788  31101

               Accuracy : 0.9025
                 95% CI : (0.9011, 0.904)
    No Information Rate : 0.798
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7319

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8957
            Specificity : 0.9296
         Pos Pred Value : 0.9805
         Neg Pred Value : 0.6928
             Prevalence : 0.7980
         Detection Rate : 0.7147
   Detection Prevalence : 0.7289
      Balanced Accuracy : 0.9127

       'Positive' Class : 0

   > mc_ari_NB
   [1] 0.6198068
```

Figure 99: Performance of Naive Bayes

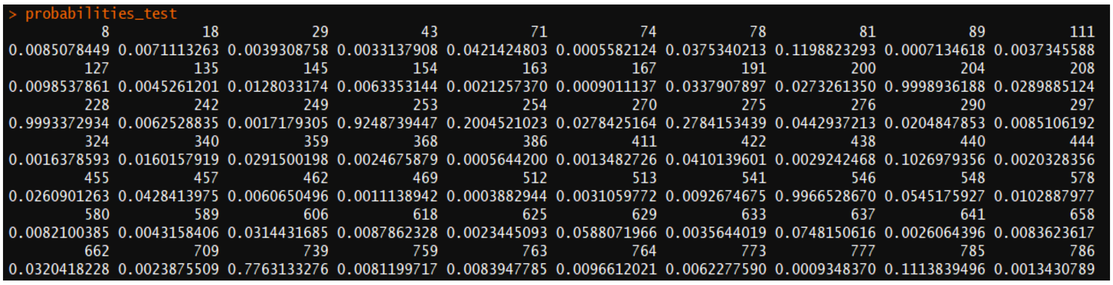
From the figure 99 we observe that Naïve Bayes classifier does not perform well on our data as it has low f1 score and ARI. The value of ARI is lower than the range [0.65,0.8) which means that the recovery is poor. Something that attracted our interest is that Naïve Bayes identified correctly more “real” churners (31.101) in comparison with the logistic regression (approach 1) that found 24.625. In addition it predicted incorrectly 13.788 no churners as churners whereas the logistic regression predicted incorrectly 3.708 no churners as churners. In a way we could say that the Naïve Bayes classifier is biased in favor of churners. The above output confirms the reason of poor performance on data which is, as we have mentioned before (page 38), the (strong) dependence between the predictors. (In our case there is indeed a very high correlation between the variables “start_trans_end_data” and “diff_maxd_mind_days”.

4.2.5.2.3 An additional try

In R studio we tried to implement the random forest classifier on our data using the data “randomForest()” but unfortunately, it was not completed due to memory error.

4.2.5.3 Conclusions

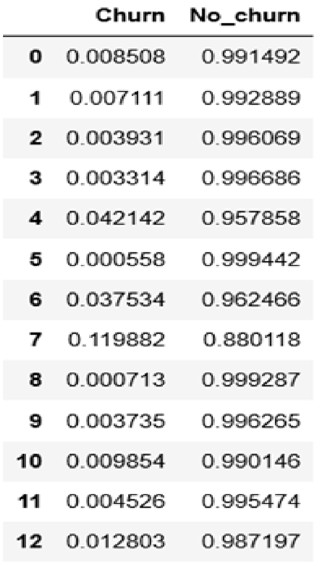
Studying the performance of the two classifiers, the logistic regression classifier was the winner (1st approach). So the last step was to apply the best classifier on the test set so as to predict the wanted probability. We dropped the variable “start_trans_end_data” from the test set and continued to the prediction. For the prediction we used the command “predict()” by setting “response” to the argument “type” so as to get the probability of being someone churner. A sample of the output is below (figure 100):



