

# **To Detect Fake Identities in Twitter using Machine Learning Models**

**at**

**Sathyabama Institute of Science and Technology  
(DEEMED TO BE UNIVERSITY)**

Submitted in partial fulfillment of the requirements for the award of Bachelor  
Engineering Degree in Computer Science and Engineering By

**PEDDADA CHETANROOP (38110107)**

**SURAPARAJU RUTHWIK PREETHAM (38110471)**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING  
SCHOOL OF COMPUTING**

# **SATHYABAMA**

**INSTITUTE OF SCIENCE AND TECHNOLOGY (DEEMED TO BE UNIVERSITY)**

**Accredited with Grade “A” by NAAC I 12B**

**Status by UGC I Approved by AICTE**

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI,**

**CHENNAI - 600 119**

**March – 2022**



# SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade “A” by NAAC

(Established under Section 3 of UGC Act, 1956)

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI– 600119

[www.sathyabama.ac.in](http://www.sathyabama.ac.in)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

## BONAFIDE CERTIFICATE

This is to certify that this Professional Training Report is the bonafide work of **Peddada Chetanroop (38110107)**, **Suraparaju Ruthwik Preetham (38110471)** who carried out the project entitled “**To Detect Fake Identities in Twitter using Machine Learning Models**” under our supervision from November 2021 to March 2022.

Internal Guide

**Dr S GOWRI**

Head of the Department

Submitted for Viva voce  
Examination held on

Internal Examiner

External Examiner

## DECLARATION

I, **Peddada Chetanroop (38110107)** and **Suraparaju Ruthwik Preetham (38110471)** hereby declare that the project entitled “**To Detect Fake Identities in Twitter using Machine Learning Models**” done by me under the guidance of **Dr S Gowri** at **Sathyabama Institute of Science and Technology** is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

**DATE:**

**PLACE: CHENNAI**

**SIGNATURE OF THE CANDIDATE**

## ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to Board of Management of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. SASIKALA, M.E., Ph.D., Dean**, School of Computing and **Dr. S. VIGNESHWARI, M.E., Ph.D., and Dr. Lakshmanan M.E., Ph.D.**, Heads of the Department, Dept. of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. S Gowri**, for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

## **ABSTRACT**

Social bots are considered to be the most popular type of spamming like spamming malware links, produce fake news, spread rumors to manipulate public opinion. Recently large-scale social bots have been created and are wide spread on social which have a bad impact on public and internet users' safety in all social media platforms. Bot detection aims to distinguish bots from humans to aid understanding the news or opinions. In recent times, classification of bots in social media have become more as they are populated everywhere. In this paper, we propose a decision tree classifier and a deep learning method to classify bots and humans in Twitter using Twitter API. This proposed model uses what an account has tweeted and cross reference against a bag of words model. These methods are unique that applies deep learning concepts to classification. Using real world data from twitter shows the validity of the model we proposed.

## **Table of Contents**

| <b>Chapter no.</b> | <b>TITLE</b>                     | <b>page no.</b> |
|--------------------|----------------------------------|-----------------|
|                    | <b>Abstract</b>                  | <b>5</b>        |
| <b>1</b>           | <b>Introduction</b>              | <b>6</b>        |
| <b>2</b>           | <b>Literature Review</b>         | <b>8</b>        |
|                    | 2.1 Social bot detection methods | <b>8</b>        |
| <b>3</b>           | <b>System Architecture</b>       | <b>12</b>       |
| <b>4</b>           | <b>4.1 Existing System</b>       | <b>13</b>       |
|                    | <b>4.2 Proposed System</b>       | <b>14</b>       |
| <b>5</b>           | <b>Conclusion</b>                | <b>15</b>       |
|                    | <b>Source Code</b>               | <b>16</b>       |

# Chapter 1

## 1. Introduction

Online social networks represent a global platform through which people share and promote products, links, opinions and news. By 2007, Twitter had almost 330 million active users and by 2015, the number of users had grown to 1.3 billion. The data sharing feature of social networks allows users to share links, images, videos, however this feature is commonly used by spammers and fraudsters. Social bots are programs that automatically generate content, share it via a particular social network, and interact with the users. A study found that social bots in Twitter are 15% of all accounts which is equivalent to 48 million accounts, generating 35% of content that is posted on Twitter.

Studies aimed to address the problem associated the use of automated accounts on social networks, which can spread spam, worms, and phishing links or manipulate legitimate accounts by hijacking and deceiving users. Malicious accounts under bot-master, who controls a group of social bots or distribute spam or manipulate behaviors on a given social network. Social bots have also played a significant role in the uprisings that occur in the aftermath of major events such as elections or conflicts. The malicious activities of bots during events can be used to spread spam. In addition, they can also cause financial harm.

The activities of social bots also impacted the social graph of Online Social Networks (OSN's) because the large number of non-genuine social relationships. If social bots successfully infiltrate users accounts, they can harvest social bot private data and subsequently use It for phishing and spamming activities. In addition, they can aggregate information from the web to impersonate others, replicate human behaviors, and influence people by ranking and retweeting. In addition to essential misleading users, social bots can damage the ecosystem of the social network by establishing fake relationships and poisoning the network content.

In attempt to limit the threats posed by social bots, researches have proposed different methods by which social bots can be detected and blocked. The majority of the studies in this domain to date have focused on studying behavior patterns. For example, a recent study was performed, proposed a dynamic metric to measure the change in user's activities as a means of identifying the strategies employed by spammers. It is important to note that not all social botcan be classified as malicious accounts. Some even explicitly state their nature in the profile of the account. Social bots that operate without malicious intent may serve positive purposes, such as managing news feeds or acting as customer care responders. The problem we are concerned with in this paper is undisclosed social bots that have malicious intentions. As outlined above, these social bots can pose fundamental financial, social, political, and security risks. They have become increasingly sophisticated in their designs and capabilities to avoid social bot detection techniques. There is a requirement to gain in-depth insights into the capabilities and limitations

of the social bot detection techniques that are currently in use on the Twitter social network platform. By comparing and evaluating the existing approaches, we can develop an understanding of the different solutions that are available based on the selected features and trained classifiers, and can subsequently apply this understanding to identify which technologies achieve the best accuracy and detection results

## Chapter 2

### 2. Literature Review

In this section, we focus on studying the detection techniques that are commonly employed to detect social bots or fake accounts on the Twitter social network platform.

#### 2.1 Social bot detection methods:

Generally, social bot detection on social networks is performed by one or more of the three common methods mentioned earlier:

- Graph-based,
- Crowdsourcing, and
- Machine Learning.

The graph-based method involves using the social graph of a social network to understand the network information and the relationships between edges or links across accounts to detect bot activity. The crowdsourcing method involves using expert annotators to identify, evaluate, and determine social bot behaviors. Finally, the machine learning method involves developing algorithms and statistical methods that can develop an understanding of the revealing features or behavior of social network accounts in order to distinguish between human- and computer-led activity. In this section, we provide an overview of the three methods that have been used by researchers to detect social bot accounts on Twitter. In each subsection, we discuss the related studies and the datasets, detection mechanisms, classifiers involved, and the process by which the results were validated of predetermined metrics. They presented a model of the strength of Sybil attacks based on the  $\alpha$  parameter, which represented the ratio of total weight of the edge in the non-Sybil region against that of the Sybil region. They assumed the worst-case scenario of Sybil attacks in which there was no interaction between the two regions

### A. Graph Base detection

Social network graphs are commonly employed to understand and distinguish between users' relationships on social networking platforms. Three social graph-based methods are typically employed to detect social bots and malicious accounts [1].

