Data Science Workshop:
Twitter Bot Detection

Team Members:

Nataly Koren - 308446624

Michal Wasserlauf - 203264775

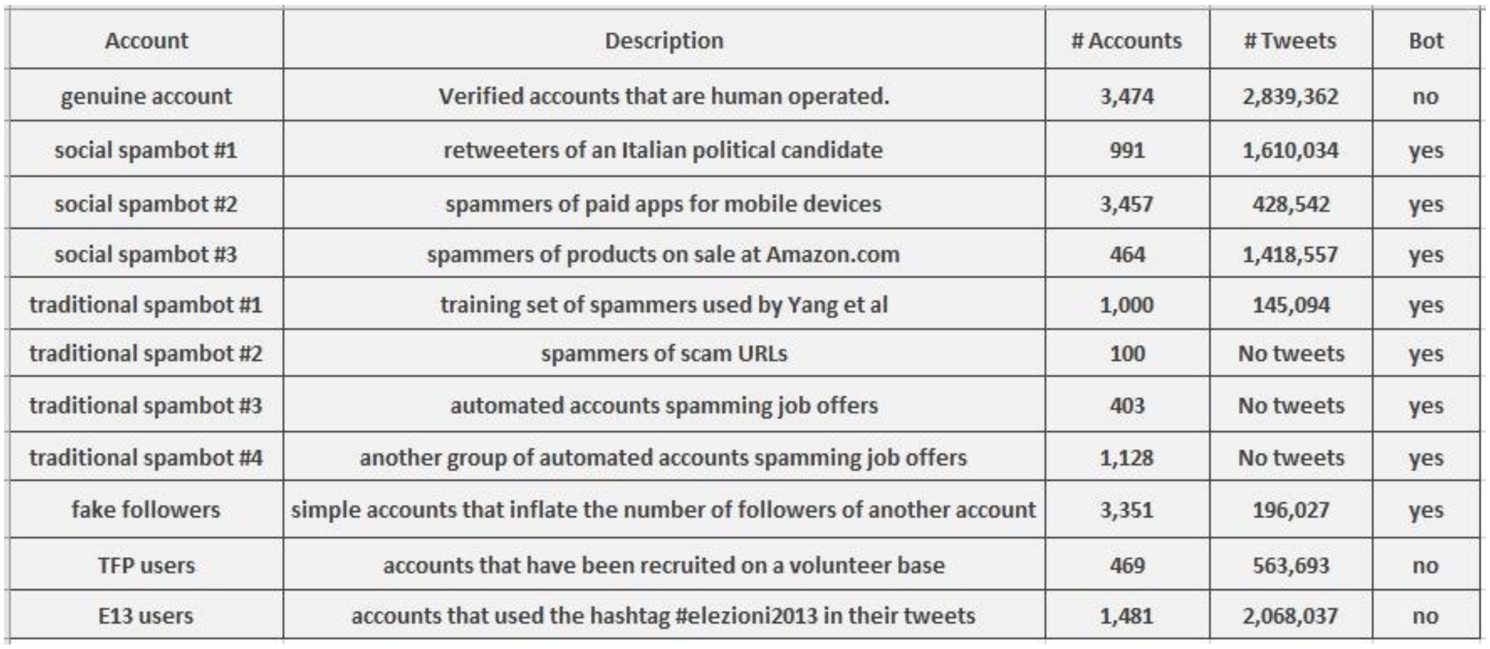
Gil Ariel - 302329446

Chen Mordoch - 203425780

Twitter Bot Detection

1. Dataset Description:

Our dataset was driven mostly from a research explained in a paper called **"**The paradigm-shift of social spambots" (https://arxiv.org/pdf/1701.03017.pdf) which was submitted to the International World Wide Web conference. Due to imbalance of target value, we also used a part of another dataset used in the researchers' former research. Our data consists of different account types from 2009 to 2014, for each we have different **users' information** file and **tweets** file:



