BUSINESS ANALYTICS
Master of Science

“A buyer’s journey”

Course: Big Data Content Analytics

Supervisors
Professor | H.Papageorgiou
Lab Assistant | G.Perakis

Team Members
Dimitriou Marialena | f2821809
Katsaraki Chrysavgi | f2821814

September 2019
Academic Year 2018-2019
Spring Quarter – Full Time

Table of Contents

"A buyer's journey"

1. Description

We are interested in analyzing the online behavior of users. We want to find out their "journey" before their final transactions, the items they have seen before their final purchases and through this analysis to come up with a model that predicts if an item will be bought and how to cluster the products bought based on some features. We also want to take as much information as possible (given this dataset) about the customers' profiles. This information is very important to e-commerce stores since it can enable them to activate more effective marketing campaigns by better understanding their customers and their shopping habits and also make valuables changes to their site in order to enhance user experience and overall user satisfaction.

2. Mission

Our goal is to predict whether a user is going to buy something and if he is going to buy, which items will be bought. This information is pretty valuable to an e-commerce store as it can indicate not only the items to recommend to the user but also how to encourage the user to buy by providing perhaps some special promotions or discounts and better targeted marketing campaigns to enhance the overall return on ad spend (ROAS).

The data we used came from a challenge organized by the annual ACM RecSys (Recommender System) Conference of 2015. That year's edition of the RecSys Challenge provided a dataset constructed by YOOCHOOSE GmbH.

YOOCHOOSE is a company providing a service in e-commerce, which offers a SaaS- solution to help online shops generate a personalized shopping experience for their customers via personalized product recommendation, search results and newsletters. On the one hand, many small and mid-sized e-commerce businesses use services from companies like YOOCHOOSE to outsource the implementation and operation of their recommender system. On the other hand, recommender system companies, like YOOCHOOSE, are recording user's activities, computing statistical models and delivering recommendations on demand.

This dataset contained a collection of sessions from a retailer, where each session is encapsulating the click events that the user performed in the session. For some of the sessions, there are also buy events; meaning that the session ended with the user buying something from the web shop. The data was collected during several months in the year of 2014, reflecting the clicks and purchases performed by the users of an on-line retailer

in Europe. To protect end user's privacy, as well as the retailer, all numbers have been modified. According to the ACM RecSys Challenge, the retailer is a big e-commerce business in Europe selling all kinds of stuff such as garden tools, toys, clothes, electronics and much more.

3. Data

The training data consists of two files: the clicks dataset and the buys dataset.

The clicks dataset consists of 33.003.945 rows and has the following attributes:

Table 1 – Clicks Dataset

Attribute	Description
Session ID	The id of the session. In one session there are one or many clicks
Timestamp	The time when the click occurred
Item ID	The unique identifier of the item
Category	The category of the item

The buys dataset consists of 1.150.753 rows has the following attributes:

Table 2 – Buys Dataset

Attribute	Description
Session ID	The id of the session. In one session there are one or many clicks
Timestamp	The time when the click occurred
Price	The unique identifier of the item
Category	The price of the item
Quantity	How many of this item were bought

One of the main issues we faced with our data was that the metadata of the items were not of high quality, leaving us with not many options as to how to interpret the results and what to search for.

Only five percent of the visits in the training data contains at least one buying event, thus the data are significantly imbalanced. Furthermore, there were no user IDs in the data that will enable a connection between the sessions, so we have to rely on the session ID and to presume that each session refers to a user. As a result, we have to keep in mind that we have limited information on to how evaluate the results based on the content of the information provided.

