electronics

Article

TweezBot: An AI-Driven Online Media Bot Identification Algorithm for Twitter Social Networks

Rachit Shukla [1], Adwitiya Sinha [1,*] and Ankit Chaudhary [2]

[1] Department of Computer Science & Engineering and Information Technology, Jaypee Institute of Information & Technology, Noida 201304, Uttar Pradesh, India; rachit.shukla28@gmail.com
[2] Department of Computer Science, University of Missouri, St. Louis, MO 63121, USA; dr.ankit@ieee.org
* Correspondence: mailtoadwitiya@gmail.com

Abstract: In the ultra-connected age of information, online social media platforms have become an indispensable part of our daily routines. Recently, this online public space is getting largely occupied by suspicious and manipulative social media bots. Such automated deceptive bots often attempt to distort ground realities and manipulate global trends, thus creating astroturfing attacks on the social media online portals. Moreover, these bots often tend to participate in duplicitous activities, including promotion of hidden agendas and indulgence in biased propagation meant for personal gain or scams. Thus, online bots have eventually become one of the biggest menaces for social media platforms. Therefore, we have proposed an AI-driven social media bot identification framework, namely *TweezBot*, which can identify fraudulent Twitter bots. The proposed bot detection method analyzes Twitter-specific user profiles having essential profile-centric features and several activity-centric characteristics. We have constructed a set of filtering criteria and devised an exhaustive bag of words for performing language-based processing. In order to substantiate our research, we have performed a comparative study of our model with the existing benchmark classifiers, such as Support Vector Machine, Categorical Naïve Bayes, Bernoulli Naïve Bayes, Multilayer Perceptron, Decision Trees, Random Forest and other automation identifiers.

Keywords: cyber security; online social networks; social media bots; machine learning; bot detection; data analytics

check for updates

Citation: Shukla, R.; Sinha, A.; Chaudhary, A. *TweezBot*: An AI-Driven Online Media Bot Identification Algorithm for Twitter Social Networks. *Electronics* **2022**, *11*, 743. https://doi.org/10.3390/electronics11050743

Academic Editor: Arkaitz Zubiaga

Received: 6 January 2022
Accepted: 24 February 2022
Published: 28 February 2022

1. Introduction

Twitter is a microblogging site which was founded in 2006 with the motive to create an online platform for interactive discourse. The users can post and interact with each other in a series of messages which are called *tweets* [1]. Tweets were originally comprised of 140 characters, but with the growing popularity of Twitter in the domain of social media, the limit was eventually increased to 280 characters for non-CJK languages, that is, the languages consisting of Chinese, Japanese and Korean characters. As of early 2019, it was reported that Twitter handles a phenomenal 330 million active monthly users, and confronts a traffic of almost 500 million tweets every day on the site. With the incredible amount of tweets sent out every day, amounting to billions every year, the chance of various forms of cyber threats also began to erupt. This includes fraud, scam, spamming, and duplicitous behaviour over online social media platforms [2]. In order to push a positive agenda, or to gain an advantage of monetary or social type, there are certain automated software applications that control a user account and are popularly called *Twitter Social Media Bots* [3]. With the help of Twitter API, these bots often autonomously perform mainstream actions, such as re-tweeting, tweeting, liking, following, un-following, or sending direct messages to other accounts. Some of the bots have positive intentions, and are mostly used by organisations to provide automated responses to user queries, customer grievances, etc. The good bots help in broadcasting helpful messages and information in times of distress, and some create useful and meaningful content to engage the user base. But in recent times, aggressive marketing and advertising has led to a flux of bots using the Twitter API in a malicious manner. This infers that a Twitter bot can circumvent the API rate limits and violate the user privacy guidelines laid down by Twitter; lately, they have been used for spamming, deception, and posing as an impostor.

In a fast-paced world where digital marketing and digital consumerism is increasing with every passing second, our research perspective is to propose a method to identify malicious Twitter bots and reduce the impact of cyber threats caused by such fraudulent profiles.

Research Highlights. Our research initiative is to tackle the menace of cyber threats posed by online bots on Twitter social media. The significant contributions and highlights of our research are summarized as follows:

- Our proposed *TweezBot* model aims at revealing the automated tweeting behavior of online Twitter bots.
- The proposed classifier is a multi-layer condition-based model developed using artificial intelligence for bot identification.
- Several string-based and numeric profile-centric attributes, such as profile name, description, location, listed count, verification status, etc. form the condition base.
- The comparative study is performed with the benchmark machine and deep learning models, including the following: decision tree, random forest, Bernoulli Naïve-Bayes classifier, categorical Naïve-Bayes, support vector machine, and multi-layer perceptron.
- Performance analysis is also conducted with publicly available standard automation identifier APIs, such as Botometer, BotSentinel, and TweetBotOrNot.
- Results obtained from the existing classifiers and APIs are used for comparative examination and performance of *TweezBot* in terms of accuracy, recall, precision, F1-score, Cohen-Kappa score, area under ROC curve, and other.

The remaining sections are organized as follows: Section 2 provides an in-depth survey of related work and current research trends. Section 3 outlines the research methodologies, and is followed by the proposed framework in Section 4. Further, experimental outcomes are highlighted in Section 5, providing the dataset description, exploratory analysis and comparative studies. Finally, our research findings with conclusion and future scopes are discussed in Section 6.

2. Related Work

There have been several relevant studies conducted in the area of social media bot identification. In a study conducted in [4], the authors have approached the social bot detection problem as a classification problem. They have used extensive data pre-processing and feature extraction operations on the labelled data obtained from Twitter accounts that have been suspended by Twitter on the basis of the assumption that these accounts were malicious in nature and were showing bot behaviour or suspicious activity. The authors have used Twitter Streaming API to stream data and have performed pre-processing with 62 features for each Twitter user, which are then fed to the machine learning classifier for training. The text similarity feature for calculating the sentiment related features is calculated by taking 100 random tweets of each user, which are compared using a term frequency–inverse document frequency (TF-IDF) vectorizer. The vectorizer was used to tokenize the document and calculate the weights for each term. In another study [5], Twitter metadata was used to derive 13 unique characteristics. In the dataset provided, every single feature was a good representative for the variance present. Correlation matrix and PCA (principal component analysis) was used to check the association and standing of the variables, providing more rich insights [6]. The authors have employed an unlabelled data set and created K-means clusters, which is a machine learning methodology, formally unsupervised [7]. This method has already been used to track down problematic bots in Twitter campaigns. The authors have picked daily activity, i.e., the tweets, retweet percentage, and daily favourite count as clustering features. Two was chosen as the number of clusters. Using a 3D scatter plot for methodology, they derived those bots that had a high value of retweet percentage and daily favourites. It is also inferred that the bot accounts are small or moderate in number, but the participation rate is soaring extremely.

Furthermore, in the research conducted in [8], it is proposed that a similarity-based approach could address this gap by complementing existing supervised and unsupervised methods. The authors present an approach called Bot-Match that evaluates social media embeddings that use a semi-supervised recursive algorithm, which in this instance is nearest neighbour search. The authors have developed a logistic regression classifier to detect random strings using characteristics such as n-grams of characters and string entropy, which has a recognition rate of 94.25 percent for random string user accounts on Twitter. As part of a planned intimidation effort, they can tag 4312 accounts with a random string screen name. In yet another contribution [9], it was found that in order to identify legitimate users from among the bots, two methods are introduced, both of which are based on Natural Language Processing (NLP). A feature extraction methodology is proposed in the first method for detecting accounts that send automated messages. The subset of characteristics picked is given into machine learning algorithms after applying feature selection techniques and dealing with skewed datasets. A deep learning architecture is proposed in the second method to determine if tweets were posted by real users or generated by bots. Additionally, important to the learning is Benford's law [10], which was important to understand the studies made in [11]. The researchers created and developed software to collect account metrics from Vkontakte, bought bots and collected eight bot datasets, gathered ten genuine user datasets, ran Kolmogorov-Smirnov tests for each measure in each dataset, and compared bot and real user findings. Bots could be deemed a considerable key in such profiles and accounts if the p-value of agreement between profile distribution and Benford's distribution is smaller than a set threshold.

In [12], the authors have used a supervised machine learning strategy to detect bots, which relies on an exclusive feature known as the bot score to assess a user's bot probability. Users were sorted into several clusters based on their motion using an unsupervised machine learning methodology. The AUC was significantly higher than in earlier research, with a fold cross-validation score of 0.9626, accuracy of 0.9743, recall score of 0.9717, and F1 measure of 0.9730. Furthermore, the authors introduced a novel type of hash chain that outperformed similar methods while being flexible enough to run on a range of systems. The selective dynamic caching solution lowered workloads dramatically while needing less storage. The most important conclusion is that the algorithm can give vital security to social networking sites in a flexible and adaptive manner, perhaps assisting in the fight against dangerous bots and cyber-attacks. Another study on an independent twitter bot detector was proposed in [13]. The authors describe a language-independent method for classifying each tweet as either autogenerated (AGT) or human-generated (HGT). AGT is a kind of a tweet in which the majority or all of the natural language data is created automatically by a bot or other machine. The method proposed classifies a tweet based on solely the metadata that arrives with every tweet, and we use metadata parameters that are independent of language and geography. The empirical section of the study reveals high success rates. In a related context, a recurrent neural network related study was proposed in [14]. The authors have employed bidirectional Long Short-term Memory (BiLSTM) to effectively record information across tweets to aid human users in recognizing who they are conversing with on Twitter. Addressing the problem of incorrect classification and results (false positives and false negatives) produced by a Botometer, the authors of [15] shed light on how to handle this in an effective way. The paper demonstrates that Botometer scores are inaccurate for estimating bots, particularly when used in a different language. It also suggests that the thresholds are prone to fluctuation, even when applied conservatively, resulting in false negatives (i.e., bots being mistaken for people) and false positives (i.e., humans being classed as bots). Most social science studies that employ the technology will incorrectly count a significant number of human users as bots, and vice versa [16–26]. There is more research on revealing the communities of such profiles on social media applications [27,28]. Motivated by the above contributions in the related domain, our research is aligned to the similar context of detecting bots hidden under the veil of human-like behavior using artificial intelligence and considering several filters based on basic and derived profile-centric attributes from Twitter. For this, our design considers building a multi-layer set of conditions based on AI-driven initiatives for achieving better performance in bot identification.