The first method is based on trust propagation, which evaluates whether the trust relationship that exists between two graph objects is strong or weak. The second method is graph clustering, by which related nodes of a social graph are grouped based on similar characteristics such as users' distance. The third method involves studying graph metrics and properties, where probability distribution, scale-free graph structure, and centrality metric measures are addressed in a social graph. In this subsection, we present three graph-based Sybil detection systems that were evaluated using datasets from Twitter. These three systems were chosen because they involved the analysis of a Twitter dataset. They are presented in chronological order, starting with the most recent work.

SybilWalk [2] is a proposed Sybil detection method that employs a random walk-based method on an undirected social graph. The idea of the random walk method is to label legitimate users with benignness scores and Sybil users with badness scores. Therefore, these scores will help to classify users into two classes: legitimate and Sybil. In addition, this method can rank all users as a means of identifying top-ranked accounts that are likely to be Sybils. The authors of SybilWalk assumed that the graph satisfies the homophily property, for which two linked nodes tend to share the same label. They labeled the legitimate node badness score with 0 and the Sybil node with 1 and employed a directed Twitter graph dataset that was obtained from a previous study [3]. To evaluate their experiment, the authors used the Area Under the Receiver Operating Characteristic Curve (AUC) as a standard metric to measure the quality of their ranking method, which they awarded with a score of 0.96. They also presented the classification results of Sybil and legitimate nodes in the form of the false positive rate (FPR) and false negative rate (FNR) of 1.3% and 17.3% respectively.

Mehrotra et al. proposed a method to detect fake followers using social graph-based features that relate to the centrality of all nodes in the graph [4]. They claimed that their proposed method can be applied to all social networking platforms. They employed five datasets, two of which were of the legitimate follower type and the remaining three of fake followers. They used six features of centralities of graph-based centrality measurements for the purposes of the classification. After computing the centrality measures of all the given nodes in the graph, they applied three classifiers: Artificial Neural Networks, Decision Tree, and Random Forest. The random forest classifiers scored the highest accuracy of 95%, with the precision of 88.99%, and recall of 100%.

TrueTop [5] is another influence measurement system that employed a graph-based approach to test Sybil resilience. The authors employed a synthetic simulation of users on Twitter to avoid violating the platform's terms of service. They employed four datasets to implement the system, evaluate the accuracy of the model, and test Sybil resilience against a set

## B. Crowdsourcing

As previously described, the crowdsourcing approach to social bot detection involves leveraging human detection to identify patterns across given account profiles or the content shared by human and social bot accounts. The role of the human is to distinguish between bot accounts and human accounts. For example, DARPA held a Twitter bot challenge competition in March 2015 [6]. The competition involved identifying influential bots that supported pro-vaccination discussions on Twitter to serve as ground truth. Teams were asked to submit their guesses to a web server, which calculated and presented the results in real-time. All teams used human judgment to identify bots after using their own implementation of bot detection techniques based on the features they had chosen to inform their guesses. Three teams out of the six scored the highest in terms of their ability to detect the bot accounts.

Four annotators were tasked with classifying and labeling Twitter profiles into two categories, bot or human, and providing a justification for their choice [7]. They were provided with a list of attributes or features to consider, such as creation date of account, number of tweets, number of favorited tweets, etc. To ensure the reliability of the annotation experiments, the researchers employed Cohen's kappa ( $\kappa$ ) coefficient and average pairwise inter-annotator agreement across all dataset bands. A study by Alarifi et al. involved 10 volunteers rating and labeling 2000 random accounts to build a ground truth dataset. They evaluated the accuracy and reliability of their labeling process by injecting their ground truth dataset with a subset of 1020 Twitter bot accounts from another ground truth reference. They achieved an accuracy of 96% with 4% error rate during the labeling process.

## C. Machine Learning

A framework for bot detection on Twitter was proposed by Kantepe and Ganiz, who applied machine learning algorithms after an extensive process of data preprocessing and feature extraction [8]. They employed the Twitter API and Apache Spark to collect the data. They collected 1,800 Twitter accounts and extracted 62 features that were categorized into three types: User Features, Tweet Features, and Periodic Features. They used percentages of 60- 40% and 70-30% and 90-10% for the training set and test set. They used four classifiers: logistic regression, multinomial naive-byes, support vector machine, and gradient-boosted trees. The highest accuracy result was 86% for gradient-boosted trees, and the F1 score was 83%.

BeDM [9] is a proposed behavior-enhanced deep learning model for bot detection in social networks. It implements convolutional neural network (CNN) layers with LSTM long-short-term memory and hidden layers to capture the latent temporal patterns of users' tweet history and

behavior. They used a public dataset in their experiment and an additional collection of 1000 tweets using the Twitter API. They employed tenfold cross-validation with measuring precision, recall, and F1 scores to evaluate their model and compare it against similar baselines. The BeDM performance achieved the highest in the F1 score at 87.32% in comparison to alternative baselines. The recall score was 86.26%, and the precision was 88.41%.

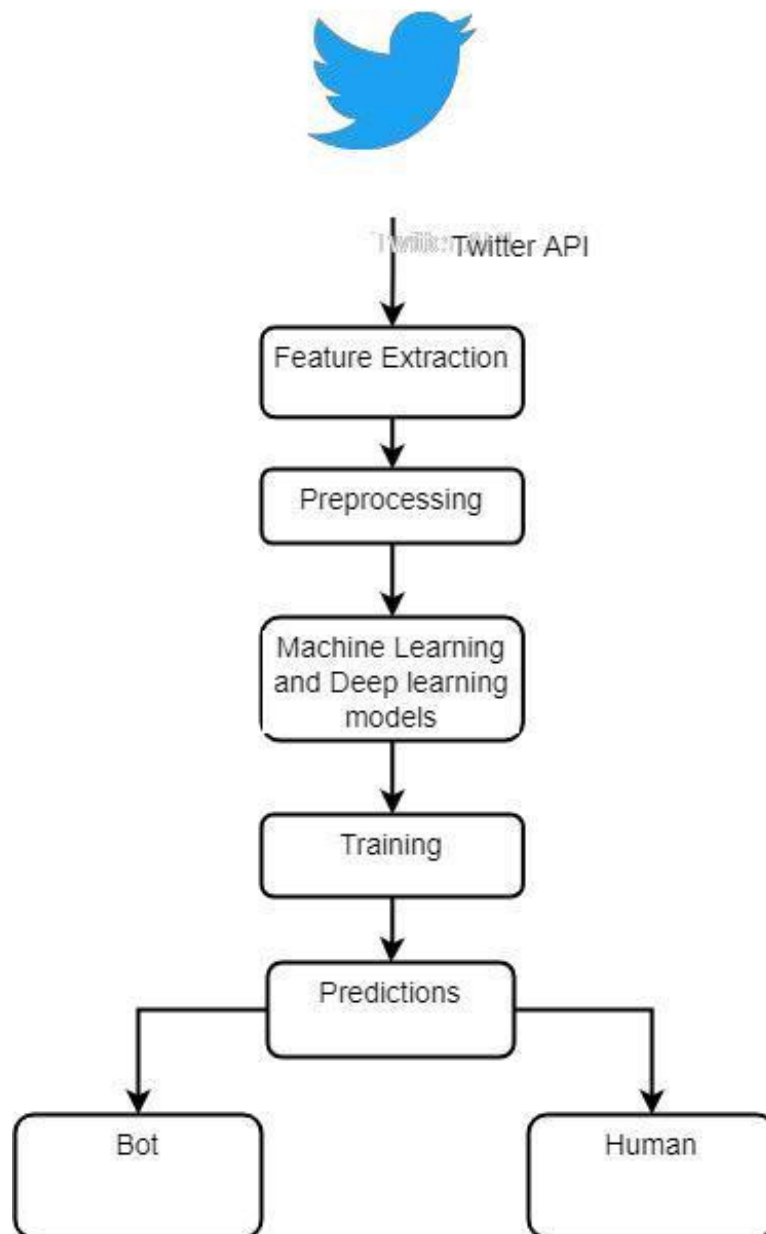
Gilani et al. classified Twitter accounts into automated agents and human users [7]. They used their own platform, Stweeler, to collect their data. They collected 2.5 and 3 million tweets a day and partitioned their dataset into four subsets: 10M, 1M, 100 K, and 1K, each of which represented the popularity of the account based on the number of followers. They used human annotation for the labeling process and Cohen's kappa coefficient to maintain the reliability of the annotator judgments. The total number of accounts included in the testing phase was 3,536 across the four bands. After performing a statistical calculation, the authors extracted 15 features and employed the random forest classifier. They performed three sets of experiments through which they ran fivefold cross-validation by training and testing. The accuracy rate was 86.44%, precision was 85.4%, recall was 82.2%, and F-measure was 83%. They found 6 features scored the highest among the 15 features.

