UNIVERSITÀ DI PISA

An analysis of the Twitter dataset

Data Mining Report

Acciaro, Gennaro Daniele
g.acciaro@studenti.unipi.it
635009

Carfì, Giacomo
g.carfi1@studenti.unipi.it
520951

Peluso, Christian
c.peluso5@studenti.unipi.it
641346

November 12, 2022

Contents

1 Introduction

The purpose of this report is to describe and motivate the choices made to analyze the dataset concerning members of the social network named Twitter. The data provided are essential regarding the subscribers of this platform, with some information and tweets written by them. This dataset contains the real behavior of the users, indeed the real challenge of this project is to handle and extrapolate as many as possible details from the tweets, in order to better classify each type of user.

The processes for achieving our goal are grouped into 4 tasks:

1. **Data Understanding:** which consists of unpacking the dataset and realizing the kind of data we are dealing with, capturing if there are semantic or syntactic errors, duplicates, outliers, or missing values, and realize how to deal with them. Keeping this in mind we will try to illustrate the distribution of data and figure out how to process the data in the following sections;

2. **Data Preparation:** here we will try to use the knowledge extracted in order to front off the highlighted problems, filling the missing values gaps and create new indices that can sum up the characteristics of each user;

3. **Clustering:** in this section, we will focus on the most important indices eliminating the less representative. On these features we will use different clustering algorithms in order to detect the number of types of users and describe common features above them;

4. **TASK 3:** TBA

5. **TASK 4:** TBA

2 Data Understanding

Bearing in mind the dimension of the files, we decided to take advantage from some libraries, mainly Pandas 1.3.5 for the elasticity and efficiency that it offers to manage heavy dataframes and Numpy 1.21.6 for the function manipulation that we can apply on the data. For illustration purposes we adopted MatPlotLib 3.2.2 for the basic plotting functions and SeaBorn 0.11.2 for the personalization capabilities.

First of all, in Section 2.1, we faced the data printing of some rows for each column in order to understand what is the content of the cells and what their type should be. Then we checked if there were missing values or <NA>. Finally, in Section 2.2, we assigned column-compatible types in order to properly plot the distributions of the features making it possible to perform data comprehension and project planning, this process has been done respectively to bot & non-bots users.

2.1 Data Semantics

The whole dataset that we have is divided into 2 different comma-separated-values files: users.csv and tweets.csv. The former represents users registered in the social network and contains a total of **11508** rows while the latter contains information about tweets and contains **13664696** records. In Table 1 we provide information about the type of the feature presents:

Data Type	**Tweets Features**	**Users Features**
Nominal	id, user_id	id, lang
Numeric	retweet_count, reply_count, favorite_count, num_hashtags, num_url, num_mentions	
Interval	created_at	created_at
String	text	name
Binary		bot

Table 1: Attributes type for tweets and users.

Let us now look at more detailed information regarding the various features. For users we have:

1. *id* is an incremental value aim uniquely identify the user, there are 11508 integer values, the majority of them are 9 and 10 digits, this indicates that an order of magnitude of 10^8 users have been registered earlier.

2. *name* is the nickname of a user, there are 11507 values of type object and only 11361 unique entries, there is 1 NaN value and some duplicated names.

3. *statuses_count* the count of the tweets made by the user at the moment of data crawling. In this phase we can notice that only real users have NaN values for this feature.

4. *lang* the user's language selected, there are no missing values and they are of type object; but as we stated there are meaningless and redundant values.

5. *created_at* the timestamp at which the profile was created, the values are spread between 2012 and 2020 so we can suppose, from this first analysis, that all the users are coherent with the Twitter creation at the moment of the data crawling. As we can see most of the users have been created between 2017 and 2019.

6. *bot* a binary variable that indicates if a user is a bot or a genuine one.

For tweets we have instead:

1. *id* an incremental value in order to uniquely identify the user, the data frame is composed by *13664696*, but we have only *11672136* unique IDs, therefore we have some duplicates, missing values, and other incoherent values. Naturally, there are many more tweets than users so the digits for identifying each tweet are far more, as we have seen that most of them present in this data set are composed of 18 digits.

2. *user_id* is an unique ID that associates each tweet with a user. There are 222286 unique values but the **most of these values are not numbers**, so we believe that this column will be useful to carry out a deep cleaning. There are *217283* NaN, it's not possible to identify the users that originated these tweets without text comprehension.

3. *retweet_count)* is an integer that indicates how many times a specific tweet has been retweeted; there are *647878* NaN.

4. *reply_count* is an integer that indicates how many replies a specific tweet has received, the number of NaN we found is *647878*.

5. *favorite_count* this attribute indicates the number of likes received by a tweet. This column has 647542 NaN values.