3. Research Methodologies

Our research is focused on identifying bot profiles on one of the most widely used social media platforms, i.e., Twitter. We have proposed a novel framework to identify the features that could be appropriately used for revealing the bots based on the Twitter-centric profiles gathered. Data science techniques were applied to gain a better insight of the attributes present in the dataset for the distinction between identifying bots and non-bots.

3.1. Correlation Statistics

The initial phase of our research begins with correlation analysis of the possible Twitter-specific features of the online users. In order to assist the feature analysis, we have considered a Spearman rank-order coefficient to examine the relation of all the individual features with respect to the target variable. The correlation formula is expressed as follows, in Equation (1).

$$\rho = 1 - \frac{6\sum q_i^2}{n(n^2-1)} \tag{1}$$

In Equation (1), ρ denotes the Spearman's rank-order coefficient, and q_i refers to the difference between the two ranks in an observation. The number of observations is depicted as n. Subsequently, there are several benchmark AI-based classifiers that have been further employed for comparative study.

3.2. Decision Trees

This classifier is one of the popular benchmarks in the supervised machine learning category that often provides fairly good solutions to classification problems. The training samples in a decision tree are stored in an array that could be sparse or dense. The shape of the array is defined as number of samples and features. The class tags for the training samples are held in an array y of negative and non-negative values. If a target is an outcome with range of values 0, 1, . . . , $t-1$; therefore, in regard to a particular node f, the classification formulation can be denoted as in the following equation:

$$p_{ft} = 1/N_f \sum_{y \in U_f} I(y = t) \tag{2}$$

In this equation, the data at node f is represented by U_f, where N_f is the number of samples. If f is a terminal node, then the predicted probability feature for the particular region is denoted by p_{ft} with respect to the observations in class t for the node f.

3.3. Random Forest Classifier

By fitting a specified number of decision tree classifiers on sub-samples of the dataset, a meta estimator that uses the law of averages to enhance projected accuracy and restrict over-fitting is called a Random Forest classifier. If the value of the bootstrap condition is true or in default, the sub-sample size is regulated by the maximum number of samples argument; otherwise, the entire dataset is utilised to create each tree. Being an ensemble learning technique, in most of the binary as well as binary classification problems, it provides reliable solutions with good accuracy [16]. Hence, we have considered it as an existing counterpart to compare the results of our proposed model.

3.4. Bernoulli Naïve Bayes

Bernoulli Naïve Bayes uses Naive Bayes training and classification models to classify and train data, which has been distributed in accordance with the multivariate Bernoulli distributions. [17]. Although there may be several features, each is believed to be a binary-valued component or variable. As a result, samples in this class must be represented as feature vectors that are binary-valued. The decision rule is expressed in Equation (3), where Q is the probability of class V_i for dependent variable J.

$$Q(V_i \mid J) = Q(i \mid J)V_i + (1 - Q(i \mid J))(1 - V_i) \tag{3}$$

3.5. Categorical Naïve-Bayes

The categorical Naive Bayes method is well suited for the categorical distribution of discrete and distinguishing features [18]. Each feature's categories are chosen from a categorical distribution. Here, the probability of category k of a feature i in a selected class c is shown in Equation (4) as following.

$$P(x_i = \ k \mid y = c \, ; \, \alpha) = \frac{N_{kic} \ + \alpha}{N_c \ + \alpha n_i} \tag{4}$$

As highlighted in the aforementioned equation, N_{kic} is the amount of times group k is present in the samples x_i that belong to the class c. Moreover, N_c refers to the total amount of samples fitting to the class c, α is the Laplace smoothing parameter, and n_i is the amount of accessible categories.

3.6. Support Vector Machine

The Support Vector Machine (SVM) generates a hyperplane in an N-dimensional space, where N denotes the number of parameters used to differentiate data points present. There are numerous hyperplanes that can be chosen to split the two varieties of data points. The objective is to discover a plane with the greatest margin, or the greatest remoteness between data points from both classes. It helps in maximizing the margin distance, while giving some support and making it easier to classify the subsequent data points that are present.

$$W^l + b = 0 \tag{5}$$

Here, W denotes the weight vector, l is the transpose of the weight vector, and b is the bias. In Equation (6), there two vectors, of which one is the positive (e^+) and the other is a negative (e^-). Additionally, O is the weight vector being orthogonal to the decision boundary, with the support vectors defining the margin.

$$\left(e^+ - e^-\right) \cdot \hat{O} = \left(e^+ - e^-\right) \cdot \frac{O}{||O||} = e^+ \cdot \ \frac{O}{||O||} - e^- \cdot \ \frac{O}{||O||} \tag{6}$$

Here, $\hat{O}$ is the unit vector, and $||O||$ is the strength of the vector. Being a categorical predictor, SVM is suitable for addressing the binary classification problem of bot detection [19].

3.7. Multi-Layer Perceptron

The multi-layer perceptron is a neural network technique which comes under the umbrella of supervised deep learning [20]. It employs a cognitive approach to learn a function, $g(*) = R^m \ \rightarrow R^z$, by training on a set of data, as m is the quantity of dimensions we are using for input and z is the quantity of channels or dimensions for output. The advantages of the multi-layer perceptron are mainly the ability to learn and deploy non-linear models and acquire knowledge about the models in real-time with a partial fitting function. The multi-layer perceptron algorithm used by the MLP classifier class uses backpropagation instead of front-propagation, as seen in Equation (7):

$$y = \varphi\left(w^T x + B\right) \tag{7}$$

As expressed above in Equation (7), w denotes the vector that consists of the weights, x is the vector that contains the inputs, B is the bias present, and φ is the activation function, which is inherently non-linear.

4. Proposed Framework: *TweezBot* Model

This section illustrates our proposed bot detection framework, termed as TweezBot, which is a multi-layer condition-based model designed exhibiting an AI-driven initiative. Our model begins with feature selection based on thorough analytics for identifying the profiles that are likely to be bots. This is followed with the model building and validation with several performance evaluation parameters and existing counterpart techniques. The infographic in Figure 1 highlights the flow diagram to illustrate the procedure of our proposed framework.

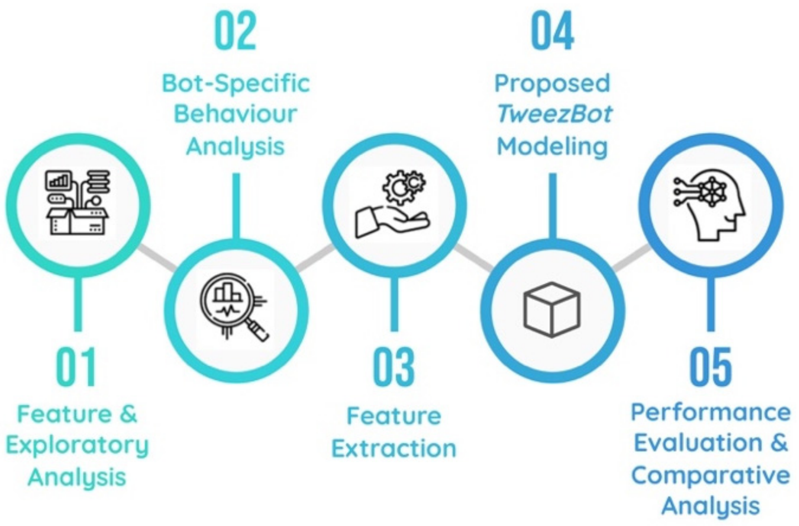
02
Bot-Specific Behaviour Analysis
04
Proposed *TweezBot* Modeling
01
Feature & Exploratory Analysis
03
Feature Extraction
05
Performance Evaluation & Comparative Analysis

Figure 1. Infographic depicting our model flow diagram.

The feature analysis was performed on the dataset for understanding the wholeness and missingness of the data. These features are further organized in multiple sets of conditions to filter out the bot profiles (Table 1). Subsequently, correlation statistics was applied to understand the relations between various attributes of the Twitter object, i.e., the social media user. Moreover, exploratory data analysis was conducted to identify the various attributes of a bot profile against a non-bot profile, for better understanding of the characteristics of a bot profile, and the different ways of mimicking a genuine human user. Finally, data analytics was applied to explore the kind of attributes that are prevalent or common to predict bot behavior.