```
> probabilities_test
           8           18           29           43           71           74           78           81           89          111 
0.0085078449 0.0071113263 0.0039308758 0.0033137908 0.0421424803 0.0005582124 0.0375340213 0.1198823293 0.0007134618 0.0037345588 
         127          135          145          154          163          167          191          200          204          208 
0.0098537861 0.0045261201 0.0128033174 0.0063353144 0.0021257370 0.0009011137 0.0337907897 0.0273261350 0.9998936188 0.0289885124 
         228          242          249          253          254          270          275          276          290          297 
0.9993372934 0.0062528835 0.0017179305 0.9248739447 0.2004521023 0.0278425164 0.2784153439 0.0442937213 0.0204847853 0.0085106192 
         324          340          359          368          386          411          422          438          440          444 
0.0016378593 0.0160157919 0.0291500198 0.0024675879 0.0005644200 0.0013482726 0.0410139601 0.0029242468 0.1026979356 0.0020328356 
         455          457          462          469          512          513          541          546          548          578 
0.0260901263 0.0428413975 0.0060650496 0.0011138942 0.0003882944 0.0031059772 0.0092674675 0.9966528670 0.0545175927 0.0102887977 
         580          589          606          618          625          629          633          637          641          658 
0.0082100385 0.0043158406 0.0314431685 0.0087862328 0.0023445093 0.0588071966 0.0035644019 0.0748150616 0.0026064396 0.0083623617 
         662          709          739          759          763          764          773          777          785          786 
0.0320418228 0.0023875509 0.7763133276 0.0081199717 0.0083947785 0.0096612021 0.0062277590 0.0009348370 0.1113839496 0.0013430789 
```

Figure 100: Predicted probabilities on test set

In order to create a more user-friendly output, we exported the estimated probabilities to a csv file (”probabilities_test.csv”) and read them on windows jupyter notebook creating a pandas data frame with two columns (Churn/No Churn) and as values the probabilities of belonging to each category. A sample of the formed pandas dataframe is the following (figure 101):



	Churn	No_churn
0	0.008508	0.991492
1	0.007111	0.992889
2	0.003931	0.996069
3	0.003314	0.996686
4	0.042142	0.957858
5	0.000558	0.999442
6	0.037534	0.962466
7	0.119882	0.880118
8	0.000713	0.999287
9	0.003735	0.996265
10	0.009854	0.990146
11	0.004526	0.995474
12	0.012803	0.987197

Figure 101: Final results on jupyter notebook

4.2.5.4 Some extra notes

After identifying a bank possible churners, should take action as soon as possible.
To regain that lost customer base, business should firstly study their transactions on the branches. It is more common for clients to visit a bank branch in order to make their transactions than entering to the bank website. Secondly, the former should check-analyze the reasons that led these clients reduce the number of their logins.

However, sometimes a bank just has to let some customers go; it should consider that it cannot keep everybody happy. The bank should take into account the profitability of its clients in order to decide which ones to let go. Some customers that are not frequent users may offer great amounts of gains to the bank. These clients are worth retaining. In conclusion, a bank should not check only to reduce its customer churn rate, but rather should examine to increase the gain at the same time. (Netigate, 2019)

Chapter 5 Limitations & Extensions

5.1 Limitations

- In accordance with the Bank's policy it was not allowed to publish data
- The main problem that we faced was that we received the final real data during the last three weeks of the project duration.