Chu et al. studied the features related to tweeting behavior, tweet content, and account properties [10] to detect the automation of bots. They categorized accounts into human, bot, and cyborg according to the investigated features. Their classification system incorporated an entropy-based component to detect regularity of timing to measure automation, a spam detection component in the form of a Bayesian classification that detected text patterns as a means of detecting spam, account properties, and a random forest classifier as a decision maker. They collected data covering 512,407 accounts using the Twitter API. They constructed their ground truth sample by using 6000 accounts divided equally per human, bot, and cyborg. They extracted 8 features and implemented a random forest classifier with tenfold cross validation. They employed a confusion matrix to measure the system performance and achieved an average score of 96%.

Alarifi et al. analyzed the detection features of bot accounts. They collected data that consisted of 1.8 million accounts and then randomly selected 2000 accounts for the sample after manually labeling them into human, bot, and hybrid accounts [11]. They employed two feature methods to extract effective features and used principal component analysis and correlation-based methods. They selected eight features to evaluate their models by applying four machine learning algorithms: decision tree, Bayesian network, support vector machine, and multilayer artificial neural network. For the purpose of performance measurements, they used six indicators: detection rate, error rate, TP/FP, precision, recall, and F-measure. Both random forest and Bayes net classifiers performed better with 88% and 86.74% in sequence

## Chapter 3

Flowchart to classify bots and human accounts from Twitter.



## Chapter 4

### 4.1 Existing System

The survey on existing system shows that the three existing methods to classify bots as discussed above are Graph Approach, crowdsourcing and Machine Learning algorithms. Machine learning methods are optimal when compared to the rest of the methods. Most of the papers use tree-based approach, Bayes theorem and random classifier. Random Classifier is most accurate and commonly used as it is very accurate in achieving more accuracy. Although it is prone to overfitting, Bayes theorem and Random forest classifiers is best with low number of features. SVM depends on selective kernel, as it reduces the error rate in classification process. Likewise, Neural networks results depend on the data provided; when the samples are large, the model performs well.

| Method           |                                       | Advantages                                                                                                                     | Disadvantages                                                                                                                                                         |
|------------------|---------------------------------------|--------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Graph Based      | Trust Propagation                     | <ul style="list-style-type: none"> <li>- Good and easy to rank users</li> <li>- Good to represent two trust classes</li> </ul> | <ul style="list-style-type: none"> <li>- Depends on assumptions</li> <li>- Sensitive to collection</li> </ul>                                                         |
|                  | Centrality and Properties             | <ul style="list-style-type: none"> <li>- Good to measure influence nodes</li> <li>- Graph visualization</li> </ul>             | <ul style="list-style-type: none"> <li>- Large network will be computationally cost</li> <li>- Depends on assumptions</li> <li>- Sensitive to neighborhood</li> </ul> |
| Crowdsourcing    | Annotation task to build ground truth | <ul style="list-style-type: none"> <li>- Using human intelligence</li> <li>- Good accuracy</li> </ul>                          | <ul style="list-style-type: none"> <li>- Paid task usually</li> <li>- Time consuming</li> </ul>                                                                       |
| Machine Learning | Tree- Based                           | <ul style="list-style-type: none"> <li>- Performance accuracy</li> <li>- Widely used</li> </ul>                                | <ul style="list-style-type: none"> <li>- Overfitting problem</li> </ul>                                                                                               |
|                  | Bayes Theorem                         | <ul style="list-style-type: none"> <li>- Easy to implement</li> <li>- Good performance</li> </ul>                              | <ul style="list-style-type: none"> <li>- Less performance in case of large features</li> <li>- Learning time</li> </ul>                                               |
|                  | Support Vector Machine                | <ul style="list-style-type: none"> <li>- Good accuracy based on kernel choice</li> <li>- Solves regression problem</li> </ul>  | <ul style="list-style-type: none"> <li>- Training and testing is affected by data size</li> <li>- Sensitive to kernel choice</li> </ul>                               |
|                  | Neural Network                        | <ul style="list-style-type: none"> <li>- Performance accuracy</li> </ul>                                                       | <ul style="list-style-type: none"> <li>- Training time is slow</li> <li>- Need large samples to perform well</li> </ul>                                               |



## 4.2 Proposed System

In this section, we propose a machine learning models with data collected using twitter API. The algorithms used are decision tree classifier, neural network. Model with neural network is found to be more accurate with the training accuracy of 90% and testing accuracy of 96%.

### Neural Networks:

#### A. Dataset

The dataset is taken using honeypot method from Twitter API. The dataset contains accounts that are labelled bot or not including some features.

#### B. Metrics

Evaluation criteria are precision, recall and F1 score which are commonly used in bot detection. For each split, trained models with 80% data, cross-validation with 10% and the remaining is used for testing.

#### C. Performance

In the experiments, we conducted a series of tests with different parameter and layers with activation functions. The experiment results are shown below:

| Number of hidden layers* | Output layer activation function | Precision | Recall | F1           |
|--------------------------|----------------------------------|-----------|--------|--------------|
| 1                        | Sigmoid                          | 74.06     | 75.1   | 73.2         |
| 2                        | Sigmoid                          | 86.21     | 84.63  | 84.61        |
| 2                        | Softmax                          | 68.76     | 67.33  | 67.80        |
| 3                        | Sigmoid                          | 88.12     | 86.90  | <b>86.54</b> |

\*activation function in hidden layers is relu

### Decision Tree Classifier:

The dataset and evaluation metrics are same as previous model (neural network). A tree can be categorized into two variables: direction and fiendishness. Choices are made based on data sharing.

Performance:

The Area under the curve for training is 95.6 and for testing is 0.93 when trained with decision tree classifier. The metrics are given below:

| Training accuracy | Testing accuracy | Precision | Recall | F1     |
|-------------------|------------------|-----------|--------|--------|
| 88.24             | 87.85            | 91.11     | 83.69  | 97.725 |

## Chapter 5

### Conclusion:

In this documentation we proposed two methods to classify accounts in social media accounts by bots or human. The results show the performance and effectiveness with high f1 scores. In future work, we can plan to study other social media platforms with behavior modelling for bot classification.