Several significant features are considered as a set of conditions to build the prediction base. This includes the account verification status, location of the user, and presence of certain specific phrases in screen name, user description, name, and profile status. These conditions help us in extracting the unique features of a bot and a non-bot. We also apply a Bag-of-Words to process the string-based features for probable bot identification. BoW refers to the corpus of words that is used to perform feature engineering by parsing tokens that are likely to be prevalent in bots, or accounts exhibiting bot behavior. For example, the word ‘bot’ has been included in the bag-of-words and then checked in the screen name, profile name, and description attributes in the dataset. For example, a user named @mattlieberisbot from our dataset will be flagged as a bot, as the screenname contains one of the words from the BoW. This will further increase its weightage as a bot profile. Finally, we also consider the listed count and the existence of an extended profile as well as a profile image. Using these attributes as conditions, we formulate our model (Algorithm 1).

Table 1. Symbols & description of variables for implementation of proposed model.

Symbol	Description
n	Number of users in dataset D
D	Twitter dataset collected
D_{tr}	Training Dataset
D_{ts}	Testing Dataset
t	Dataframe used for attributes of a Twitter object
t_{id}	Id of a twitter user in a dataframe
t_{focnt}	Followers count of a twitter user
t_{frcnt}	Friends count of a twitter user
t_{url}	Link provided in t_{desc}
t_{loc}	Location of user profile
t_{ver}	Verification status imparted by Twitter
t_{name}	Name of the twitter user in the dataset
t_{desc}	Description of the twitter user
t_{sname}	Screen name of the twitter user
t_{status}	Statuses count of the twitter user
BoW	Bag of words containing objectionable phrases
$t_{hasprofimg}$	Existence of profile image of Twitter user
$t_{hasextprof}$	Existence of extended profile of Twitter user
$t_{lstcount}$	Listed count of Twitter user
t_{bot_pred}	Predicted status of profile as bot or non-bot
t_{bot_lbled}	Actual labelled status of a profile as bot or non-bot
A_{tr}	Training accuracy computation
A_{ts}	Testing accuracy computation

Algorithm 1: Proposed *TweezBot* Algorithm

Input: Twitter dataset D
Procedure:

```
1:  $n = |D|$    # count of users in Twitter dataset
2:  $D := \{u_1, u_2, \ldots, u_{n-1}, u_n\}$    # dataset with users
3:  $D_{tr} \leftarrow$ training dataset
4:  $D_{ts} \leftarrow$ testing dataset
5:  $t_{bot\_lbled} \leftarrow$ initial target labels
6:  for each $u$ in $D_{tr}$ :
7:      $t = \{u.t_{name},\ u.t_{desc},\ u.t_{sname}\}$
8:      for $w$ in $BoW$:
9:          if $w$ exists in $t$:
10:             $u.t_{bot\_pred} = tru$
11:             break
12:         else:
13:             $u.t_{bot\_pred} = false$
14:         end-if
15:     end-for
16:     if $((u.t_{ver}) || (u.t_{hasprofimg}) || (u.t_{hasextprof}) || (u.t_{loc}) || ((u.t_{lstcount}) > 1000))$ :
17:         $u.t_{bot\_pred} = false$
18:     end-if
19:     $D_{tr,u} := D_{tr,u} \cup u.t_{bot\_pred}$    # append computed bot status as feature for $u$
20: end-for
```

Output: D_{tr} , $u.t_{bot\_pred}$ for each user $u \in D$

The following section summarizes our results based on the findings interpreted by the performance of our proposed approach. We have conducted exploratory analysis, data analytics, and feature engineering results. Further, we have performed comparative analysis with several existing machine learning and deep learning classifiers.

5. Experimental Outcomes

The base premise of *TweezBot* is to identify the social media bot profiles from the online user profiles on Twitter. There are several Twitter datasets available with bot and non-profiles [21,22]. We have extracted the Twitter-based dataset from Kaggle with 2789 distinct user profiles labelled with binary classes represented by bots and non-bots [23]. Our dataset contains the attributes relative to a Twitter user, which includes the user ID, user handle, user screen name, location, verification status, extended profile, count of friends and followers, total favorite count and listed count. Such exhaustive dataset allows us to have a model based on diverse attributes and variety. Our dataset used for our experimentation comprises two classes of social media profiles that are fairly balanced, with 47.22% bots and 52.78% non-bot profiles, respectively. Our proposed model processes the profile attributes of Twitter user objects to extract several useful pieces of information, including its verification status, existence of profile image, extended profile attributes, and location. This set of attributes acts as a primary base condition that has been fed in all the analysis and feature engineering. The usage of listed count attributes by the user object, assists in tuning the performance of our classifier and negates the biased cases in which non-bots have been identified as bots. Another key feature of our approach in differentiating bots from non-bots is the collection of lewd and explicit strings obtained after researching and scoping certain bot accounts. This corpus of words is termed as a bag of words. For experimentation, the training and testing split was performed with the standard 7:3 ratio.

5.1. Feature Analysis

For feature analysis, the missingness of the data provided us with a significant lead in revealing the bot profiles. In Figure 2, the heatmap is highlighted to show the null values in our labelled dataset for each attribute. The graph shows the missingness, particularly in the location, description, and URL attribute. These attributes with null values are not dropped, as they have served as key features in detecting bot behavior and were useful for important deductions.

Moreover, to find the correlation between the attributes and understand the relations, we used Spearman's rank-order correlation coefficient for some valuable insights among the different attributes with respect to the target variable, i.e., *bot* status (Figure 3).

For the Spearman correlation review, further illustration to assist feature extraction is shown in Figure 4. It shows the correlation between the user id, existence of profile image, statuses, and default profile with respect to the target classification variable. It is apparent from the graph that there is a strong correlation between verification status, listed count, and number of friends and followers. As a result, we cannot perform correlation for categorical attributes; hence, we have considered the screen-name, description, and name directly in our feature engineering during training. The correlation study justifies the choice of input variables for model training.

5.2. Exploratory Data Analysis

For our exploratory data analysis, we have employed several crux parameters to examine the difference in the natures of a bot and a non-bot profile. This includes inspection of relevant attributes and features that eventually become prominent in our model building phase. Figure 5 illustrates that the bot users often try to conceal their personal information. This is apparent from the graph where most of the profile-based parameters for bots are missing. This indicates collusion in the online social network caused by such suspicious users. A smaller amount of information on the public domain actually helps these bots to rapidly disappear after fraudulent indulgence.

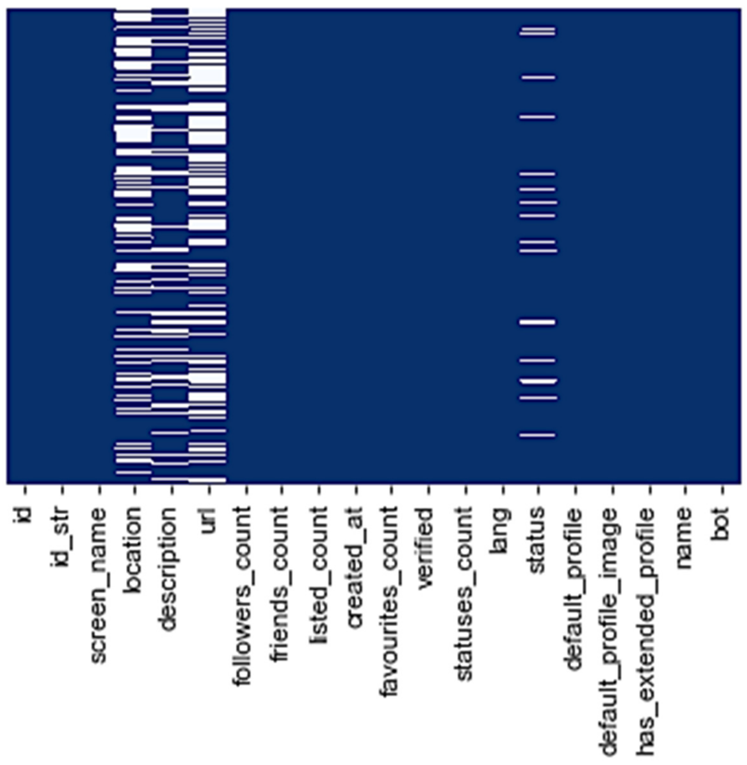

Figure 2. Analysis of Missingness in Bots Dataset.

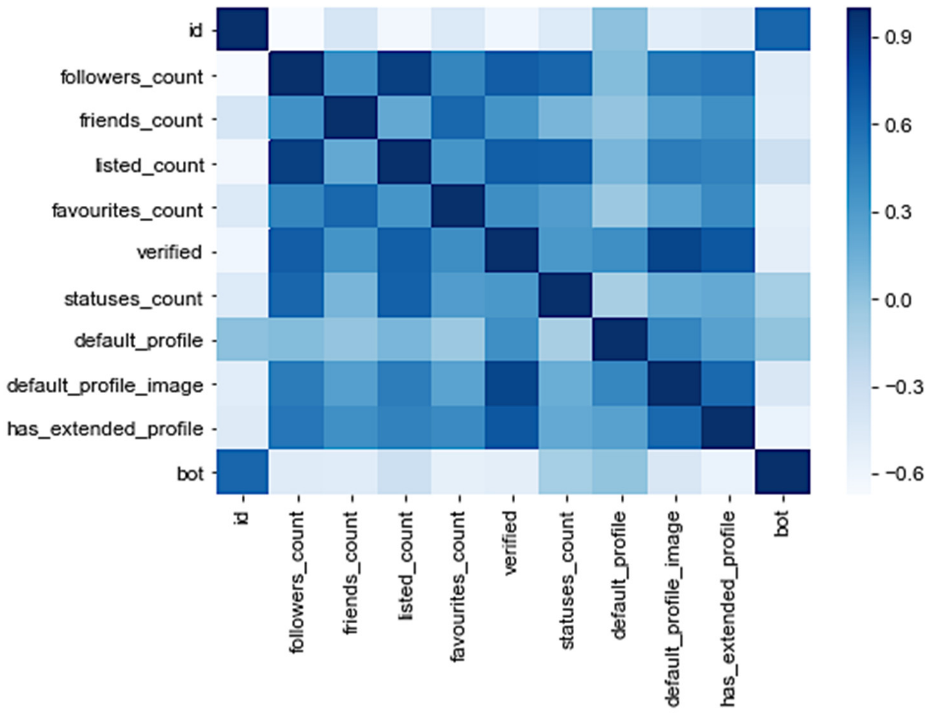

Figure 3. Spearman Correlation of Twitter Parameters.

