Bots in Twitter

Data Mining project

Erriquez Domenico

Ferraro Domenico

Khanlari Elshan

University of Pisa

2022/23

Table of contents

1 Introduction

The goal of this project is to perform several data mining tasks. We have two datasets, a dataset of users and a dataset of tweets. Before doing any analysis, a more explanatory analysis is needed so we can get a feel for the dataset. Data understanding is then the first step that we will do, followed by a data cleaning and preparation phase. These require more intensive work and attention.

2 Users dataset: data understanding and cleaning

In the first phase, the dataset will be analyzed as it is, without making any changes. The goal is to understand the quality of it. The second phase will be to fix all the errors and replace the missing values.
The users dataset contains **11.508** total observations and **5** features: name, lang, bot, created_at, statuses_count. The index has a int64 data type.

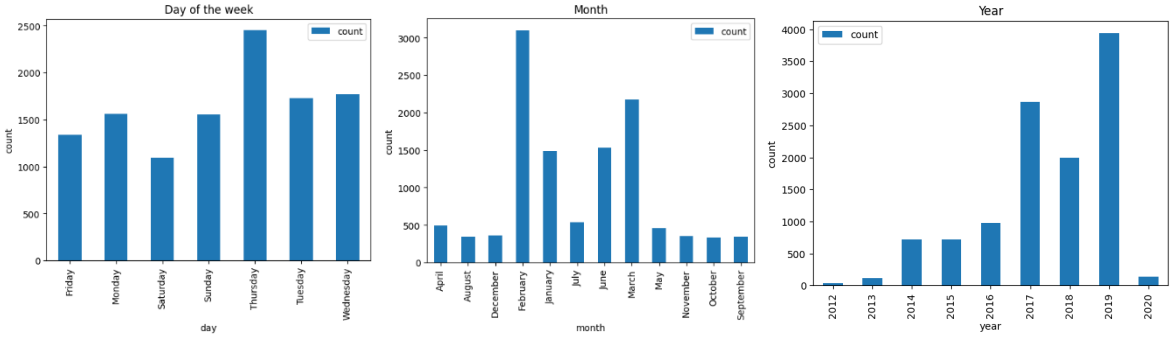
Attribute	Type
Name	Object
Lang	Object
Bot	int64
Created_at	Object
Statuses_count	Float64

Data type of Users Dataset's attributes.

2.1 Feature understanding

- **User_id**: it is an *integer* that identifies the user. There are **11.508** unique values and there are no missing values. There are no duplicate values.
- **Name**: it is a *string* that represents the name of the user. There are **11.361** unique values. Because the unique names are less than the number of the unique identifiers, there are multiple users with the same name. The number of missing values is just one.
- **Lang**: it is a *string* that represents the language selected by the user. There are **26** unique values and there are no missing values. The 3 most used languages are: English (9970), Italian (906) and Spanish (319). There are **3** invalid entries: 2 Select Language… and 1 xx-lc. These invalid entries are there only for genuine users and not for bots.
- **Bot**: it is a *binary integer* which indicates if a user is a bot or a genuine user. The value 0 indicates that the user is a real one, while the value 1 indicates that it is a bot. There are no missing values. The number of users that are bots is **6.116**, and the number of genuine users is **5.392**.
- **Created_at**: it is the date in which the user profile was created. The format is *%Y-%m-%d %H:%M:%S*. In the dataset it is an *object* data type. To use it as real date, a parsing to a date type is

needed. There are no missing values. The range of the data is from 2012-01-24 to April 2020-04-21. The most frequent day of the week when the users have been created is Thursday (2453), while the less common day is Saturday (1093). On the dataset 4 months have a frequency much higher compared to other months, more precisely the most frequent months are the following: February (3098), March (2178), June (1530) and January (1483). From 2012 to 2016 there was a slight growth in the number of users. From 2017 to 2020 there is a sharp increase in the number of users, where the highest one is 2019 with the count of 3935.



From left to right, the number of created users by day of the week, month and year.