Account	Description	# Accounts	# Tweets	Bot
genuine account	Verified accounts that are human operated.	3,474	2,839,362	no
social spambot #1	retweeters of an Italian political candidate	991	1,610,034	yes
social spambot #2	spammers of paid apps for mobile devices	3,457	428,542	yes
social spambot #3	spammers of products on sale at Amazon.com	464	1,418,557	yes
traditional spambot #1	training set of spammers used by Yang et al	1,000	145,094	yes
traditional spambot #2	spammers of scam URLs	100	No tweets	yes
traditional spambot #3	automated accounts spamming job offers	403	No tweets	yes
traditional spambot #4	another group of automated accounts spamming job offers	1,128	No tweets	yes
fake followers	simple accounts that inflate the number of followers of another account	3,351	196,027	yes
TFP users	accounts that have been recruited on a volunteer base	469	563,693	no
E13 users	accounts that used the hashtag #elezioni2013 in their tweets	1,481	2,068,037	no

Features description by Twitter API:

User dataset: https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/user-object.html

Tweets dataset: https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/tweet-object.html

2. Dataset Analysis Summary - characteristics and the nature of the datasets:

- As mentioned above, we have information about each user, such as name, followers count, statuses count, profile background color and location. Also we have tweets for most users and data related to tweets such as retweets count, hashtags count and URLs count.
- The data contains 16,318 records, which divide unevenly by target value: 10,894 records of bots, and 5,424 records of genuine users. The data was labeled in an internal verification process as a part of the research.
- For user data we have 43 features and for each user we have 763 tweets in average.
- About 74% of the accounts are in English, 16% in Italian, 2% in Spanish and the rest belong to other languages such as Portuguese, Russian, French, and Japanese.

- Null Values – about 36% of our data consists of null values, mostly concentrated in specific columns consisting a majority of null values.
- Correlations -
    - High correlation between the count of friends, followers and listed count (~0.8).
    - For basic user features, without engineering, no significant correlation with the target value (the highest is around 0.3 for status and favorites count).

3. Problem Formulation and Goals:

We will predict whether a twitter account is likely to be a spambot or not.
Since this is a binary classification problem, we chose those measures (which appeared in the paper mentioned above as well): Precision, recall, F1 score, Accuracy, MCC and Specificity. Our goal was to find a model maximizing all those measures, focusing on achieving high accuracy (since it indicates how well we label the accounts) and high specificity (since there’s a high price for labeling a genuine account as spambot).

4. Point of Focus:

In our work we focused on extracting data from text features as much as we can, and improve usage of user data. In other words, we focused on text processing methods, some are simple as extracting word count and marking if text contains a URL, and some are more advanced, such as BoW. In addition, we tried to use user features in a way that improves feature-target correlation. To compare with work done in the mentioned research, see “related work” below.

5. Description of the solution:

a. Baseline:

For baseline we used only user features, without any optimization, i.e. using features the way we got them on data files and fill null values by a default value that matches the data type. For example, numeric values got ‘0’ or ‘-1’ as default. String and text features were replaced by length. By using random forest we got the following results:

Recall	Precision	F1 score	Accuracy	MCC	Specificity
0.51	0.65	0.31	0.5	0.03	0.02

b. Data Preparation and Cleaning:

Features removal: We removed columns with more than 16,000 missing values. We have 16,318 samples, therefore we believe that columns with more than 16,000 missing values won’t produce any significant knowledge on the data. We also removed columns that we found irrelevant for our problem (dataset name, crawled at etc.).

Filling missing values: to begin with, we filled the missing values as follows:

Numeric features: replace missing values by zero. If zero is a valid value, we replaced by some other value that is not valid for the particular feature.

Text features: replaced missing values by empty string.

- On Feature Engineering section, for each feature type, we took other strategies for replacing the missing values (indicated by zero or empty string).

c. **Feature Engineering:**

- **Numeric features:** the numeric features suffers from extreme values. For each numeric feature, we took 3 main strategies to increase correlation:
    1. Binning to equal width buckets.
    2. Binning to unequal width buckets, using feature distribution.
    3. Binary binning by most frequent value (1- contains most frequent value, 0 - else).

  We searched for the binning option that maximizes correlation to the target feature. Binning to equal width buckets didn’t yield any significant result, whereas binning to unequal width buckets and binary binning lead to satisfactory results. We believe this is due to the fact that numeric feature’s distribution is very different between bot accounts to genuine accounts.

- **String features:**

  Date columns: we parsed the date columns to create 7 new numeric features: day of the week, month, day in month, hour, minute, second, year.

