Computers & Security 91 (2020) 101715

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

A one-class classification approach for bot detection on Twitter

Jorge Rodríguez-Ruiz[a,*], Javier Israel Mata-Sánchez[b], Raúl Monroy[c], Octavio Loyola-González[d], Armando López-Cuevas[e]

[a] Tecnologico de Monterrey, School of Engineering and Sciences Av. Carlos Lazo No. 100, Álvaro Obregón, Ciudad de México, 01389, México
[b] Tecnologico de Monterrey, School of Engineering and Sciences Av. Eugenio Garza Sada 2501 Sur, Monterrey, N.L., 64849, México
[c] Tecnologico de Monterrey, School of Engineering and Sciences Carretera al Lago de Guadalupe Km. 3.5, Atizapán de Zaragoza, Estado de México, C.P. 52926, México
[d] Tecnologico de Monterrey, School of Engineering and Sciences Vía Atlixcáyotl No. 2301, Reserva Territorial Atlixcáyotl, Puebla, 72453, México
[e] Tecnologico de Monterrey, School of Engineering and Sciences Av. General Ramón Corona No. 2514, Zapopan, Jalisco, 45138, México

ARTICLE INFO

Article history:
Received 18 July 2019
Revised 20 December 2019
Accepted 3 January 2020
Available online 7 January 2020

Keywords:
Twitter bot detection
Supervised classification
One-class classifiers
Anomaly detection
Social networks

ABSTRACT

Twitter is a popular online social network with hundreds of millions of users, where n important part of the accounts in this social network are not humans. Approximately 48 million Twitter accounts are managed by automated programs called bots, which represents up to 15% of all accounts. Some bots have good purposes, such as automatically posting information about news and academic papers, and even to provide help during emergencies. Nevertheless, Twitter bots have also been used for malicious purposes, such as distributing malware or influencing the perception of the public about a topic. There are existing mechanisms that allow detecting bots on Twitter automatically; however, these mechanisms rely on examples of existing bots to discern them from legitimate accounts. As the bot landscape changes, with the bot creators using more sophisticated methods to avoid detection, new mechanisms for discerning between legitimate and bot accounts are needed. In this paper, we propose to use one-class classification to enhance Twitter bot detection, as this allows detecting novel bot accounts, and requires only from examples of legitimate accounts. Our experiment results show that our proposal can consistently detect different types of bots with a performance above 0.89 measured using AUC, without requiring previous information about them.

1. Introduction

Twitter is a micro-blogging social network that allows posting short messages of up to 280 characters called tweets. Twitter's users can interact with one another by replying to tweets, mentioning other users in their tweets, or reposting the message of another user; an action called retweeting. Furthermore, users can follow each other to keep up with the tweets posted by other users. For all registered users, the social platform allows accessing its services through a web page, mobile applications, and an application programming interface (API). The latter mode of access has allowed for an ecosystem of applications that enhances the user's experience at consuming and aggregating content. However, this has also allowed the creation of applications that control accounts and automate the posting of tweets (Chu et al., 2010).

Automation in Twitter can be performed at multiple degrees, depending on the use case. Fully automated accounts, called bots, can be used for retweeting exciting and relevant content for specific communities or providing an aggregation of tweets about a topic (Haustein et al., 2016; Lokot and Diakopoulos, 2016). Partly automated accounts, called cyborgs, allow humans to automate specific tasks, such as scanning for mentions about a brand to provide timely customer support, posting at certain times, and automatically respond to individual messages. The degree of automation has separated cyborgs into human-assisted bots, and bot-assisted humans (Chu et al., 2012), making the latter more difficult to differentiate from non-automated accounts (Varol et al., 2017). Terms and conditions provided by Twitter allow for automation and bots, but it prohibits the use of bots for malicious purposes (Twitter, 2017), such as disseminate spam (Fu et al., 2018) or malware (Ji et al., 2016), try to manipulate the perception about a political candidate to sway the electorate towards one side or another, and manipulating the information of an event to change the public's perception of it or obscuring the facts (Ferrara et al., 2016).

Detection of bots is paramount for the Twitter platform to suspend accounts that infringe on the terms and conditions. This de-

* Corresponding author.
E-mail addresses: jorger@tec.mx (J. Rodríguez-Ruiz), A00815374@itesm.mx (J.I. Mata-Sánchez), raulm@tec.mx (R. Monroy), octavioloyola@tec.mx (O. Loyola-González), acuevas@tec.mx (A. López-Cuevas).

https://doi.org/10.1016/j.cose.2020.101715
0167-4048/

tection uses different mechanisms, such as, accepting reports from users or flagging accounts that show suspicious behaviors such as posting the same tweet content while mentioning different users. However, bots are getting more sophisticated to achieve their goals better and avoid detection, being programmed to interact with others and mimic behaviors shown by legitimate accounts (Cresci et al., 2019). Furthermore, third parties can also benefit from bot detection, as it allows identifying and measuring the engagement an account has with real persons (Wald et al., 2013), or even detect command and control channels for malware botnets (Pantic and Husain, 2015). Given the volume of accounts and tweets, it is necessary to have automated methods for detecting bots, as even legitimate accounts can exhibit bot behaviors sometimes, making it difficult for a human to discern between them (Varol et al., 2017).

Automated bot detection works under the premise that the behavior expressed by the account of a human differs from one of a bot. These differences can be measured using representative features such as the statistical distribution of the words used in tweets, posting rate per day, number of followers, among others (Loyola-González et al., 2019). Using the extracted features, multi-class classifiers have been trained with examples of legitimate and bot accounts for classifying suspicious accounts. Most of the research in Twitter bot detection focuses on creating new feature representations and applying novel classifiers.

A limitation of using multi-class classifiers is the requirement of representative training examples from legitimate and bot accounts. Although the examples from the legitimate class are relatively easy to obtain, the collection of representative examples from bot accounts is not a trivial task, as these need extensive verifications to assure the account is not of a legitimate user (Yang et al., 2013). Furthermore, Cresci et al. (2017) have recently proposed that there exist different types of bots and that each type has its characteristic behavior. These new types of bots could be challenging to detect by supervised classifiers if the behaviors found in the training examples are too different. Taking into account that the type of bots will continue to evolve in the future, with bot creators modifying the behaviors to avoid detection (Cresci et al., 2019), a new strategy for automatic bot detection is needed.

In this paper, we use the one-class classification approach for bot detection on Twitter. One-class classifiers have the advantage of not requiring examples of anomalous behavior, which in this case is the behavior of bot accounts, to characterize the accounts and discern between bots and humans. Because the way one-class classifiers are trained, they can be used to detect new types of bots that show deviations with respect to legitimate account behaviors. We compare the performance of our proposal regarding popular multi-class and one-class classifiers reported on in the literature. Using our proposed approach, we show that one-class classifiers can consistently discern between legitimate and bot accounts with an AUC value above 0.89, while other state-of-the-art classifiers cannot consistently discern between legitimate accounts and bots of different types than the ones used in training the classifiers.

The remainder of the paper is organized as follows. In Section 2, we give an overview of related research regarding bot detection in social networks. Section 3 outlines our proposal of using one-class classification for bot detection on Twitter. Section 4 contains a discussion of our experimental methodology and our experimental setup. In Section 5, we provide an in-depth discussion about the obtained results. Finally, in Section 6, we provide the conclusions of this paper as well as further work.

2. Related work

Bot detection on Twitter is based on the premise that accounts managed by a human show a different behavior regarding accounts managed by bots; regardless of the machine's level of automation or sophistication. Several feature representations have been proposed for obtaining an accurate bot detection model (Cresci et al., 2017; Davis et al., 2016; Loyola-González et al., 2019; Martinez-Romo and Araujo, 2013). Such features can be divided depending on the source of information such as the content, account, or inter-account interactions.

Content: Tweets can contain up to 280 characters, which can be used to post text, mention other users, post URLs, and images. By parsing the text and analyzing the other elements, telltale signs of a bot can be found. One of the earliest features used was the number of URLs posted, the numbers of unique URLs, and the number of mentions to other users per tweet (Ahmed and Abulaish, 2013; Gilani et al., 2017; Lee et al., 2010), because bots were commonly used to spam links to malicious pages using the same message but mentioning different users. The text can also be analyzed to characterize bot behavior, as the language and phrase construction may differ between bots and legitimate users (Davis et al., 2016; Varol et al., 2017). The construction of the phrases and words can also be used to separate between bots and legitimate users, as the language used could be different (Davis et al., 2016) or even enhance the chances of people interacting with the bots (Wald et al., 2013).

Sentiment: Further analysis of the text of the tweet can be done to extract the sentiment, and measure how the poster of the tweet feels about the topic at hand. While these characteristics are related to the content, they require further analysis and do not allow for direct extraction (Loyola-González et al., 2019). Dickerson et al. (2014) found that bots accounts express more consistent sentiments for a topic, as they are commonly used to support or oppose a topic. Sentiment features can be used to analyze individual tweets (Ferrara et al., 2016; Loyola-González et al., 2019), or complete accounts depending on the average sentiment expressed (Dickerson et al., 2014).

Account Information: These features are obtained from the account used to post a tweet, and have the advantage of not suffering significant changes over time, as the information does not change with every tweet. For this reason, these features can be measured with less frequency. Features, such as the account age, if there was a biography or description of the account, and if the account had an image could be used to detect bots, as these accounts could be empty and be relatively new (Wald et al., 2013). However, bot accounts sometimes stay dormant until needed, and are filled with false information to avoid detection (Cresci et al., 2017). One of the most useful features of this type is the friends to followers ratio as legitimate accounts who follow other humans should have a similar number of friends and followers, as human relationships tend to be reciprocal (Chu et al., 2012). While friends to follower ratio is a widely used feature (Davis et al., 2016; Lee et al., 2010; Loyola-González et al., 2019), it can be subverted as sophisticated bots unfollow friends that do not reciprocate or follow other bots that are guaranteed to follow back (Chu et al., 2012).

Account Usage: These features measure how the user or bot uses the account, and how it interacts with the Twitter service. Bots can have distinct posting patterns than a legitimate user, like posting at regular intervals or posting in volumes that would be impossible for humans (Chu et al., 2012). Furthermore, bots generally access the Twitter platform via the API or other ways that allow automated

programs to post, while legitimate users tend to use the web or mobile interfaces (Chu et al., 2012). Account usage features can also obtain single metrics from aggregating content ones, such as analyzing the difference between messages, as bots could post the same message to different users (Lee et al., 2010).