- **Statuses_count**: it is a *float* that represents the number of tweets made by the user at the moment of data crawling. There are **399** missing values. There are no negative values. The distribution has a long tail, so a logarithmic transformation has been applied on the values. Furthermore, it has been analyzed separately the distribution for the real users and for bots. By plotting a histogram of real users and bots, we can see that the distribution is different between real users and bots. The average number of tweets made by bots it is **1.185**, while for real users it is **11.638**.

2.2 Type casting

Most of the features have correct data type, so a casting is not needed. The only feature with an incorrect data type was the *created_at* feature. We casted it from object to pandas date type.

2.3 Data cleaning

The first thing to do is to transform to lowercase all the letters, since otherwise there are equivalent values considered as different (e.g., “en-gb” and “en-GB” in lang). We also removed leading, trailing and double spaces where present because they could lead as well to the same kind of problems. After these transformations there are 24 unique languages instead of 26. Looking at the tweets made by the users with an invalid language, we figured out that we have some of their tweets in the tweets dataset. They wrote in English their tweets on the social media, so we assumed that these users are English speakers and we transformed their *lang* attribute into “en”. We removed the languages specifications, i.e. we aggregated the languages “en-au” and “en-gb” to “en” and the languages “zh-cn” and “zh-tw” to “zh”.

2.4 Replace missing values

At this step in the users dataset there are only two attributes with missing values: name and statuses count. The only missing **name** attribute has been transformed to “Unknown”. The missing **Statuses count** values have been replaced by the median which was calculated by aggregating the users by the attribute Bot.

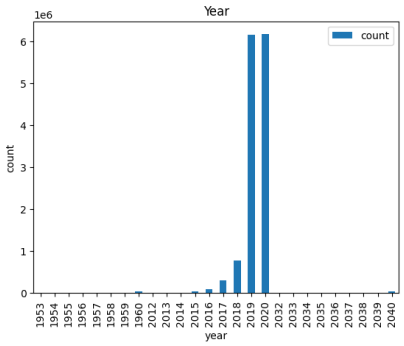
3 Tweets dataset: data understanding and cleaning

The tweets dataset contains **13.664.696** total observations and **10** features: id, user_id, retweet_count, reply_count, favorite_count, num_hashtags, num_urls, num_mentions, created_at and text.
In this task we will focus on analyzing the dataset and on fixing missing values, removing duplicates and fixing the errors.

3.1 Feature understanding

In this dataset each feature is an *object* data type. As we will see later in the analysis it will be needed to apply a proper casting to each of them.

- **Id**: it identifies the tweet. There are **2** missing values.
- **User_id**: it is the identifier of the user who wrote the tweet. There are **222.286** unique values and there are **217.283** missing values
- **Retweet_count**: it is the number of retweets of the tweet. For some tweets we have a numerical value but there are some that are strings. There are **437.134** missing values.
- **Reply_count**: it is the number of replies of the tweet. There are **647.878** missing values.
- **Favorite_count**: it is the number of favorites (likes) received by the tweet. There are **647.542** missing values.
- **Num_hashtags**: it is the number of hashtags used in the tweet. There are **1.057.524** missing values.
- **Num_urls**: it is the number of urls in the tweet. There are **648.623** missing values.
- **Num_mentions**: it is the number of mentions in the tweet. There are **854.165** missing values.
- **Created_at**: it is the date when the tweet was created. There are no missing values. The range of dates is from 1953 to 2040. Before 2012 there are dates that are from 1953 to 1960, so they are all mistakes. After 2020 there are dates from 2032 to 2040, so even them are mistakes as well. The number of tweets grows year by year. In 2012 there are fewer tweets, while in 2020 (6.1 million) there are the highest number of tweets.



- **Text**: it is the text of the tweet. There are **537.721** missing values.

All the features that should contain only integer values have some records that are semantically wrong even though they are integers. During the next phase of data cleaning a threshold for each attribute will be chosen to set to NaN all those records that are above it.

3.2 Type casting

Most of the features have an incorrect data type, so a casting is needed. We casted the *created_at* from object to pandas date type. All the other features but the text must be casted to integer. This was done in multiple steps because there may be some string as well as integers and floating values. We initially set each string value to *NaN* and then we casted the feature to a numerical data type. This converted each object data type into float64 data type. Furthermore, every float record (i.e, with floating point value) has been set to NaN. Later, after replacing the missing value, we will cast all the float64 attributes to the more appropriate int64 data type.