6. *num_hashtags* is a numeric attribute that indicates the number of hashtags written in the tweet. There are *1057524* NaN values, these are almost 10% of the dataset, so it is better to see fill them once selected the one corresponding to a user ID present in the User data frame.

7. *num_urls* this column is showing the number of links that are present in a tweet, can only take integer values. There are *648623* NaN.

8. *num_mentions* Numeric attribute, can only assume integer values. It means the number of accounts tagged in the tweet. There are *854165* NaN.

9. *created_at* it tells when the tweet was written. It is an object attribute that must be converted into date-time type. There are no NaN.

10. *text* it represents the text of the tweet, it is a string attribute. We found that are *537721* null values.

2.2 Data Visualization

The aim of this section is to provide some insights about the data we are dealing with. As we can see from the graphs on Figure 1, there are very high values in the numeric type attributes present in the dataframe tweets. These values are clearly outliers that will be fixed in the Section 3.
We can also see from the graphs regarding the date of creation of tweets in Figure 2a that there are years relative to date that are before the creation of Twitter, such as 1953, and years in the future, such as 2040.

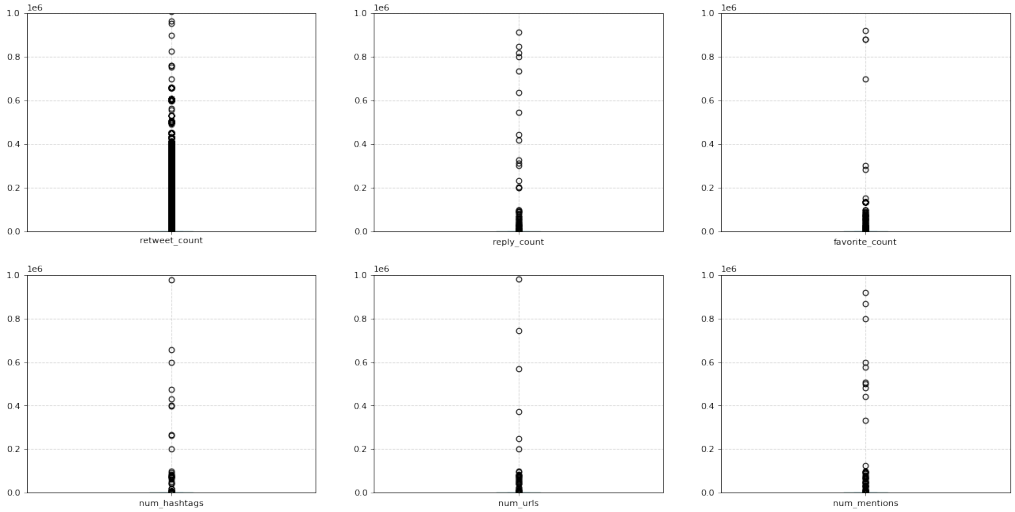
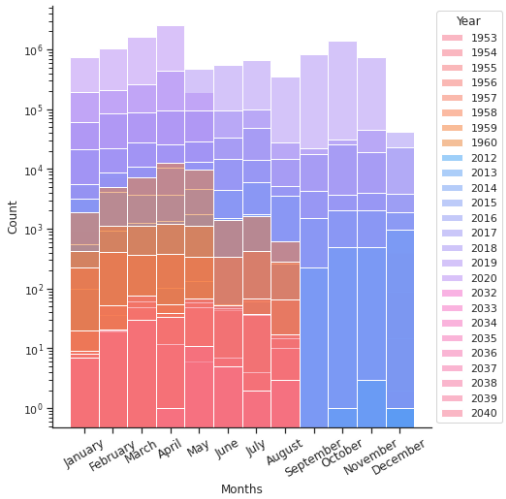


Figure 1: Box plot of numeric type features in the tweets dataframe.

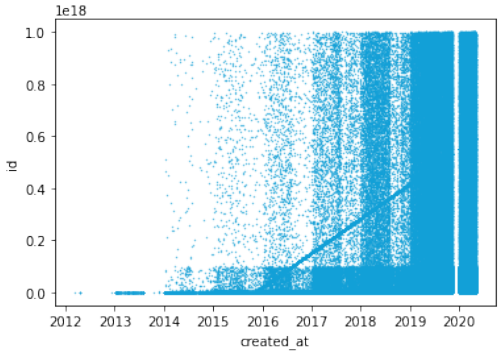
Since there are some tweets with incoherent dates we decided to plot the distribution of IDs in correlation with the dates of creation, taking into account only the coherent dates (Fig. 2b). We can see that there is a dense region that indicates the growth of the ID with the passing of the days. This is a hint that indicates that the ID is an incremental number, handled by the database.