### References

- [1] K. S. Adewole, N. B. Anuar, A. Kamsin, K. D. Varathan and S. A. Razak, "Malicious accounts: dark of the social networks," *Journal of Network and Computer Applications*, vol. 79, pp. 41-67, 2017.
- [2] J. Jia, B. Wang and N. Z. Gong, "Random walk based fake account detection in online social network," *Dependable Systems and Networks (DSN)*, pp. 273-284, 2017.
- [3] H. Kwak, C. Lee, H. Park and S. Moos, "What is Twitter, a social network or a news media," *Proceedings of the 19th international conference on World wide web*, pp. 591-600, 2010.
- [4] A. Mehrotra, M. Sarreddy and S. Singh, "'Detection of fake twitter followers using graph," *Contemporary Computing and Informatics (IC3I)*, pp. 499-504, 2016.
- [5] "Truetop: A sybilresilient system for user," *IEEE/ACM Transactions on Networking*, vol. 24, pp. 2834-2846, 2016.
- [6] V. Subrahmanian, A. Azaria, S. Durst, V. Kagan, A. Galstyan and K. Leman, "The darpa twitter bot challenge," *Computer*, vol. 49, pp. 38-46, 2016.
- [7] Z. Gilani, E. Kochmar and J. Crowcroft, "Classification of twitter accounts into automated".
- [8] M. Kantepe and M. C. Ganiz, "Preprocessing framework for twitter bot detection," *Computer Science and Engineering (UBMK), 2017 International Conference on. IEEE*, pp. 630-634, 2017.
- [9] C. Cai, L. Li and D. Zengi, "Behavior enhanced deep bot detection in social media," *Intelligence and Security Informatics (ISI), 2017 IEEE International Conference on. IEEE*, pp. 128-130, 2017.
- [10] Z. Chu, S. Gianvecchio, H. Wang and S. Jajodia, "'Detecting automation of twitter accounts: Are you a human, bot, or cyborg?," *IEEE Transactions on Dependable and Secure Computing*, vol. 9, pp. 811-824, 2012.
- [11] A. Alarifi, M. Alsaleh and A. Al-Salman, "Twitter turing test: Identifying social machines," vol. 372, pp. 332-346, 2016.

## Source Code

### Preprocessing:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
%matplotlib inline
file = 'training_data.csv'

training_dataset = pd.read_csv(file)

def heatmap(dataset):
    plt.figure(figsize = (10, 6))
    sns.heatmap(dataset.isnull(), yticklabels=False, cbar = False, cmap = 'viridis')
    plt.tight_layout()
    plt.show()

#missing columns represent the missing data.
print(heatmap(training_dataset))

#seperating bots and non-bots from dataset
bots = training_dataset[training_dataset.bot==1]
nonbots =
training_dataset[training_dataset.bot==0]
print(bots.shape)

#friends vs followers for bots
```

```
plt.figure(figsize=(10,5))
```

```

plt.subplot(2,1,1)
plt.title('Bots friends vs Followers')
sns.regplot(bots.friends_count, bots.followers_count, color='red', label='Bots')
plt.xlim(0, 100)
plt.ylim(0, 100)
plt.tight_layout()

#friends vs followers for humans
plt.subplot(2,1,2)
plt.title('NonBots Friends vs Followers')
sns.regplot(nonbots.friends_count, nonbots.followers_count, color='blue', label='NonBots')
plt.xlim(0, 100)
plt.ylim(0, 100)
plt.tight_layout()
plt.show()

# imbalance in data
bots['friends_by_followers'] = bots.friends_count/bots.followers_count
print(bots[bots.friends_by_followers<1].shape)

nonbots['friends_by_followers'] = nonbots.friends_count/nonbots.followers_count
nonbots[nonbots.friends_by_followers<1].shape plt.figure(figsize=(10,5))
plt.plot(bots.listed_count, color='red', label='Bots')
plt.plot(nonbots.listed_count, color='blue', label='NonBots')
plt.legend(loc='upper left')
plt.ylim(10000,20000)
print(bots[(bots.listed_count<5)].shape)

```

```

bots_listed_count_df = bots[bots.listed_count<16000]
nonbots_listed_count_df = nonbots[nonbots.listed_count<16000]

bots_verified_df = bots_listed_count_df[bots_listed_count_df.verified==False]
bots_scrennname_has_bot_df =
bots_verified_df[(bots_verified_df.screen_name.str.contains("bot", case=False)==True)].shape

plt.figure(figsize=(12,7))

plt.subplot(2,1,1)
plt.plot(bots_listed_count_df.friends_count, color='red', label='Bots Friends')
plt.plot(nonbots_listed_count_df.friends_count, color='blue', label='NonBots Friends')
plt.legend(loc='upper left')

plt.subplot(2,1,2)
plt.plot(bots_listed_count_df.followers_count, color='red', label='Bots Followers')
plt.plot(nonbots_listed_count_df.followers_count, color='blue', label='NonBots Followers')
plt.legend(loc='upper left')

#bots[bots.listedcount>10000]

condition = (bots.screen_name.str.contains("bot",
case=False)==True)|(bots.description.str.contains("bot",
case=False)==True)|(bots.location.isnull())|(bots.verified==False)

bots['screen_name_binary'] = (bots.screen_name.str.contains("bot", case=False)==True)
bots['location_binary'] = (bots.location.isnull())
bots['verified_binary'] = (bots.verified==False)
bots.shape

```

```

condition = (nonbots.screen_name.str.contains("bot", case=False)==False)|
(nonbots.description.str.contains("bot", case=False)==False)
|(nonbots.location.isnull()==False)|(nonbots.verified==True)
nonbots['screen_name_binary'] = (nonbots.screen_name.str.contains("bot", case=False)==False)
nonbots['location_binary'] = (nonbots.location.isnull()==False)
nonbots['verified_binary'] = (nonbots.verified==True)
nonbots.shape
# df.corr(method='spearman')
plt.figure(figsize=(8,4))
sns.heatmap(df.corr(method='spearman'), cmap='coolwarm', annot=True)
plt.tight_layout()
plt.show()

```

## Neural Networks:

```

file= open('kaggle_data/training_data.csv', mode='r', encoding='utf-8', errors='ignore')

training_data = pd.read_csv(file)

bag_of_words_bot = r'bot|b0t|cannabis|tweet me|mishear|follow
me|updates every|gorilla|yes_ofc|forget' \

r'expos|kill|clit|bbb|butt|fuck|XXX|sex|truthe|fake|anony|free|virus|funky|RNA|kuck|jargon' \

r'nerd|swag|jack|bang|bonsai|chick|prison|paper|pokem|xx|freak|ffd|dunia|clone|genie|bbb'\

r'ffd|onlyman|emoji|joke|troll|droop|free|every|wow|cheese|yeah|bio|magic|wizard|face'

training_data['screen_name_binary'] =
training_data.screen_name.str.contains(bag_of_words_bot, case=False, na=False)

```

```
training_data['name_binary'] = training_data.name.str.contains(bag_of_words_bot, case=False,
na=False)
training_data['description_binary'] = training_data.description.str.contains(bag_of_words_bot,
case=False, na=False)
training_data['status_binary'] = training_data.status.str.contains(bag_of_words_bot, case=False,
na=False)
```

```
training_data['listed_count_binary'] = (training_data.listed_count>20000)==False
features = ['screen_name_binary', 'name_binary', 'description_binary', 'status_binary',
'verified', 'followers_count', 'friends_count', 'statuses_count', 'listed_count_binary', 'bot']
```

```
#variables
```

```
test_size = 0.2
```

```
validation_size = 0.5
```

```
#splitting into feature and labels
```

```
x = training_data[features].iloc[:, :-1]
```

```
y = training_data[features].iloc[:, -1]
```

```
x = np.asarray(x).astype(np.float32)
```

```
#splitting dataset into training and and remaining from
```

```
sklearn.model_selection import train_test_split
```

```
x_train, x_remaining, y_train, y_remaining = train_test_split(x, y, test_size= test_size,
random_state= 10)
```

```
#splitting into validation and test datasets
```

```
x_valid, x_test, y_valid, y_test = train_test_split(x_remaining, y_remaining, test_size =
validation_size, random_state = 10)
```