3.3 Data cleaning

As we did for the users dataset, the first thing to do is to transform all the letters to lowercase, since otherwise there may be equivalent values considered as different. We also removed leading, trailing and double spaces where present. All the duplicates have been dropped, where for duplicates it means all the records that have all the attributes with the same value (except for Id). Now we will transform these errors to NaN. The following are the threshold chosen for each attribute.

Retweet_count: The threshold chosen for retweet count is 4.100.000. This is because looking at this page of Wikipedia, we found out that the tweet published before April 2020 with most retweets had 4.000.000 retweet.

Reply_count: The threshold chosen for reply count is 200.000, this is because looking at the tweets most famous (with more retweets and likes) they had several replies that was around 100.000-150.000. So, 200.000 seems to be a reasonable threshold and all the records above will be considered an error.

Favorite_count: The threshold chosen for favorite count is 4.200.000, because by looking at this page of Wikipedia, we found out that the tweet published before April 2020 with most like had 4.100.000 likes.

Num_hashtags, **num_urls**, **num_mentions**: The threshold chosen for all the three attributes is 40, this is because also looking the distribution it seemed to be a reasonable threshold.

Created_at: All the dates before 2012-01-01 and after 2020-12-31 have been settled to NaN.

Text: The texts with only spaces have been replaced with NaN.

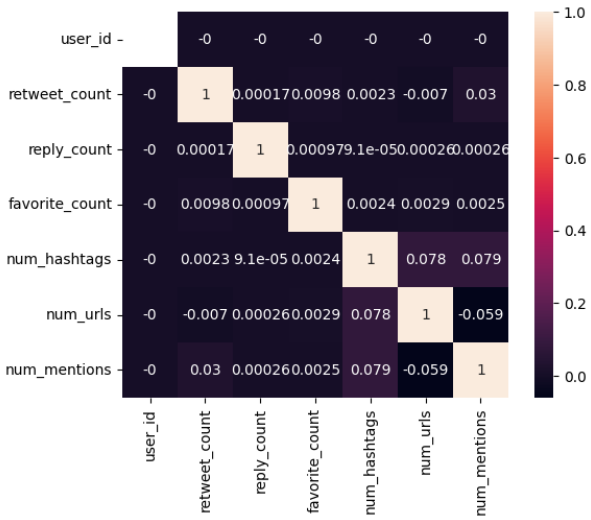
3.4 Replace missing values

At this point we choose to drop the rows that have at least 5 over 9 missing values. They were **656.680**. The last step for the data cleaning section is to replace the missing values. For the records that had the feature *user_id* with value NaN, we replace it with "-1". The missing *text* have been replaced with an empty string.

The records that had the feature *created_at* with value NaN were replaced with “1900-12-12 12:12:12”. For all the other numeric values, looking at the distribution they all had a skewed distribution. Since the outliers are still present, we choose to replace the missing values with the median. The median for each attribute is calculated grouping by *user_id*, for example if a tweet has a missing favorite count value, we replace it with the median computed by looking at the other tweets made by the same user and with a valid favorite count value.

3.5 Conclusion data understanding and data cleaning

At this point in the tweets dataset there aren’t any missing values and any errors. The dataset still contain outliers. From the picture below it is possible to see that there is no correlation between the attributes.