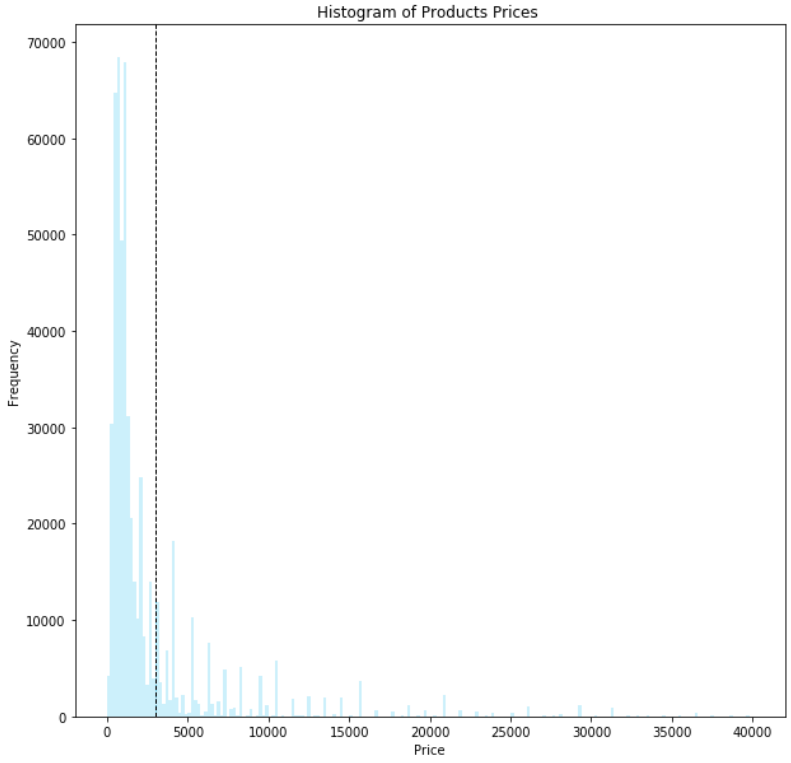
3.1 Data Preparation / Engineering

The dataset used in our models contains information about the total number of clicks in each session and the total time, in seconds, between the first click and the last click. The Session ID column is not needed since it doesn't provide any valuable information and it represents more the user of each session. We also created the feature "Total_Buys", a binary feature indicating whether the session ended with a buy or not which will be used as our target variable in the models for the prediction. This column was produced from the ''Buys" column, where NaN values were replaced with 0 and values greater than zero (indicating a buy) were replaced with 1.

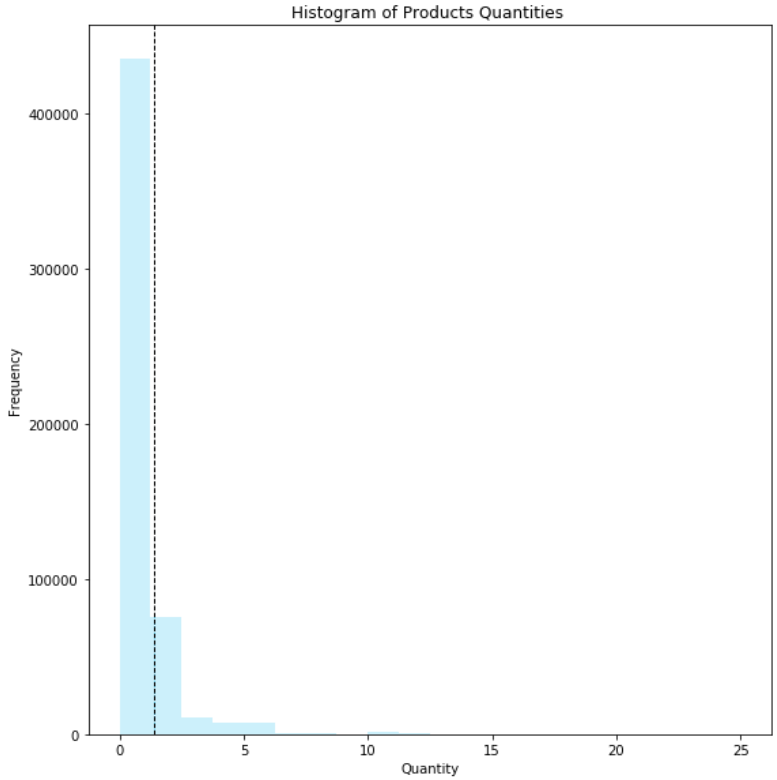
3.2 Data Visualization

We also performed some data visualization in order to better understand our data. In Graph 1, we can see the distribution of the Price variable of our buys dataset. As we can observe we have a very skewed distribution that showcases that the majority of prices are quite low whereas there are some really extreme ones. This is also true about the Quantity distribution that we can see in Graph 2. This is also a very right-skewed distribution. Specifically, a right-skewed distribution points to fact that the picture is not symmetric around the mean. For a right-skewed distribution, the mean is typically greater than the median. Also, the tail of the distribution on the right hand (positive) side is longer than on the left-hand side. So, we see how not symmetric our data is.