Social Network: These features measure the interactions between different accounts, as a single account may not express suspicious behavior, but when analyzed in group bots, could be found. Examples of these features can be a measurement of the difference in text or topic of accounts, as groups of bots tend to post about the same topic (Keller et al., 2017); the difference in posting times of messages, as bots tend to retweet the same message in short time spans; or the sequence of actions as bots may perform the same actions in the same order while legitimate users do not (Cresci et al., 2017).

Using features of these different categories, the bot detection task has been approached using supervised and unsupervised classification. In this section, we discuss the most prominent mechanisms that have been used for bot detection in Twitter, considering the feature representation it has made use of and the reported performance.

Regarding performance, papers commonly report their results using three measurements, derived from the number of the bot and legitimate accounts correctly and incorrectly classified. The first measurement is accuracy, which is the percentage of the correctly classified bot and legitimate accounts from all the accounts on the testing datasets. A classifier with high accuracy makes fewer errors when detecting bots or legitimate accounts. The second measurement is the F1 score (Morstatter et al., 2016), which is the harmonic mean of the percentage of correctly classified bot accounts from all the accounts classified as a bot, known as precision; and the percentage of correctly predicted bot accounts of all the bot accounts in the testing dataset, known as recall. Precision allows measuring how many accounts would be marked as a bot when they are not, while recall measures the percentage of bots found from all. F1 score is useful as it provides a measurement of how good a classifier is to find bots correctly. However, some classifiers can be adjusted to provide better bot detection or decrease the number of legitimate accounts incorrectly found as bots. To see the full operating range of a classifier, the Receiver Operating Characteristics (ROC) curve is used, which plots the recall against the percentage of incorrectly classified accounts as bots from all the legitimate accounts in the testing dataset, known as False Positive Rate (FPR). ROC curves, for this problem, show the relationship between correctly classified bots and incorrectly classified legitimate accounts at different configurations of the classifier. To provide a single numerical measure of the performance, the Area Under the Curve (AUC) (Huang and Ling, 2005) is measured. AUC is the third used measurement when testing classifiers to detect bots. It has also been the case that some authors (Cresci et al., 2016; Loyola-González et al., 2019) provide the Matthews correlation coefficient (MCC) (Boughorbel et al., 2017) as a complementary metric, which is a correlation coefficient between the observed and predicted classifications. The most used metrics are F1 and AUC for checking different operating values of the classifiers (Cresci et al., 2017; Loyola-González et al., 2019; Miller et al., 2014; Yang et al., 2013). Furthermore, these two measurements are considered balanced measurements and are robust against problems where there are fewer examples of one class or the other (Japkowicz and Shah, 2011). Since class imbalance is common in the Twitter bot detection problem, as examples of legitimate accounts are easier to obtain, these measurements provide a better measurement of the performance.

2.1. Supervised approaches

Lee et al. (2010) proposed one of the earliest methods for Twitter bot detection. They created a classifier called Decorate, which uses content, tweet account, and account usage features, on a dataset collected by them. This method achieved an F1 value of 0.88. Similarly, Wang (2010) approaches the problem as Lee et al., but obtains the features only from the last 20 tweets in the account, and uses a naïve Bayes classifier, obtaining an F1 value of 0.91.

In a later study, Ahmed and Abulaish (2013) used the Weka data mining tool (Hall et al., 2009a) to test the performance of three classifiers: naïve Bayes, J48, and Jrip. By restricting each classifier to different feature subsets, they found that account usage features are more discriminative than tweet content ones. Regarding the performance of the classifiers, Jrip attained the best average performance, achieving a recall of 0.987.

Research with the focus on improving the performance obtained, have exploited more account usage features. Chu et al. (2012) claim to have achieved 96% of recall, using Random Forest (Breiman, 2001) as the classifier, when including usage features such as how the client logs into the Twitter platform or the client regularity of tweet posting. Yang et al. (2013) improved the F1 score previously obtained by Decorate and Bayesian network classifiers by using features of type tweet usage and social network. They obtained an F1 equal to 0.9 using Random Forest, while the previously mentioned naïve Bayes and Decorate methods obtained an F1 of 0.88 and 0.83, respectively.

In order to optimize the amount of data needed to be gathered, Wang et al. (2015) focuses on the detection of spam tweets by relying only on content features. In their experiments, two different datasets were used, but the best results were obtained using the Social Honeypot Dataset created by Lee et al. (2011). Lee's dataset consists of 22,223 spammers and 19,276 legitimate users, along with their most recent tweets. Random Forest was the best performing algorithm, capable of detecting bots with an F1 score of 0.946.

Gilani et al. (2017) adopted a different methodology, which included the multimedia elements in the tweets to the account usage features. This author separated the accounts into different datasets, depending on the number of followers. By using a Random Forest classifier, he achieved an F1 close or equal to 1.0 in the best case, i.e., bots accounts with more than 10 million followers. In these experiments, it was also found that using their representation it is more difficult to detect a bot among accounts with less than one thousand followers. In such cases, Gilani et al. method obtained an F1 of 0.84.

Cresci et al. (2016) proposes a DNA-like analysis of the tweet usage account. They hypothesized that not only are the actions that allow discerning if an account is a bot but the sequence of the actions as well. Cresci et al. assign a letter to each tweet of an account, depending on the types of tweets shared. Next, a DNA fingerprint of an account is obtained by creating a string with all the assigned letters of each tweet, appearing in chronological order. To test the approach, they calculate the length of the longest common substrings of the bot and legitimate accounts, considering bots those accounts with a common substring with all the others of the same class higher than a threshold. With this method, Cresci et al. obtain an F1 of 0.97.

Loyola-González et al. (2019) emphasize the need for understandable classification models and propose to approach Twitter bot detection using contrast-pattern based classifiers. Because such models have high explanatory power, they might help experts in different scenarios. For example, they can support decision making, or be used to forewarn an account holder about suspicious activity. The second contribution of Loyola et al. research is the

introduction of a new feature model, which extends an already existing model with features out of Twitter account usage and tweet content sentiment analysis. The proposed feature representation provided the lowest standard deviation, among all the tested classifiers. Regarding classification algorithms, the top-ranked (using AUC score as criteria) (LCmine+Hell)+Filt+PBC4cip, and (LCmine+Hell)+PBC4cip obtained an average AUC of 0.9976.

Currently, state of the art for Twitter bot detection is the Botometer tool, proposed by Yang et al. (2019). Botometer uses a Random Forest classifier (Breiman, 2001) over 1200 features from all the categories. Their results are reported using bots of different types, obtaining a score of up to 1.0 AUC, when classifying political bots. Overall, this method achieves a cross-validation performance of 0.97 AUC.

2.2. Unsupervised approaches

Most of the approaches for Twitter Bot detection rely on supervised classifiers. However, unsupervised classification methods have also been used to detect groups of related bots, and have even been tested on supervised datasets to obtain measurements of performance.

Ahmed and Abulaish (2013) proposed to find groups of suspected spam campaign accounts using the Markov clustering algorithm (Van Dongen, 2008). To find groups of spammers, first, they created an undirected fully connected graph where each node is a suspected spam account. This graph contains the weight of the links, which is the similarity between accounts using tweet usage features; then, the clustering algorithm is applied. On the downside, while they claim that campaigns were successfully detected, they do not give any performance measure.

Miller et al. (2014) uses a clustering approach to find groups based on features of either tweet account or account usage. Since account usage varies over time, Miller et al. use DenStream (Wan et al., 2009) and StreamKMmeansCC (Ackermann et al., 2012), which are techniques adapted for data streams. The authors claim to have obtained an F1 of 0.88 using this approach.

Chavoshi et al. (2016) analyzed the behavior of Twitter users and observed that it is highly unlikely that several users retweet a message in less than 20 seconds after it is posted, or that they post a message with a related trending topic in the same second. To show this, they use social network features together with a lag-sensitive hashing technique and a warping-invariant correlation measure to organize the accounts in clusters of abnormally correlated accounts. The results showed that, when the accounts are used to post for the same trending topic in a short time frame, they find groups of bots for Twitter with a precision of 0.94.

2.3. Summary of existing approaches

Most of the approaches for Twitter bot detection rely on the use of supervised classifiers, with Random Forest being the one that is currently most used, as can be seen in Table 1, and has helped achieve an AUC of 1.0 for specific types of bots (Yang et al., 2019). Furthermore, while all features have been useful to the correct classification of legitimate and bot accounts, the features that are more discriminative are the account usage ones (Ahmed and Abulaish, 2013). A summary of the performance, type, and features used by each approach are given in Table 1.

While Twitter bot detection proposed mechanisms have been successful, all the existing approaches, require labeled examples of bot accounts in order to extract the characteristic behavior. Furthermore, recent studies have shown that the bot landscape is not homogeneous, and not all bots perform the same type of actions, follow the same objectives, or show the same behavior (Cresci et al., 2017). Moreover, some bots are programmed to emulate the

Table 1
Previous works comparison.

Author	Performance	Method	Relevant Details	Approach
Lee et al. (2010)	F1 = 0.88	Decorate classifier	Use content, tweet account, and account usage features.	Supervised
Wang (2010)	F1 = 0.91	Naive Bayes	Similar approach to Lee et al., but using only last 20 tweets of each account.	Supervised
Ahmed and Abulaish (2013)	Recall = 0.987	Jrip	Features explored are interactions-driven, tweets-driven and URL-driven.	Supervised
Chu et al. (2012)	Recall = 0.96	Random Forest	Includes usage features, such as how the client logs in and regularity of tweet posting	Supervised
Yang et al. (2013)	F1 = 0.9	Random Forest	Uses features of type tweet usage and social network	Supervised
Wang et al. (2015)	F1 = 0.94	Random Forest	Content and sentiment of the tweet; information and account use.	Supervised
Gilani et al. (2017)	F1 = 1.0	Random Forest	Separates the accounts according to the number of followers. F1 = 1 for accounts with number of followers $\geq$ 10 Millions	Supervised
Gilani et al. (2017)	F1 = 0.84	Random Forest	For accounts with number of followers $<$ 1,000.	Supervised
Cresci et al. (2016)	F1 = 0.97	DNA-like analysis (Longest Common Substring)	Implements a DNA-like analysis of the tweet usage account	Supervised
Loyola-González et al. (2019)	AUC = 0.99	Contrast-patterns based classifiers	Uses contrast-patterns based classifiers along with features out of Twitter account usage and tweet content sentiment analysis.	Supervised
Yang et al. (2019)	AUC = 1	Botometer (Random Forest-based)	Uses over 1200 features from all the categories. Trains with different types of bots. Best AUC performance was obtained when classifying political bots.	Supervised
Miller et al. (2014)	F1 = 0.88	DBScan y K-means++	Uses a clustering approach to find groups based on features of either tweet account or account usage.	Unsupervised
Chavoshi et al. (2016)	Precision = 0.94	Lag-sensitive hashing technique and a warping-invariant correlation measure	This method considers cross-correlating user activities and requires no labeled data.	Unsupervised

behavior shown in real accounts, including the social interaction aspect. These social bots are even hard to discern from legitimate user accounts for human verifiers, as shown by Cresci et al. (2017). Given that the strategies to avoid detection might change even further, Twitter bot detection methods that do not rely on existing bots are necessary, as the typical behavior extracted from these accounts might become obsolete.