From the graph regarding the length of tweets in Figure 2c, we can see that most tweets are 0 to 150 characters in length. There are some very long tweets that reach 420 characters but this is clearly an anomaly since the maximum length of characters that can be written in a tweet is 140 till November 2017 and 280 thereafter.

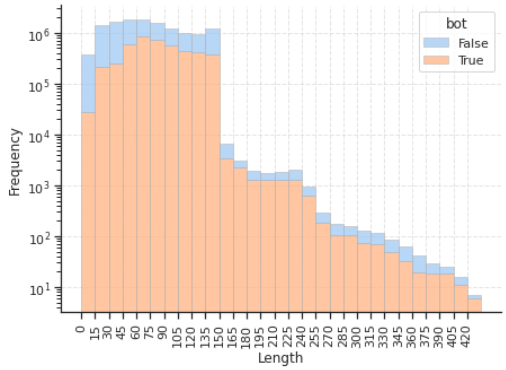
Finally, as we can see in the Figure 2d most of the bots were created in 2019 and 2017, before mainly there were usual members of Twitter.



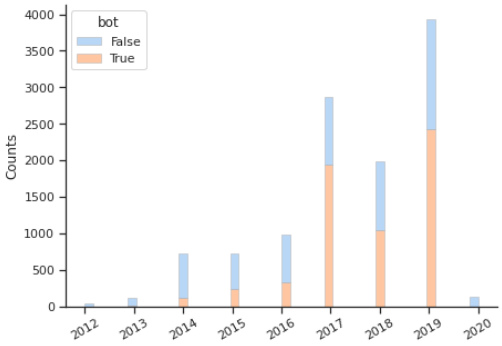
(a) Creation of the tweets [hue=year]



(b) Creation of the tweets over the ID



(c) Length of the tweets [hue=bot]



(d) Subscription period of the user [hue=bot]

Figure 2: Here we are 4 graphs concerning the initial untouched distribution, as we can see there are multiple values out of coherent ranges.

3 Data Preparation

In this section we have identified and filled missing values and outliers, in order to be able to re-plot the distributions, create indices and in general be able for the next scope of data clustering and classification.

3.1 Outliers Management

Firstly we assigned the correct type to the attributes, converting the columns for each contained attribute. In this way numeric attributes, if the value to be corrected cannot be converted to a number (because it is a textual string, for example, and therefore an error) it is replaced with a NaN.

After that, we removed tweets that have a *user_id* that does not exists in the user dataframe and deleted from the tweets those records that had the *text* field empty since we consider these record meaningless. Clearly, we also removed duplicates in both dataframes. We found that the tweet dataframe contains 1952099 duplicates corresponding to about 14% of the entire dataframe while the user dataframe contains no duplicates.

We then continued the analysis by performing the outliers search by analyzing the distribution of the data on the tweets dataset.
Infinite and negative values were removed by substituting it with NaN. As we can see in the Table 2 we noticed that for values on the 99% quantile maximum we reach changes very rapidly. We can eliminate very large values by choosing a specific quantile for each of the attributes.

Quantile	#retweet	#reply	#favorite	#hashtags	#urls	#mentions
0.750000	1.0e+00	0.0e+00	0.0e+00	0.00e+00	0.00e+00	1.00e+00
0.900000	2.6e+01	0.0e+00	1.0e+00	1.00e+00	1.00e+00	1.00e+00
0.950000	4.5e+02	0.0e+00	2.0e+00	1.00e+00	1.00e+00	2.00e+00
0.970000	1.5e+03	0.0e+00	3.0e+00	2.00e+00	1.00e+00	2.00e+00
0.990000	7.4e+03	0.0e+00	7.0e+00	3.00e+00	1.00e+00	3.00e+00
0.995000	1.7e+04	0.0e+00	1.2e+01	4.00e+00	1.00e+00	4.00e+00
0.999000	1.1e+05	0.0e+00	4.6e+01	7.00e+00	2.00e+00	7.00e+00
0.999900	3.9e+05	2.3e+01	3.2e+03	1.10e+01	3.00e+00	1.10e+01
0.999990	3.3e+06	7.9e+03	2.2e+04	2.61e+03	3.54e+03	5.78e+03
0.999999	1.8e+10	1.3e+37	8.2e+09	5.13e+11	4.90e+10	9.99e+09

Table 2: We show the maximum values that we have considering a specific percentile.

From these values we can clearly select the quantiles with exaggerated values and cut the field from them, indeed in the next table 3 we show the percentages we consider. Values that are larger than a certain threshold will be replaced with NaN and then filled by the mean of the user computed on that counter.

Feature	Percentile
#retweet, #url, #mentions	99.9900%
#reply, #favorite, #hashtags	99.9990%

Table 3: Here we can see the percentiles choosed for compute the cut, the corresponding value can be identified in the table above.