  At first we believed there may be a correlation between the created time (hour, minute and seconds) to the bot accounts. However, we didn’t find any significant relation except for the created month. We assume it can be related to ordinary events taking place in a certain month (for example elections). For the month feature, we created a binary binning by most frequent months of the bot accounts creation.

  General text features: each general text feature (for example: profile_background_image_url) was replaced by 5 numeric features:
    1. Unique value: in order to keep the original feature distribution, we created a unique mapping from each feature value to a certain number.
    2. Replace missing values in unique mapping by mode: on the resulted feature of step 1 above, we tried to replace the missing values by the new feature’s mode. There are several features with missing values for bot accounts only. Therefore, in this case we replaced the missing values by the bot’s mode value, else, replaced by the feature’s mode.

3. Replace missing values in unique mapping by distribution: on the resulted feature of step 1, we tried to replace the missing values by sampling from the new feature distribution.
4. Binning by is most common: For features we detected a single value that appears much more in relation to other values, we created a new binary feature indicating if the sample contains the most frequent value or not.
5. Replace a string with length: replacing each string by its length.

Location feature: the location feature was binned by 6 most frequent values. The location feature has high variance since it is being inserted by the user. It contains values from actual cities to invented places and sometimes gibberish as well. The bins that were defined were: Italy, USA, Asia and Oceania, Europe without Italy, America without USA, Others. For each bucket we have a list of locations it contains, helping us matching location to bucket.

Color features: each color feature was replaced by 10 numeric features:

1. All general numeric text features mentioned above were added, except for the length feature (text length is fixed).
2. Binning by main colors: when trying to parse the colors for each feature, we found a wide range of different colors. Therefore we decided to define 12 main colors: black, white, blue, red, green, pink, brown, purple, grey, yellow, orange and turquoise. We mapped the main colors to numeric values. Then for each color, we performed binning by calculating the nearest color to it (defined as the color with the minimum distance to the current color). In total, a new feature with 13 unique values was created (one for Nan).
3. Replace missing values in color binning by mode: on the result of the previous step, we tried to replace the missing values by the new feature's mode. Since all missing values of are of bot accounts, we replaced the missing values by the new feature's bot mode.
4. Replace missing values in color binning by distribution: on the resulted feature of step 2, we tried to replace the missing values by sampling from the new feature's distribution.
5. Binning by top 3 colors: After binning by main colors (step 2), we noticed the new feature correlation was not high enough. Therefore we performed more specific binning to top 3 colors for each feature (based on the binning performed on step 2). The top 3 colors do not include the nan values, hence the total binning will produce 4 unique values (0 for the missing values, 1,2,3 for top 3 colors).
6. Replace missing values in top color binning by mode: same as step 3, on the results of step 5.

7. Replace missing values in top color binning by distribution: same as step 4, on the results of step 5.

- **Analyze results:**

When we worked on the features engineering section, we wanted to test the influence of the new created features. To test it, we created a results analyzer. The results analyzer compares the current models results to previous results and calculates improvements measures.

Text Processing

- **Description**

  Each user has a textual field named 'description', containing a description of the account. For it we used basic and advanced processing methods.

  - Our first mission was to translate description text field in order to create a single-language dictionary for full understanding of the data. We used Yandex translation service (which was limited by characters per Yandex key). Before creating a dictionary we preprocessed each text field by lowercase all strings, remove punctuation and numbers, spelling correction, removing stop words and lemmatization. Afterwards we created a Bag of Words and extracted for each description the number of appearances of each word. Then we run a tree model in order to find most important features, and used only ten of them, because we didn't want to affect the weight of other features we have.
  - Another process we used is language detection, since we noticed there are many descriptions that are not in the declared language of the account. Afterwards we created a binary feature called 'different_lang' which was filled with '1' if the detected language is different than the declared one, and '0' otherwise.
  - In addition, we used basic processing techniques to extract more from 'description', such as find number of words in the text, find number of stopwords, find hashtags count, etc.

- **Tweets**

  The Tweets allow us to learn about a user from another aspect - examining behavioral patterns or different behavior between bots and genuine users.

  We have created 5 additional numerical features and one additional binary feature.

  Our tweets-based numerical features are proportion features, calculated for each user:

  - p_retweet - number of total retweets of user's tweets, divided by the number of this user's tweets.
  - p_hashtags - number of total hashtags appearing in user's tweets, divided by the number of this user's tweets.