Graph 1 - Price Distribution

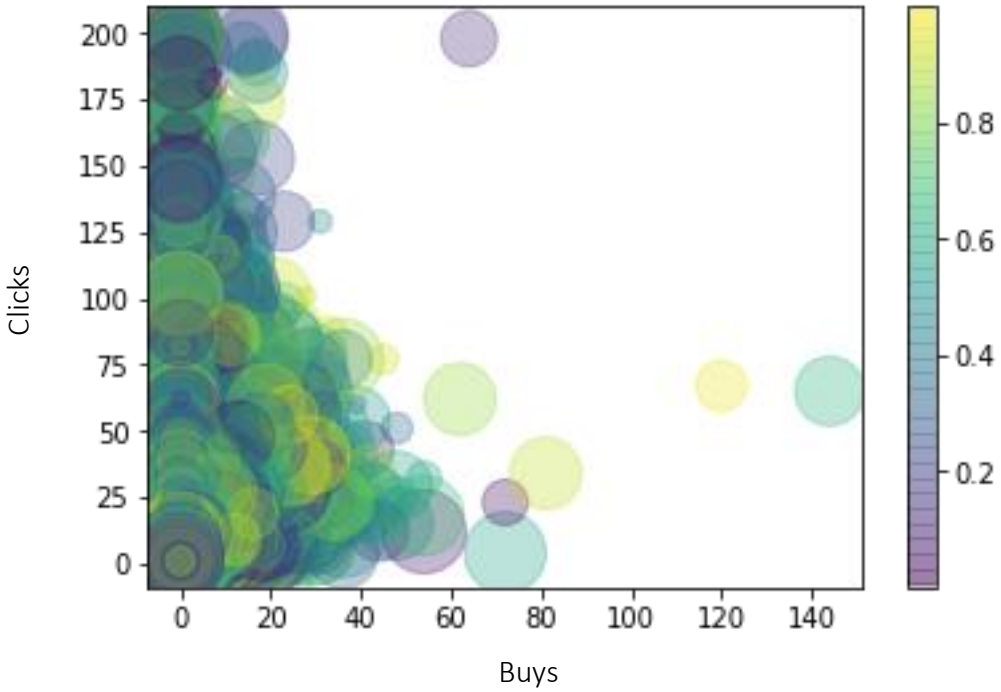


Graph 2 - Quantity distribution



To further proceed, we wanted to see the relationship between the clicks and the actual buys. Graph 3 represents the scatter plot between those two features. We can observe that the more the clicks, the less amount of buys we have. This is actually a behavior that we will later on also observe in our clustering. We can see that large amount of clicks probably reveals a more exploratory intend, rather than the need to buy.

Graph 3- Clicks vs. Buys scatter plot



4. Feature Engineering

The primary goal of this analysis is to better understand customer behavior. We will focus on two main aspects. The first task is to predict whether a customer will eventually buy or not a product. The second pillar is to focus on customer behavior and thus categorize customers when possible. In order to perform this analysis, we have to greatly engineer our data to have them ready for our models.

4.1 Are they going to buy?

The biggest problem was the large amount of data we had. We had to find a way to transform and edit this huge amount of rows with our conventional laptops. For this purpose, a big part of the "heavy work" was accomplished through the bash terminal. The extended code can be found in Appendix A. We can see below a view of the dataset in its raw format. The columns are divided by space and there are no headers present. So, we had to perform multiple changes in order to bring our data into a useful and meaningful for us, format.

Table 3 – Raw data

1 2014-04-07T10:51:09.277Z 214536502 0
1 2014-04-07T10:54:09.868Z 214536500 0
1 2014-04-07T10:54:46.998Z 214536506 0
1 2014-04-07T10:57:00.306Z 214577561 0
2 2014-04-07T13:56:37.614Z 214662742 0
2 2014-04-07T13:57:19.373Z 214662742 0
2 2014-04-07T13:58:37.446Z 214825110 0
2 2014-04-07T13:59:50.710Z 214757390 0
2 2014-04-07T14:00:38.247Z 214757407 0

4.1.1 Steps to salvation

We will describe below in a quite brief way, the basic steps that we followed in order to clean and transform our data. We divided our actions into three parts, depending on the dataset that we used.

1. Clicks dataset
    a. Count the rows of our dataset.
    b. Keep only the column containing the Session IDs in order to facilitate our future purposes.
    c. Count the times each Session ID appeared.
    d. Sort the dataset.
2. Buys dataset
    a. Count the rows of the dataset.
    b. Keep only column that contains Product ID.
    c. Count times each Product ID appeared.
    d. Sort the dataset.
    e. Combine the two datasets and thus create the merged dataset.
    f. Add suitable column names.