```

import tensorflow as tf

ann = tf.keras.models.Sequential()
ann.add(tf.keras.layers.Dense(units = 10, activation = 'relu'))
ann.add(tf.keras.layers.Dense(units = 5, activation = 'relu'))
ann.add(tf.keras.layers.Dense(units = 1, activation = 'sigmoid'))

ann.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
hist = ann.fit(x_train, y_train, validation_data = (x_valid, y_valid), batch_size = 32, epochs
= 300)

y_pred = ann.predict(x_test)
threshold = 0.5
y_pred = pd.DataFrame(y_pred, columns=["y_pred"])
y_pred = np.where(y_pred > threshold, 1, 0)
from sklearn.metrics import accuracy_score, confusion_matrix

print(confusion_matrix(y_test, y_pred))
accuracy_score(y_test, y_pred)

```

## Desicion Trees:

```

file= open('kaggle_data/training_data.csv', mode='r', encoding='utf-8', errors='ignore')

training_data = pd.read_csv(file)

bag_of_words_bot = r'bot|b0t|cannabis|tweet me|mishear|follow
me|updates every|gorilla|yes_ofc|forget' \

r'expos|kill|clit|bbb|butt|fuck|XXX|sex|truthe|fake|anony|free|virus|funky|RNA|kuck|jargon' \

```

```
r'nerd|swag|jack|bang|bonsai|chick|prison|paper|pokem|xx|freak|ffd|dunia|clone|genie|bbb\
```

```
r'ffd|onlyman|emoji|joke|troll|droop|free|every|wow|cheese|yeah|bio|magic|wizard|face'
```

```
training_data['screen_name_binary'] =  
training_data.screen_name.str.contains(bag_of_words_bot, case=False, na=False)  
  
training_data['name_binary'] = training_data.name.str.contains(bag_of_words_bot, case=False,  
na=False)  
  
training_data['description_binary'] = training_data.description.str.contains(bag_of_words_bot,  
case=False, na=False)  
  
training_data['status_binary'] = training_data.status.str.contains(bag_of_words_bot, case=False,  
na=False)
```

```
training_data['listed_count_binary'] = (training_data.listed_count>20000)==False  
  
features = ['screen_name_binary', 'name_binary', 'description_binary', 'status_binary',  
'verified', 'followers_count', 'friends_count', 'statuses_count', 'listed_count_binary', 'bot']
```

```
#variables
```

```
test_size = 0.2
```

```
validation_size = 0.5
```

```
#splitting into feature and labels
```

```
x = training_data[features].iloc[:, :-1]
```

```
y = training_data[features].iloc[:, -1]
```

```
x = np.asarray(x).astype(np.float32)
```

```
#splitting dataset into training and and remaining from
```

```
sklearn.model_selection import train_test_split
```

```
x_train, x_remaining, y_train, y_remaining = train_test_split(x, y, test_size= test_size,  
random_state= 10)
```

```
#splitting into validation and test datasets
```

```
x_valid, x_test, y_valid, y_test = train_test_split(x_remaining, y_remaining, test_size =  
validation_size, random_state = 10)
```

```
from sklearn.tree import DecisionTreeClassifier
```

```
from sklearn.metrics import accuracy_score, roc_curve, auc
```

```
from sklearn.model_selection import train_test_split
```

```
X      =
```

```
training_data[features].iloc[:, :-1] y =
```

```
training_data[features].iloc[:, :-1]
```

```
dt = DecisionTreeClassifier(criterion='entropy', min_samples_leaf=50, min_samples_split=10)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=101)
```

```
dt = dt.fit(X_train, y_train)
```

```
y_pred_train = dt.predict(X_train)
```

```
y_pred_test = dt.predict(X_test)
```

```
print("Trainig Accuracy: %.5f" %accuracy_score(y_train, y_pred_train))
```

```
print("Test Accuracy: %.5f" %accuracy_score(y_test, y_pred_test))
```

## **Demo:**

### **Class:**

```
from sklearn.tree import DecisionTreeClassifier
```

```
from sklearn.metrics import accuracy_score, roc_curve, auc
```

```

from sklearn.model_selection import train_test_split
import seaborn as sns
import matplotlib.pyplot as plt

class TwitterBotClassifier:
    def __init__(self, training_dataset, demo_dataset) -> None:

        self.training_dataset = training_dataset
        self.demo_dataset = demo_dataset

        self.test_size = 0.3 #used for splitting dataset into training and testing

        self.bag_of_words_bot = r'bot|b0t|cannabis|tweet me|mishear|follow me|updates
every|gorilla|yes_ofc|forget' \

        r'expos|kill|clit|bbb|butt|fuck|XXX|sex|truthe|fake|anony|free|virus|funky|RNA|kuck|jargon
'\

        r'nerd|swag|jack|bang|bonsai|chick|prison|paper|pokem|xx|freak|ffd|dunia|clone|genie|bbb'\

        r'ffd|onlyman|emoji|joke|troll|droop|free|every|wow|cheese|yeah|bio|magic|wizard|face'

    def create_features(self, dataset):

        dataset['screen_name_binary'] =
dataset.screen_name.str.contains(self.bag_of_words_bot, case=False, na=False)

        dataset['name_binary'] = dataset.name.str.contains(self.bag_of_words_bot,
case=False, na=False)

        dataset['description_binary']
=dataset.description.str.contains(self.bag_of_words_bot, case=False, na=False)

        dataset['status_binary'] = dataset.status.str.contains(self.bag_of_words_bot,
case=False, na=False)

```

```

dataset['listed_count_binary'] = (dataset.listed_count>20000)==False

return dataset

def initialisation(self) -> None:
    self.training_dataset = self.create_features(self.training_dataset)

    self.features = ['screen_name_binary', 'name_binary',
'description_binary', 'status_binary', 'verified', 'followers_count',
                    'friends_count', 'statuses_count', 'listed_count_binary', 'bot']

    self.to_train = self.training_dataset[self.features]

def split_dataset(self) -> None:
    x = self.to_train.iloc[:, :-1]
    y = self.to_train.iloc[:, -1]

    self.xtrain, self.xtest, self.ytrain, self.ytest = train_test_split(x, y, test_size =
self.test_size, random_state = 101)

def train_with_training_dataset(self) -> None:
    decision_tree_classifier = DecisionTreeClassifier(criterion='entropy',
min_samples_leaf=50, min_samples_split=10)
    self.dt = decision_tree_classifier.fit(self.xtrain, self.ytrain)
    self.y_pred_train = self.dt.predict(self.xtrain)
    self.y_pred_test = self.dt.predict(self.xtest)

def print_accuracy(self) -> None:
    print(f'Training Accuracy : {accuracy_score(self.ytrain, self.y_pred_train)}')

```

```

print(f'Testing Accuracy : {accuracy_score(self.ytest, self.y_pred_test)}')

def plot_roc(self):
    sns.set(font_scale=1.5)
    sns.set_style("whitegrid", {'axes.grid' : False})

    scores_train = self.dt.predict_proba(self.xtrain)
    scores_test = self.dt.predict_proba(self.xtest)

    y_scores_train = []
    y_scores_test = []
    for i in range(len(scores_train)):
        y_scores_train.append(scores_train[i][1])

    for i in range(len(scores_test)):
        y_scores_test.append(scores_test[i][1])

    fpr_dt_train, tpr_dt_train, _ = roc_curve(self.ytrain, y_scores_train, pos_label=1)
    fpr_dt_test, tpr_dt_test, _ = roc_curve(self.ytest, y_scores_test, pos_label=1)

    plt.plot(fpr_dt_train, tpr_dt_train, color='darkblue', label='Train
AUC: %5f %auc(fpr_dt_train, tpr_dt_train))
    plt.plot(fpr_dt_test, tpr_dt_test, color='red', ls='--', label='Test
AUC: %5f %auc(fpr_dt_test, tpr_dt_test))
    plt.title("Decision Tree ROC Curve")
    plt.xlabel("False Positive Rate (FPR)")
    plt.ylabel("True Positive Rate (TPR)")
    plt.legend(loc='lower right')
    plt.show()

```