Then, we proceed our analysis by checking whether there are syntactic or semantic errors in the values of each feature. Regarding the *lang* field, we normalized all values to lowercase in order to merge values such as "*en-gb*" and "*en-GB*". Then we removed some errors such as the nonexistent values like "*select language...*" and "*xx-lc*". We changed them into "*en*" language because we extracted the tweets of these users and we noticed that they were written in English. There were missing IDs in tweets dataframe, we decided to replace them by creating new IDs using the `.fillna('ffill')` method of Pandas. Speaking about the incoherent data values we analyzed the distribution of the IDs

through the time, we assumed that the incoherent dates can be substituted thanks to them. The assumption is since the IDs are an incremental number, like we can see from 2b, we can extract all the tweets with coherent dates, than pick the days from each of those and assign a representative ID choose with the median (trying to avoid outliers or modified tweets), in the end we used the closest id to select the correct day and substitute it to the wrong one.
To fill missing values on *num_hashtags* and *num_urls* we decided to check with regular expression on "text "attribute of the tweet if hashtags or urls are present and, if they are present, we count them.
The null value in the name field are been filled with the **UNKNOWN** string, and because during the data understanding phase we saw that only real users have NaN values in the *statuses_count* feature, to clear this feature we calculated the mean for actual users only and substituted it for the NaN values.
Finally, we fixed the length of the tweets. During data understanding phase we saw that there are some tweets which have more than 280 character. This problem occurs for alphabets that use different characters, because them are encoded with more than 1 character, so we fixed the problem encoding and decoding the text field properly.

3.2 Indicators

In order to properly develop the clustering and classification part, we extended the users dataframe with the following indicators calculated with respect to the single user.

number_of_tweets Indicates how many tweet were published in total by the user.

tweets_2019 Number of tweets published in 2019 for that user.

tweets_2020 Number of tweets published in 2020 for that user.

likes_sum Total number of likes received by the user for its tweets.

likes_max Maximum number of likes received by the user on its tweets.

likes_mean Average number of likes received on a tweet by the user.

time_delta_sec Average time in seconds between two consecutive tweets.

favorite_count_entropy Entropy on favorite count for the user.

num_hashtags_entropy Entropy on number of hashtags for the user.

created_at_entropy Entropy on created at count for the user.

mean_length Average of the length of the tweets published by the user.

max_length Maximum of the length of the tweets published by the user.

number_of_special_chars Number of special characters written by the user on its tweets.

num_hashtags_mean Average number of hashtags in the users's tweets.

num_hashtags_max Maximum number of hashtags in the users's tweets.

num_urls_mean Average number of urls in the users's tweets.

num_urls_max Maximum number of urls in the users's tweets.

num_mentions_max Average number of mentions in the users's tweets.

num_mentions_mean Maximum number of mentions in the users's tweets.

user_number_of_retweets How many retweets made the user.

3.3 Results of data preparation

As described previously the work done to aim to get coherent and summarized data to perform clustering algorithms.
Keeping this in mind we have performed some plots to point out the differences with Section 2.2; the first is the matrix of box plots (Fig. 3), following the histogram of $created\_at$ values (Fig. 4a), the scatter plot of the IDs (Fig. 4b) finally the histogram on the text lengths (Fig. 4c).
At the end there is the correlation matrix, we used it to choose the best representative indices (Fig. 5).