This final dataset contains:

Session ID: The id of the Session

Clicks: Number of total clicks in the session

Buys: Number of total buys in the session

3. Time

a. From dataset clicks, we kept only the columns Session ID and Date.

b. We replaced the dot with space character.

c. We kept only the two columns and excluded the one containing the milliseconds.

d. We added appropriate headers to the columns.

At this point, we were done with the bash part. At least we thought so. We eventually discover an abnormal behavior on how the bash "translated" a file. Instead of keeping the records as they are, bash added spaces before the counted numbers. In our bash configuration, each space represents a column. So, this had as a result that wherever there was a sum with more than one digits, the columns were fewer. In order to resolve this issue, we used again the bash and replace the spaces with comma and then proceed with Python for further transformation.

Once the data were in the aforementioned format, we then performed yet another layer of cleaning, using python. At first, we coped with fixing the time variable. We decided that we will keep the information about the total spend time between the first and the last click of each session. In order to achieve this, we had to load the whole clicks dataset and keep only the columns with the Session ID and Date. Then, we converted the date into the appropriate format and after some more processing we eventually calculated the time spend. As for the mistake that we observed we treated the data accordingly in order to finally have the desired outcome.

The final result was the construction of two informative dataframes. The first contains the sessions' information with 9.249.730 rows. This dataset consists of, each unique session id the number of the products that were clicked on each specific session, the sequence of these products as well as the number of buys that were eventually purchased. The second dataset was built for holding the products' information. This dataset contains the product id, the price of each unit and the total quantity of the purchased items and also the different sessions that this item was purchased. This dataset contains 20.790 rows of distinct product ids.

5. Methodology and Results

5.1 Prediction

Since we are interested in predicting whether a session is going to end with a buy or not our problem can be solved as a classification problem. Our target variable "Buys" is a binary variable and we can start our analysis with a binary classification model with the Keras library for deep learning. This binary classification model is a neural network model and to evaluate the model we used stratified k-folds cross-validation. This is a resampling technique that will provide an estimate of the performance of the model. It does this by splitting the data into k-parts, training the model on all parts except one which is held out as a test set to evaluate the performance of the model. This process is repeated k-times and the average score across all constructed models is used as a robust estimate of performance. It is stratified because it will look at the output values and attempt to balance the number of instances that belong to each class in the k-splits of the data.

To use Keras models with scikit-learn, we will use the KerasClassifier wrapper. This class takes a function that creates and returns our neural network model. It also takes arguments that it will pass along to the call to fit() such as the number of epochs and the batch size.

Our model will have a single fully connected hidden layer with the same number of neurons as input variables, meaning the features "Clicks" and "Difference". The weights are initialized using a small Gaussian random number and the Rectifier function is used as the activation function. The output layer contains a single neuron in order to make predictions. It uses the sigmoid activation function in order to produce a probability output in the range of 0 to 1 that can easily and automatically be converted to class values.

Finally, we are using the logarithmic loss function (binary_crossentropy) during training, which is the preferred loss function for binary classification problems. The model also uses the efficient Adam optimization algorithm for gradient descent and accuracy metrics will be collected when the model is trained.

Before creating the binary classification model, we applied a Logistic Regression classifier and a Random Forest classifier to our data in order to compare the results with the binary classification model. To split the dataset into train and test we used the train_test_split function from Scikit-Learn library. In Table 4, you can see how the data were split in X_train, X_test, y_train and y_test.

Table 4 – How the data were split

Datasets	Shape
X_train	(6197318, 2)
X_test	(3052411, 2)
y_train	(6197318, 1)
y_test	(3052411, 1)

As we mentioned previously, first we applied a logistic regression and the results had high accuracy. We suspected that this may be a result of overfitting and in order to further analyze the case, we also applied a random forest classifier. The results of the second model were very close to the results of the first model both resulting in very high accuracy, as you can see in Table 5.

Table 5 – Results of Logistic Regression and Random Forest Classifier

Model	Accuracy
Logistic Regression	0.9436
Random Forest Classifier	0.9423

5.2 Neural Network

Our next step was to create the neural network using Keras following the steps we mentioned before. Unfortunately, due to hardware limitations and the issues we faced with our data preparation causing us a significant delay we were unable to get the results on time but you can find the code for the procedure we followed in the predictions.ipynb.