3. One-class classification for Twitter bot detection

One-class classification is a branch of supervised classification, where the training set only contains examples of one class. Unlike binary classification, where the classifiers require examples of two classes to find differences between them, one-class classifiers only require examples of the normal class in order to learn representations of it and identify if a new example belongs to that class or not. One-class classification is commonly used for problems where the objective is finding deviations from normal or expected behavior, called anomaly detection. In this type of problem, it is common that there are a plethora of examples showing normal behavior, but the occurrence of anomalies is rare, or the type of deviations from normality can change from time to time. Furthermore, in supervised classification, a new instance will always be labeled as one of the pre-defined categories, even if it belongs to another, unknown category. This is detrimental when little information is provided about specific classes, which is the case in tasks as fraud detection (Di Martino et al., 2012), fault detection in an engine (Bartkowiak, 2011), personal-risk detection (Rodríguez et al., 2016), and malware detection (Jha et al., 2001). Thus, one-class classifiers are the best suited to solve problems where the examples of abnormalities are scarce or nonexistent.

When approaching Twitter bot detection as a supervised classification problem, it must be taken into account that behaviors of bots are not homogeneous. For example, there are types of bots that try to emulate social behaviors to avoid detection and gain followers (Cresci et al., 2017; Wald et al., 2013). Considering this, we propose to approach Twitter bot detection as a one-class classification problem instead of a binary classification one. Using a one-class classifier to obtain a model of the behavior observed in the legitimate Twitter accounts has the advantage that it would be able to detect bots that deviate from the legitimate account behavior regardless of the bot type, giving it the capability of detecting entirely new types of bots.

To test our proposal, we use a dataset that contains different types of bots. We hypothesize that one-class classifiers will be able to distinguish the bots and the legitimate users consistently, while the binary classifiers will only be able to distinguish between bots and legitimate users when, at the testing phase, the bots are of the same type as the ones used when training the classifier. In the next section, we discuss the experimental setup to test our hypothesis.

4. Experimental setup

Our experimental setup contains four parts: the datasets used to test the hypothesis that one-class classifiers can achieve a good performance regardless of the type of bot to detect; the features extracted to observe representative behavior of legitimate and bot accounts; the classifiers used to measure the performance, and the experiments to be performed to test our hypothesis. In this section, we describe each of these components.

4.1. Dataset

In a recent study, Cresci et al. (2017) provided quantitative evidence that a paradigm-shift exists in the design of Twitter bots. Along with their results, a public dataset that contains different types of bots, collected by them and other researchers, was released. This dataset, summarized in Table 2, provides information about 9386 accounts, where 3474 of them are genuine, and the rest are divided between 4 different types of bots. Cresci et al.'s dataset provides the individual tweets posted, each one with metadata such as posting time, the method used for posting the tweet, how many times have been retweeted, among others. Furthermore, the dataset provides also information from the account, such as the number of friends and followers, Twitter handle, if the image is the default one, and time since the account was created, among others. Each of the accounts underwent manual verification to check if the type was correct.

The information from the legitimate accounts was obtained by doing a random sample of accounts and interacting with them by asking a question in natural language. From the sampled accounts, 3474 responded to the message and were marked as genuine. The accounts that did not respond to the message, either because they were a bot, or because the user did not want to response were not considered for the dataset. From these accounts, at the time of Cresci et al. (2017) study, 3.3% have been deleted by the user, and 0.1% were suspended for infringing the terms and conditions.

Yang et al. (2013) provided the traditional bot accounts. In their collection process, Yang et al. used the Twitter public timeline, which showed all the tweets that were posted to the service. Using accounts from this feed as a seed, they recursively sampled accounts via their social connections, collecting up to the latest 40 tweets. Using this method, Yang et al. obtained information from 485,721 accounts and nearly 14 million tweets. From these accounts and tweets, the URLs posted were checked using Google Safe Browsing, which served as a blacklist of malicious or spam URLs. Furthermore, they created a honeypot that visited the URLs and marked it as malicious if it could lead to the creation or modification of files, processes, or the registry. By using this automated method, and a manual revision for the potentially suspicious accounts, they found 2060 accounts that at least 10% of their tweets contained malicious URLs. Cresci et al. provide in their dataset 1000, which at the time of their study, 2.5% have been deleted and 8.6% suspended. While these accounts infringed on the terms and conditions of Twitter, less than 12% were eliminated, which highlights the necessity of having detection systems for bots, especially malicious ones.

Cresci et al. collected the social bot accounts by monitoring accounts on Twitter and finding relationships between the accounts that pointed to a single bot operator. These accounts had an account picture stolen from other accounts, fake biographies, and

Table 2
Databases used in Cresci et al. (2016).

Database	Description	Accounts	Tweets	Year
Genuine accounts	Verified accounts that are human-operated	3474	8,377,522	2011
Social spambots #1	Retweeters of an Italian political candidate	991	1,610,176	2012
Social spambots #2	Spammers of paid apps for mobile devices	3457	428,542	2014
Social spambots #3	Spammers of products on sale at Amazon.com	464	1,418,626	2011
Traditional spambots #1	Training set of malware spammers used by Yang et al. (2013)	1000	1,418,626	2009

interacted with other accounts by having friends and followers that are legitimate users. The tweets of these accounts generally post images, phrases, and videos to simulate being a legitimate account, but did not interactions initiated by other users. The first type of social bot accounts was used to enhance the Twitter reach of a political candidate in Rome's 2014 mayor election, allowing him to modify engagement metrics. These bots retweeted the candidate tweets within minutes of being posted. The second type of social bot accounts were found because they used the common hashtag #TALNTS, which is an application to connect users with an artist. These bots promoted the application, and from time to time mentioned a user to encourage him to purchase the app. The third type of social bot posted URLs to Amazon, inviting users to buy promoted products on the page. While some of the accounts from these types, as reported by Cresci et al., were suspended (4.6% for the first type, 3.8% for the second, and 0% for the third), these suspended percentage is less than half of the accounts that post traditional spam or malicious URLs. This could indicate that it is harder to detect bot accounts that incorporate social components to mimic legitimate user accounts.

4.2. Selected classifiers

To test our hypothesis, we have to measure the performance of binary classifiers, and then one-class classifiers when classifying the same testing dataset. The classifier's performance depends on the underlying assumptions, features' type, and the statistical distribution of each feature. Because of these reasons, it is important to test our hypothesis using a set of diverse classifiers, where each of them has different internal working mechanisms.

4.2.1. Binary classifiers

Ten of the most widely used binary classifiers were chosen to perform the bot detection task. Their performance will be compared since they have shown to be robust in several domains. These classifiers are Bayes Network (Ben-Gal, 2008), J48 (Bhargava et al., 2013; Quinlan, 1993), Random Forest (Ho, 1995), Adaboost (Freund and Schapire, 1997; Hastie et al., 2009), Bagging (Breiman, 1996), KNN (Altman, 1992), Logistic Regression (Yu et al., 2011), MLP (Hinton, 1989; Rumelhart et al., 1988), Naïve Bayes (Zhang, 2004), and SVM (CJC, 1998). All of them were trained with a particular type of bot (i.e., social 1), and tested with the same type of bot. Additionally, they were tested with the other types of bots not used in training. The implementations of this classifiers are the ones provided in the Weka Data Mining tool (Hall et al., 2009b).

The selected classifiers that are used in our experiments are representative of the ones used in the literature, as can be seen in Table 1. For our comparisons, we do not include Lee et al. (2010) Decorate, as it is not widely used and other methods have surpassed its performance; and Cresci et al. (2016) DNA analysis, as the implementation is not publicly available and it has been recently shown that this method is easily circumvented (Cresci et al., 2019). We also do not include Loyola-González et al. (2019) contrast pattern method, as this is devised to perform classification per tweet instead of per account, which falls outside of the scope of this work. Lastly, we do not use the commercial Botometer tool (Yang et al., 2019), as it requires from 1200 features that it automatically extracts from a given Twitter username from an active account, and not all the accounts in the dataset are currently active, as they have been found to infringe the Twitter's terms and conditions. Furthermore, since the commercial tool is already trained, it would not allow for a fair comparison of the performance in the experiments. Nevertheless, we do include Random Forest in our selected classifiers, as it is the basis for Botometer, as mentioned by Davis et al. (2016).

4.2.2. One-class classifiers

In our selection of one-class classifiers, we consider the ones belonging to the state-of-the-art in anomaly detection. For this reason, we select Bagging-TPMiner (Medina-Pérez et al., 2017) and Bagging-RandomMiner (Camiña et al., 2018), which have been used for masquerade-detection, and OCKRA (Rodríguez et al., 2016), a classifier used for personal-risk detection. These three classifiers are ensemble-based (Opitz and Maclin, 1999; Polikar, 2006; Rokach, 2010). For comparison, we also include one-class versions of Support Vector Machines and naïve Bayes. Since these classifiers are less known than binary classifiers, we briefly discuss how they work.

- **BTPM** *Bagging-TPMiner* (Medina-Pérez et al., 2017) is a one-class classifier ensemble consisting of multiple instances of the same base classifier, TPMiner. TPMiner finds Typical Patterns in the training phase via clustering, where the Typical Patterns are the centroids of a cluster. In this method, the set of centroids maximizes the similarity of objects in a cluster and the dissimilarity of objects outside that cluster. BTPM is made robust by making an ensemble where attribute bagging is used. Then, at the testing phase, a threshold is used to indicate if the distance of a new object, to a typical pattern, represents a normal or anomalous example.
- **BRM** *Bagging-RandomMiner* (Camiña et al., 2018) Bagging-RandomMiner is a one-class classifier ensemble consisting of multiple instances of the same base classifier, RandomMiner. This classifier follows the same notion of Bagging-TPMiner of finding representative objects, but instead of an exhaustive search of typical patterns, it selects objects randomly in the normal class.
- **OCKRA** *One-Class K-means with Randomly-projected features Algorithm* (Rodríguez et al., 2016), OCKRA for short, is an ensemble of one-class classifiers, each of which is based on k-means++ (Arthur and Vassilvitskii, 2007). This algorithm uses the centroids found by the clustering algorithm as representative objects and classifies new examples depending on the distance to the found centroids. A threshold is used as a criterion to determine if the example is normal or anomalous.
- **ocSVM** *One-Class Support Vector Machine* (DMJ, 2002), uses the notion of binary SVM where a hyperplane that maximizes the distance between classes is found. However, since there is only one class in the training set for ocSVM, the hyperplane maximizes the distance between the objects of that class and the origin.
- **Naïve Bayes** *Naïve Bayes* (DMJ, 2002), works using the Bayes Theorem to calculate the probability of an example belonging to a class, and assuming that all events are independent. Unlike binary naïve Bayes, where the highest probability indicates the class, in one-class naïve Bayes, only the probability of belonging to the normal class is considered, and a threshold is used to determine if the probability is considered normal or anomalous.