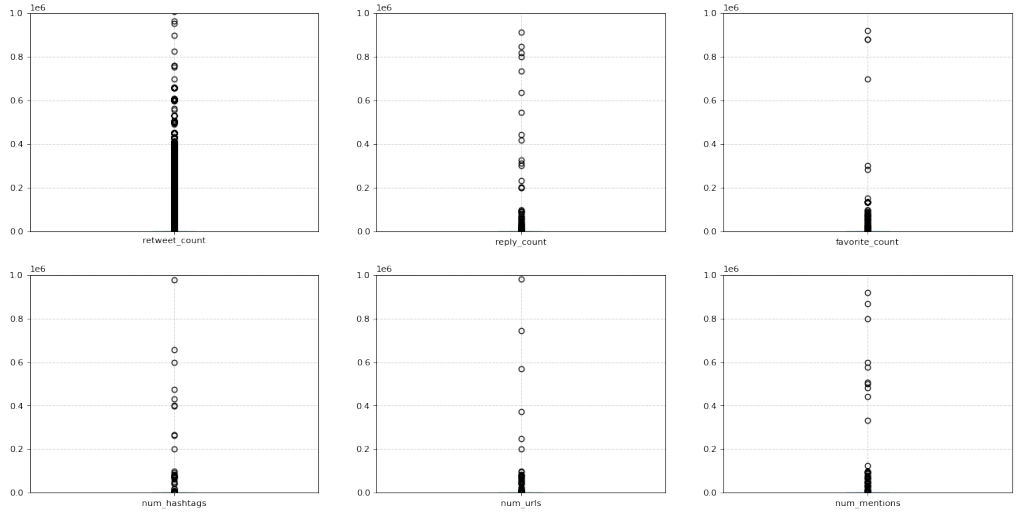
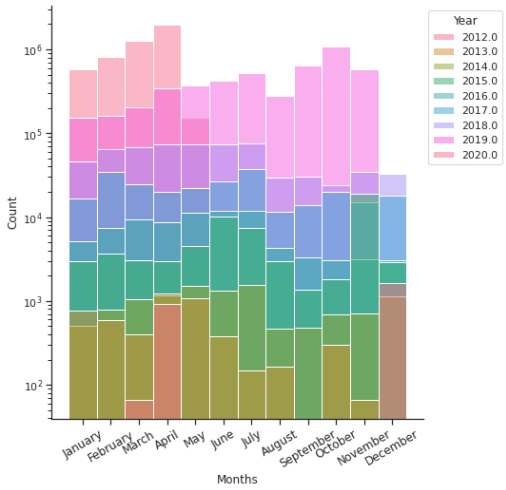
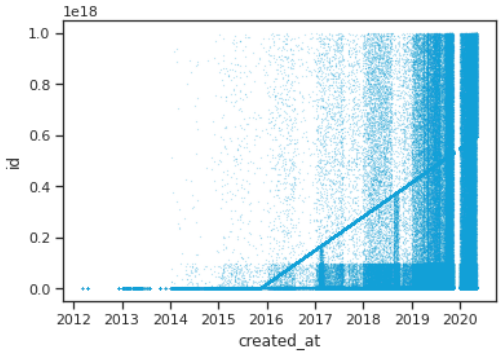


Figure 3

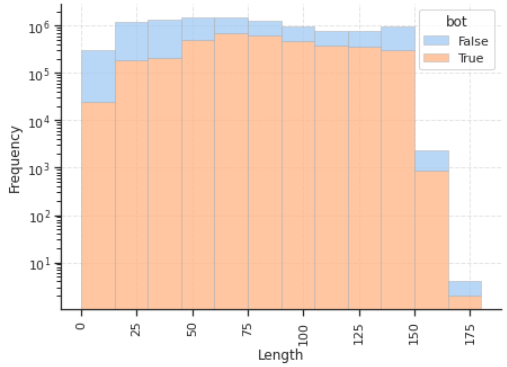
It is clear that the preparation section has scaled down the counter fields; here the scales are very small compared to the previous box plot (Fig. 1).



(a) Creation of the tweets [hue=year]



(b) Creation of the tweets over the ID



(c) Length of the tweets [hue=bot]

Figure 4: As we can see now the data have all coherent frames and the curvature of the data was left as unchanged as possible.