5.3 Clustering

5.1.1 Product 'placement'

The second focus was on product clustering. The dataset did not provide any information about the category in which each product belongs to. We also had no information about the nature of the online store, except for the fact that it is in the retail domain. We first tried to categorize a bit our products in order to better understand the profile behind each session. For this purpose, we used our buys dataset to extract as much information as possible about the products. Having a very unbalanced and not so trustworthy dataset to work with, we tried to implement our ideas in the best possible way. We found out that there were inconsistencies as well as a lot of missing information. To be more specific out of the total 1.150.753 only about the 50.000 were actually unique sessions. Out of them, we could use about 20.700 distinct product codes that had enough information about both the quantity and the price. For categorizing our products into clusters, we decided to use information that described their total quantity bought, in all of the sessions (out of the

20+K we used) that they were purchased, as well as the number of different sessions that they were bought in along with the price per 1 unit. In order to create the "Different Sessions that the products were bought" column, we grouped by the product code and counted the total number of sessions it appeared. For the "Total quantity" we grouped by the product code and the add the quantities. Then we combined these two columns with our "Price" column and manufactured our clustering dataset.

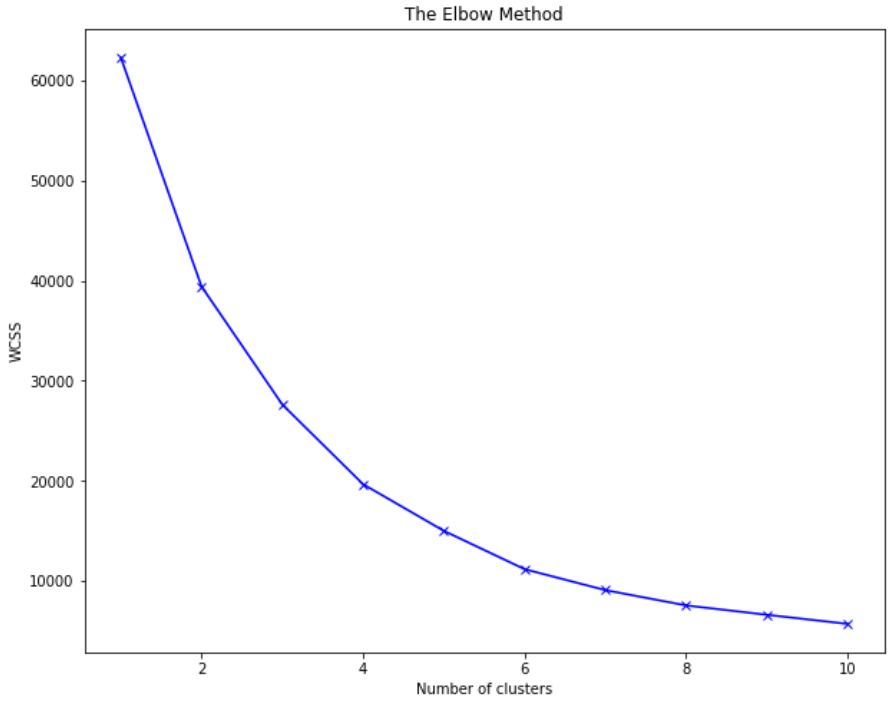
We chose to use K-means as our clustering method. In every clustering method, the goal is to create clusters that contain very similar objects on the inside but have many differences to other clusters outside. The aim of K-Means is to minimize the distance between the points within a cluster with their centroid.

At first, we should choose an optimal number of clusters. For this purpose, we used the diagram of the Elbow method. The Elbow method is a technique that chooses a number of clusters so that adding another cluster doesn't improve much better the total within sum of squares. It works as follows:

1. Compute clustering algorithm (e.g., k-means clustering) for different values of k varying k from 1 to 10 clusters.
2. For each k, calculate the total within-cluster sum of square (wss).
3. Plot the curve of wss according to the number of clusters k.
4. The location of a bend (knee) in the plot is generally considered as an indicator of the appropriate number of clusters.

From Graph 4, below we concluded that an optimal number could be four clusters. We also tried with three, five and six clusters, but the results were not that much satisfactory.