```

def predict(self):
    self.demo_dataset = self.create_features(self.demo_dataset)

    self.to_predict = self.demo_dataset[self.features[:-1]]

    predictions = self.dt.predict(self.to_predict)

    return predictions

```

### **Main:**

```

training_dataset = pd.read_csv('kaggle_data/training_data.csv')
demo = pd.read_csv('kaggle_data/v1.csv')
from TwitterBotClassification import TwitterBotClassifier
classifier = TwitterBotClassifier(training_dataset, demo)
classifier.initialisation()
classifier.split_dataset()
classifier.train_with_training_dataset()
# classifier.print_accuracy() classifier.plot_roc()

demo['predictions'] = classifier.predict()

print(demo[['id', 'screen_name', 'name', 'predictions']])

```

## Plagarism Report:

# KL34

*by* Green Official

---

**Submission date:** 24-Mar-2022 12:17AM (UTC-0700)

**Submission ID:** 1791632623

**File name:** Paper\_on\_Twitter\_Bots\_vs\_Humans01.pdf (684.5K)

**Word count:** 1706

**Character count:** 9309

# To Detect Fake Identities in Twitter using Machine Learning Models.

Pedada Chetanroop, Ruthwik Preetham

<sup>3</sup> [REDACTED], Computer Science, [REDACTED]

Sathyabama Institute of Science and Technology, Computer Science, India

[p.chetanroop0509@gmail.com](mailto:p.chetanroop0509@gmail.com), [rutwikritu@gmail.com](mailto:rutwikritu@gmail.com)

**Abstract:** Social bots are considered to be the most popular type of spamming like spamming malware links, produce fake news, spread rumours to manipulate public opinion. Recently large-scale social bots have been created and are wide spread on social which have a bad impact on public and internet users' safety in all social media platforms. Bot detection aims to distinguish bots from humans to aid understanding the news or opinions. In recent times, classification of bots in social media have become more as they are populated everywhere. In this paper, we propose a decision tree classifier and a deep learning method to classify bots and humans in Twitter using Twitter API. This proposed model uses what an account has tweeted and cross reference against a bag of words model. These methods are unique that applies deep learning concepts to classification. Using real world data from twitter shows the validity of the model we proposed.

**Keywords:** Social Media, Twitter, decision trees, Neural networks, accuracy metrics.

## 1. Introduction

Bots are automated software that scrape or spam, interact with human on a social media platform. Users often tend to use social bots to manipulate opinions, rapid spam of rumours and also rate a product with fake reviews. According to twitter, over 15% accounts are bots. As the social media evolve, the issues caused by bots are obvious.

Bots in twitter control the account via twitter API, to automate actions like follow users, unfollow, tweet, retweet, and reply to direct messages. Few bots in this media are bots that help the user to schedule a tweet, help the assist by providing valuable insights over API. Such bots are termed as hybrid bots. For these specific reasons researchers are in design of advanced methods to classify social media bots, hence bot detection is a very valuable research problem.

The challenges <sup>5</sup> bot detection have been identified by many teams in the past. Common methods used in classifying are: [REDACTED]

[REDACTED] uses social graphs of social networks to get the network information and nodes for multiple accounts to classify bot activities. In crowdsourcing [1], an expert annotator identifies, evaluates and classifies accounts. The machine learning involves training large datasets with multiple algorithms and statistical methods to distinguish human or computer led activities across social media platforms.

## 2. Objectives

- Survey current strategies on Machine learning and deep learning to classify bots.
- Classify twitter accounts from human and bots
- The authenticity of people constantly questioning if it is a fake or not, this classifier authenticates and solves the problem.

## 3. Methodology

There are papers focusing on different techniques and methods that are targeted to classify social media accounts on twitter media platform.

The methods used are Graph Based Approach, Crowdsourcing, Machine Learning Algorithms [1].

Graph based approach uses nodes and samples of benign nodes for classification. Three social media based methods are employed namely trust propagation, graph clusters, and graph metrics. Crowdsourcing approach is a detection method that involves an annotator to find relations between accounts posted by bot or human. <sup>6</sup> [REDACTED] annotator [REDACTED] here [REDACTED] differences in bots and human accounts. This method involves a lot of guesses from the annotators. Machine Learning techniques uses a large amount of datasets that have many features. Using these models, patterns on feature of twitter accounts can be found and the probability that those accounts being humans or bots can be calculated. [1]

## 4. Literature Survey

### 4.1 Graph Based Detection

The authors [2] proposed three methods to detect accounts. The methods include confidence spread to determine if the connection between two attributes are strong. Next process includes grouping the attributes with similar properties. The last method is based on probability. [3] proposed in . This method calculates into two fields, legitimate and Sybil. The authors assumed the graph the attributes of nodes property. is of 0 and the Sybil node is with score 1.

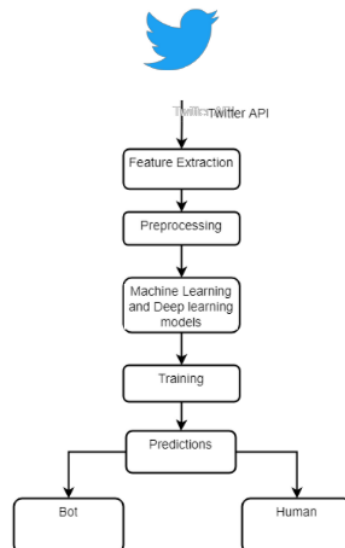
### 4.2 Crowdsourcing

[4] held a Twitter Bot contest to find bots that support vaccination discussions on Twitter. The teams participated used judgement for classification with individual methods. [5] construction of ground truths datasets with four features to classify and labelling accounts into categories – human and bots. They were provided with attributes to determine like data, number of tweets, and favoured tweets, researchers used Cohen's Kappa (k) to annotated accounts in twitter.

### 4.3 Machine Learning

The preprocessing techniques for platforms was proposed in [6], by training algorithms multiple times after preprocessing and feature extraction. With API from Twitter and Apache Spark, collected over eighteen hundred accounts with sixty-two features. The highest classifier among the algorithms with of 0.83. [7] [1] classification uses convolution neural networks with classify history and behaviour. The authors experimented with publicly available dataset and also used new tweets with Twitter API. The main aim for BeDM is to capture latent features by fusing content and behaviour information. This paper is first to apply deep learning techniques to classify bots. The authors of [8] proposed system BotorNot, which uses random forest classifier [1] to classify social media bots. They used thousand features from six classes with analysed users, connections, media and features. The AUC score was 95%. Further in 2014, the authors improved their work, with extended training data, with less AUC than previous results. [9] with randomly selected accounts manually classifying accounts into program or human with annotators. Optimal extraction using two functions. Properties dependent on the correlation method. Finally extracted 8 traits to score. A Model using a learning algorithm with naive based decision trees, Bayesian networks, SVM and ANN [1]. The more accurate was scored by a random forest classifier with 88% of accuracy.

## 5. System Architecture



## 6. Existing System

The survey on existing system shows that the three existing methods to classify bots as discussed above are Graph Approach, crowdsourcing and Machine Learning algorithms. Machine learning methods are optimal when compared to the rest of the methods. Most of the papers use tree-based approach, Bayes theorem and random classifier. Random Classifier is most accurate and commonly used as it is very accurate in achieving more accuracy. Although it is prone to overfitting, Bayes theorem and Random forest classifiers is best with low number of features. SVM depends on selective kernel, as it reduces the error rate in classification process. Likewise, Neural networks results depend on the data provided; when the samples are large, the model performs well.