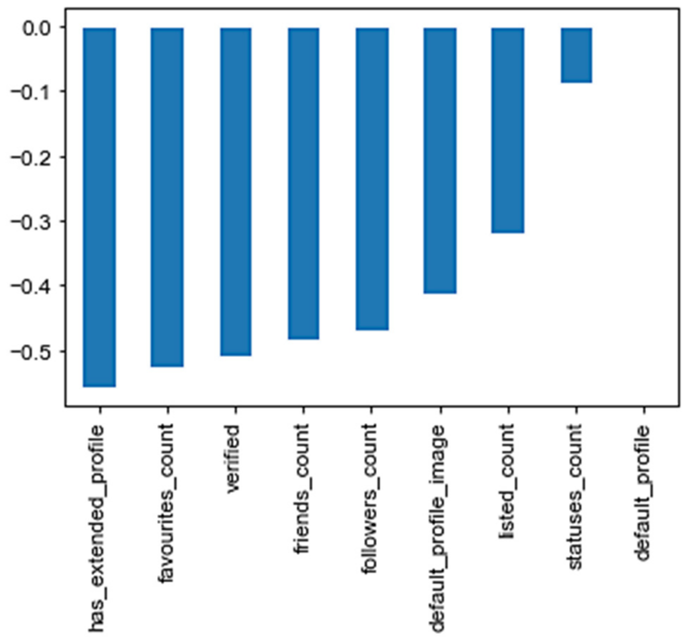

Figure 4. Study of Correlated Parameters.

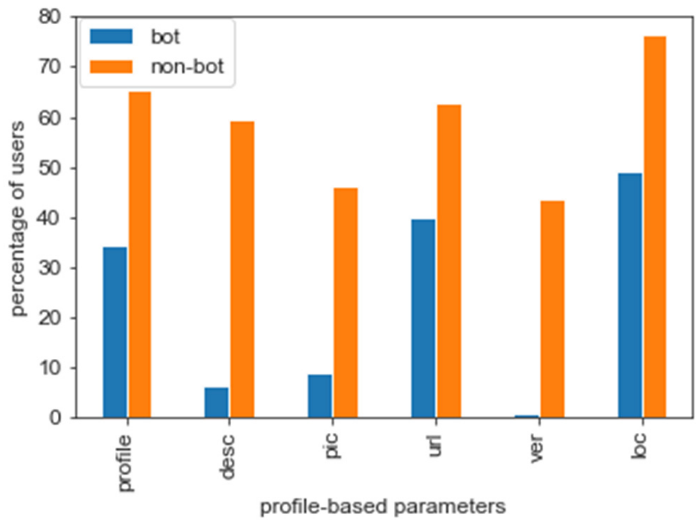

Figure 5. User Quantity based on profile parameters.

Further, Figure 6 depicts the study of bot characteristics based on statuses and favorite count. The statuses count signifies the number of posts and activities on the feed. The favorites count tells us how many times a certain post has been favorited by a user. The graph shows extremely low favorites counts for the bots on their feeds. In another analysis, shown in Figure 7, the number of followers is spiked for bots with extremely low favorites counts, while the non-bots exhibit a uniform behavior. The bot profiles often indulge in buying followers from online black-marketeers to mimic an influential public status.

In Twitter, friends are the specific users that a certain user may choose to follow (i.e., following). Figure 8 shows that in the case of the bots, the friend count has a concentration towards the bottom-left, indicating a low following, whereas there is a uniform ratio in the case of non-bots as the number of followers are not exorbitantly high, with friends being relatively uniform till a certain degree. In yet another study of friends and statuses count in Figure 9, it is evident that the bots take parts in short periods of time, and are extremely active during an ongoing event or trend, as depicted by the low number of statuses; while non-bots post over an extended timeline, which could span from months to years, thereby exhibiting normal behavior.

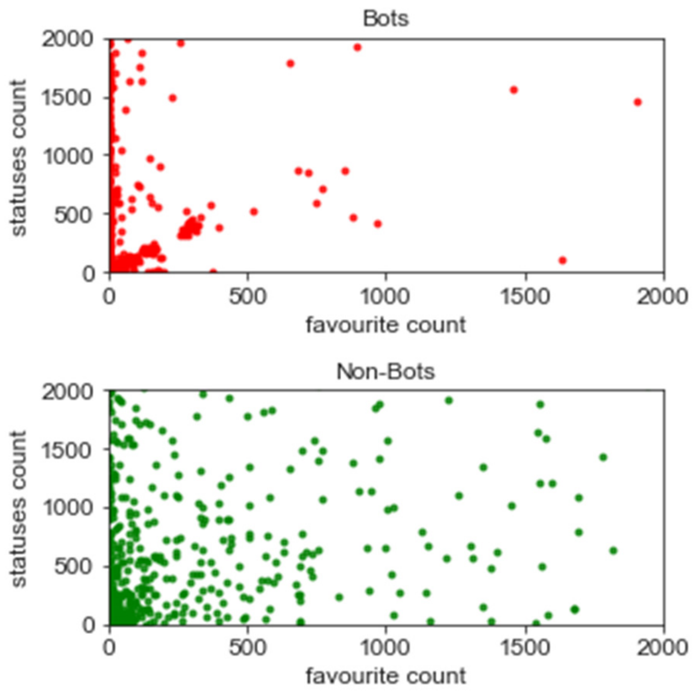

Figure 6. Statuses against Favorites count for Bots and Non-bots.

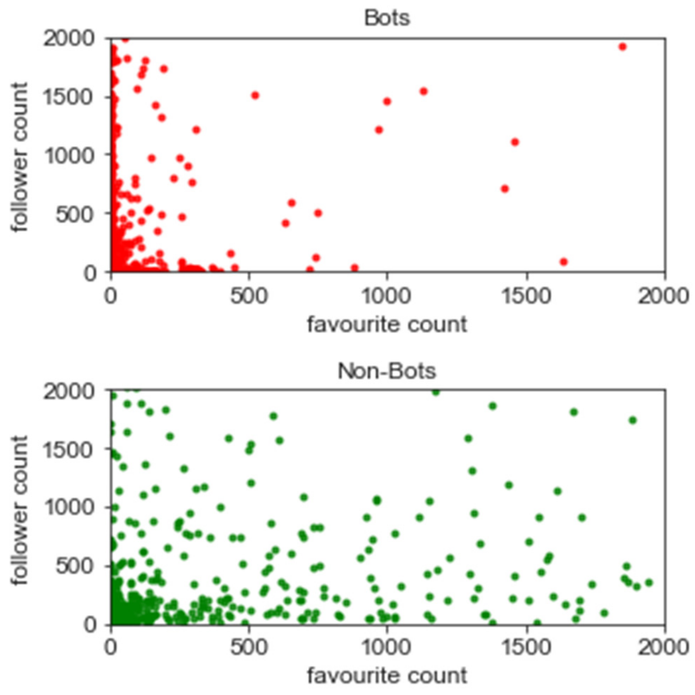

Figure 7. Follower against Favorites count for Bots and Non-bots.

In Figure 10, a key observation was found that bots do not experience an increase in their follower count even when they are posting more and staying extremely active, as a part of their attempt to mimic natural behavior. Meanwhile, for the non-bots, the follower count increases till a certain rate for a number of statuses.

5.3. Assessing Bot Behaviour

This section provides an extended study on scrutinizing bot behavior, in order to form a sound basis for developing a set of effective filters for our proposed *TweezBot*. Figure 11 illustrates that the number of users is more with an increasing number of follower counts, thus solidifying our basis of bot and non-bot differentiation. Further, in Figure 12, the non-bots have clearly been added to a number of lists, leading to the increase in the listed count feature, while bots have not added to many lists because of their suspicious nature and lack of credibility, though some bots might also often show non-malicious behavior.