Graph 4 - Elbow method for products clusters



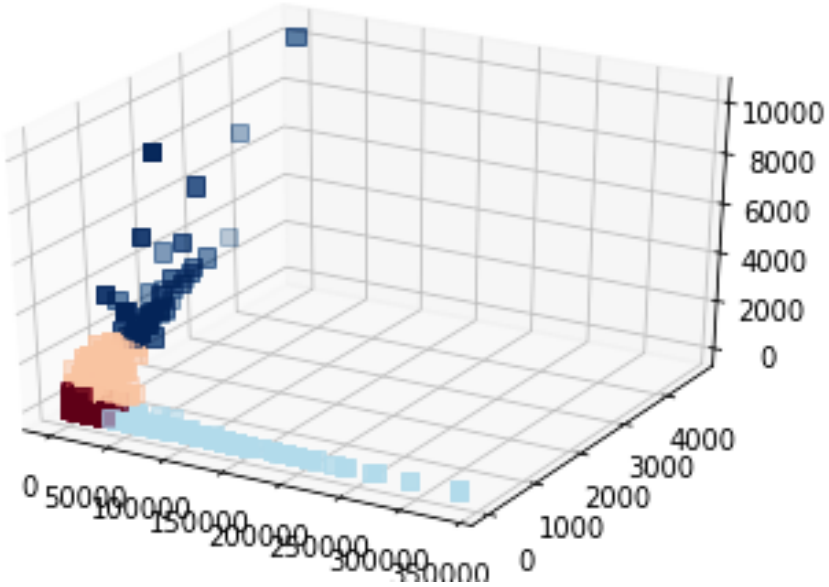
Our final clusters had the characteristics presented in table 6 below. We can see that the first cluster contains products that are relatively pricy and have been purchased quite a few times but the quantities bought are rather a lot. Products in this category are probably not extremely expensive and also bought in batches in the majority of the time. They also seem to not have a need for frequent recharging. The biggest part of our products are members of this cluster. The second cluster has the second-lowest price and it appears in a huge number of transactions. As for the total quantity we can see that is very high. So, within this category of products, it is very likely to be bought together. So maybe this is a product that needs replacement quite frequently and is quite necessary also. The third cluster includes products with a different significantly higher price. Here we observe that the sessions that these particular products were purchased are very few as well as not in large quantities. This category of products is probably a luxury category that is affordable and preferred by few. The last category includes products with extremely large number of frequencies in different sessions as well as indications that is approximately purchased as pairs. Here we observe the smaller price as well. These can be products that are necessary and need to be replaced as soon as they have finished and a very large portion of the products belong to this cluster.

Table 6 - Cluster characteristics

Cluster	Price/unit	No Sessions	Quantity	No of Members
1	4735	19.7	30.3	8400018
2	3650.3	468.5	742.7	195118
3	62416.3	11	12.7	14209
4	1665.1	1694.7	2806.6	640384

Below we can also see a graphical representation of our clusters.