4.3. Feature selection

Cresci et al. (2017) provides, for each account, 40 features of the account information type. Furthermore, for each tweet, it provides a vector of 25 content features, including the full text of the tweet. However, these features are a combination of text, nominal, and numeric features, while the one-class classifiers selected only work with numerical features. For this reason, our feature vector includes only numerical features that are representative of the behavior expressed in a Twitter account.

Table 3
Feature vector obtained per each Twitter account.

Characteristics	Description	Type
retweets	Ratio between retweet count and tweet count.	Account Usage
replies	Ratio between reply count.	Account Usage
favoriteC	Ratio between favorited tweet and tweet count.	Account Usage
hashtag	Ratio between hashtag count and tweet count.	Account Usage
url	Ratio between url count and tweet count.	Account Usage
mentions	Ratio between mention count and tweet count.	Account Usage
intertime	Average seconds between postings.	Account Usage
ffratio	Friends-to-followers ratio.	Account Information
favorites	Number of tweets favorited in this account.	Account Usage
listed	Number of listed tweets in the account.	Account Information
uniqueHashtags	Ratio between unique hashtag count and tweet count.	Account Usage
uniqueMentions	Ratio between unique mention count and tweet count.	Account Usage
uniqueURL	Ratio between unique urls count and tweet count.	Account Usage

Obtaining a large number of features for each account often leads to better classification performance. Nevertheless, feature extraction is a time-consuming process, and not all the features provide the same amount of useful information. Based on the previous works, we chose to make a feature vector per account, which contains only the features that have been proven to allow for better representation of legitimate account, and that also provide better discerning capabilities from bot accounts (Ahmed and Abulaish, 2013). Our selected features are of the account usage and account information types. These features were obtained from the original information of the accounts provided by Cresci et al., alongside the aggregation of statistical information of each user's corresponding tweets. Table 3 describes the 13 features obtained for each account.

4.4. Experiments

In our experiments, we are working based on the hypothesis of Cresci et al. (2017) that there are different types of bots, each with its characteristic behavior. Cresci et al. propose that bots that emulate the social behavior of users are more difficult to detect than traditional bots. However, they do not test if different types of bots affect the performance of the classifiers. If the behaviors are different from each other, then a classifier trained to detect bots of one type will not be able to detect, with the same performance, bots of another type. Working under this hypothesis, we have constructed, for each bot type, a training dataset that contains 90% of the legitimate account examples and 90% of the bot account examples. The testing dataset consists of the left 10% of legitimate and bot account examples. This allows us to test the classifiers with accounts that it has not previously classified. To avoid the variations due using different legitimate account examples, the training datasets for each type of bot contain the same 90% legitimate account examples, and the examples that change are only those of the bot accounts. For statistical validity, we repeat this process ten times with different 10% of bots and legitimate account examples, resulting in the 10-fold cross-validation procedure.

The first experiment aims to test the hypothesis of Cresci et al. (2017) and measure the performance of binary classifiers when discerning between bots and legitimate accounts. For this purpose, we have trained the selected binary classifiers using the training datasets containing each one of the bot's types, then tested each trained model with all the different testing datasets. For example, if a classifier is trained using a dataset that contains 90% of legitimate accounts and 90% of the social1 bot accounts examples, then we have tested such trained classifier using 4 different testing sets, where each one contains the same 10% of the legitimate account examples, but has 10% of social1 bots, social2 bots, social3 bots, and traditional bots account examples, respectively. This allows us to verify how a classifier performs when discerning between bots of a different type it was trained on.

The second experiment tests how the classifiers perform when trained with multiple types of bots, and then are used to detect one type of bot that was not used in the training set. With this, we are emulating the behavior of state-of-the-art classifiers that obtain examples of different types of bots to train their classifiers, such as botometer (Yang et al., 2013). For this experiment, each classifier is trained with 90% of the legitimate examples, and 90% of three types of bots. The testing set contains 10% of the legitimate account examples, and 10% of the other type of bot.

For the third experiment, we aim to test the performance of one-class classifiers when trained with legitimate user account examples and classifying the different types of bots. For this purpose, we follow the same methodology as experiment one, using the same datasets. Since in this experiment we are using one-class classifiers, at the training phase, the classifiers ignore the 90% of the bot account examples and only use the 90% of the legitimate user account examples to construct the model. Each trained classifier is evaluated using the same testing datasets used in the first experiment.

5. Experimental results and discussion

Most one-class classifiers require a threshold to determine if the example is considered as normal or anomalous. The threshold to select is application- and implementation-dependent, based on the needs of the user, such as increasing the detection power of anomalies, or reducing the number of false positives. For this reason, we use ROC curves to study the performance of the classifiers on different thresholds. We use the AUC of the ROC curve, as it summarizes the performance of a classifier into a single metric (Batista et al., 2004).

The AUC performance of a classifier is equivalent to the probability that the classifier give a higher similarity to the training class to an anomalous example than a normal one (Bekkar et al., 2013; Fawcett, 2006). Furthermore, the AUC value can describe performance correctly in the case of datasets, where there are more objects of one class than another (Provost and Fawcett, 1999). Since we have fewer examples of each bot type than,legitimate account examples in our experiments, we face a class imbalance problem. and because of that, choosing AUC as the performance measure is appropriate. In our results, we present the average AUC obtained by each classifier, when tested against a validation set containing genuine and bot activity, along with the AUC standard deviations.

To analyze the statistical differences between the performance of the classifiers, we use the Friedman test with a Bergmann-Hommel and Schaffer post-hocs (Garcia and Herrera, 2008). For a graphical representation of the statistical results, we use *Critical Difference (CD) diagrams* (Demšar, 2006) to show the results, such as the one shown in Fig. 1, because they present the rank measured by the Friedman test of the measurement, in this case, the

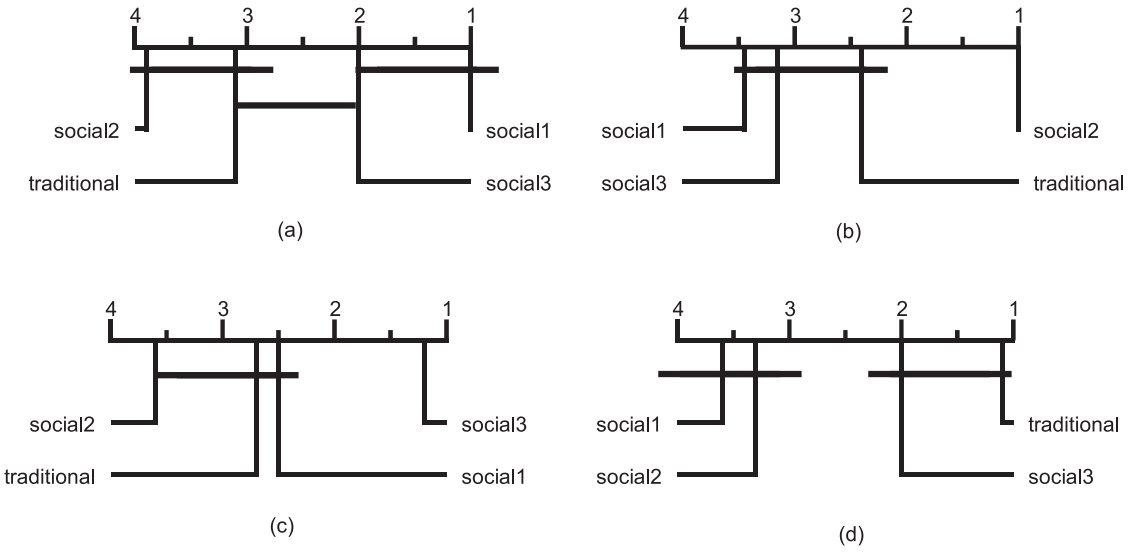

Fig. 1. CD diagram of the performance when the classifiers are trained with one type of bot and tested with another.

AUC value, when testing. CD diagrams also show both the quantity and significance of the difference. Note that the top-ranked algorithm appears rightmost, and a thick line joins statistically similar classification results for a set of classifiers.

5.1. Experiments using binary classifiers

In this experiment set, we validate that different types of bots have different behaviors, as tested in Cresci et al. (2017). For this purpose, we trained classifiers with examples of legitimate accounts and one type of bots, and test them using different types of bots. If different classifiers cannot correctly discern between bots and legitimate users when tested against activity of bots other than from those they were trained with, then the classifier learned bot behaviors that are not useful to detect another type of bot. To test this idea, we propose the following hypothesis that will be tested with the Friedman Test and the Schaffer post-hoc.

H_0: Binary classifiers trained with genuine activity and activity of one from three types of bots exhibit similar performance when tested against a dataset containing genuine activity and activity of a bot of a different type, for all combinations of bot types.

H_1: It is not the case that H_0 holds

In Table 4, we can observe that all the classifiers trained with examples of legitimate and social1 bot accounts show the best AUC when performing classification of bots of the same type. The mean AUC of the classifiers for social1 bot accounts is 0.97, with a standard deviation of 0.007. In Fig. 1(a), we can observe that there is no statistically significant difference between the classifiers testing social1 and social3. This could indicate that social1 and social3 types of bots have elements in common that allow categorizing an account as a bot, even if they are from different types, as mentioned by Cresci et al. (2017). However, for the social2 and traditional datasets, the CD diagram shows that there is enough evidence to reject the null hypothesis.

In Table 5, we show the results obtained after training the classifiers with examples of legitimate and social2 bot accounts. When used to discern between legitimate accounts and those of the same bot type, the performance of the classifier is excellent, as a mean AUC of 0.991 and a standard deviation of 0.003. However, when testing the datasets with different types of bots, the results are less than 0.6 on average. Furthermore, in Fig. 1(b), we can observe that there is enough evidence to reject the null hypothesis, and the differences in performance are statistically significant.