## 7. Proposed System

In this section, we propose a machine learning models with data collected using twitter API. The algorithms used are decision tree classifier, neural network. Model with neural network is found to be more accurate with the training accuracy of 90% and testing accuracy of 96%.

## 8. Classification Techniques

### 8.1 Neural Networks:

#### A. Dataset

The dataset used is taken using honeypot method from Twitter API. This dataset contains accounts that labelled bot or not including some features.

#### B. Metrics

Evaluation criteria are precision, recall and F1 score which are commonly used in bot detection. For each split, trained models with 80% data, cross-validation with 10% and the remaining is used for testing.

#### C. Performance

In the experiments, we conducted a series of tests with different parameter and layers with activation functions. The experiment results are shown below:

| Number of hidden layers* | Output layer activation function | Precision | Recall | F1           |
|--------------------------|----------------------------------|-----------|--------|--------------|
| 1                        | Sigmoid                          | 74.06     | 75.1   | 73.2         |
| 2                        | Sigmoid                          | 86.21     | 84.63  | 84.61        |
| 2                        | Softmax                          | 68.76     | 67.33  | 67.80        |
| 3                        | Sigmoid                          | 88.12     | 86.90  | <b>86.54</b> |

\*activation function in hidden layers is relu

### 8.2 Decision Tree Classifier:

The dataset and evaluation metrics are same as previous model (neural network). A tree can be categorized into two variables: direction and fiendishness. Choices are made based on data sharing.

Performance:

The Area under the curve for training is 95.6 and for testing is 0.93 when trained with decision tree classifier. The metrics are given below:

| Training accuracy | Testing accuracy | Precision | Recall | F1     |
|-------------------|------------------|-----------|--------|--------|
| 88.24             | 87.85            | 91.11     | 83.69  | 97.725 |

## 9. Conclusion

In this paper we proposed two methods to classify accounts in social media accounts by bots or human. The results show the performance and effectiveness with high f1 scores. 3 can study other social media platforms with behaviour modelling for bot classification.

## References

- [1] E. Alothali, N. Zaki, E. A. Mohamed and H. Alashwa, "Detecting Social Bots on Twitter: A Literature Review," *2018 13th International Conference on Innovations in Information Technology (IIT)*, pp. 175-180, 2018.
- [2] K. S. Adewole, N. B. Anuar, A. Kamsin, K. D. Varathan and S. A. Razak, "Malicious accounts: dark side of social networks," *Journal of Network and Computer Applications*, vol. 79, pp. 41-67, 2017.
- [3] J. Jia, B. Wang and N. Z. Gong, "Random walk based fake account detection in online social networks," *2017 47th Annual IEEE/IFIP International Conference on. IEEE*, vol. Dependable Systems and Networks (DSN), pp. 273-284, 2017.
- [4] V. Subrahmanian, A. Azaria, S. Durst, V. Kagan, A. Galstyan, L. Zhu, E. Ferra, A. Flammini and F. Menczer, "The darpa twitter bot challenge," *Computer*, vol. 49, pp. 38-46, 2016.
- [5] Z. Gilani , L. Wang and J. Crowcroft, "Classification of twitter accounts into automated".
- [6] M. Kantepe and M. C. Ganiz, ""Preprocessing framework for twitter bot detection," *2017 International Conference on. IEEE*, pp. 630-634, 2017.
- [7] C. Cai, L. Li and D. Zengi, "Behavior enhanced deep bot detection in social media," in *Intelligence and Security Informatics (ISI)*," *2017 IEEE International Conference on. IEEE*, pp. 128-130, 2017.
- [8] C. A. Davis, O. Varol, E. Ferrara, A. Flammini and F. Menczer, "Botornot: A system to evaluate social bots," *Proceedings of the 25th International Conference Companion on World Wide*, pp. 273-274, 2016.
- [9] A. Alarifi, M. Alsaleh and A. Al-Salman, "Twitter turing test: Identifying social machines," vol. 372, p. 2016, 332-346.

**5%**

SIMILARITY INDEX

**3%**

INTERNET SOURCES

**5%**

PUBLICATIONS

**1%**STUDENT PAPERS

---

PRIMARY SOURCES

---

**1**

Eiman Alothali, Nazar Zaki, Elfadil A. Mohamed, Hany Alashwal. "Detecting Social Bots on Twitter: A Literature Review", 2018 International Conference on Innovations in Information Technology (IIT), 2018

Publication

**2%****2**

Submitted to University of Exeter

Student Paper

**1%****3**

"Mobile Computing and Sustainable Informatics", Springer Science and Business Media LLC, 2022

Publication

**1%**

4

Mariam Orabi, Djedjiga Mouheb, Zaher Al Aghbari, Ibrahim Kamel. "Detection of Bots in Social Media: A Systematic Review", Information Processing & Management, 2020

Publication

1%

5

[scalab.dimes.unical.it](http://scalab.dimes.unical.it)

Internet Source

1%

6

[www.ijrte.org](http://www.ijrte.org)

Internet Source

1%

# To Detect Fake Identities in Twitter using Machine Learning Models.

**Pedada Chetanroop, Ruthwik Preetham**

Sathyabama Institute of Science and Technology, Computer Science, India

Sathyabama Institute of Science and Technology, Computer Science, India

[p.chetanroop0509@gmail.com](mailto:p.chetanroop0509@gmail.com), [rutwikritu@gmail.com](mailto:rutwikritu@gmail.com)

**Abstract:** Social bots are considered to be the most popular type of spamming like spamming malware links, produce fake news, spread rumours to manipulate public opinion. Recently large-scale social bots have been created and are wide spread on social which have a bad impact on public and internet users' safety in all social media platforms. Bot detection aims to distinguish bots from humans to aid understanding the news or opinions. In recent times, classification of bots in social media have become more as they are populated everywhere. In this paper, we propose a decision tree classifier and a deep learning method to classify bots and humans in Twitter using Twitter API. This proposed model uses what an account has tweeted and cross reference against a bag of words model. These methods are unique that applies deep learning concepts to classification. Using real world data from twitter shows the validity of the model we proposed.

**Keywords:** Social Media, Twitter, decision trees, Neural networks, accuracy metrics.

## 1. Introduction

Bots are automated software that scrape or spam, interact with human on a social media platform. Users often tend to use social bots to manipulate opinions, rapid spam of rumours and also rate a product with fake reviews. According to twitter, over 15% accounts are bots. As the social media evolve, the issues caused by bots are obvious.

Bots in twitter control the account via twitter API, to automate actions like follow users, unfollow, tweet, retweet, and reply to direct messages. Few bots in this media are bots that help the user to schedule a tweet, help the assist by providing valuable insights over API. Such bots are termed as hybrid bots. For these specific reasons researchers are in design of advanced methods to classify social media bots, hence bot detection is a very valuable research problem.

The challenges in bot detection have been identified by many teams in the past. Common methods used in classifying are: Graph based, crowdsourcing and machine learning.

Graph based uses social graphs of social networks to get the network information and nodes for multiple accounts to classify bot activities. In crowdsourcing [1], an expert annotator identifies, evaluates and classifies accounts. The machine learning involves training large datasets with multiple algorithms and statistical methods to distinguish human or computer led activities across social media platforms.

## 2. Objectives

Survey current strategies on Machine learning and deep learning to classify bots.

Classify twitter accounts from human and bots

The authenticity of people constantly questioning if it is a fake or not, this classifier authenticates and solves the problem.

## 3. Methodology

There are papers focusing on different techniques and methods that are targeted to classify social media accounts on twitter media platform.

The methods used are Graph Based Approach, Crowdsourcing, Machine Learning Algorithms [1].