- The project's nature was such that we had to execute the queries many times and this was a handling procedure from an authorized person.
- The data concerns real clients of the bank, so, it was needed to be anonymized so as to be utilized, something that required time.
- The large amount of data was way too heavy, as a result, the windows operating system was too slow in executing some commands or was not capable to execute some ones.
- Due to a computers' problem, the installation of some packages-libraries in Ubuntu was impossible and we had to alternate windows-Ubuntu.
- The Pyspark has not many packages as far as the visualization is concerned, so we had to convert pyspark data frames to pandas ones.

5.2 Extensions

- Our transactional data did not include monetary values which is a feature with strong influence. Consequently, someone could reinforce the model's fidelity adding them as a feature in the clustering phase.
- As additional features that could be added as input to the clustering phase would be demographic characteristics, the type of utilizing device combined with the age of the clients, etc. so as analysts have a more correct behavioral segmentation of clients.
- It could be a separation of clients to businesses and individuals because of the different behaviors.

- The internet banking transactions could be combined with the transactions of users at branches because there is quite a large part of the population that is not familiar with the internet and technology in general.
- Having someone at his disposal more resources, they could implement more algorithms that were utilized in our research but did not work such as PAM (K - Medoids) or Clara (Clustering Large Applications, extension to K-medoids methods for dealing with large amount of data) about the procedure of clustering and RandomForest or DecisionTrees for the classification phase.

References

Forecast Analytics. 2015. *How to Define Churn in a Non-Contractual Business.* [Online] Available from: <https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=2ahUKEwis7Y3Z9b7lAhXGo4sKHbznDlkQFjAAegQIABAC&url=https%3A%2F%2Fforecastanalytics.world%2F%3Fexport%3Dpdf%26post_id%3D1554%26force&usg=AOvVaw3UTyJbyq76MCLsde240PZJ> [Accessed 20 August 2019]

Foss, B., Stone, M. 2002. *CRM in financial services: A practical guide to making customer relationship management work*. British Library Cataloguing in Publication Data, Inc. London, UK

Lejeune, M. 2001. *Measuring the impact of data mining of churn management.* MCB UP Ltd, Vol. 11 No. 5, pp. 375-387

Netigate. 2019. *Customer Churn Meaning (and what to do about it*). [Online] Available from: <https://www.netigate.net/articles/customer-satisfaction/customer-churn-meaning/> [Accessed 25 Oct 2019> [Accessed 20 August 2019]

Gallo, A. 2014. *The Value of Keeping the Right Customers.* [Online] Available from:<https://hbr.org/2014/10/the-value-of-keeping-the-right-customers> [Accessed 25 Oct 2019]

Wonder, S. 2018. *Why Retention Marketing Is More Important Than Customer Acquisition*. [Online] Available from: <https://blog.influenceandco.com/why-retention-marketing-is-more-important-than-customer-acquisition> [Accessed 21 August 2019]

Fighting Churn. 2019. *How To Create a Churn Dataset.* [Online] Available from: <https://fightchurnwithdata.com/create-churn-data-set/> [Accessed 22 August 2019]

Wikipedia. 2019. *Customer attrition.* [Online] Available from: <https://en.wikipedia.org/wiki/Customer_attrition> [Accessed 22 August 2019]

Galetto, M. 2016. *What is Customer Churn?* [Online] Available from: <https://www.ngdata.com/what-is-customer-churn/> [Accessed 25 August 2019]

Bonnie, E. 2017. *Churn Rate: How to Define and Calculate Customer Churn*. [Online] Available from:
<https://clevertap.com/blog/churn-rate/> [Accessed 26 August 2019]

