**Detection of social bots on VKontakte social network**

**Beldiushkin Filipp**

# Abstract

This report shows how to detect social bots on social media in general and namely in VKontakte. Detection of social bots is done by employing classical machine learning methods, as well as graphical approaches, based on a user’s profile and their network of friends.

# Introduction

Social bots are usually created in social media to spread spam and fake news. Fake news is often disseminated to influence politicians, elections, and financial markets. Traditional information sources are not suitable for this task, as ordinary people tend to believe ‘ordinary people like me’, not the media ‘affiliated with government or corporations’. That is why social media are used to spread fake news — here every user has their own trusted ‘circle of friends’.

A way to detect social bots on VKontakte social network, in particular by analysing the friendly connections of a user’s friends, will be discussed in this report. The main hypothesis is as follows: social bots choose friends from random users, while real users choose friends from their social circle.

# Data

Open API of the VKontakte social network was used to collect the data a user indicated on his or her profile and the friends a person has.

1797 ids of social bots were downloaded from gosvon.net website. 9969 ids of real users were selected randomly from VKontakte.

Data of 2 types was collected for each account:

1. Data indicated by a user in their profile (listed in Appendix A)
2. Graph features extracted from the network of friends

The main attention in this report is paid to the graph of the friend connections. The graph was built using the VK API as follows: for each of the 11766 accounts (denoted by V1 set), their friends were taken (denoted by V2 set). For each of the accounts of the V2 set, their friends were taken (denote by V3 set). After that, for each account (1 specific account denoted by v), from set V1 a graph Gv is built, which consists of the node v itself, and its adjacent nodes, which are taken from set V2. Node v is connected to all the nodes in set V2 and all nodes in V2 are also connected to those in both V2 and V3. The edges of this graph are friend connections. Due to the symmetry of the friend connection, the edges of the graph are undirected, and the graph itself is undirected.

Then graph features were calculated for each Gv. The features (listed in Appendix A) describe quantitively the structure of the VKontakte user’s friend network.

There are two examples shown in Figure 1 and Figure 2

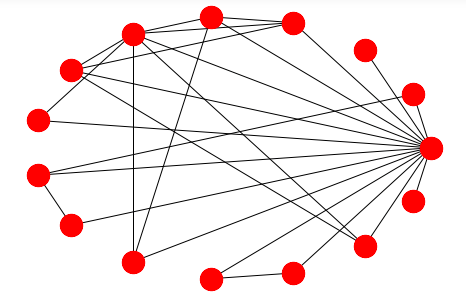


Figure 1 – Graph Gv for a real user

# 

# Изображение выглядит как красный Автоматически созданное описание

# Figure 2 –Graph Gv for a social bot

# Approach

The main hypothesis is that the nature of a VKontakte user can be determined by examining the friend network of the user.

The efficiency of using graph features is to be compared to the efficiency of profile features.

The algorithm of comparison is as follows:

1. Classification model is built
2. The importance of each feature is measured with the Permutation Importance algorithm
3. The place of graphical features in the importance ranking is recorded

3 methods were used to build classification models:

1. Logistic Regression
2. Random Forest
3. Gradient Boosting

Permutation Importance is the following algorythm:

1. A validation dataset is fixed
2. The quality of the model is calculated on the validation dataset
3. One feature is removed from the dataset forming a revised dataset
4. The quality of the model is calculated on the revised dataset
5. The difference between the qualities is calculated

Thus, Permutation Importance algorithm demonstrates that the feature is important if its removal leads to the significant drop of quality.

# Experiment

# Random Forest algorithm showed the best result (Table 1)

|  |  |
| --- | --- |
| **Method of modelling** | roc\_auc score |
| Logistic Regression | 0.876 |
| **Random Forest** | 0.981 |
| Gradient Boosting | 0.975 |

Table 1

Top 5 features according to Permutation Importance in Logistic Regression:

| **Weight** | **Feature** |
| --- | --- |
| 0.0701 ± 0.0053 | average\_neighbor\_degree |
| 0.0289 ± 0.0017 | closeness\_centrality |
| 0.0419 ± 0.0034 | subscriptions |
| 0.0178 ± 0.0013 | relation |
| 0.0166 ± 0.0012 | transitivity |

Top 5 features according to Permutation Importance in Random Forest:

| **Weight** | **Feature** |
| --- | --- |
| 0.2950 ± 0.0062 | average\_neighbor\_degree |
| 0.0747 ± 0.0025 | pages |
| 0.0419 ± 0.0034 | has\_photo |
| 0.0101 ± 0.0009 | subscriptions |
| 0.0088 ± 0.0003 | friends |