Graph based approach uses nodes and samples of benign nodes for classification. Three social media based methods are employed namely trust propagation, graph clusters, and graph metrics. Crowdsourcing approach is a detection method that involves an annotator to find relations between accounts posted by bot or human. The main aim of annotator is here to find the differences in bots and human accounts. This method involves a lot of guesses from the annotators. Machine Learning techniques uses a large amount of datasets that have many features. Using these models, patterns on feature of twitter accounts can be found and the probability that those accounts being humans or bots can be calculated. [1]

## 4. Literature Survey

### 4.1 Graph Based Detection

The authors [2] proposed three methods to detect accounts. The methods include confidence spread to determine if the connection between two attributes are strong. Next process includes grouping the attributes with similar properties. The last method is based on probability. [3] proposed a Sybil detection method that employs a random walk based method in an undirected graph. This method calculates scores into two fields, legitimate and Sybil. The authors assumed the graph the attributes of nodes to share the same property. The legitimate node is of score 0 and the Sybil node is with score 1.

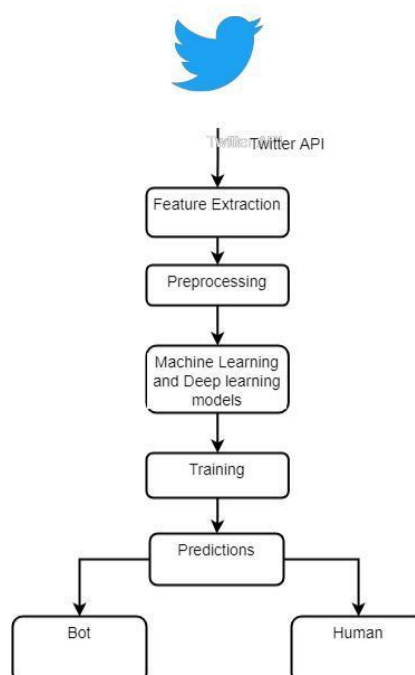
### 4.2 Crowdsourcing

[4] held a Twitter Bot contest to find bots that support vaccination discussions on Twitter. The teams participated used judgement for classification with individual methods. [5] construction of ground truths datasets with four features to classify and labelling accounts into categories – human and bots. They were provided with attributes to determine like data, number of tweets, and favoured tweets, researchers used Cohen's Kappa (k) to annotated accounts in twitter.

### 4.3 Machine Learning

The preprocessing techniques for classification of bots and humans in social media platforms was proposed in [6], by training algorithms multiple times after preprocessing and feature extraction. With API from Twitter and Apache Spark, collected over eighteen hundred accounts with sixty-two features. The highest classifier among the algorithms was 86% for gradient boosted trees with f1 score of 0.83. [7] [1] classification uses convolution neural networks with LSTM long-short-term memory with hidden layers to classify history and behaviour. The authors experimented with publicly available dataset and also used new tweets with Twitter API. The main aim for BeDM is to capture latent features by fusing content and behaviour information. This paper is first to apply deep learning techniques to classify bots. The authors of [8] proposed system BotorNot, which uses random forest classifier [1] to classify social media bots. They used thousand features from six classes with analysed users, connections, media and features. The AUC score was 95%. Further in 2014, the authors improved their work, with extended training data, with less AUC than previous results. [9] with randomly selected accounts manually classifying accounts into program or human with annotators. Optimal extraction using two functions. Propoties dependent on the correlation method. Finally extracted 8 traits to score. A Model using a learning algorithm with naïve based decision trees, Bayesian networks, SVM and ANN [1]. The more accurate was scored by a random forest classifier with 88% of accuracy.

## 5. System Architecture



## 6. Existing System

The survey on existing system shows that the three existing methods to classify bots as discussed above are Graph Approach, crowdsourcing and Machine Learning algorithms. Machine learning methods are optimal when compared to the rest of the methods. Most of the papers use tree-based approach, Bayes theorem and random classifier. Random Classifier is most accurate and commonly used as it is very accurate in achieving more accuracy. Although it is prone to overfitting, Bayes theorem and Random forest classifiers is best with low number of features. SVM depends on selective kernel, as it reduces the error rate in classification process. Likewise, Neural networks results depend on the data provided; when the samples are large, the model performs well.

## 7. Proposed System

In this section, we propose a machine learning models with data collected using twitter API. The algorithms used are decision tree classifier, neural network. Model with neural network is found to be more accurate with the training accuracy of 90% and testing accuracy of 96%.

## 8. Classification Techniques

### 8.1 Neural Networks:

#### A. Dataset

The dataset used is taken using honeypot method from Twitter API. This dataset contains accounts that labelled bot or not including some features.

#### B. Metrics

Evaluation criteria are precision, recall and F1 score which are commonly used in bot detection. For each split, trained models with 80% data, cross-validation with 10% and the remaining is used for testing.

#### C. Performance

In the experiments, we conducted a series of tests with different parameter and layers with activation functions. The experiment results are shown below:

| Number of hidden layers* | Output layer activation function | Precision | Recall | F1           |
|--------------------------|----------------------------------|-----------|--------|--------------|
| 1                        | Sigmoid                          | 74.06     | 75.1   | 73.2         |
| 2                        | Sigmoid                          | 86.21     | 84.63  | 84.61        |
| 2                        | Softmax                          | 68.76     | 67.33  | 67.80        |
| 3                        | Sigmoid                          | 88.12     | 86.90  | <b>86.54</b> |

\*activation function in hidden layers is relu

### 8.2 Decision Tree Classifier:

The dataset and evaluation metrics are same as previous model (neural network). A tree can be categorized into two variables: direction and fiendishness. Choices are made based on data sharing.

Performance:

The Area under the curve for training is 95.6 and for testing is 0.93 when trained with decision tree classifier. The metrics are given below:

| Training accuracy | Testing accuracy | Precision | Recall | F1     |
|-------------------|------------------|-----------|--------|--------|
| 88.24             | 87.85            | 91.11     | 83.69  | 97.725 |

## 9. Conclusion

In this paper we proposed two methods to classify accounts in social media accounts by bots or human. The results show the performance and effectiveness with high f1 scores. In future work, we can plan to study other social media platforms with behaviour modelling for bot classification.

## References

- [1] E. Alothali, N. Zaki, E. A. Mohamed and H. Alashwa, "Detecting Social Bots on Twitter: A Literature Review," *2018 13th International Conference on Innovations in Information Technology (IIT)*, pp. 175-180, 2018.
- [2] K. S. Adewole, N. B. Anuar, A. Kamsin, K. D. Varathan and S. A. Razak, "Malicious accounts: dark side of social networks," *Journal of Network and Computer Applications*, vol. 79, pp. 41-67, 2017.
- [3] J. Jia, B. Wang and N. Z. Gong, "Random walk based fake account detection in online social networks," *2017 47th Annual IEEE/IFIP International Conference on. IEEE*, vol. Dependable Systems and Networks (DSN), pp. 273-284, 2017.
- [4] V. Subrahmanian, A. Azaria, S. Durst, V. Kagan, A. Galstyan, L. Zhu, E. Ferra, A. Flammini and F. Menczer, "The darpa twitter bot challenge," *Computer*, vol. 49, pp. 38-46, 2016.
- [5] Z. Gilani , L. Wang and J. Crowcroft, "Classification of twitter accounts into automated".
- [6] M. Kantepe and M. C. Ganiz, "'Preprocessing framework for twitter bot detection," *2017 International Conference on. IEEE*, pp. 630-634, 2017.
- [7] C. Cai, L. Li and D. Zengi, "Behavior enhanced deep bot detection in social media," in *Intelligence and Security Informatics (ISI)*," *2017 IEEE International Conference on. IEEE*, pp. 128-130, 2017.
- [8] C. A. Davis, O. Varol, E. Ferrara, A. Flammini and F. Menczer, "Botornot: A system to evaluate social bots," *Proceedings of the 25th International Conference Companion on World Wide*, pp. 273-274, 2016.
- [9] A. Alarifi, M. Alsaleh and A. Al-Salman, "Twitter turing test: Identifying social machines," vol. 372, p. 2016, 332-346.