Correlation matrix of Tweets dataset

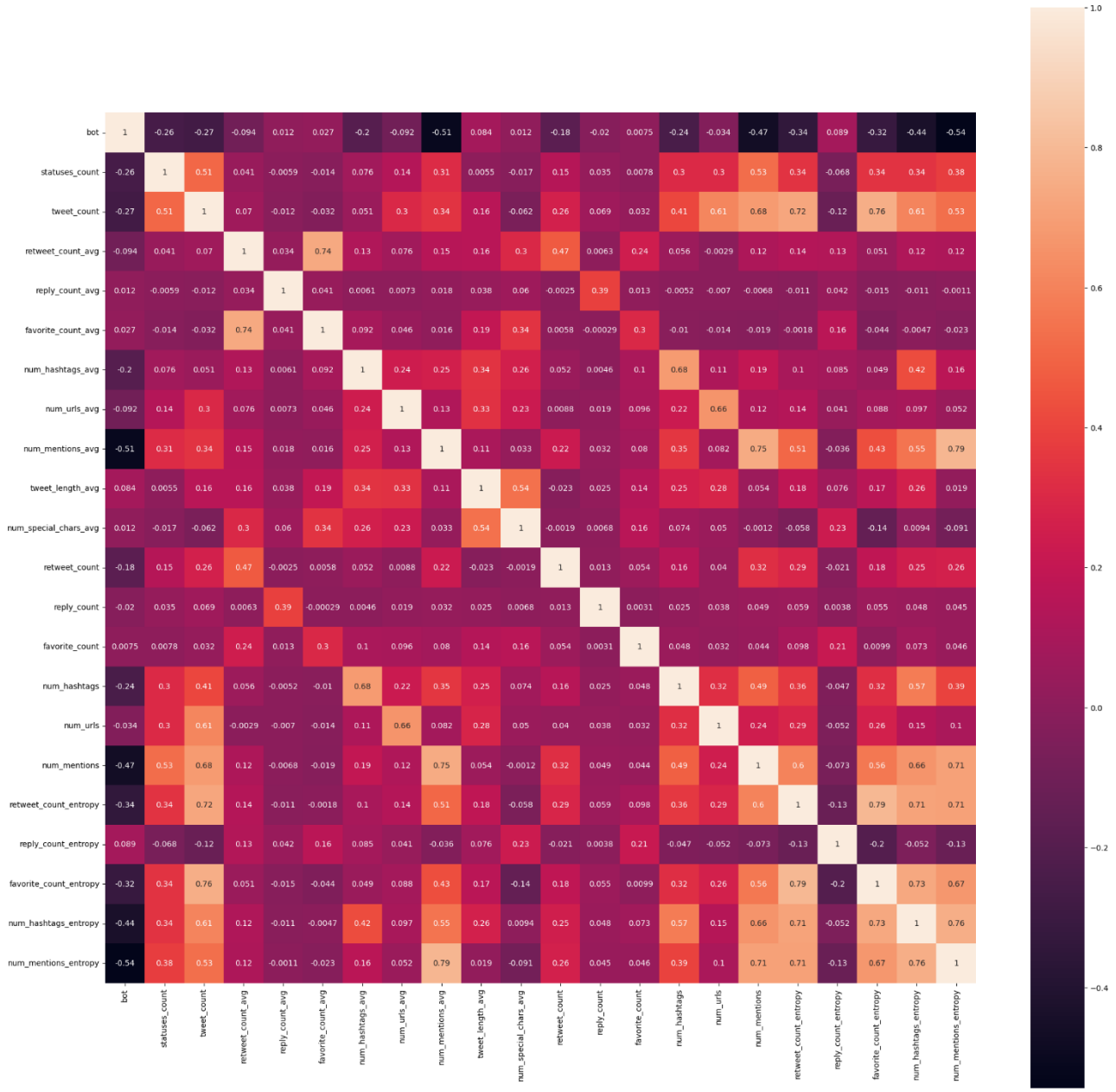
4 Data Preparation

This section’s goal is to create interesting new features for describing the user and his/her behavior. The new indicators are the following with their respective meaning and how they have been computed:

- **retweet_count:** It is the total number of retweets of the user’s tweets. It is computed by summing the number of retweets of the tweets made by the user
- **reply_count:** It is the total number of replies of the user’s tweets. It is computed by summing the number of reply_count of the tweets made by the user
- **favorite_count:** It is the total number of likes received by of the user’s tweets. It is computed by summing the number of favorite_count of the tweets made by the user
- **num_hashtags:** It is the total number of hashtags that the user’s wrote in its tweets. It is computed by summing the number of num_hashtags of the tweets made by the user
- **num_urls:** It is the total number of urls that the user’s wrote in its tweets. It is computed by summing the number of num_urls of the tweets made by the user