Top 5 features according to Permutation Importance in Gradient Boosting:

| **Weight** | **Feature** |
| --- | --- |
| 0.2849 ± 0.0055 | average\_neighbor\_degree |
| 0.1399 ± 0.0033 | has\_photo |
| 0.1147 ± 0.0012 | pages |
| 0.0871 ± 0.0039 | closeness\_centrality |
| 0.0158 ± 0.0012 | friends |

# 5 Conclusion

The results of the experiment show that the graph analysis (namely average\_neigbor\_degree feature) gives the best signal that allows to tell a social bot from a real user. This confirms the original hypothesis that social bots choose friends from random users, while real users choose friends from their social circle, as average\_neighbor\_degree feature indicates how many friends of the studied users are actually friends with each other.

Social bots can be detected with high accuracy with the help of the built models, as, unlike forgering the profile data, forgering the friends network is a considerably more complicated task.

# Appendix

The list of the features used in modelling and their description is given below.

Profile features:

1. has\_photo – information about whether the user has a profile photo. Return values: 1 — has, 0 — does not have.
2. sex - sex.Return values: 1 —female, 2 —male, 0 — not indicated.
3. has\_mobile – information whether user has mobile app. Return values: 1 — has, 0 —does not have.
4. followers\_count - the quantity of followers.
5. contacts – indication of phone number on profile. Return values: 1 —indicated, 0 — not indicated.
6. relatives - indication of the list of relatives on profile. Return values: 1 —indicated, 0 — not indicated.
7. relation – indication of whether a person is in relationship or not on profile. Return values: 1 —indicated, 0 — not indicated.
8. personal – indication of life position on profile. Return values: 1 —indicated, 0 — not indicated.
9. activities – indication of activities on profile. Return values: 1 —indicated, 0 — not indicated.
10. music – indication of favorite music on profile. Return values: 1 —indicated, 0 — not indicated.
11. movies – indication of favorite movies on profile. Return values: 1 —indicated, 0 — not indicated.
12. tv – indication of favorite tv shows on profile. Return values: 1 —indicated, 0 — not indicated.
13. books – indication of favorite books on profile. Return values: 1 —indicated, 0 — not indicated.
14. about – indication whether ‘about myself’ field on profile was filled. Return values: 1 — filled, 0 — not filled.
15. quotes – indication of favorite quotes on profile. Return values: 1 —indicated, 0 — not indicated.
16. albums – the quantity of albums.
17. audios – the quantity of audios.
18. followers – the quantity of followers.
19. friends – the quantity of friends.
20. pages – the quantity of pages user indicated as interesting.
21. photos – the quantity of photos.
22. subscriptions – the quantity of subscriptions.
23. videos – the quantity of videos.
24. age – age.
25. city – city id.
26. country – country id.

Graph features:

1. avg\_cl - the average clustering of the graph of the user's friends. It is the average of the sum of the results of division of all existing links between the neighbors of an edge and all possible links between them for all nodes in the social graph of the user.
2. trans - the transitivity of the graph of the user's friends. It is the relation of the tripled number of triangles to the number of triples of nodes in the social graph of the user.
3. deg\_centr - the average of the degree centrality, which is the relation of the degree of the node to the number of nodes in the graph minus 1, for every node in the social graph of the user.
4. average\_neighbor\_degree - the average of the average neighbor degree for every node in the social graph of the user.
5. average\_degree\_connectivity – the average of the average nearest neighbor degree, which is the relation of the sum of the degrees of the neighbors of a node to the degree of it, for all nodes in the social graph of the user.
6. k\_nearest\_neighbors - the average of the average neighbor degree for every node in the social graph of the user.
7. degree\_centrality – the average of the degree centrality, which is the relation of the degree of the node to the number of nodes in the graph minus 1, for every node in the social graph of the user.
8. closeness\_centrality – the average of the closeness centrality, which is the reciprocal of the sum of the length of the shortest paths between the node and all other nodes, for every node in the social graph of the user.
9. betweenness\_centrality – the average of the betweenness centrality, which is the sum of the relations of the number of shortest path length for all combinations of 2 nodes that pass through the given node v to the number of shortest path length for all combinations of 2 nodes, for every in the social graph of the user.
10. diameter - the longest path between any points in the social graph of the user.
11. average\_shortest\_path\_length – the average of the lengths of the shortest path lengths between all pairs of nodes in the graph.