Table 6 shows the results obtained by using the classifiers trained with examples of legitimate and social3 bot accounts. As happens in the other cases, the best classification results are when the classifiers are used to discern between legitimate accounts and bot accounts of the same type, with a mean for social3 bots of 0.963 and a stable behavior from all classifiers, as shown with the standard deviation value of 0.009. Furthermore, the CD diagram in

Table 4
Results of testing different types of bots when classifiers have used a training dataset containing legitimate users and bots of type social1 account examples. In black, the results of testing the same type of bot as in the training set.

Classifier	social1	social2	social3	traditional	Mean	Standard Deviation
Bayes Network	**0.972**	0.5	0.959	0.76	0.798	0.221
J48	**0.977**	0.517	0.957	0.733	0.796	0.216
Random Forest	**0.98**	0.496	0.96	0.719	0.789	0.228
Adaboost	**0.979**	0.497	0.948	0.772	0.799	0.221
Bagging	**0.975**	0.497	0.954	0.78	0.802	0.221
KNN	**0.974**	0.491	0.584	0.566	0.654	0.217
Logistic Regression	**0.969**	0.74	0.957	0.588	0.814	0.184
MLP	**0.97**	0.488	0.714	0.526	0.675	0.22
Naïve Bayes	**0.958**	0.477	0.937	0.523	0.724	0.259
SVM	**0.96**	0.482	0.951	0.658	0.763	0.234
Mean	0.971	0.519	0.892	0.663		
Standard Deviation	0.007	0.079	0.132	0.104		

Table 5
Results of testing different types of bots when classifiers have used a training dataset containing legitimate users and bots of type social2 account examples. In black, the results of testing the same type of bot as in the training set.

Classifier	social1	social2	social3	traditional	Mean	Standard Deviation
Bayes Network	0.5	**0.989**	0.514	0.502	0.626	0.242
J48	0.68	**0.992**	0.596	0.835	0.776	0.175
Random Forest	0.502	**0.996**	0.508	0.511	0.629	0.245
Adaboost	0.502	**0.993**	0.518	0.589	0.651	0.231
Bagging	0.633	**0.993**	0.535	0.673	0.709	0.198
KNN	0.493	**0.99**	0.494	0.499	0.619	0.247
Logistic Regression	0.511	**0.987**	0.907	0.604	0.752	0.23
MLP	0.519	**0.989**	0.513	0.529	0.638	0.234
Naive Bayes	0.498	**0.99**	0.498	0.5	0.622	0.246
SVM	0.511	**0.986**	0.509	0.505	0.628	0.239
Mean	0.535	0.991	0.559	0.575		
Standard Deviation	0.065	0.003	0.126	0.109		

Table 6
Results of testing different types of bots when classifiers have used a training dataset containing legitimate users and bots of type social3 account examples. In black, the results of testing the same type of bot as in the training set.

Classifier	social1	social2	social3	traditional	Mean	Standard Deviation
Bayes Network	0.59	0.599	**0.963**	0.774	0.732	0.176
J48	0.712	0.553	**0.97**	0.774	0.752	0.172
Random Forest	0.909	0.497	**0.966**	0.78	0.788	0.209
Adaboost	0.88	0.496	**0.968**	0.787	0.783	0.205
Bagging	0.722	0.497	**0.968**	0.756	0.736	0.193
KNN	0.91	0.491	**0.959**	0.621	0.745	0.226
Logistic Regression	0.971	0.984	**0.97**	0.741	**0.917**	0.117
MLP	0.887	0.496	**0.966**	0.695	0.761	0.21
Naive Bayes	0.812	0.497	**0.956**	0.643	0.727	0.2
SVM	0.494	0.484	**0.942**	0.891	0.703	0.248
Mean	0.789	0.559	0.963	0.746		
Standard Deviation	0.155	0.153	0.009	0.078		

Table 7
Results of testing different types of bots when classifiers have used a training dataset containing legitimate users and bots of type traditional account examples. In black, the results of testing the same type of bot as in the training set.

Classifier	social1	social2	social3	traditional	Mean	Standard Deviation
Bayes Network	0.503	0.501	0.942	**0.995**	0.735	0.27
J48	0.5	0.502	0.503	**1**	0.626	0.249
Random Forest	0.501	0.502	0.504	**0.999**	0.627	0.248
Adaboost	0.5	0.5	0.5	**0.999**	0.625	0.25
Bagging	0.5	0.502	0.503	**0.999**	0.626	0.249
KNN	0.489	0.48	0.928	**0.952**	0.712	0.263
Logistic Regression	0.501	0.502	0.511	**0.999**	0.628	0.247
MLP	0.536	0.561	0.933	**0.963**	0.748	0.231
Naive Bayes	0.523	0.729	0.929	**0.985**	0.792	0.21
SVM	0.489	0.481	0.939	**0.922**	0.708	0.257
Mean	0.504	0.526	0.719	0.981		
Standard Deviation	0.015	0.075	0.227	0.027		

Fig. 1(c) shows that the difference in performance is statistically significant when testing all the other datasets. In Table 4, we could see that the behavior of accounts of type social1 could be used to identify bots of type social3 correctly for some classifiers, such as KNN or Logistic Regression. However, while classifiers trained to discern bots of type social3 can also help to discern bots of type social1, the AUC obtained is generally lower, shown with a mean of 0.789.

Lastly, in Table 7, we show the results obtained when using classifiers trained with examples of legitimate and traditional bot accounts. Traditional bot accounts are easy to discern from legitimate accounts when the same type of bot is used for training, as shown by the AUC values, obtaining a mean AUC for all classifiers of 0.981, with a standard deviation of 0.027, which is an order of magnitude higher than for the other types of bots. However, the CD diagram in Fig. 1(d) shows that when testing a dataset with social3 bots with a classifier trained with examples of traditional bots, there is not enough evidence to reject the null hypothesis. Table 7 shows that some classifiers can correctly discern between legitimate users and social3 bots when using traditional bots to train, but that does not happen for other types of bots.

5.2. Experiments using multi-class classifiers

In this set of experiments, we trained the classifiers with 3 out of 4 types of bots and tested them using a testing dataset containing genuine behavior and activity of bots of the other, remaining type. This simulates more closely how classifiers can be used to detect bots, with a more comprehensive set of examples that encompasses multiple behaviors which could be

Table 8
Results of testing different types of bots when classifiers have used a training dataset containing legitimate users and bots of different types. In black, the best AUC obtained by a classifier for testing an specific type of bot.

Classifier	social1	social2	social3	traditional	Mean	Standard Deviation
Bayes Network	0.578	0.539	0.957	0.774	0.712	0.193
J48	0.874	0.88	0.887	0.902	0.886	0.012
Random Forest	0.935	0.496	0.957	0.826	0.804	0.213
Adaboost	0.922	0.515	0.942	0.869	0.812	0.2
Bagging	0.928	0.499	0.954	**0.933**	0.829	0.22
KNN	0.893	0.537	0.941	0.609	0.745	0.202
Logistic Regression	**0.956**	**0.973**	**0.957**	0.726	**0.903**	0.118
MLP	0.858	0.717	0.953	0.729	0.814	0.112
Naïve Bayes	0.81	0.609	0.943	0.625	0.747	0.16
SVM	0.533	0.528	0.937	0.873	0.718	0.218
Mean	0.829	0.629	0.943	0.786		
Standard Deviation	0.15	0.171	0.021	0.113		

used to discern between legitimate and bot accounts correctly. To determine if there is a difference between the performance of the different classifiers when using them to discern between legitimate accounts and a bot that was not used in the training set, we use the following hypothesis.

H_0: Multi-class classifiers trained with genuine activity and activity of three types of bots exhibit similar performance when tested against a dataset containing genuine activity and activity of a bot of a different type, for all combinations of bot types.
H_1: It is not the case that H_0 holds.

In Table 8, we show the AUC obtained by each classifier. Using the mean AUC obtained by each classifier for each bot type, we can observe that classifying social3 is easier for the different classifiers, as it allows for an average AUC of 0.943. However, this value is 0.02 lower than when using a classifier that only detects bots of type social 3, as shown in Table 6. The worst average AUC is when the previously unseen bot is of type social2, as the average AUC is 0.629; this corroborates the results in Table 5, where the examples of this type of bot were not enough to discern the bot behavior of other types.

In Fig. 2, we can observe that for the classifiers trained with social1, social2, and traditional, there is enough evidence to reject the null hypothesis when testing social3, which means that the difference in performance is statistically significant. However, for the other cases, there is not enough evidence to reject the null hypothesis, meaning the contrary.

Regarding the performance per classifier, we can observe, in Table 8, that the classifier with the highest AUC for this set of experiments is Logistic Regression, with an average AUC of 0.903. The classifier with the second-best performance is J48, with an average AUC of 0.886. J48 has a more stable performance than other classifiers, regardless of the unseen type of bot in the testing dataset. This can be observed by its standard deviation of 0.012, which is one order of magnitude lower than the other classifiers.

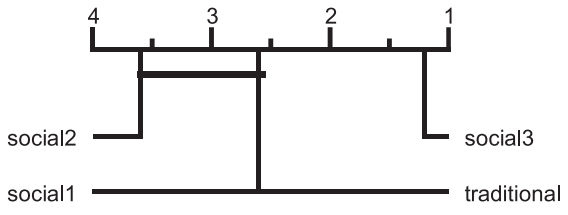

Fig. 2. CD diagram of the performance when the classifiers are trained with three types of bots and tested with another.

Except for J48, all the other classifiers have at least one case where the performance is drastically lower.

5.3. Experiments using one-class classifiers

One-class classifiers only require examples of legitimate user accounts in their training phase. Since the training dataset does not need to be modified, we present the results of the selected one-class classifiers when trained with legitimate accounts, and are tested with datasets that contain legitimate and bot account examples of only one type.

In Table 9, we show the results obtained by using one-class classifiers to classify query examples by using testing datasets containing different bot examples. We can observe in bold numbers the highest AUC obtained by a classifier for each type of bot. OCKRA shows better performance when detecting traditional bots, Naïve Bayes when detecting bots of type social1, and Bagging-TPminer shows a higher performance when detecting bots of type social2 and social3. To determine if the differences in performance are statistically significant, we use the following hypothesis.

H_0: One-class classifiers trained with genuine activity exhibit a similar performance when tested against a dataset containing genuine and bot activity.
H_1: It is not the case that H_0 holds.

In Fig. 3, we show the CD diagram of the results of comparing the different one-class classifiers. In this case, Bagging-TPminer is the best-ranked algorithm being rightmost, followed by OCKRA. However, the Friedman (1937) test indicates that there is not enough evidence to reject the null hypothesis, making the differences in the classifiers' performance without statistical significance.

5.3.1. Comparison

One advantage of our experimental setup is that the results of the one-class and multi-class classifiers can be directly compared, as the training and testing datasets are the same, with the difference that one-class classifiers ignore the examples of bot accounts in the training stage. For our comparison, we selected the results of the two best-ranked classifiers when trained with one and two classes, and use the AUC values obtained to compare the classifiers.