Graph 5 – Clusters of Products

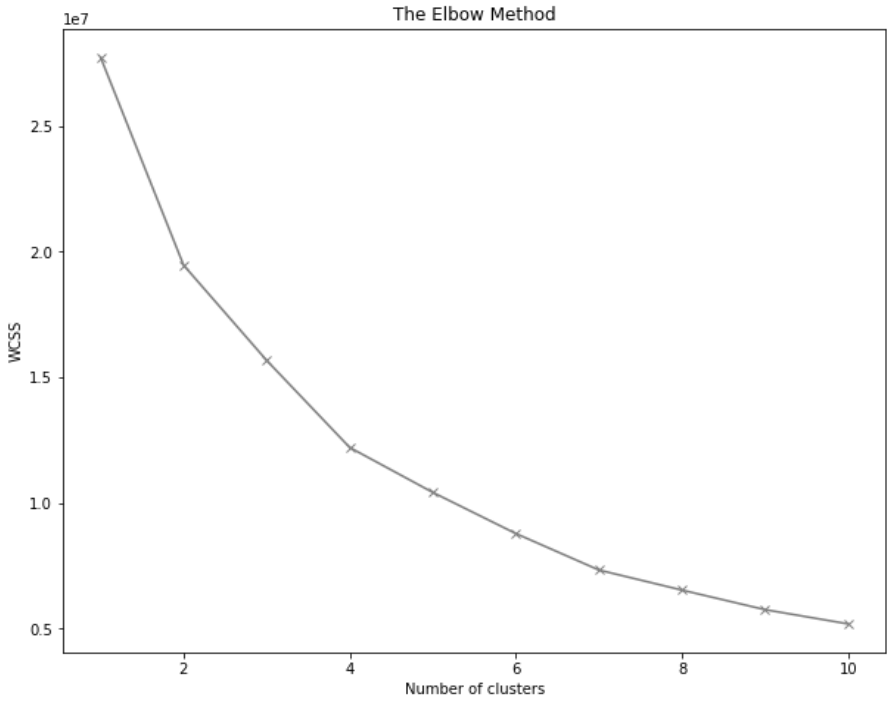


5.1.2 Explore the behavior of a Session

Following the above-described methodology, we also tried to identify and describe the "behavior" of the sessions. This dataset does not provide any information about the actual users that performed the purchases. Nevertheless, we thought that we can have an idea through the session exploration, as it is the depiction of the user's behavior. For this purpose, we took the dataset used in the prediction part. We also took into consideration the total price spend on each particular session. So, our final dataset consisted of the number of total clicks in each session, the number of total buys in each session, as well as the number of total monetary units, spend.

The elbow method recommended to use again four clusters in order to better describe our data. We can see the graph below for further understanding.

Graph 6 - Elbow method for sessions clusters



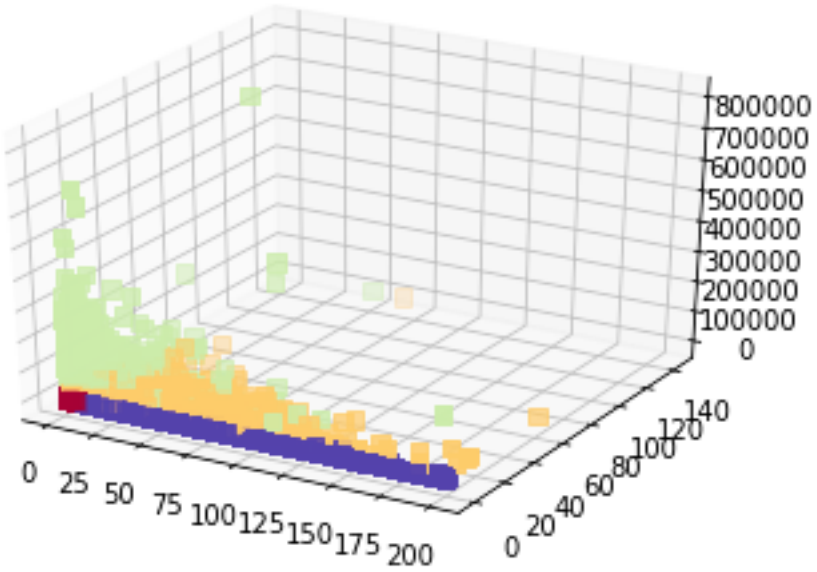
So, we end up with four clusters with the characteristics that are described in Table 7. We can see that we have a variety of session profiles. The first cluster which is the most numerous cluster showcases sessions that were a bit random. Maybe somebody just entered the page to see one or two products without further investigation and without a need to actually buy something. They are the “Casual viewers”. The second cluster is also the second most popular one, which seems to contain users that are in the search of buying something and spend some time in searching. Most of the time though, they do not make any purchase. They are probably “Bargain Hunters” and wait for the right time to make a buy. The third cluster is probably a medium category that contains casual buys. The intent behind seems to be the purchase of the right product. We can say that we have “Intent to buy “ here. The last cluster which has the fewer members are the “Elite buys”. We see here very targeted behavior to very pricy products. We also see here determination of buying and knowledge of what they are looking for.

Table 7 - Cluster characteristics

Cluster	Clicks	Buys	Total Spend	No of Members
1	2.6	0	36	8178449
2	10.5	0.1	32.5	839687
3	12.2	3.4	4238.9	213321
4	6.4	2.4	38512.3	18272

Here, in Graph 7, we can see our clusters and cluster members.

Graph 7 – Cluster of Sessions



6. Hardware Specifications

The hardware specifications of our equipment were as follows:

- Installed RAM: 8GB
- System type: 64-bit operating system
- Processor: Intel Core i5 8th Gen

7. Members/Roles

Our team consisted of two people, Marialena Dimitriou and Chrysavgi Katsaraki.

Marialena Dimitriou is a graduate of the School of Applied Mathematical and Physical Sciences from the National Technical University of Athens. She has a specialization in Applied Analysis and Statistics and previous working experience in the digital marketing sector. Currently, she is working as a data warehouse developer at a multinational technology consulting firm.