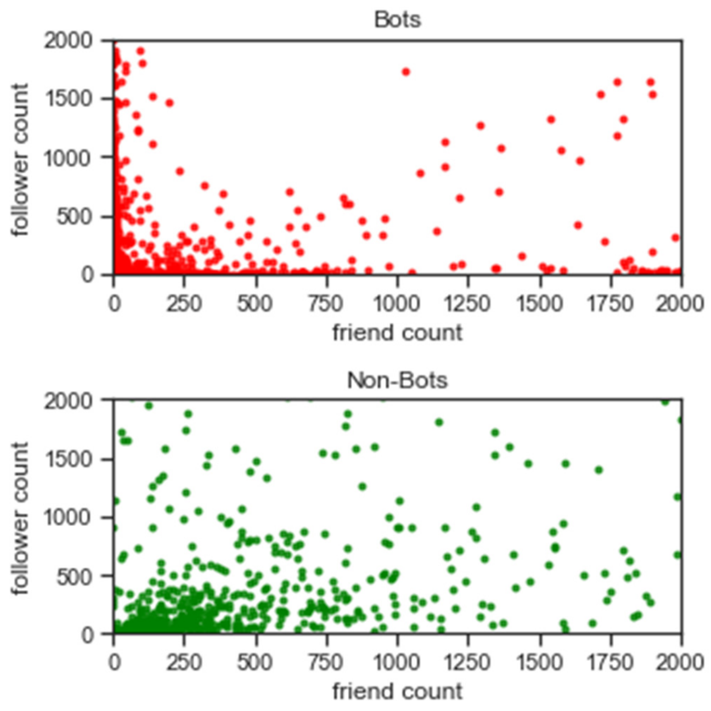

Figure 8. Follower against Friend count for Bots and Non-bots.

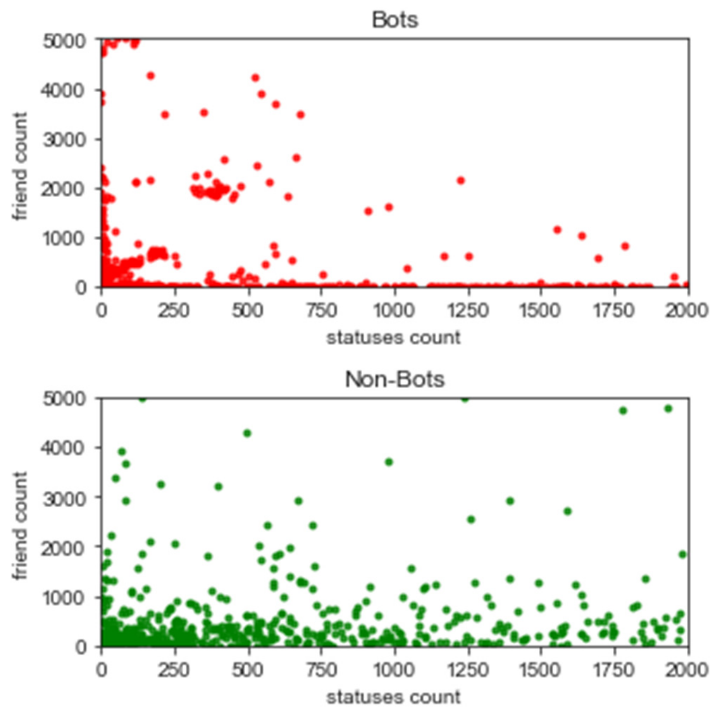

Figure 9. Friend count against statuses count for Bots and Non-bots.

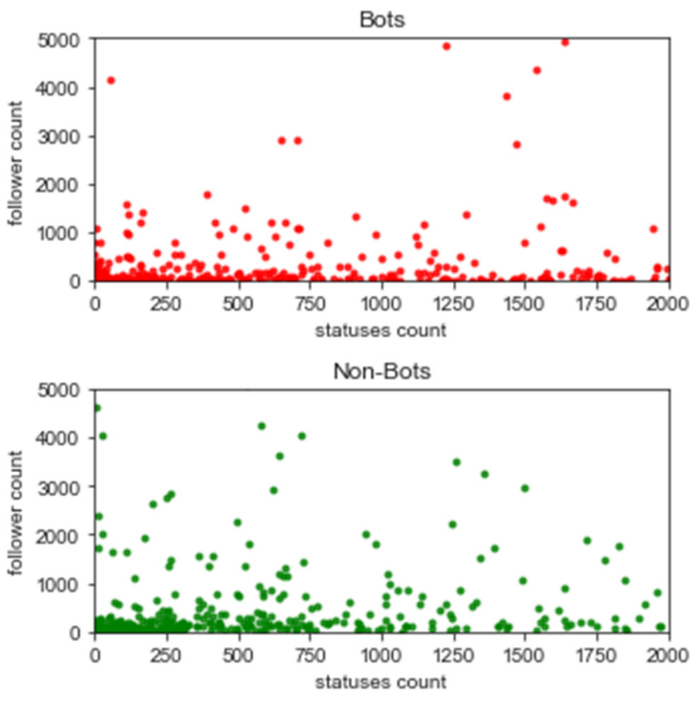

Figure 10. Follower count against statuses count for Bots and Non-bots.

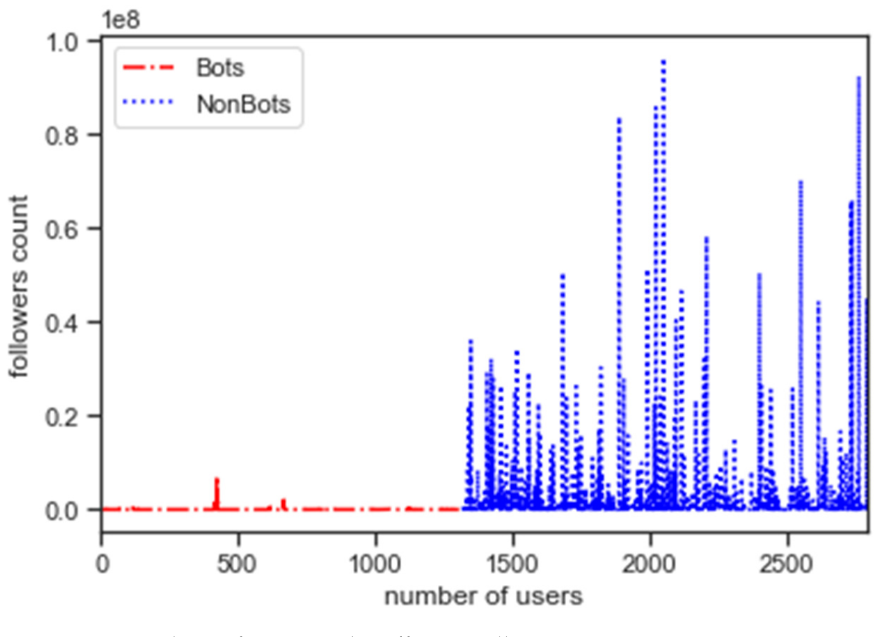

Figure 11. Numbers of Users with Different Follower Count.

Figures 13 and 14 here help us understand how the bots and non-bots are differentiated in terms of their follower and friend counts. As we can see in Figure 13, the number of users after a certain number has a high number of friends count, which is different to how bots operate.

Additionally, in Figure 14, the follower counts of some users, mainly non-bots, is exorbitantly high, while the bots have a similar follower count. This is another key characteristic of the bots and non-bots and their patterns in following and followers.

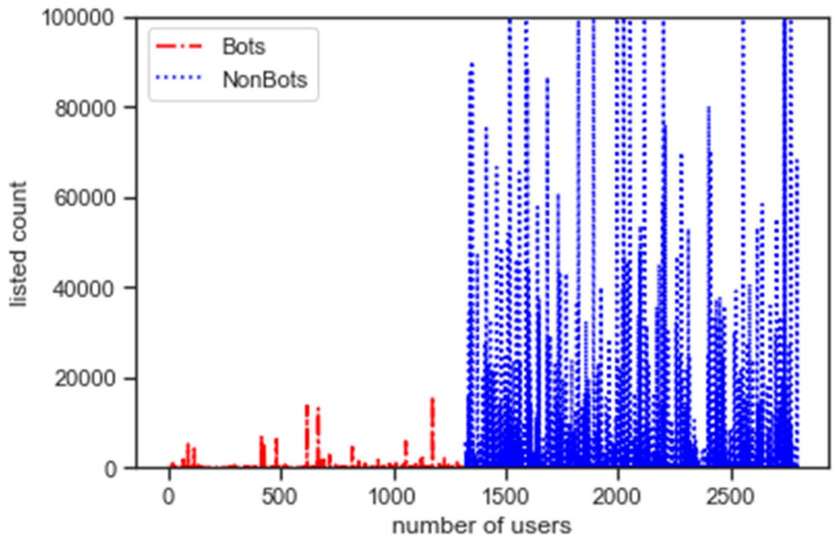

Figure 12. Numbers of Users with Variation of Listed Count.

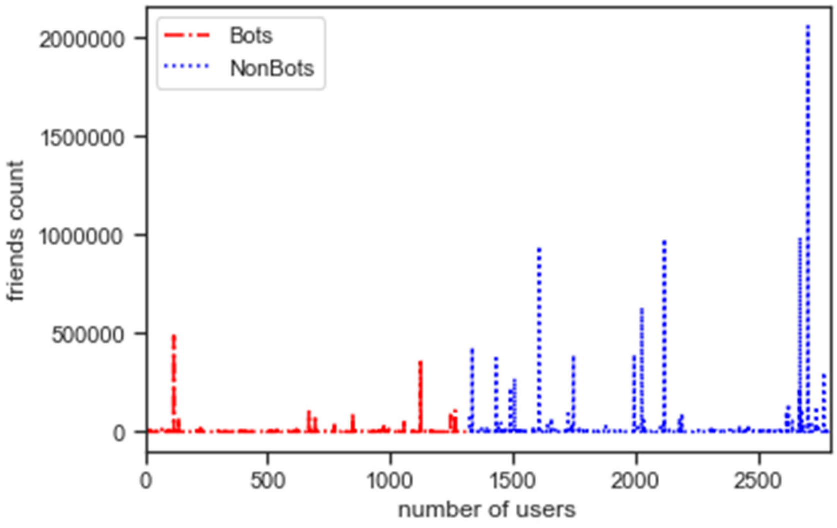

Figure 13. Numbers of Users against friends count.