In Table 10, we contrast the performance of the classifiers by using the AUC scores. In the rows where the training and testing datasets contain bots from the same type, we mark the first column with bold letters. For each row, we also mark with bold the AUC value of the classifier that obtained better performance than the others. The first thing to notice is that, when the training set contains the same type of bot account examples as the testing

Table 9
Results of testing different types of bots when classifiers have used a training dataset containing legitimate users examples. In black, the highest performance attained by a classifier per testing dataset.

Classifier	social1	social2	social3	traditional	Mean	Standard Deviation
OCKRA	0.843	0.906	0.924	**0.916**	0.897	0.037
BTPM	**0.904**	**0.952**	**0.934**	0.893	**0.921**	0.027
BRM	0.724	0.807	0.791	0.892	0.803	0.069
ocSVM	0.871	0.893	0.851	0.898	0.878	0.021
Naive Bayes	0.995	0.791	0.849	0.883	0.88	0.086
Mean	0.868	0.87	0.87	0.896		
Standard Deviation	0.099	0.069	0.059	0.012		

Table 10
Comparison of binary and one-class classifiers performance. In black, the highest performance attained by a classifier per testing dataset.

Bots in Training/Testing	J48	LogReg	OCKRA	BTPM
s1/s1	**0.977**	0.969	0.843	0.904
s1/s2	0.517	0.740	0.906	**0.952**
s1/s3	**0.957**	**0.957**	0.924	0.934
s1/t	0.733	0.588	**0.916**	0.893
s2/s1	0.680	0.511	0.843	**0.904**
s2/s2	**0.992**	0.987	0.906	0.952
s2/s3	0.596	0.907	0.924	**0.934**
s2/t	0.835	0.604	**0.916**	0.893
s3/s1	0.712	**0.971**	0.843	0.904
s3/s2	0.553	**0.984**	0.906	0.952
s3/s3	**0.970**	**0.970**	0.924	0.934
s3/t	0.774	0.741	**0.916**	0.893
t/s1	0.500	0.501	0.843	**0.904**
t/s2	0.502	0.502	0.906	**0.952**
t/s3	0.503	0.511	0.924	**0.934**
t/t	**1.000**	0.999	0.916	0.893
Mean	0.738	0.778	0.897	**0.921**
Standard deviation	0.196	0.210	0.033	**0.024**

set, binary classifiers perform better than the others, with J48 obtaining the best average performance of the four compared classifiers in the 4 cases, out of 16, where the training and testing bots are of the same type. However, when the training and testing datasets contain different types of bots, one-class classifiers generally perform better than binary classifiers, with OCKRA obtaining better performance than the others in 3 of 16 cases, and BTPM in 6 of 16 cases.

Regarding the average performance of the classifier in all cases, we can observe that Bagging-TPMiner achieves the highest mean performance of all the classifiers shown in Table 10, with an average AUC value of 0.921, followed by OCKRA with an average AUC value of 0.897. The one-class classifiers are at least 0.1 higher than the better performing binary classifiers. Furthermore, the performance of one-class classifiers is more stable than binary classifiers, as the standard deviations using the one-class approach are lower than 0.04, while for binary classifiers are higher than 0.19. To test if the differences in the performance of the classifiers are statistically significant, we use the following hypothesis.

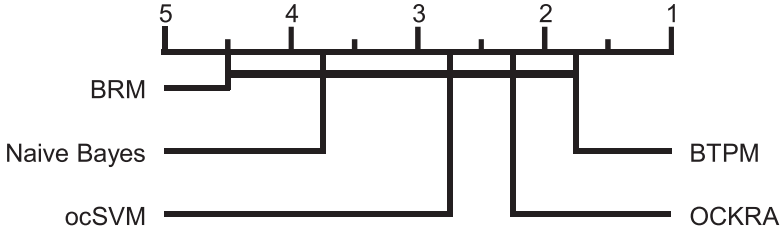

Fig. 3. CD diagram of one-class classifiers performance for Twitter bot detection.

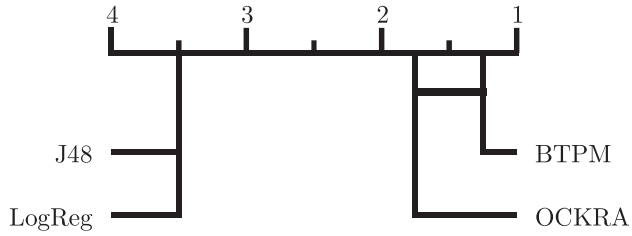

Fig. 4. CD diagram of classifiers performance for Twitter bot detection.

H_0: One-class classifiers and binary classifiers exhibit a similar performance when trained with examples of genuine activity, and one of three bot types for the binary classifiers, and are tested using a dataset containing genuine activity and activity from a bot of different type.

H_1: It is not the case that H_0 holds.

In Fig. 4, we can observe the CD diagram that compares classifiers when discerning between legitimate and bot accounts. This diagram shows that one-class classifiers are better ranked than binary classifiers. Furthermore, between binary and one-class classifiers, there is enough evidence to reject the null hypothesis, making the differences in performance statistically significant. This is because in 9 of 16 cases, the best performing classifier is either OCKRA or BTPM, and they achieve higher AUC values than their binary counterparts, with differences of even 0.4 in the AUC values. In the CD diagram, we can also observe that for our experiments, the binary classifiers do not have statistically significant differences between them, nor do the one-class classifiers.

5.4. Discussion

Twitter bots have been used for malicious purposes, such as distributing malware (Ji et al., 2016; Yang et al., 2013), disseminate fake news (Shao et al., 2018), falsely increasing the engagement metrics between accounts and artificially extending messages to change people perceptions (Castillo et al., 2019; Cresci et al., 2017), faking grassroots political movements (Ratkiewicz et al., 2011), and interfering with social movements (Chavoshi et al., 2016; Suárez-Serrato et al., 2016). Detecting bots is paramount to stop malicious activities, or at least reduce them, while also allowing people to be

better informed when making decisions. While the gold standard to determine if an account was a bot could be if the account were suspended (Morstatter et al., 2016), the Twitter platform can take time to detect and suspend accounts that infringe on the terms and conditions. For example, Cresci et al. (2017) found that less than 10% of 1000 accounts that included URLs recurrently to malicious sites were suspended, while Chavoshi et al. (2016) found that bots found using methods in the state-of-the-art had a suspension rate between 20% and 40%. Considering that bot accounts are not suspended efficiently and that bot detection could take extensive manual verifications, there is a need for automated bot detection.

Methods for automated bot detection in the state-of-the-art, rely on classifiers and information extracted from known bot accounts for training. However, there is evidence that bots are becoming more sophisticated, as their creators include elements that mimic real accounts, for example, adding social interaction capabilities (Cresci et al., 2017). Another concern is the prevalence of malicious cyborgs, accounts that combine bot and human behavior. Varol et al. (2017) showed that cyborgs that behave more closely to humans, are challenging to detect for automated bot detection mechanisms and human annotators alike. The attackers can use this knowledge to design bots that escape detection (Freitas et al., 2016). Given the importance of detecting new types of bots, in our first and second experiments, we test how different classifiers would perform when used to detect a bot of a different type than previously encountered ones.

From the first experiment with binary classifiers, we can observe that classifiers can detect bot accounts from a previously known type and with previously seen behaviors with high performance. However, when classifiers are used to detect bots of a different class than the used in training, the performance is significantly lower. Only in some cases, like with traditional and social3, and social1 and social3, classifiers can detect bots that belong to other classes, without a significant loss of performance. Furthermore, there are some types of bots, like the ones in the social2 dataset, that are not similar to other ones and would be better to avoid detection from automated methods. In the second experiment, where classifiers are trained with multiple types of bots, like state-of-the-art methods (Yang et al., 2013), there are bot types that can be detected with a high performance (social3), while others cannot (social2).

Classifiers trained with multiple bots have lower detection performances than the ones that focus on a single type of bot. Given the results obtained in our first and second experiments and the continuous sophistication of bots, we can observe that it is probable that new types of bots, with distinct behaviors than previously seen, will avoid being detected by current automated bot detection methods. This could result in successful amplification of messages, or a malware dissemination campaign being successful for an extended period.

Our proposal in this paper is to use one-class classifiers to detect bot accounts of an unknown type, training classifiers that do not rely on the previous bot examples but only in the behavior of legitimate accounts. In our experiments, we found that one-class classifiers have a higher performance than other types of classifiers when detecting bot accounts with behaviors not present in the training set. However, classifiers designed to detect a specific type of bot had a higher performance than one-class classifiers. Given these results, we can conclude that one-class classifiers can complement existing approaches, helping with the detection of new bots, while state-of-the-art classifiers are better when focused on previously known bot types.

An implementation of an automated bot detection system by using one-class and binary classifiers would need to include updating mechanisms that take into account changes in the legitimate behaviors, such as those that result by changes in the platform. For example, the increase of characters could result in different behaviors regarding mentions or URLs. To manage these changes, a meta-learning system could be used that incorporates scaffolding techniques from learning theory (Pratama et al., 2016), especially managing when to learn and how to learn. Such a system could then be updated regularly, and train new binary classifiers that are specialized in detecting single types of bots, that were initially found with the one-class classifiers, while selecting correctly the type of classifier and features that better characterize each type of bot accounts.

Cyborgs are another element that is needed to be considered in the implementation of a hybrid system. Varol et al. (2017) found that human-assisted bots are more likely to be detected by classifiers than bot-assisted humans. By using a hybrid system, it could happen that neither the one-class nor the binary classifiers would be able to detect a cyborg account, as it might not resemble other types of bots, and it could mimic the behavior of legitimate users well-enough that the one-class classifier marks the account as such. Thus, there would be a need for other components that allow discerning between legitimate users and cyborgs, such as using aggregated data from multiple accounts to check if the suspected cyborg behaves similar to others, which could indicate they belong to the same bot creator (Varol et al., 2017).

6. Conclusions and future work

Twitter bot detection using supervised classification relies on the hypothesis that features representative of the behavior of legitimate accounts can be extracted, and that this behavior is different from the one expressed by bot accounts. Recent studies propose that there are different types of bots accounts, where each group of bots shows different behaviors, which makes it harder to discern between them.

In this paper, we have corroborated that a supervised classification approach is a viable option for Twitter bot detection, as our results achieve a performance higher than 0.95 AUC when discerning between bots and legitimate user accounts. However, we have also shown that binary supervised classifiers cannot correctly detect novel or different bot accounts from the ones that were used to train the classifier, observing a loss of performance of almost 0.4 in the AUC measurement.