Chrysavgi Katsaraki is a graduate of the Departments of Statistics and Insurance Science from the University of Piraeus. She has previous working experience in the insurance industry, the biomedical industry as a researcher and in the digital marketing sector. Currently she is working as a data warehouse and business intelligence consultant for a large multinational technology firm.

We were both actively involved in almost all phases of our project, collaborating and discussing all aspects of the analysis and the methodology we should follow. Chrysavgi was responsible for the data cleaning, preparation and data engineering and she was also engaged with the data modeling and evaluation. Marialena was responsible for the data modeling and the interpretation of the results, as well as the documentation of the requirements specification.

8. Bibliography

The ACM Conference on Recommender Systems. (n.d.). Retrieved from https://recsys.acm.org/recsys15/

Towards Data Science. (n.d.). Retrieved from https://towardsdatascience.com/understanding-neural-networks-19020b758230

9. Time Plan

By the time we started the project we had already devoted much time at studying neural networks and various deep learning algorithms in order to facilitate the work on our project and to be better prepared. Our original plan was to spend 1-2 weeks in data preparation, 1 week in data modeling and 1 week in evaluating the models and interpreting the results. The original plan was followed with some delays in the data preparation phase. The time we devoted in this phase was significantly longer than expected and we had to iterate through the pipeline, from data preparation to modeling many times during the project as many issues kept arising. This resulted in a limited time to devote to the data modeling and evaluation.

10. Contact Person

The contact person would be Marialena Dimitriou (marialenadim@gmail.com, +306978311945).

11. Comments

The overall collaboration and communication of the team was exceptional. With hard work and a great team effort we managed to overcome any obstacle that occurred in our project. Our main difficulties had to do with the data cleaning and preparation of the datasets. It was a very difficult task and many unexpected errors occurred later in our analysis resulting in many delays in our time plan. After this project we have gained more valuable insights in an end-to-end big data analytics case and we have developed expertise in all phases of a business analytics case pipeline. Our key takeaway is the importance of having good quality data, a rational and well thought time plan taking into consideration any possible delays and of course the importance of the technical specifications of the equipment used because projects like this require much memory and heavy, time consuming computations.

Appendix A

```
## At first we choose our directory with cd command

$ cd /c/Users/xkats/Desktop/BigDataContent


## Count the rows of the dataset clicks

$ wc -l yoochoose-clicks.txt

33003944


## Add headers

$ sed -i '1 i\Session,Date,Time,Product_id,category' yoochoose-clicks.txt


## Create time dataset

$ awk -F "\"*,\"*" '{print $1 , $2}' yoochoose-clicks.txt > time.txt


## Change the dot to space

$ sed -i 's/\./ /g' time.txt


## Add Headers to columns in time dataset

$ sed -i '1 i\Session Date Time' time.txt


## Keep the two columns and output a new table tim.

$ awk -F "\"* \"*" '{print $1 , $2}' time.txt > tim.txt


## Erase T (not used )

xkats@LAPTOP-0FAEUNH2 MINGW64 ~/Desktop

$ sed -i 's/T/,/g' yoochoose-clicks.txt
```

```
## Perform some checks on buys dataset

## To take only two columns

$ awk -F "\"*,\"*" '{print $1 , $3}' time_to.csv > t.csv

### Find a record

$ awk -F , ' $1 == "\"11276101\"" {print;}' yoochoose-buys.txt


## Count the Buys

## awk -F "\"* \"*" '{print $1 , $3}' yoochoose_buys.txt> t.xtx

## First output to to e.txt

awk -F ',' '{print $1}' t.txt | sort | uniq -c > e.txt


## Count the clicks

### Keep column 1

$ awk -F "\"*,\"*" '{print $1}' yoochoose-clicks.txt > cli.txt

# Count

$ awk -F ',' '{print $1}' cli.txt | sort | uniq -c > e_cl.txt


# Sort both datsets


## Combine clicks+buys

$ awk 'FNR==NR{k[$2]=$2;next} {print $1,$2,k[$1]}' e_sorted.txt e_c_sorted.txt > mer.txt

#Add tittle

$ sed -i '1 i\Clicks Session Buys' mer.txt


### For the mistake we had to come back follow the same code but perform this change before the merge stageand preform the actual merge in python.

$  sed -e "s/\s\{1,\}/,/g" e_c_sorted.txt > e_c_sorty.txt
```