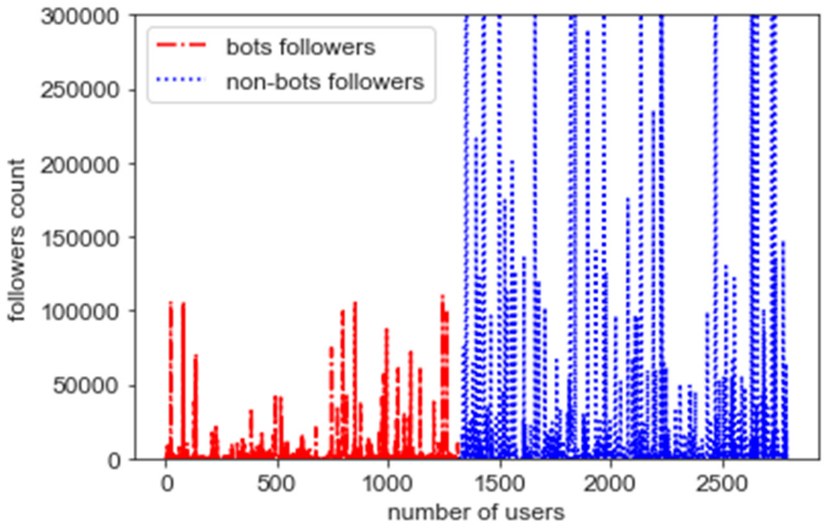

Figure 14. Numbers of Users against follower count.

5.4. Comparative Performance Evaluation

The various machine learning classifiers and standard APIs, including namely Botometer [24], BotSentinel [25], and TweetBotOrNot [26]. The machine learning models used for comparative analysis include Random Forest, Decision Tree, Bernoulli Naïve Bayes, Categorical Naïve Bayes, Support Vector Classifier and Multi-layer Perceptron (ANN) [27,28]. These classifiers are widely accepted as standard benchmarks for addressing binary classification problems involving large-scale multi-parameter datasets [29–31]. Decision Trees are well-known for their wide applicability in critical decision-making processes; additionally, Random Forest allows for great accuracy in handling large datasets, and takes less training time. SVM allows the segregation of the N-dimensional space into classes so that the data can be classified into correct category. All the variants of Naïve Bayes classifiers allow accurate binary and multi-class classifications. Additionally, we explored Multilayer Perceptron, which is used for binary classification through a feedforward mechanism through hidden layers, which can allow differentiation and the learning of more complex patterns. This forms the rationale behind selecting the benchmark classifiers for comparative study.

Apart from the existing benchmarks, we have also performed an accuracy analysis of our proposed *TweezBot*, with variation in the size of the total volume of data. Initially, our dataset contained 47.22% bots and 52.78% non-bot profiles. For experimentation, we have scaled our dataset over varying number of bots & non-bots while training and testing phases to check class imbalances. As apparent from Table 2, our proposed bot identification algorithm provides adequate performance in case of certain variation in the fraction of bots and non-bot profiles in the selective dataset. In addition, as the size of the dataset increases, the proposed *TweezBot* attains a maximum accuracy of 99.0049%, thereby justifying that our dataset has no class imbalance problems.

Table 2. Accuracy of Machine Learning Classifiers & Proposed *TweezBot*.

S.No.	Dataset Size	Fraction of Bot & Non-Bot		Training Accuracy	Testing Accuracy
1	25%	Bot	42.23%	0.979729	0.97009
2		Non-Bot	57.77%		
3	50%	Bot	52.34%	0.970833	0.985621
4		Non-Bot	47.66%		
5	75%	Bot	54.03%	0.984137	0.989867
6		Non-Bot	45.97%		
7	100%	Bot	47.22%	0.982749	0.990049
8		Non-Bot	52.78%		

The Receiver Operating Characteristics (ROC) curves for the aforementioned align with their training and testing AUC scores. The proposed bot classifier *TweezBot* provides comparatively better results on the training and the test data (Figure 15). On obtaining the ROC curve, where the Training AUC (i.e. area under curve) and Test AUC recorded for training and testing was 0.981675 and 0.996606, respectively, the training accuracy on the model is 0.9827498 while the testing accuracy for it is 0.99004975.

Similarly, in order to obtain a better understanding of how well the model performed when pitted against the classifiers and learning algorithms we used in the aforementioned sections, the comparative study of *TweezBot* with Random Forest, Decision Tree, Bernoulli Naïve Bayes, Categorical Naïve Bayes, Multi-layer perceptron (ANN) and Support Vector Machine has been shown for the ROC, for both the training set and test set in Figures 16 and 17.

The following Table 3 shows the comparative analysis of the *TweezBot* against two tree-based algorithms, two Naïve Bayes models, an SVM classifier, and a multi-level perceptron. Hence, the testing and training scores are exhaustively summarized for different types of machine leaning and deep learning models. Random forest, being an ensemble model, can be seen providing better results than decision tree. Additionally, the categorical version is considered suitable for performing the prediction of a target variable that is categorically distributed. However, our *TweezBot* has outperformed all its existing counterparts in terms of accuracy and AUROC scores while training and testing.

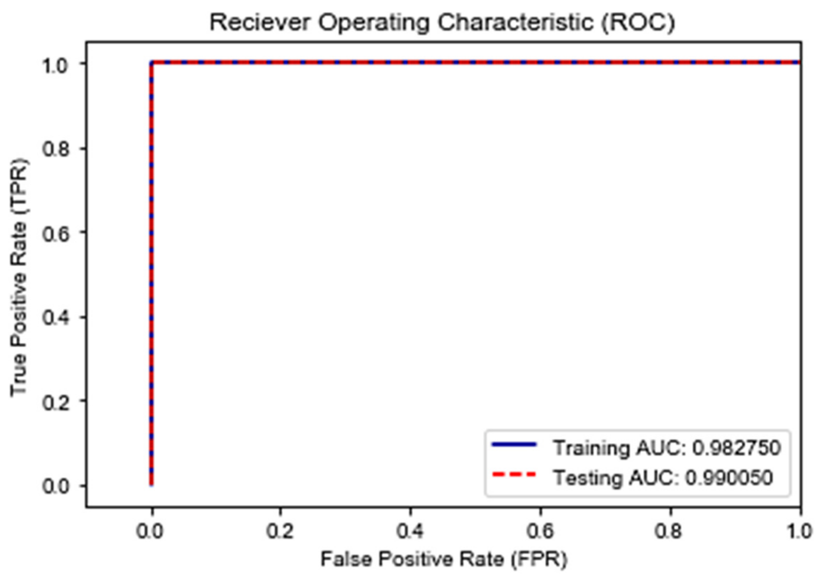

Figure 15. Proposed *TweezBot* AUC ROC Score.

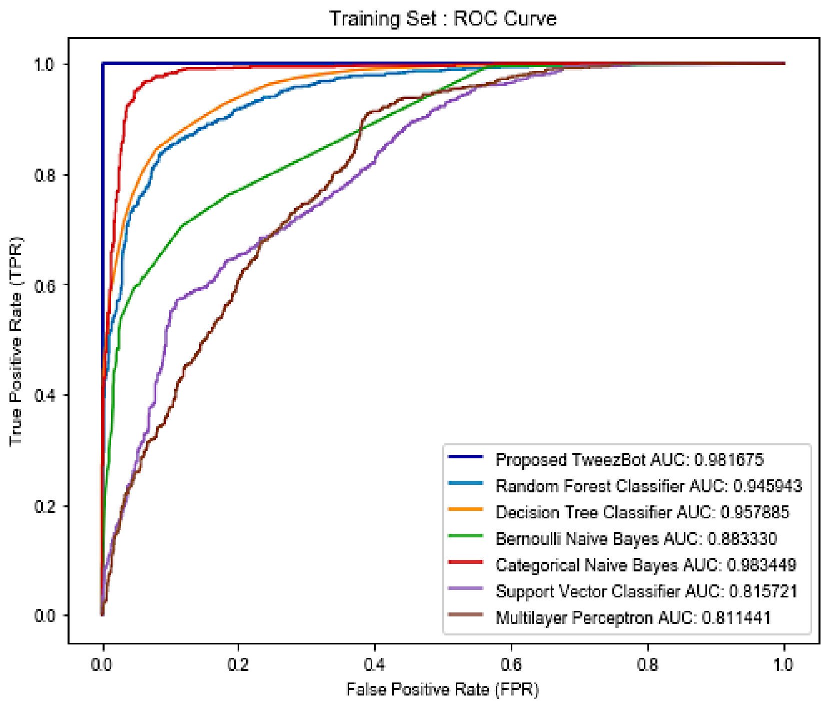

Figure 16. Comparative Analysis of Accuracy for Various Classifiers while Training.