When the classifiers are trained with different types of bots, multi-class classifiers cannot for all cases correctly discern between bot and legitimate accounts, when the bot to be classified is from a different type than the ones used in training. To solve this problem, we propose to approach Twitter bot detection as an anomaly detection problem and use one-class classifiers. The results of our experiments showed that, in general, the best performing classifier for each type of bot achieved a score above 0.9 AUC, when trained only with examples of legitimate bot accounts.

Given our results, one-class classifiers could serve as an early-warning system, detecting anomalous patterns of account behavior, which could represent a new bot. The addition of a one-class classification approach to the current bot detection methods which rely on binary or multi-class classifiers would make a more robust system that could detect bots correctly with known behaviors, and also find possible suspect accounts with more certainty. However, as the gap of the behavior between normal and anomalous accounts closes due to the advances in IA, or other techniques used by the bot creators (Cresci et al., 2019), it is possible that no method would be entirely sufficient for detecting online bots.

As future work, the presented proposal could be extended to take advantage of the detection capacity of the one-class approach, and aim to characterize the groups of detected Twitter bots. This characterization is useful not only for broadening the protection offered to online users but to document new waves of bots and update binary classifiers, as these have performed better when

looking for a known type of bot. This could also be a useful method for detecting cyborgs, as the individual accounts might avoid detection by mimicking legitimate user behaviors. However, collectively, if the cyborgs have similar behaviors between them, it could mean a new group of semi-automated accounts.

A key aspect of one-class methods is the need for quality characterizations of the normal (genuine) class. Because of that, continued efforts need to be done to collect genuine user's information recurrently. Furthermore, bot accounts could be modified to mimic the behavior expressed in legitimate accounts to avoid detection, similar to presentation attacks in biological and behavioral biometric systems (González-Soler et al., 2017; Tolosana et al., 2019), or created by following behavior patterns generated using adversarial networks (Cresci et al., 2019). Since the bot creators leverage the knowledge of the normal class to avoid detection, it is necessary to test the resilience of the existing methods to these advanced attacks.

Since current cyborg detection requires specialized knowledge to determine if an account uses automation techniques (Castillo et al., 2019), another line of future work has to do with automating the detection of cyborgs and determining the amount of automation an account uses via behavior features. Furthermore, since cyborgs tend to be confused by detectors with accounts managed by multiple people, corporate accounts, or accounts where the owner tweets in different languages (Varol et al., 2017), a hybrid system could create multiple types of one-class classifiers, each for different types of legitimate accounts, to perform a better detection anomalies that could represent a bot or a cyborg.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the Google, Inc., under the APRU Project "AI for Everyone" led by Jiro Kokuryo (Keio, Japan) and Toby Walsh (UNSW, Australia). J.I. Mata-Sánchez is supported by a scholarship 885845 awarded by CONACyT.

References

Ackermann, M.R., Märtens, M., Raupach, C., Swierkot, K., Lammersen, C., Sohler, C., 2012. StreamKM++: a clustering algorithm for data streams. J. Exp. Alg. 1–2. 17:2.4:2, Available from: http://doi.acm.org/10.1145/2133803.21844504:2.30.

Ahmed, F., Abulaish, M., 2013. A generic statistical approach for spam detection in online social networks. Comput. Commun. 36 (10), 1120–1129. Available from: http://www.sciencedirect.com/science/article/pii/S0140366413001047.

Altman, N.S., 1992. An introduction to kernel and nearest-neighbor nonparametric regression. Am. Stat. 08 (46), 175–185.

Arthur, D., Vassilvitskii, S., 2007. K-means++: the advantages of careful seeding. In: Proceedings of the Eighteenth Annual ACM Symposium on Discrete Algorithms Society for Industrial and Applied Mathematics;, pp. 1027–1035.

Bartkowiak, A., 2011. Anomaly, novelty, one-class classification: acomprehensive introduction. Int. J. Comput. Inf.Syst. Ind. Manag. Appl. 12 (3).

Batista, G., Prati, R., Monard, M.C., 2004. A study of the behavior of several methods for balancing machine learning training data. SIGKDD Explor. 06 (6), 20–29.

Bekkar, M., Djema, H., Alitouche, T.A., 2013. Evaluation measures for models assessment over imbalanced data sets. J. Inf. Eng. Appl. 3 (01), 27–38.

Ben-Gal, I., 2008. Bayesian Networks. American Cancer Society. Available from: https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470061572.eqr089.

Bhargava, N., Sharma, G., Bhargava, R., Mathuria, M., 2013. Decision tree analysis on j48 algorithm for data mining. Proc. Int. J. Adv. Res. Comput. Sci. Softw. Eng. 3 (6).

Boughorbel, S., Jarray, F., El-Anbari, M., 2017. Optimal classifier for imbalanced data using Matthews correlation coefficient metric. PLoS ONE. 06;12:e0177678.

Breiman, L., 1996. Bagging predictors. Mach. Learn. 24 (2), 123–140. Available from: https://doi.org/10.1023/A:1018054314350.

Breiman, L., 2001. Random forests. Mach. Learn. 45 (1), 5–32. Available from: https://doi.org/10.1023/A:1010933404324.

Camiña, J.B., Medina-Pérez, M.A., Monroy, R., Loyola-González, O., LAP, V., LCG, G., 2018. Bagging-randomminer: a one-class classifier for file access-based masquerade detection. Mach. Vis. Appl.. Available from: https://doi.org/10.1007/s00138-018-0957-4.

Castillo, S., Allende-Cid, H., Palma, W., Alfaro, R., Ramos, H.S., C, G., et al., 2019. Detection of bots and cyborgs in Twitter: a study on the chilean presidential election in 2017. In: International Conference on Human-Computer Interaction Springer, pp. 311–323.

Chavoshi, N., Hamooni, H., Mueen, A., 2016. Identifying correlated bots in Twitter. In: Spiro, E., Ahn, Y.Y. (Eds.), Social Informatics. Springer International Publishing, Cham. 14–21.

Chu, Z., Gianvecchio, S., Wang, H., Jajodia, S., 2010. Who is tweeting on Twitter: human, bot, or cyborg? In: Proceedings of the 26th Annual Computer Security Applications Conference. ACSAC '10. New York, NY, USA: ACM, pp. 21–30. Available from: http://doi.acm.org/10.1145/1920261.1920265.

Chu, Z., Gianvecchio, S., Wang, H., Jajodia, S., 2012. Detecting automation of twitter accounts: are you a human, bot, or cyborg? IEEE Trans. Depend. Secure Comput. 9 (6), 811–824.

CJC, B., 1998. A tutorial on support vector machines for pattern recognition. Data Min. Knowl. Discov. 2 (2), 121–167. Available from: https://doi.org/10.1023/A:1009715923555.

Cresci, S., Di Pietro, R., Petrocchi, M., Spognardi, A., Tesconi, M., 2016. DNA-inspired online behavioral modeling and its application to spambot detection. IEEE Intell. Syst. 31 (5), 58–64.

Cresci, S., Di Pietro, R., Petrocchi, M., Spognardi, A., Tesconi, M., 2017. The paradigm-shift of social spambots: evidence, theories, and tools for the arms race. In: Proceedings of the 26th International Conference on World Wide Web Companion International World Wide Web Conferences Steering Committee;, pp. 963–972.

Cresci, S, Petrocchi, M, Spognardi, A, Tognazzi, S., 2019. Better Safe Than Sorry: An Adversarial Approach to Improve Social Bot Detection. In: Proceedings of the 10th ACM Conference on Web Science. WebSci'19. Association for Computing Machinery, New York, NY, USA, pp. 47–56. Available from: https://doi.org/10.1145/3292522.3326030.

Davis, C.A., Varol, O., Ferrara, E., Flammini, A., Menczer, F., 2016. Botornot: a system to evaluate social bots. In: Proceedings of the 25th International Conference Companion on World Wide Web;, pp. 273–274.

Demšar, J., 2006. Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. 7, 1–30.

Di Martino, M., Decia, F., Molinelli, J., Fernández, A., 2012. Improving electric fraud detection using class imbalance strategies. In: International Conference on Pattern Recognition Applications and Methods;, pp. 135–141.

Dickerson, J.P., Kagan, V., Subrahmanian, V., 2014. Using sentiment to detect bots on Twitter: are humans more opinionated than bots? In: Proceedings of the 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining. IEEE Press, pp. 620–627.

DMJ, T., 2002. One-Class Classification: Concept Learning in the Absence of Counter-Examples.

Fawcett, T., 2006. Introduction to ROC analysis. Pattern Recognit. Lett. 06 (27), 861–874.

Ferrara, E., Varol, O., Davis, C., Menczer, F., Flammini, A., 2016. The rise of social bots. Commun. ACM 59 (7), 96–104. Available from: http://doi.acm.org/10.1145/2818717.

Freitas, C., Benevenuto, F., Veloso, A., Ghosh, S., 2016. An empirical study of socialbot infiltration strategies in the twitter social network. Soc. Netw. Anal. Min. 6 (1). 23.

Freund, Y., Schapire, R.E., 1997. A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. Syst. Sci. 55 (1), 119–139. Available from: http://www.sciencedirect.com/science/article/pii/S002200009791504X.

Friedman, M., 1937. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. J. Am. Stat. Assoc. 01 (32), 675–701.

Fu, Q., Feng, B., Guo, D., Li, Q., 2018. Combating the evolving spammers in online social networks. Comput. Secur. 72, 60–73.

Garcia, S., Herrera, F., 2008. An extension on"statistical comparisons of classifiers over multiple data sets"for all pairwise comparisons. J. Mach. Learn. Res. (9) 2677–2694.

Gilani, Z., Kochmar, E., Crowcroft, J., 2017. Classification of Twitter accounts into automated agents and human users. In: Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017. ASONAM '17, New York, NY, USA: ACM, pp. 489–496. Available from: http://doi.acm.org/10.1145/3110025.3110091.

González-Soler, L.J., Chang, L., Hernández-Palancar, J., Pérez-Suárez, A., Gomez-Barrero, M., 2017. Fingerprint presentation attack detection method based on a bag-of-words approach. In: Iberoamerican Congress on Pattern Recognition. Springer, pp. 263–271.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H., 2009a. The WEKA data mining software: an update. SIGKDD Explor. Newslett. 11 (1), 10–18. Available from: http://doi.acm.org/10.1145/1656274.1656278.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H., 2009b. The WEKA data mining software: an update. SIGKDD Explor. 11 (1), 10–18.

Hastie, T.J., Rosset, S., Zhu, J.Q., Zou, H., 2009. Multi-class AdaBoost. International Press. 349 – 360, 2.