Forecast Analytics. 2019. *How to Define Churn in a Non-Contractual Business?* [Online] Available from:
<https://forecast.global/2019/01/01/how-to-define-churn-in-a-non-contractual-business/> [Accessed 26 August 2019]

Αραμπατζής, Τ. 2008. *Τι είναι το CRM και πως μπορεί να βοηθήσει μια επιχείρηση.* [Online] Available from:
<https://epixeirein.gr/2008/04/04/crm-epixeirisi/> [Accessed 27 August 2019]

Aminuddin. 2011. *What is Information Technology*. [Online] Available from: <http://danish-amin.blogspot.com/2011/01/what-is-information-technology.html> [Accessed 27 August 2019]

TECHONESTOP. 2019. *What is Analytical CRM.* [Online] Available from: <https://techonestop.com/what-is-analytical-crm> [Accessed 27 August 2019]

Grow Digital Team. 2018. *Τι Είναι το Customer Lifetime Value και Γιατί Θεωρείται Ως Ένα από τα Σημαντικότερα Metrics για τις Επιχειρήσεις*. [Online] Available from: <https://www.grow-digital.gr/ti-einai-to-customer-lifetime-value/> [Accessed 27 August 2019]

Rouse, M. 2019. *Customer loyalty*. [Online] Available from: <https://searchcustomerexperience.techtarget.com/definition/customer-loyalty> [Accessed 25 August 2019]

Hongjiu, G. 2013. *Data Mining in the Application of E-Commerce Website*. Intelligence Computation and Evolutionary Computation. Advances in Intelligent Systems and Computing, vol 180. Springer, Berlin, Heidelberg, pp 493-497 [Online] Available from:
<https://link.springer.com/chapter/10.1007/978-3-642-31656-2_70> [Accessed 26 August 2019]

Laney, D., 2001. 3*D data management: Controlling data volume, velocity and variety*. [Online] Available from:
<https://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf> [Accessed 26 August 2019]

Rijmenam, M. 2012. *Why The 3V's Are Not Sufficient To Describe Big Data*. [Online] Available from:
<https://datafloq.com/read/3vs-sufficient-describe-big-data/166> [Accessed 26 August 2019]

Datafloq. 2019. 5 *Major Data Mining Techniques Being Used by Big Data*. [Online] Available from:
<https://datafloq.com/read/5-major-data-mining-techniques-being-used-big-data/3352> [Accessed 27 August 2019]

Lejeune, M. 2001. *Measuring the impact of data mining on churn management*. [Online] Available from:
<https://www.researchgate.net/publication/220146790_Measuring_the_impact_of_data_mining_on_churn_management> [Accessed 27 August 2019]

Berry, M., Linoff, G. 2004. *Data Mining Techniques* (Second Edition) Wiley Publishing, Inc., Indianapolis, Indiana

Mohammadian, M., Makhani, I. 2016. *RFM-Based customer segmentation as an elaborative analytical tool for enriching the creation of sales and trade marketing*

strategies. International Academic Journal of Accounting and Financial Management Vol. 3, No. 6, 2016, pp. 21-35. [Online] Available from:
<http://iaiest.com/dl/journals/5-%20IAJ%20of%20Accounting%20and%20Financial%20Management/v3-i6-jun2016/paper3.pdf> [Accessed 28 August 2019]

PUTLER. 2019. *RFM Analysis For Successful Customer Segmentation*. [Online] Available from:
<https://www.putler.com/rfm-analysis/> [Accessed 28 August 2019]

Han, J., Kamber, M., Pei, J. 2012. *Data Mining Concepts and Techniques (Third edition)*. Morgan Kaufmann is an imprint of Elsevier Inc. [Online] Available from:
<http://myweb.sabanciuniv.edu/rdehkharghani/files/2016/02/The-Morgan-Kaufmann-Series-in-Data-Management-Systems-Jiawei-Han-Micheline-Kamber-Jian-Pei-Data-Mining.-Concepts-and-Techniques-3rd-Edition-Morgan-Kaufmann-2011.pdf>
[Accessed 2 September 2019]

Priy, S. 2019. *Clustering in Machine Learning.* [Online] Available from:
<https://www.geeksforgeeks.org/clustering-in-machine-learning/> [Accessed 2 September 2019]

Yoseph, F., Malim, N., AlMalaily, M. 2019. *NEW BEHAVIORAL SEGMENTATION METHODS TO UNDERSTAND CONSUMERS IN RETAIL INDUSTRY.* International Journal of Computer Science & Information Technology. [Online] Available from:
<http://aircconline.com/ijcsit/V11N1/11119ijcsit04.pdf> [Accessed 2 September 2019]

Bock, T. 2019. *What is Hierarchical Clustering?* [Online] Available from:
<https://www.displayr.com/what-is-hierarchical-clustering/> [Accessed 2 September 2019]

SlideShare. 2015. *Clustering Partioning methods.* [Online] Available from:
<https://www.slideshare.net/Krish_ver2/32-partitioning-methods> [Accessed 2 September 2019]

GeeksforGeeks. 2019. *Basic Concept of Classification (Data Mining)*. [Online] Available from: <https://www.geeksforgeeks.org/basic-concept-classification-data-mining/> [Accessed 3 September 2019]

Lucidworks. 2019. *Clustering Algorithms and Classification Techniques for More Precise Results.* [Online] Available from: <https://lucidworks.com/ai-powered-search/clustering-algorithm/> [Accessed 5 September 2019]

Joshi, P. 2018. *Generative VS Discriminative Models*. [Online] Available from: <https://medium.com/@mlengineer/generative-and-discriminative-models-af5637a66a3> [Accessed 7 September 2019]

Chioka. 2014. *Explain to Me: Generative Classifiers VS Discriminative Classifiers.* [Online] Available from: <http://www.chioka.in/explain-to-me-generative-classifiers-vs-discriminative-classifiers/> [Accessed 8 September 2019]

Wikipedia. 2019. *RFM (customer value).* [Online] Available from: <https://en.wikipedia.org/wiki/RFM_(customer_value)> [Accessed 8 September 2019]

Rathi, A. 2018. *Dealing with Noisy Data in Data Science.* [Online] Available from: <https://medium.com/analytics-vidhya/dealing-with-noisy-data-in-data-science-e177a4e32621> [Accessed 15 September 2019]

Nist Sematech. 2018. *What are outliers in the data?* [Online] Available from: <https://www.itl.nist.gov/div898/handbook/prc/section1/prc16.htm> [Accessed 15 September 2019]

Sharma, N. 2018. *Ways to Detect and Remove the Outliers.* [Online] Available from: <https://towardsdatascience.com/ways-to-detect-and-remove-the-outliers-404d16608dba> [Accessed 15 September 2019]

Wikipedia. 2019. *Interquartile range.* [Online] Available from: <https://en.wikipedia.org/wiki/Interquartile_range> [Accessed 15 September 2019]

STHDA. 2018. *Model Selection Essentials in R. Stepwise Regression Essentials in R.* [Online] Available from: <http://www.sthda.com/english/articles/37-model-selection-essentials-in-r/154-stepwise-regression-essentials-in-r/> [Accessed 16 September 2019]

Github. 2018. *In Depth: k-Means Clustering.* [Online] Available from: <https://jakevdp.github.io/PythonDataScienceHandbook/05.11-k-means.html> [Accessed 16 September 2019]

DATA STUFF. 2019. *K-Means Clustering: Unsupervised Learning for Recommender Systems.* [Online] Available from: <http://www.datastuff.tech/machine-learning/k-means-clustering-unsupervised-learning-for-recommender-systems/> [Accessed 17 September 2019]

Lucidwords. 2019. *Clustering Algorithms and Classification Techniques for More Precise Results.* [Online] Available from: <https://lucidworks.com/ai-powered-search/clustering-algorithm/> [Accessed 17 September 2019]

StackExchange. 2019. *Why does k-means clustering algorithm use only Euclidean distance metric?* [Online] Available from: <https://stats.stackexchange.com/questions/81481/why-does-k-means-clustering-algorithm-use-only-euclidean-distance-metric> [Accessed 18 September 2019]

Dabbura, I. 2018. *K-means Clustering: Algorithm, Applications, Evaluation Methods, and Drawbacks.* [Online] Available from: <https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a> [Accessed 18 September 2019]

Yse, D. 2019. *The Anatomy of K-means.* [Online] Available from: <https://towardsdatascience.com/the-anatomy-of-k-means-c22340543397> [Accessed 18 September 2019]

Wikipedia. 2019. *Elbow method (clustering).* [Online] Available from: <https://en.wikipedia.org/wiki/Elbow_method_(clustering)> [Accessed 19 September 2019]