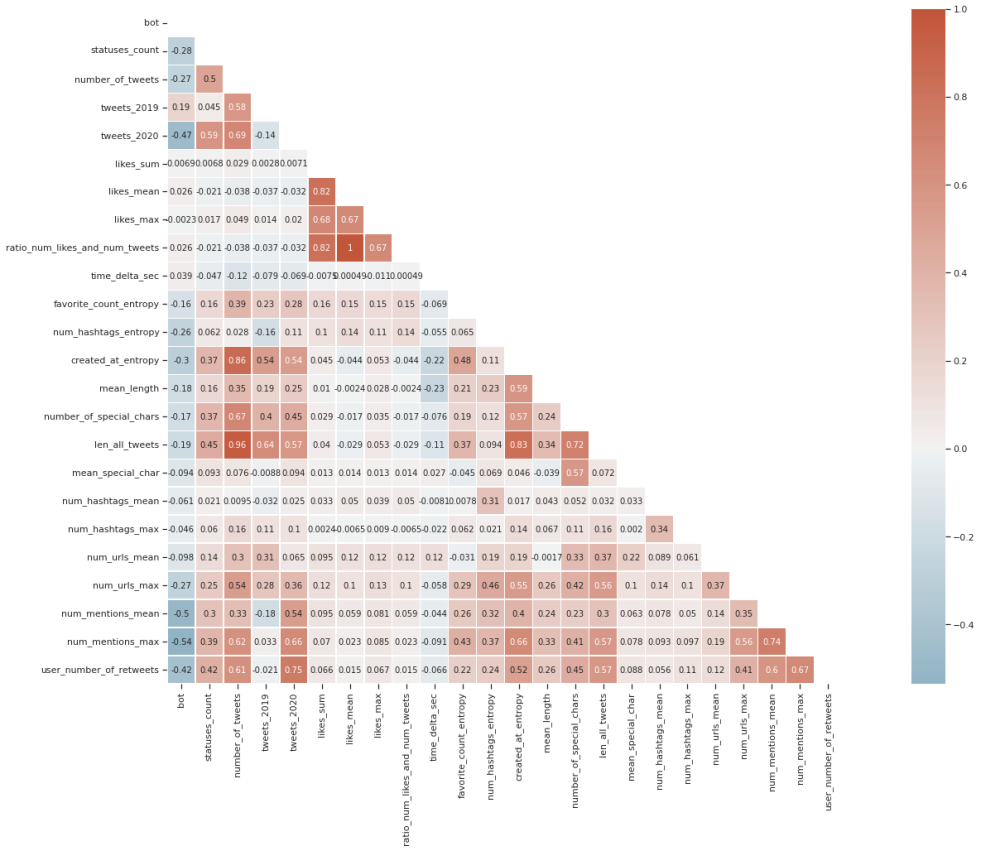


Figure 5: Correlation triangular matrix on whole the indices representing the users.

As we can see different indices have a high positive correlation and some even negative stated that we removed once for each of these pairs and choose a few indices that were most representative of the uniqueness of the meaning.

4 Clustering

4.1 Preprocessing

Since clustering algorithms perform poorly when many features are taken into consideration, so, we consider only a few attributes to avoid the Curse of Dimensionality problem. Clustering was performed **only** on the following features: "*likes_sum*", "*number_of_tweets*", "*user_number_of_retweets*", "*time_delta_sec*", we thought that these attributes gives us general hints on the usage of Twitter by his members. Moreover, these indicators are not correlated with each other.

In order to extract meaningful information about groups of users representing a certain behavior, we thought it was necessary that they must have written a minimum number of tweets equal to 10. Therefore, we eliminated (just for the clustering phase) all users who did not meet this parameter. The total number of users considered now is **11085** users (out of **11508**, we deleted the 0.036% of the total).

In light of the fact that the values we have within the features have very different orders of magnitude between them, we used log-transformation to try to normalize them to each other. After this, we used the MinMaxScaler.

4.2 K-Means

K-Means is an unsupervised learning algorithm for clustering. The main parameter in K-Means is k, the (fixed) number of clusters, we found this parameter using an external library (kneed[1]) which provides an easy way to perform the Knee method.
Each cluster is defined by a centroid and the main aim of K-Means is to minimize the sum of distances between the data point and their corresponding clusters, first assigning to each point a centroid and then fixing the centroids minimizing the distances. With respect to the SSE graph (Figure 6), the Kneed library found $k = 6$, and the value of inertia with these values is 213.98.

[1] https://pypi.org/project/kneed/

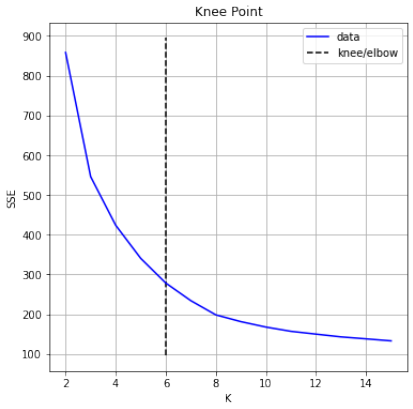


Figure 6

4.2.1 Analysis

Following we will describe some consideration per each cluster in each configuration.

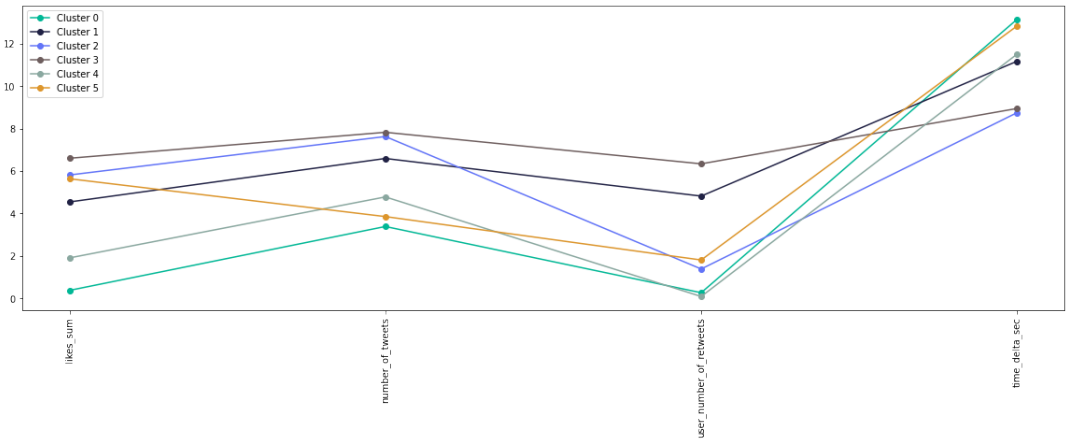


Figure 7: Distance of each cluster among the chosen attributes.