Haustein, S., Bowman, T.D., Holmberg, K., Tsou, A., Sugimoto, C.R., Lariviére, V., 2016. Tweets as impact indicators: examining the implications of automated "bot" accounts on twitter. J. Assoc. Inf. Sci. Technol. 67 (1), 232–238.

Hinton, G., 1989. Connectionist learning procedures. Artif. Intell. 40 (1), 185–234. Available from: http://www.sciencedirect.com/science/article/pii/0004370289900490.
Ho, T.K., 1995. Random decision forests. In: Proceedings of 3rd International Conference on Document Analysis and Recognition, 1, pp. 278–282.
Huang, J., Ling, C.X., 2005. Using AUC and accuracy in evaluating learning algorithms. IEEE Trans. Knowl. Data Eng. 17 (3), 299–310.
Japkowicz, N., Shah, M., 2011. Evaluating Learning Algorithms: A Classification Perspective. Cambridge University Press.
Jha, S., Tan, K.M., Maxion, R.A., 2001. Markov chains, classifiers, and intrusion detection. Computer Security Foundation Workshop, 1(206). Citeseer.
Ji, Y., He, Y., Jiang, X., Cao, J., Li, Q., 2016. Combating the evasion mechanisms of social bots. Comput. Secur. 58, 230–249.
Keller, F.B., Schoch, D., Stier, S., Yang, J., 2017. How to manipulate social media: analyzing political astroturfing using ground truth data from south korea. In: Eleventh International AAAI Conference on Web and Social Media.
Lee, K., Caverlee, J., Webb, S., 2010. Uncovering social spammers: social honeypots + machine learning. In: Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval.. SIGIR '10, New York, NY, USA: ACM. 435–442.
Lee, K., Eoff, B., Caverlee, J., 2011. Seven months with the devils: a long-term study of content polluters on twitter. In: International AAAI Conference on Web and Social Media.
Lokot, T., Diakopoulos, N., 2016. News bots: automating news and information dissemination on twitter. Dig. Journal. 4 (6), 682–699.
Loyola-González, O., Monroy R, R.J., López-Cuevas, A., Mata-Sánchez, J.I., 2019. Contrast pattern-based classification for bot detection on twitter. IEEE Access 7, 45800–45817.
Martinez-Romo, J., Araujo, L., 2013. Detecting malicious tweets in trending topics using a statistical analysis of language. Expert Syst. Appl. 40 (8), 2992–3000.
Medina-Pérez, M.A., Monroy, R., Camiña, J.B., García-Borroto, M., 2017. BaggingTPMiner: a classifier ensemble for masquerader detection based on typical objects. Soft Comput. 21 (3), 557–569. Available from: https://doi.org/10.1007/s00500-016-2278-8.
Miller, Z., Dickinson, B., Deitrick, W., Hu, W., Wang, A.H., 2014. Twitter spammer detection using data stream clustering. Inf. Sci. 260, 64–73. Available from: http://www.sciencedirect.com/science/article/pii/S0020025513008037.
Morstatter, F., Wu, L., Nazer, T.H., Carley, K.M., Liu, H., 2016. A new approach to bot detection: striking the balance between precision and recall. In: 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM) IEEE, pp. 533–540.
Opitz, D., Maclin, R., 1999. Popular ensemble methods: an empirical study. J. Artif. Intell. Res. 11, 169–198. Available from: https://doi.org/10.1613/jair.614.
Pantic, N., Husain, M.I., 2015. Covert botnet command and control using twitter. In: Proceedings of the 31st Annual Computer Security Applications Conference ACM;, pp. 171–180.
Polikar, R., 2006. Ensemble based systems in decision making. IEEE Circ. Syst. Mag. 6 (3), 21–45. Third.
Pratama, M., Lu, J., Lughofer, E., Zhang, G., Anavatti, S., 2016. Scaffolding type-2 classifier for incremental learning under concept drifts. Neurocomputing 191, 304–329.
Provost, F., Fawcett, T., 1999. Analysis and visualization of classifier performance: comparison under imprecise class and cost distributions. In: Proceedings of the Third International Conference on Knowledge Discovery and Data Mining, 12, pp. 43–48.
Quinlan, J.R., 1993. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
Ratkiewicz, J., Conover, M., Meiss, M., Gonçalves, B., Patil, S., A, F., et al., 2011. Truthy: mapping the spread of astroturf in microblog streams. In: Proceedings of the 20th International Conference Companion on World Wide Web ACM, pp. 249–252.
Rodríguez, J., Barrera-Animas, A., Trejo, L., Medina-Pérez, M., Monroy, R., 2016. Ensemble of one-class classifiers for personal risk detection based on wearable sensor data. Sensors 16 (10). 1619.
Rokach, L., 2010. Ensemble-based classifiers. Artif. Intell. Rev 02 (33), 1–39.
Rumelhart, D.E., Hinton, G.E., Williams, R.J., In: Anderson, J.A., Rosenfeld, E., 1988. Neurocomputing: Foundations of Research. MIT Press, Cambridge, MA, USA. 696–699, Available from: http://dl.acm.org/citation.cfm?id=65669.104451.
Shao, C., Ciampaglia, G.L., Varol, O., Flammini, A., Menczer, F., 2018. The spread of low-credibility content by social bots. Nat. Commun. 9 (1), 4787.
Suárez-Serrato, P., Roberts, M.E., Davis, C., Menczer, F., 2016. On the influence of social bots in online protests. In: International Conference on Social Informatics. Springer. 269–278.
Tolosana R., Gomez-Barrero M., Busch C., Ortega-Garcia J.. Biometric presentation attack detection: beyond the visible spectrum. 2019. arXiv:190211065.
Twitter, 2017. Automation Rules — Twitter Help Center. Available from: https://help.twitter.com/en/rules-and-policies/twitter-automation.
Van Dongen, S., 2008. Graph clustering via a discrete uncoupling process. SIAM J. Matrix Anal. Appl. 30 (1), 121–141. Available from: https://doi.org/10.1137/040608635.
Varol, O., Ferrara, E., Davis, C.A., Menczer, F., Flammini, A., 2017. Online human-bot interactions: detection, estimation, and characterization. In: Eleventh International AAAI Conference on Web and Social Media.
Wald, R., Khoshgoftaar, T.M., Napolitano, A., Sumner, C., 2013. Predicting susceptibility to social bots on twitter. In: 2013 IEEE 14th International Conference on Information Reuse & Integration (IRI) IEEE;, pp. 6–13.
Wan, L., Ng, W.K., Dang, X.H., Yu, P.S., Zhang, K., 2009. Density-based clustering of data streams at multiple resolutions. ACM Trans. Knowl. Discov. Data 3 (3), 1–14. 14: :28., Available from: http://doi.acm.org/10.1145/1552303.1552307.
Wang, A.H., 2010. Detecting spam bots in online social networking sites: a machine learning approach. In: Foresti, S., Jajodia, S. (Eds.), Data and Applications Security and Privacy XXIV. Springer, Berlin Heidelberg. 335–342.
Wang, B., Zubiaga, A., Liakata, M., Procter, R., 2015. Making the most of tweet-inherent features for social spam detection on twitter. Comput. Res. Repository. Abs/1503.07405. Available from: arXiv: 1503.07405.
Yang, C., Harkreader, R., Gu, G., 2013. Empirical evaluation and new design for fighting evolving twitter spammers. IEEE Trans. Inf. Forensics Secur. 8 (8), 1280–1293.
Yang, K.C., Varol, O., Davis, C.A., Ferrara, E., Flammini, A., Menczer, F., 2019. Arming the public with artificial intelligence to counter social bots. Hum. Behav. Emerg. Technol. 1 (1), 48–61. Available from: https://onlinelibrary.wiley.com/doi/abs/10.1002/hbe2.115.
Yu, H.F., Huang, F.L., Lin, C.J., 2011. Dual coordinate descent methods for logistic regression and maximum entropy models. Mach. Learn. 85 (1), 41–75. Available from: https://doi.org/10.1007/s10994-010-5221-8.
Zhang, H., 2004. The optimality of Naive Bayes. In: Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference, 2.

Jorge Rodríguez-Ruiz received the M.Sc. degree in computer science and the Ph.D. degree in engineering sciences from the Tecnologico de Monterrey, in 2013 and 2017, respectively. He was with HP Labs, Palo Alto, CA, USA, on Machine Learning and Cloud Computing Projects. He is currently a professor at the computing department in Tecnologico de Monterrey, a member of the GIEE-ML (Machine Learning) Research Group, and a member of the Mexican National Researchers System (SNI) level C. His research interests include biometrics, anomaly detection, and the application of machine earning and artificial intelligence for physical and information security.

Javier Israel Mata-Sánchez received the B.S. degree in music production and engineering from the Tecnologico de Monterrey, Monterrey Campus, in 2017, where he is currently pursuing the master's degree in computer science. His thesis is on regards bot detection and one class classifiers. His research interests include supervised classification, one-class classifiers, bot detection, and data science.

Raúl Monroy received the Ph.D. degree inartificial intelligence from Edinburgh University,in 1998, under the supervision of Prof. A. Bundy. He is currently a Full Professor in computing with the Tecnologico de Monterrey, and also leads GIEE-ML, a group of computer scientists interested in machine learning models. Since 1998, he has been a member of the CONACyT's National Research System, currently rank 3, a Fellow of the National Academy of Sciences, and a Founding Fellow of the Mexican Academy of Computing. His research interests include the discovery and application of machine learning techniques for anomaly detection, robot motion planning, and the uncovering and then correction of errors in either a system or its specification.

Octavio Loyola-González received the B.Eng. degree in informatics engineering and the M.Sc. degree in applied informatics from the University of Ciego de ila, Cuba, in 2010 and 2012, respectively, and the Ph.D. degree in computer science from the National Institute of Astrophysics, Optics and Electronics, Mexico, in 2017. He is currently a Research Professor with the Tecnologico de Monterrey, Puebla Campus, where he is also a member of the GIEE-ML (Machine Learning) Research Group. He is also a member of the Mexican Researchers System (Rank 1). He has been involved in many research projects about pattern recognition, which have been applied on biotechnology and dactyloscopy problems. His research interests include contrast pattern-based classification, data mining, one-class classification, masquerader detection, fingerprint recognition,and palmprint recognition.

Armando López-Cuevas graduated in electronic engineering from the Autonomous University of Nayarit, in 2005, received the M.Sc. degree in electrical engineering from the Center for Research and Advanced Studies, CINVESTAV in collaboration with the National Institute for Nuclear Research, in 2009, and the Ph.D. degree from CINVESTAV Guadalajara, in 2014. He has conducted research stays at different institutes and currently holds a position at ITESM. His current research interests include natural language processing and machine learning.