- p_mentions - number of total mentions appearing in user's tweets, divided by the number of this user's tweets.
- p_urls - number of total urls appearing in user's tweets, divided by the number of this user's tweets.
- avg_tweets_per_hour - average of the number of tweets a certain user tweets in each hour.

We do not have tweet data for the traditional spambots 2-4, so the columns above were filled with the mean value of the feature for bots.

We also wanted to create a feature that represents creativity, since we expect bots to be less creative in their tweets than genuine users. For that purpose we have decided to use **Levenshtein Distance**, which is a minimum edit distance, i.e. given two strings, the Levenshtein distance between them is the minimal number of insertions, deletions and substitutions required to get from the first string to the second.

Since this calculation is expensive, we calculated Levenshtein distance on a sample of 300 tweets maximum per user (unless the total number of tweets for a user is less than 300). It was calculated for 1000 genuine users and 1000 bot users (200 accounts of each kind of bot). At the end it sums up to 2000 accounts in total, which is around ⅛ of our dataset. For each user we calculated the variance of the Levenshtein Distances and for users not in the sample, we filled the void with the mean value of its kind. For instance, all human users were assigned with the mean value of the variances found for humans. As Said before, we don't have tweet data for traditional spambots of kind 2-4, so these were given the mean variance found for traditional spambots of kind 1. At the end, we looked at the variance for bots and genuine users and created a binary feature, where accounts with a variance of values between 500 and 750 get 1 and otherwise 0 (bot accounts are likely to be outside of this range).

d. Modeling and feature selection:

We used models fitting a binary classification problem: logistic regression, random forest, decision tree, SVM, naïve Bayes and KNN. We also created a model combining the results of the models above, deciding the classification by the majority of classifications. Since SVM and KNN didn't yield results much better than the baseline, they do not appear in our notebook. In addition, we took only features with high correlation to target, staying with 18 features.

6. Findings and statistical evaluation:

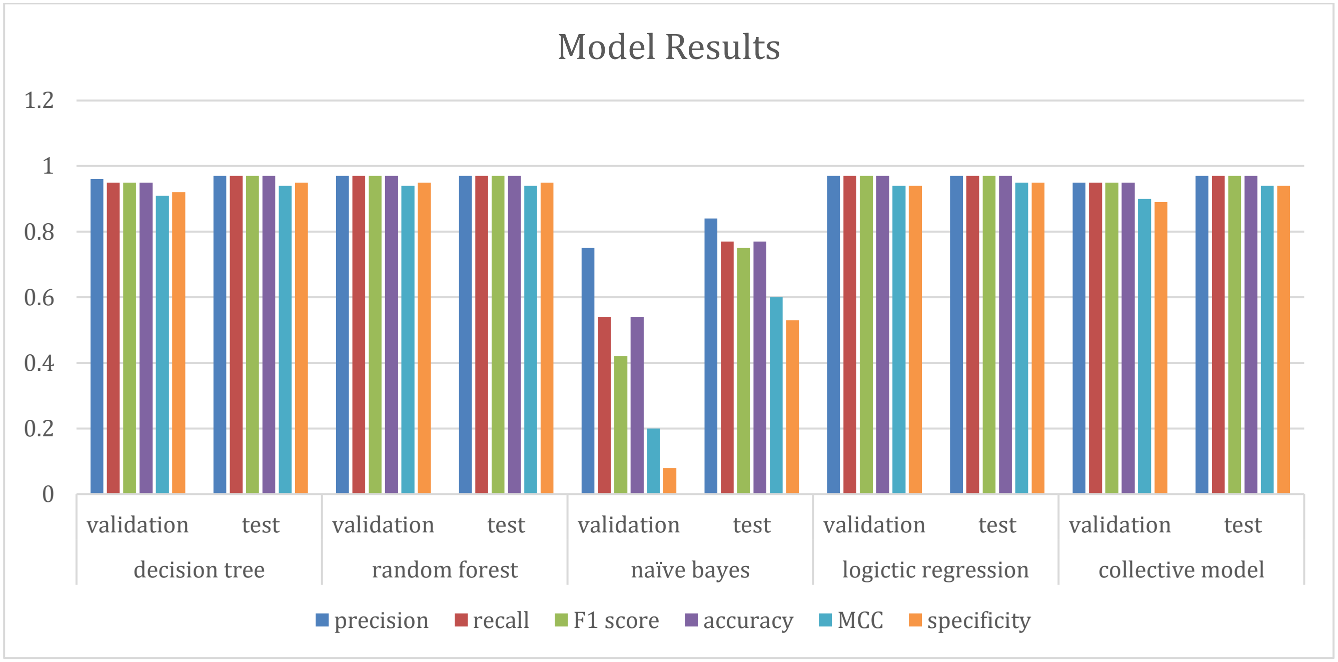
- **Data Division (train-evaluation-test)**