- **num_mentions: :** It is the total number of mentions that the user's wrote in its tweets. It is computed by summing the number of num_mentions of the tweets made by the user
- **tweets_counts:** It is the number of tweets made by the user. It is computed by counting the number of times a *user_id* appears in the Tweets dataset.
- **retweet_count_avg:** It is the average of the retweets of the tweets of the user. It is computed by summing the number of retweets of the tweets made by the user divided by the *tweets_count* of the user.
- **reply_count_avg:** It is the average of the replies of the tweets of the user. It is computed by summing the number of *reply_counts* of the tweets made by the user divided the *tweets_count* of the user.
- **favorite_count_avg:** It is the average of the likes of the tweets of the user. It is computed by summing the number of *favourite_count* of the tweets made by the user divided the *tweets_count* of the user.
- **num_hashtags_avg:** It is the average number of hashtags of the tweets of the user. It is computed by summing the number of num_hashtags of the tweets made by the user divided the tweets_count of the user.
- **num_urls_avg:** It is the average of the number of urls of the tweets of the user. It is computed by summing the number of num_urls of the tweets made by the user divided the tweets_count of the user.
- **num_mentions_avg:** It is the average of the number of mentions of the tweets of the user. It is computed by summing the number of num_mentions of the tweets made by the user divided the tweets_count of the user.
- **tweet_length_avg:** It is the average of the lenghts of the tweets of the user. It is computed by summing the length of each tweet made by the user divided the tweets_count of the user.
- **num_special_characters_avg:** It is the average of the number of special characters used in tweets of the user. It is computed by summing the number of special characters of each tweet made by the user divided the tweets_count of the user.
- by calculating the entropy for each attribute, we computed also the following new indicators: *retweet_count_entropy, reply_count_entropy, favorite_count_entropy, num_hashtags_entropy, num_urls_entropy, num_mentions_entropy, tweet_length_entropy, num_special_chars_entropy.*

We dropped all the couple of features that have more than '80%' of correlation. Leaving just one feature per correlated pair. The features dropped are *num_special_chars_entropy, tweet_length_entropy, tweet_length, num_special_chars, num_urls_entropy*.



Correlation matrix of the tweets dataset with the new indicators

5 Clustering

In this part we explore an in-depth comparison of different clustering algorithms, such as K-Means, DBScan, Hierarchical. The dataset is made by 11.508 records and 25 attributes.

5.1 Features selection

We have decided to select the features respecting the following criteria in order:

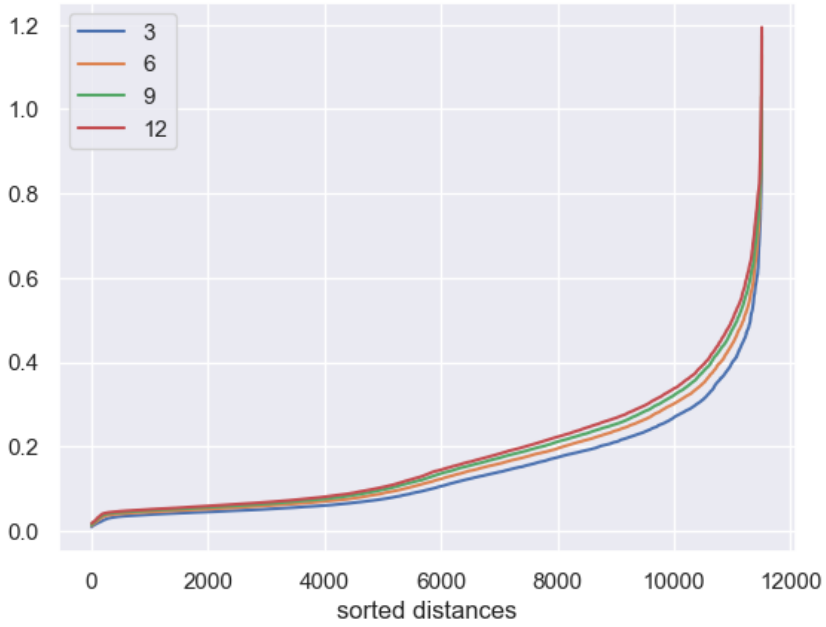
- Removed categorical features which are: *name*, *lang*, *created_at* and *bot*.
- Applied log scale to the features that have a skewed distribution that are all the attributes besides the follwing: *retweet_count_entropy*, *favorite_count_entropy*, *num_hashtags_entropy*, *num_mentions_entropy*.