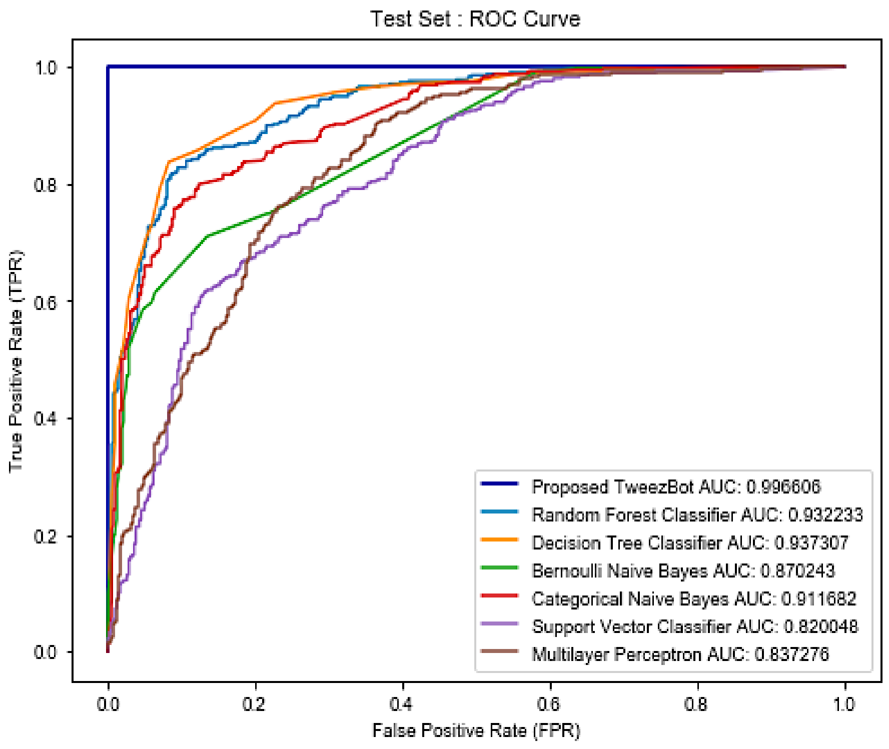

Figure 17. Comparative Analysis of Accuracy for Various Classifiers while Testing.

Table 3. Accuracy of Machine Learning Classifiers & Proposed *TweezBot*.

S.No.	Classifier	Training Accuracy [†]	Training AUROC *	Testing Accuracy [†]	Testing AUROC *
1.	Decision Tree	88.5539	0.957885	87.7381	0.937313
2.	Random Forest	87.2253	0.945545	86.0714	0.931360
3.	Bernoulli Naïve Bayes	80.0715	0.883330	78.8095	0.870243
4.	Categorical Naïve Bayes	95.0945	0.983449	81.7857	0.911682
5.	Support Vector Machine	87.2253	0.815790	86.0714	0.820042
6.	Multi-layer Perceptron	63.1068	0.811441	62.1428	0.837276
7.	Proposed *TweezBot*	98.2749	0.981675	99.0049	0.996606

[†] accuracy in terms of percentage; * area under ROC curve.

Even though accuracy is considered as the most intuitive performance measure, there are several other relevant parameters that are used to evaluate the performance of our model for better insight. Table 4 highlights the precision, recall, F1 score, Cohen-Kappa score, and ROC-AUC score, respectively. Precision denotes the ratio of correctly predicted observations to the total predicted positive observations, while recall is the ratio of correctly predicted observations to all observations in actual class. The F1 score is defined as the average of precision and recall, and sometimes is even more useful than accuracy when the data has an uneven class distribution. Furthermore, Cohen-Kappa is yet another eminent metric which is calculated on the basis of a confusion matrix. It expresses the level of the agreement between two annotators on a classification problem. There are two key aspects to consider when a machine learning model is evaluated, which includes reliability and validity. The amount of trust we have in the model capacity to deliver consistent outcomes in comparable scenarios is referred to as reliability. Validity, on the other hand, refers to the model accuracy on test data. Moreover, AUC (Area Under the Curve) or AUROC score is also considered to compute the area under the Receiver Operating Characteristics (ROC) curve. The significance of this score is that it allows us to compute the area under the curve, which leads to the summarizing of the curve information as a single value. The roc_auc_score function in the sklearn Python library was used to compute the score. This result differs from simple AUC because the latter denotes an abstract area under the curve, which is not specific to the ROC curve. AUC is more general in terms of curves, while AUROC is specifically the Area under the ROC curve. Therefore, the robustness is tested more appropriately using the roc_auc score. The results in Table 4 shows that our proposed model outperforms the benchmark classifiers. Among all the existing models, categorical Naïve Bayes proved to be a close competitor with a training recall and F1-score of 0.945708 and 0.9478827, respectively. However, our *TweezBot* model performs comparatively better with the highest training and testing precision of 0.999899 and 0.992443, respectively.

Table 4. Comparative Study of Classification Parameters of Benchmark Classifiers & Proposed *TweezBot*.

Parameters	**Decision Tree**	**Random Forest**	**Bernoulli Naïve Bayes**	**Categorical Naïve Bayes**	**SVM**	**Multilayer Perceptron Layer (ANN)**	**Proposed *TweezBot***
Training Precision	0.901891	0.895238	0.838157	0.9307036	0.895238	0.8181818	0.999899
Testing Precision	0.908854	0.891472	0.838526	0.7959641	0.891472	0.8510638	0.992443
Training Recall	0.844026	0.831858	0.704646	0.945708	0.831858	0.2588496	0.950369
Testing Recall	0.836930	0.827338	0.709832	0.8513189	0.827338	0.2877698	0.938574
Training F1_Score	0.872000	0.862385	0.765625	0.9478827	0.862385	0.3932773	0.995689
Testing F1 Score	0.871410	0.858209	0.768831	0.8227115	0.858209	0.4301075	0.989361
Training Cohen Kappa Score	0.768693	0.752054	0.594537	0.9015805	0.752054	0.2201130	0.988734
Testing Cohen Kappa Score	0.754611	0.728419	0.575707	0.6358753	0.728419	0.2392474	0.990504
Training ROC AUC Score	0.882602	0.874143	0.793918	0.9519898	0.874143	0.6047334	0.987283
Testing ROC AUC Score	0.877094	0.864023	0.787540	0.8180945	0.864023	0.6190622	0.995192

We have also performed another comparative study with some standard social media bot APIs, namely Botometer, BotSentinel, and TweetBotOrNot. For the purpose of validation, we have randomly extracted some screen names of Twitter users, and verified their automation behavioral outcomes in comparison to our proposed *TweezBot*. The scores are different as the criterion decided by the aforementioned API differs from ours, but all three of them show scores on which a profile can be deduced as bot or human behavior. The results in Table 5 highlight the automation outcomes in the form of bot and non-bot classification. It is evident from the tabular data that Botometer shows fairly good results and performs better than TweetBotOrNot. However, BotSentinel wrongly classifies most of the bots as non-bots with a low automation score. Overall, the proposed *TweezBot* outperforms all the existing bot identifiers by adequately classifying the Twitter profiles.

As is evident from our overall in-depth analysis and experimental outcomes, the proposed bot detection model, *TweezBot*, performs exceptionally well, and hence can be recommended for identifying bots on the Twitter social network.

Table 5. Comparative Evaluation of Automation Scores from Existing Bot-APIs & Proposed *TweezBot*.

S.No.	Actual Bot Status	Twitter Screen Name	Existing Social Media Bot Identifiers: Botometer Automation (Score) *	BotSentinel Automation (Score) †	TweetBotOrNot Automation (Score) †	Proposed *TweezBot* Automation Outcome ‡
1	Bot	@2181chrom_bot	Bot (4.6)	Non-Bot (0)	Bot (0.992)	1
2		@2LA1R_bot	Non-Bot (4.1)	Non-Bot (0)	Non-Bot (0.561)	1
3		@3pei_bot	Bot (4.7)	Non-Bot (0)	Bot (0.957)	1
4		@joe_ghinn_ITBot	Bot (4.3)	Non-Bot (0.13)	Bot (0.820)	1
5		@misheardly	Bot (4.6)	Bot (0.23)	Bot (0.873)	1
6		@stevehssb_ITBot	Bot (4.6)	Non-Bot (0.39)	Bot (0.764)	1
7	Non-Bot	@KellySchuberth	Non-Bot (0)	Non-Bot (0.03)	Non-Bot (0.320)	0
8		@KylieJenner	Non-Bot (0)	Non-Bot (0.03)	Non-Bot (0.254)	0
9		@Meg_Cramer	Non-Bot (0.1)	Non-Bot (0.01)	Non-Bot (0.439)	0
10		@mitchprothero	Non-Bot (0.1)	Non-Bot (0.1)	Bot (0.770)	0
11		@o_tilli_a	Non-Bot (0)	Non-Bot (0.02)	Non-Bot (0.191)	0
12		@jasonhall8675	Non-Bot (0.4)	Non-Bot (0.09)	Bot (0.683)	0

* on a scale of 5.0, † on a scale of 1.0; ‡ outcome 1 means bot, 0 means non-bot.