Looking at the graphs we can distinguish some information for each of the clusters.

Green cluster This cluster contains users that have tweeted with low frequency, received a low amount of likes and retweets. We can notice that here there are

674 genuine users and 1479 bots, so we can think that these are people that use not so much Twitter and old bots or not so useful ones.

Brown cluster The users gathered in this cluster have a tweet frequency slightly higher than the ones in the green cluster, but they have lots of likes and retweets. This may indicate that these are celebrities or appreciated bots. This is also confirmed by the low number of points present here, only 529, indeed users like this are rare.

Orange cluster This is the biggest cluster among all, in fact, we can see an average behavior both in the number of published tweets and likes received, remembering that Likes Sum is the sum of total likes received, we can immediately infer that the average likes per tweet drops to the increase of tweets. Here we find 1140 non-bots and 2511 bots

Grey cluster Grey users posted a higher number of tweets than average, as well as number of likes and retweets. Since they have a higher posting frequency we can assume that they are daily users, which is confirmed by the presence of predominantly genuine users 91

Blue and Black clusters Finally, these 2 clusters are really close in all the indices but Number of retweets, this clue says that blue users interact way less than the black one with the Twitter community. Evidence of this is the quantity of bots and real user in the clusters: the blue one is mainly composed by bots (1292 on 1384), while the black cluster consist of genuine users (2212 user on 2375).