5.2 DBScan

We performed DBSscan two times, one with the Min-Max normalization and the other one with the Z-Score normalization. This allowed us to understand which normalization technique would give the best results with our dataset. By using the Euclidean metric, we computed the distance between data points. We checked the distances between the k-th nearest values for each possible point and with *k* values of *3*, *6*, *9* and *12*. This will allow us to select the best *eps* and *min samples* values for the DBScan.

5.2.1 Min-Max normalization

To understand a good range of values for the *eps* parameter, we print the plot below to check the distances between the k-th nearest values for each possible point. Each distance is plotted in the x-axis, ordered with respect to the k-th nearest value.

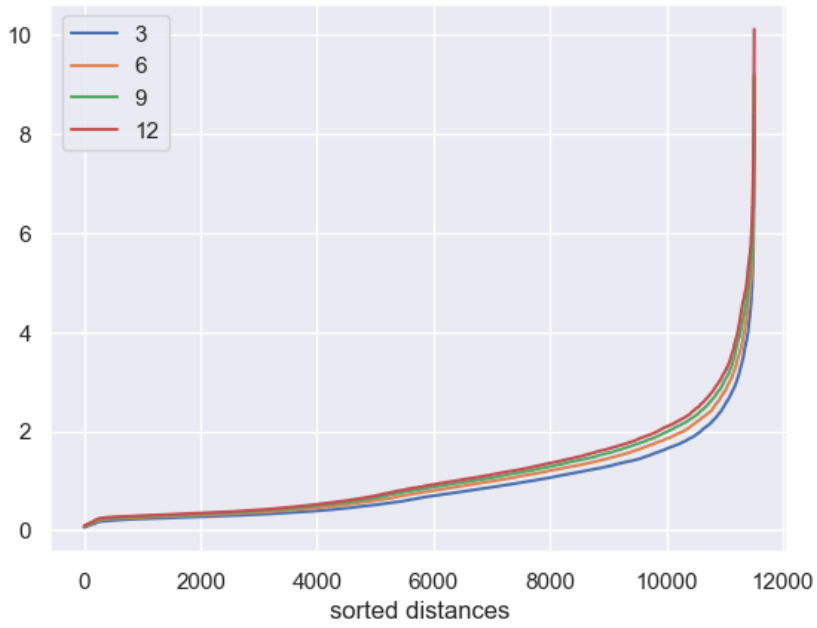


Results of the distances between the k-th nearest values for each possible point.

Looking the plot, the elbow point gave us the smaller range for optimal *eps* value. We choose the range from *0.2* to *0.6* with steps of *0.1*. We considered a *min samples* range from *3* to *21* with steps of *3*. By iterating over these combinations, we got the best silhouette score. From our results we obtained that the best epsilon was 0.2 while the best *min sample* was 15. Giving these parameters to DBScan we obtained a silhouette score equal to 0.1159, Davies Boulding equal to 0.7796, 2 clusters and 11461 outliers. So, the DBScan with MinMax normalized failed to perform clustering in the dataset. The silhouette score near to zero suggests that the clusters are overlapping.

5.2.2 Z-Score normalization

To understand a good range of values for the *eps* parameter, we applied the same strategy done with the min-max normalization. Again, each distance is plotted in the x-axis, ordered with respect to the k-th nearest value.



Results of the distances between the k-th nearest values for each possible point.

Looking the plot, the elbow point gave us the smaller range for optimal *eps* value. We choose the range from *1* to *4* with steps of *0.5*. We considered *min samples* range from *3* to *21* with steps of *3.* By iterating over these combinations, we got the best silhouette score. From our results we obtained that the best epsilon was 1.5 while the best *min sample* was 12. Giving these parameters to DBScan we obtained a silhouette score equal to 0.1869 and Davies Boulding equal to 1.6186, *21* clusters and *3635* outliers. The silhouette score near to zero suggests that the clusters are overlapping.