6. Conclusions

In recent times, social media usage has increased substantially and has led to the emergence of multiple social networking sites and applications. With the increase in traffic on social media, there has been a considerable increase in the existence of automated bots that often try to replicate and mimic user choices and behavior for commercial purposes. Such automations attempt to perform malicious activities, such as profile impersonation, hacking, and cyber stalking. To address such challenging issues, we have devised a novel AI-driven online bot detection model, namely *TweezBot*, to detect bot profiles on Twitter. For effective model training, we have used a set of highly correlated Twitter parameters so as to obtain better performance. This mainly includes the verification status imparted by Twitter, user description, extended profile, and listed count location, along with screening with a bag of words, which is often found being frequently used by online media bots. The dataset obtained from Kaggle has been used for the experimentation, and has been the basis of our research, with a training and test split ratio of 7:3. The proposed *TweezBot* has a testing accuracy of 99.0049% and a testing AUROC of 0.996606 on the dataset. Our *TweezBot* model achieved significant results over its existing counterparts. For instance, our model had a 12.84% improvement in performance over Decision Trees and a 15.02% improvement over Random Forest Classifiers. For Naïve Bayes classifiers, *TweezBot* had a good 25.62% improvement over Bernoulli Naïve Bayes and a 21.05% improvement over Categorical Naïve Bayes. For the Multilayer Perceptron, it 59.31%, the best escalation in testing accuracy, while it had a meagre improvement of 15.02% over Support Vector Machine. Therefore, based on our extensive study and comparative analysis with several existing classifiers, it can be concluded that *TweezBot* is highly capable of revealing bots on the social media platform Twitter. Regarding future research directions, the proposed model could be extended to support cross-platform identification of bots by tracking the automation in their tweeting behavior on other social networking platforms. This would provide valuable and holistic insight into better prospects. Additionally, cross-discipline AI techniques such as quantum-based learning could be used for improving bot prediction. In addition, graph-learning techniques may also be incorporated to further understanding of the networked structure within bot communities.

Author Contributions: Conceptualization, A.S. and R.S.; methodology, R.S.; software, A.S. and R.S.; validation, A.S., R.S., A.C.; formal analysis, A.S.; investigation, A.C.; resources, R.S.; data curation, A.S. and R.S.; writing—original draft preparation, R.S.; writing—review and editing, A.S.; visualization, A.S.; supervision, A.S.; project administration, A.S. and A.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interests.

References

1. Clark, E.M.; Williams, J.R.; Jones, C.A.; Galbraith, R.A.; Danforth, C.M.; Dodds, P.S. Sifting robotic from organic text: A natural language approach for detecting automation on Twitter. *J. Comput. Sci.* **2016**, *16*, 1–7. [CrossRef]
2. He, D.; Liu, X. Novel competitive information propagation macro mathematical model in online social network. *J. Comput. Sci.* **2020**, *41*, 101089. [CrossRef]
3. Jain, S.; Sinha, A. Identification of influential users on Twitter: A novel weighted correlated influence measure for COVID-19. In *Chaos, Solitons Fractals*; Elsevier: Amsterdam, The Netherlands, 2020; Volume 139, pp. 1–8. [CrossRef]
4. Kantepe, M.; Ganiz, M.C. Preprocessing framework for Twitter bot detection. In Proceedings of the 2017 International Conference on Computer Science and Engineering (UBMK), Antalya, Turkey, 5–8 October 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 630–634.
5. Anwar, A.; Yaqub, U. Bot detection in twitter landscape using unsupervised learning. In Proceedings of the 21st Annual International Conference on Digital Government Research, Seoul, Korea, 15–19 June 2020; pp. 329–330.
6. Buitinck, L.; Louppe, G.; Blondel, M.; Pedregosa, F.; Mueller, A.; Grisel, O.; Niculae, V.; Prettenhofer, P.; Gramfort, A.; Grobler, J.; et al. API design for machine learning software: Experiences from the scikit-learn project. *arXiv* **2013**, arXiv:1309.0238.
7. Chen, Z.; Subramanian, D. An unsupervised approach to detect spam campaigns that use botnets on twitter. *arXiv* **2018**, arXiv:1804.05232.
8. Beskow, D.M.; Carley, K.M. Bot-Match: Social Bot Detection with Recursive Nearest Neighbors Search. *arXiv* **2020**, arXiv:2007.07636.
9. Ilias, L.; Roussaki, I. Detecting malicious activity in Twitter using deep learning techniques. *Appl. Soft Comput.* **2021**, *107*, 107360. [CrossRef]
10. Miller, S.J. (Ed.) *Benford's Law*; Princeton University Press: Princeton, NJ, USA, 2015.
11. Kalameyets, M.; Levshun, D.; Soloviev, S.; Chechulin, A.; Kotenko, I. Social networks bot detection using Benford's law. In Proceedings of the 13th International Conference on Security of Information and Networks, Merkez, Turkey, 4–7 November 2020; pp. 1–8.
12. Barhate, S.; Mangla, R.; Panjwani, D.; Gatkal, S.; Kazi, F. Twitter bot detection and their influence in hashtag manipulation. In Proceedings of the 2020 IEEE 17th India Council International Conference (INDICON), New Delhi, India, 11–13 December 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–7.
13. Lundberg, J.; Nordqvist, J.; Laitinen, M. Towards a language independent Twitter bot detector. In Proceedings of the Digital Humanities in the Nordic Countries, Copenhagen, Denmark, 5–8 March 2019; pp. 308–319.
14. Wei, F.; Nguyen, U.T. Twitter Bot Detection Using Bidirectional Long Short-Term Memory Neural Networks and Word Embeddings. In Proceedings of the 2019 First IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA), Los Angeles, CA, USA, 12–14 December 2019; pp. 101–109.
15. Rauchfleisch, A.; Kaiser, J. The False positive problem of automatic bot detection in social science research. *PLoS ONE* **2020**, *15*, e0241045. [CrossRef] [PubMed]
16. Moosavi, A.; Rao, V.; Sandu, A. Machine learning based algorithms for uncertainty quantification in numerical weather prediction models. *J. Comput. Sci.* **2021**, *50*, 101295. [CrossRef]
17. McCallum, A.; Nigam, K. A comparison of event models for naive bayes text classification. In Proceedings of the AAAI-98 Workshop on Learning for Text Categorization, Menlo Park, CA, USA, 27 July 1998; Volume 752, pp. 41–48.
18. Metsis, V.; Androutsopoulos, I.; Paliouras, G. Spam filtering with naive bayes-which naive bayes? In Proceedings of the CEAS 2006—The Third Conference on Email and Anti-Spam, Mountain View, CA, USA, 27–28 July 2006; Volume 17, pp. 28–69.
19. Shrivastava, A.; Tripathy, A.K.; Dalal, P.K. A SVM-based classification approach for obsessive compulsive disorder by oxidative stress biomarkers. *J. Comput. Sci.* **2019**, *36*, 101023. [CrossRef]
20. Cinar, A.C. Training Feed-Forward Multi-Layer Perceptron Artificial Neural Networks with a Tree-Seed Algorithm. *Arab. J. Sci. Eng.* **2020**, *45*, 10915–10938. [CrossRef]
21. Samper-Escalante, L.; Loyola-González, O.; Monroy, R.; Medina-Pérez, M. Bot Datasets on Twitter: Analysis and Challenges. *Appl. Sci.* **2021**, *11*, 4105. [CrossRef]
22. Loyola-Gonzalez, O.; Monroy, R.; Rodríguez, J.; Lopez-Cuevas, A.; Mata-Sanchez, J.I. Contrast Pattern-Based Classification for Bot Detection on Twitter. *IEEE Access* **2019**, *7*, 45800–45817. [CrossRef]
23. Charvi Jain. 2018. Available online: Kaggle.com/charvijain27/training-data-2-csv-utfcsv (accessed on 20 January 2022).
24. Subrahmanian, V.S.; Azaria, A.; Durst, S.; Kagan, V.; Galstyan, A.; Lerman, K.; Zhu, L.; Ferrara, E.; Flammini, A.; Menczer, F. The DARPA Twitter Bot Challenge. *Computer* **2016**, *49*, 38–46. [CrossRef]

25. Bouzy, C. Towards a Language Independent Twitter Bot Detector. 2018. Available online: Botsentinel.com (accessed on 10 December 2021).
26. Kearney, M.W. TweetBotOrNot. 2018. Available online: github.com/mkearney/tweetbotornot (accessed on 1 December 2021).
27. Shuja, J.; Humayun, M.A.; Alasmary, W.; Sinky, H.; Alanazi, E.; Khan, M.K. Resource Efficient Geo-Textual Hierarchical Clustering Framework for Social IoT Applications. *IEEE Sens. J.* **2021**, *21*, 25114–25122. [CrossRef]
28. Aftab, H.; Shuja, J.; Alasmary, W.; Alanazi, E. Hybrid DBSCAN based Community Detection for Edge Caching in Social Media Applications. In Proceedings of the 2021 International Wireless Communications and Mobile Computing (IWCMC), Harbin, China, 28 June–2 July 2021; pp. 2038–2043. [CrossRef]
29. Fan, J.; Lee, J.; Lee, Y. A Transfer Learning Architecture Based on a Support Vector Machine for Histopathology Image Classification. *Appl. Sci.* **2021**, *11*, 6380. [CrossRef]
30. Nalbantov, G.; Bioch, J.C.; Groenen, P.J.F. Solving and Interpreting Binary Classification Problems in Marketing with SVMs. In *From Data and Information Analysis to Knowledge Engineering*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 566–573. [CrossRef]
31. Otani, N.; Otsubo, Y.; Koike, T.; Sugiyama, M. Binary classification with ambiguous training data. *Mach. Learn.* **2020**, *109*, 2369–2388. [CrossRef]