Scikit-Learn. 2019. *Selecting the number of clusters with silhouette analysis on KMeans clustering*. [Online] Available from: <https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html> [Accessed 19 September 2019]

Derksen, L. 2016. *Visualising high-dimensional datasets using PCA and t-SNE in Python*. [Online] Available from: <https://towardsdatascience.com/visualising-high-dimensional-datasets-using-pca-and-t-sne-in-python-8ef87e7915b> [Accessed 20 September 2019]

Scikit-Learn. 2019. *Importance of Feature Scaling.* [Online] Available from: <https://scikit-learn.org/stable/auto_examples/preprocessing/plot_scaling_importance.html> [Accessed 20 September 2019]

Galarnyk, M. 2017. *PCA using Python (scikit-learn).* [Online] Available from: <https://towardsdatascience.com/pca-using-python-scikit-learn-e653f8989e60> [Accessed 20 September 2019]

Wikipedia.2019. *Rstudio.* [Online] Available from: <https://en.wikipedia.org/wiki/RStudio> [Accessed 20 September 2019]

Vora, A. 2019. *Clustering Metrics*. [Online] Available from: <https://datastoriesweb.wordpress.com/tag/silhouette/> [Accessed 20 September 2019]

Swaminathan, S. 2018. *Logistic Regression — Detailed Overview*. [Online] Available from: <https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc> [Accessed 25 September 2019]

STHDA. 2018. *Classification Methods Essentials. Logistic Regression Essentials in R*. [Online] Available from: <http://www.sthda.com/english/articles/36-classification-methods-essentials/151-logistic-regression-essentials-in-r/> [Accessed 25 September 2019]

Global Software Support. 2019. *Random Forest Classifier – Machine Learning*. [Online] Available from: <https://www.globalsoftwaresupport.com/random-forest-classifier/> [Accessed 25 September 2019]

Yiu, T. 2019. *Understanding Random Forest.* [Online] Available from: <https://towardsdatascience.com/understanding-random-forest-58381e0602d2> [Accessed 26 September 2019]

Gandhi, R. 2018. *Naive Bayes Classifier.* [Online] Available from: <https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c> [Accessed 27 September 2019]

Boyle, T. 2019. *Dealing with Imbalanced Data*. [Online] Available from: <https://towardsdatascience.com/methods-for-dealing-with-imbalanced-data-5b761be45a18> [Accessed 27 September 2019]

Wikipedia. 2019. *Accuracy and precision.* [Online] Available from: <https://en.wikipedia.org/wiki/Accuracy_and_precision> [Accessed 28 September 2019]

Shung, K. 2018. *Accuracy, Precision, Recall or F1?* [Online] Available from: <https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9> [Accessed 28 September 2019]

MixGHD documentation. 2019. *ARI: Adjusted Rand Index*. [Online] Available from: <https://rdrr.io/cran/MixGHD/man/ARI.html> [Accessed 28 September 2019]

Solymos, P. 2019. *Hosmer-Lemeshow Goodness of Fit (GOF) Test*. [Online] Available from: <https://www.rdocumentation.org/packages/ResourceSelection/versions/0.3-5/topics/hoslem.test> [Accessed 29 September 2019]

Bartlett, J. 2014. *McFadden's Pseudo-R2 Interpretation*. [Online] Available from: <https://stats.stackexchange.com/questions/82105/mcfaddens-pseudo-r2-interpretation> [Accessed 10 November 2019]

StackOverflow. 2017. *KMeans clustering in PySpark*. [Online] Available from: <https://stackoverflow.com/questions/47585723/kmeans-clustering-in-pyspark> [Accessed 2 Oct 2019]

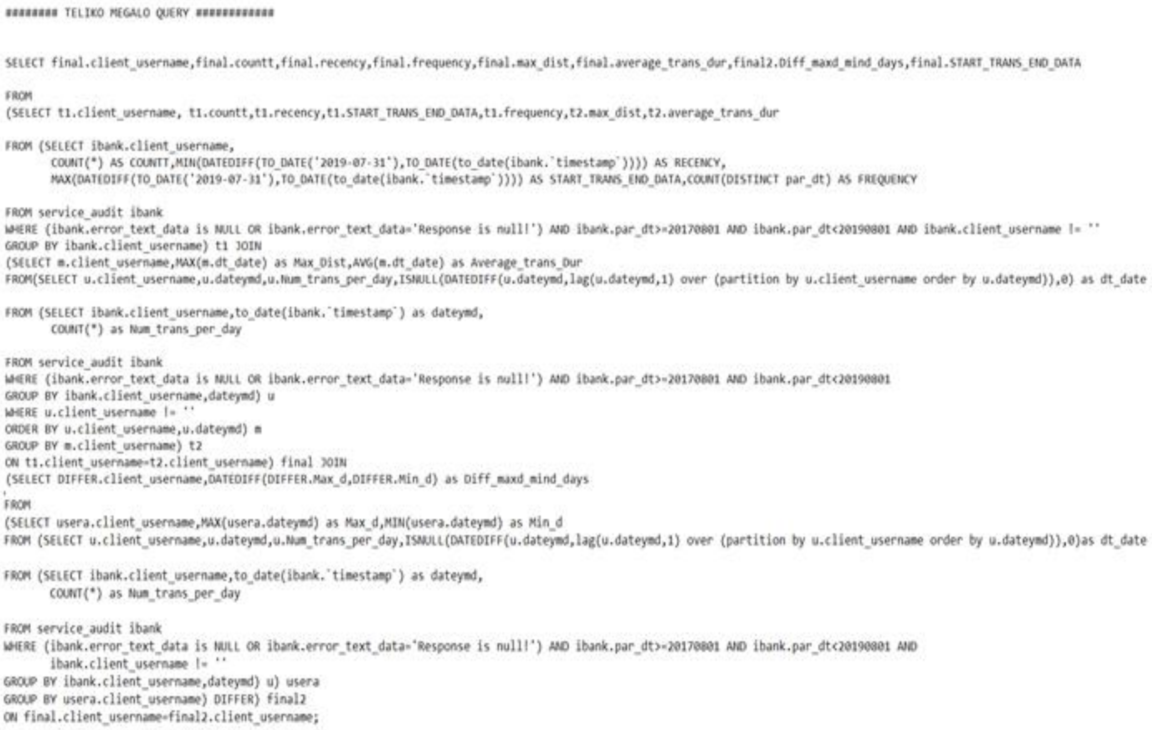
ThatWare. 2019. Naive Bayes : *The Definitive Guide*. [Online] Available from: <https://thatware.co/naive-bayes/> [Accessed 5 Oct 2019]

RapidMiner. 2009. *Interpreting k-means.* [Online] Available from: <https://community.rapidminer.com/discussion/3322/interpreting-k-means> [Accessed 25 Oct 2019]

Diffen. 2019. *Mean vs. Median.* [Online] Available from: <https://www.diffen.com/difference/Mean_vs_Median> [Accessed 25 Oct 2019]

Appendices

Impala query