The division was made based on the partition defined by the researchers. In the files given to us appeared two columns: 'test_set_1' and 'test_set_2'. When assigned '1', an account belongs to the relevant test set, otherwise it belonged to the training set. It's important to mention that the partition was made in a way guarantying each test set has equal representation of spambots and genuine users.

In addition, test set 1 contains all of 'social_spambot_1' file accounts and test_set_2 contains all of 'social_spambot_3' accounts. An even amount of genuine users was added to each test set. First test set was considered as validation set, and the other as test set.

The reason we chose to divide the data this way is that it represents reality better, and makes the detection process harder. Since in reality spambots are usually generated for a specific target (elections for example), and we usually don't see similar examples before. We also wanted a point of comparison with the papers' results.

- **Performance**



- **K-fold**

In order to ensure robustness we used k-fold cross validation, with a common choice of k=10. We used a dataset containing high correlated features and examined the results of the best model in the evaluation process – **random forest**. Cross validation concludes our model is robust.

7. Insights and applications:

We believe the final 18 features creates a robust tweeter user profile. Using those features creates a classifier that is more resistant to different bots' profiles. Furthermore, the tweets data or more specifically – the binary variance feature, yields very high and accurate results. It is strongly indicating a genuine account. By looking at the data we can see that spambots have very similar "patterns" of tweets among each user. Variance binary feature has a correlation of 0.89 to target value. It explains the high score of the best model introduced in the paper as well (see below).

8. Related work - "The paradigm-shift of social spambots":

Article's main goal was reviewing different methods available today for detecting the new evolving kind of Twitter bots - social spambots, which are very similar to genuine accounts. At the beginning, the paper mentions the emerging importance of the problem, since today twitter doesn't detect social spambots very well. Furthermore, even humans find it hard to differentiate between social spambots and genuine accounts. As a solution, researchers mention some popular techniques for handling the problem. Some of the methods are more traditional – relying in accounts' relationships, tweets timing and level of automation. These methods didn't get a high score, with an accuracy that isn't much better than a guess. Later the paper mentions emerging trends that gets much higher scores (above 90% accuracy). The methods are:

- Tamper Detection in Crowd Computations - Detecting whether a group of accounts contains a subset of malicious accounts. This methodology suggests that the statistical distribution of reputation scores is more diverse in tampered accounts. Reputation scores chosen for this case are followers count and join date (i.e. this method relies on user's data only). The detection of a tampered computation is performed by computing the Kullback-Leibler distance between the statistical distribution of a given reputation score, and if the distance crosses a threshold - the computation is tampered. The researchers found that social spambots have anomalous distribution, whereas genuine accounts have uniform span across range of values. The results of this model brings an accuracy of above 92% and a specificity of above 91% for both test sets.
- Digital DNA - Similar to previous method, this method tries to detect a subgroup of malicious accounts. Inspired by the human DNA sequence, researchers created a DNA sequence that represents each account. The DNA sequence was created from tweets data, for content of tweets and type of tweets. Then the sequences created are compared to each other to find longest common substring. Accounts that share a suspiciously long DNA are then labeled as spambots. This method relies on the assumption that spambots will have similar DNA sequences. The results of this model brings an accuracy of above 92% and a specificity of above 98% for both test sets. Detailed results can be found on *table 7* of the paper.

9. Citations:

- **Python libraries:** textblob, nltk, missingno
- **Description processing:** **https://www.analyticsvidhya.com/blog/2018/02/the-different-methods-deal-text-data-predictive-python/**
- **Levenshtein Distance:** **https://en.wikipedia.org/wiki/Levenshtein_distance**