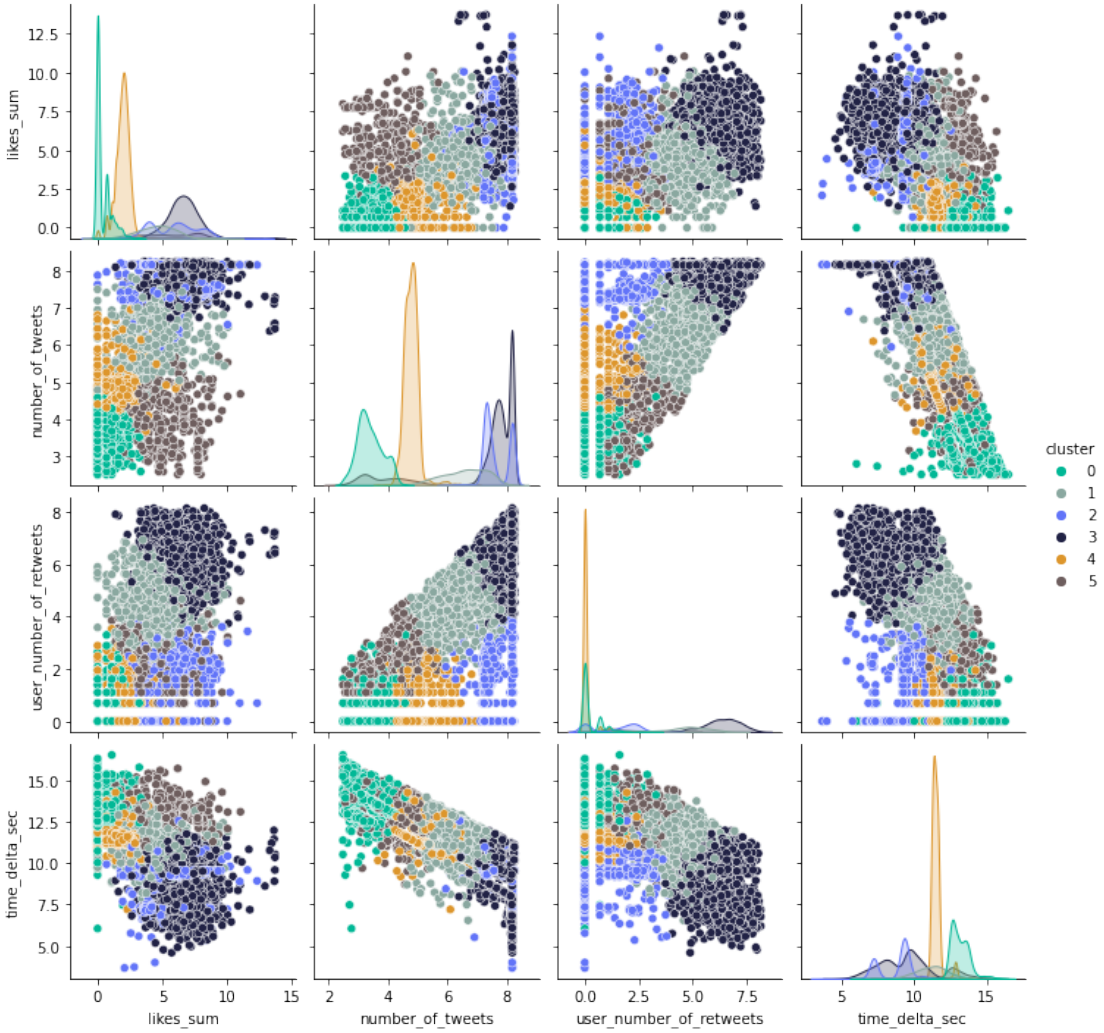


Figure 8: These are the clusters plotted in a matrix $n * n$

4.3 DBSCAN

DBSCAN is a clustering technique which can cluster any shape. It is a widely used algorithm and only relies on two parameters *epsilon* and *minimum samples*.

- **epsilon**: The maximum distance between two input data to be considered as in the same cluster.
- **minimum samples**: the minimum number of input data needed to create a cluster.

In order to find the optimal value for *epsilon*, we performed the K-NearestNeighbor algorithm to compute the average distance of every data point to its k-nearest neighbors. The best epsilon is given again using the Knee method. Figure 9 shows the result.

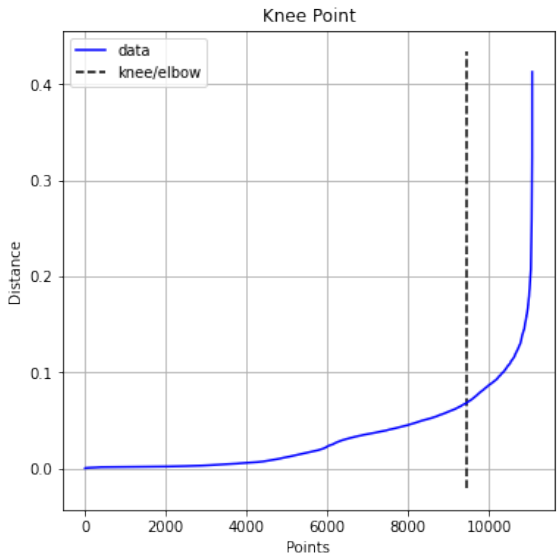


Figure 9

In particular we found that the optimal value for epsilon is 0.068, the number of clusters is 8 and the number of noise points is 2378. The seven clusters have this dimensionality:
Cluster 1: 3336 points.
Cluster 2: 1996 points.

Cluster 3: 330 points.
Cluster 4: 432 points.
Cluster 5: 1537 points.
Cluster 6: 747 points.
Cluster 7: 164 points.
Cluster 8: 356 points.
The overall silhouette value is 0,387.

4.4 Hierarchical Clustering

This is a clustering approach that aims to build a hierarchy of clusters. The result can be visualized by a plot called *dendrogram*. This algorithm doesn't require a fixed number of clusters but it's based on a linkage function, a function that describes the distance between two clusters. This function can be executed using different methods.
All methods were used on normalized data, as in previous algorithms.

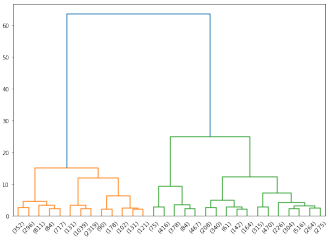


Figure 10: Ward

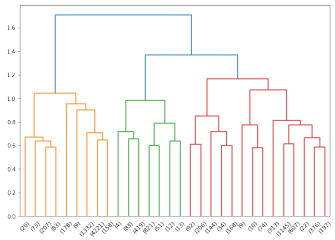


Figure 11: Complete

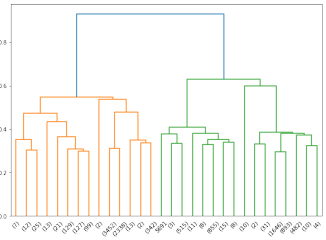


Figure 12: Average

Ward Method Through this method we identified 2 clusters, using *25* as a threshold. The silhouette score is *0,629*.

Average Method Through this method we identified *2* clusters. The silhouette score is *0,62*.

Complete Method Through this method we identified *3* clusters. The silhouette score is *0,59*.

Among all these methods we think the best one, for our objective, is the Ward method given the Silhouette score value.

4.5 Conclusions

In conclusion, we think that the obtained results of K-Means return more meaningful results regarding user behavior. Anyway, we would like to explore better the results of the all clustering algorithms, taking into account the different properties extracted by each of those, emphasizing the overlap of data and the significance that can result from it.