```
######## TELIKO MEGALO QUERY ############


SELECT final.client_username,final.countt,final.recency,final.frequency,final.max_dist,final.average_trans_dur,final2.Diff_maxd_mind_days,final.START_TRANS_END_DATA

FROM
(SELECT t1.client_username, t1.countt,t1.recency,t1.START_TRANS_END_DATA,t1.frequency,t2.max_dist,t2.average_trans_dur

FROM (SELECT ibank.client_username,
       COUNT(*) AS COUNTT,MIN(DATEDIFF(TO_DATE('2019-07-31'),TO_DATE(to_date(ibank.`timestamp`)))) AS RECENCY,
       MAX(DATEDIFF(TO_DATE('2019-07-31'),TO_DATE(to_date(ibank.`timestamp`)))) AS START_TRANS_END_DATA,COUNT(DISTINCT par_dt) AS FREQUENCY

FROM service_audit ibank
WHERE (ibank.error_text_data is NULL OR ibank.error_text_data='Response is null!') AND ibank.par_dt>=20170801 AND ibank.par_dt<20190801 AND ibank.client_username != ''
GROUP BY ibank.client_username) t1 JOIN
(SELECT m.client_username,MAX(m.dt_date) as Max_Dist,AVG(m.dt_date) as Average_trans_Dur
FROM(SELECT u.client_username,u.dateymd,u.Num_trans_per_day,ISNULL(DATEDIFF(u.dateymd,lag(u.dateymd,1) over (partition by u.client_username order by u.dateymd)),0) as dt_date

FROM (SELECT ibank.client_username,to_date(ibank.`timestamp`) as dateymd,
       COUNT(*) as Num_trans_per_day

FROM service_audit ibank
WHERE (ibank.error_text_data is NULL OR ibank.error_text_data='Response is null!') AND ibank.par_dt>=20170801 AND ibank.par_dt<20190801
GROUP BY ibank.client_username,dateymd) u
WHERE u.client_username != ''
ORDER BY u.client_username,u.dateymd) m
GROUP BY m.client_username) t2
ON t1.client_username=t2.client_username) final JOIN
(SELECT DIFFER.client_username,DATEDIFF(DIFFER.Max_d,DIFFER.Min_d) as Diff_maxd_mind_days
,
FROM
(SELECT usera.client_username,MAX(usera.dateymd) as Max_d,MIN(usera.dateymd) as Min_d
FROM (SELECT u.client_username,u.dateymd,u.Num_trans_per_day,ISNULL(DATEDIFF(u.dateymd,lag(u.dateymd,1) over (partition by u.client_username order by u.dateymd)),0)as dt_date

FROM (SELECT ibank.client_username,to_date(ibank.`timestamp`) as dateymd,
       COUNT(*) as Num_trans_per_day

FROM service_audit ibank
WHERE (ibank.error_text_data is NULL OR ibank.error_text_data='Response is null!') AND ibank.par_dt>=20170801 AND ibank.par_dt<20190801 AND
       ibank.client_username != ''
GROUP BY ibank.client_username,dateymd) u) usera
GROUP BY usera.client_username) DIFFER) final2
ON final.client_username=final2.client_username;
```