IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 26 July 2024, accepted 23 September 2024, date of publication 1 October 2024, date of current version 21 October 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3471630

RESEARCH ARTICLE

Botnet Identification on Twitter: A Novel Clustering Approach Based on Similarity

LUIS DANIEL SAMPER-ESCALANTE[1], OCTAVIO LOYOLA-GONZÁLEZ[2], RAÚL MONROY[1], AND MIGUEL ANGEL MEDINA-PÉREZ[3]

[1]Tecnologico de Monterrey, Escuela de Ingeniería y Ciencias, Carretera Lago de Guadalupe, Atizapán 52926, Mexico
[2]NTT DATA, Novus Building, 28050 Madrid, Spain
[3]Kayak Analytics, The Dome Tower, Dubai, United Arab Emirates

Corresponding author: Octavio Loyola-González (octavio.loyolagonzalez@nttdata.com)

This work was supported by the National Council of Humanities, Science, and Technology of Mexico (CONAHCyT) through the Scholarship under Grant 540975.

ABSTRACT Due to Twitter's potential reach and influence, malicious automated accounts and services have been operating and growing without control. One of the most recognizable is the bot, a piece of programming or automated system reproducing an assignment constantly over time. The bots' owners can also coordinate them into botnets to increase their capabilities and impact on the platform. In the literature, botnet detection on Twitter is a relatively under-researched topic where researchers need complete and updated collections, as many of the botnets currently reported are outdated or incomplete. Despite Twitter's best efforts to mitigate the botnets' presence by issuing restrictions and bans, they are insufficient due to their accelerated creation and expansion. Therefore, we propose a solution employing an updated botnet collection that identifies with high certainty which botnets a set of bots belongs to. Our experimentation showed that our solution labeled the botnets accurately, achieving an outstanding performance with unknown botnets in most of our experiments and when compared with other works from the literature. We also proved that our solution obtained its best scores without relying on a specific algorithm or configuration.

INDEX TERMS Botnet detection, bot similarity, clustering, Twitter.

I. INTRODUCTION

Due to Twitter's potential reach and influence, automated accounts and services have been operating and growing on Twitter even after multiple bannings performed by the platform [1], [2]. One of the most recognized publicly is the bot, a piece of programming or automated system reproducing an assignment constantly over time [3]. The owners of these bots, also known as botmasters, maintain and operate them through Command-and-Control stations [4], [5]. Botmasters can also coordinate their bots into botnets to increase their capabilities and impact on the platform [6], [7].

Fukuda et al. [8] estimated in 2022 that up to 18% of Twitter's active accounts are bots. This figure provides no information about botnets' presence on the platform. Bot detection is a class imbalance problem [9], where the presence of legitimate users (dominant class) overshadows that of bot accounts (minority class), resulting in the need for a correct classification [10]. This consideration can be extended to botnet detection, where the classes (botnets) vary in size and objectives, with the classifiers undergoing the risk of being overwhelmed by a dominant class [11], [12] (e.g., a large botnet).

It is also worth noting that while bots and thus botnets are not ill-natured by definition [13], [14], they represent a risk to the security and credibility of the platform when programmed with mischievous objectives [15], [16]. Botnets on Twitter can inflate users' statistics quickly to trigger the platform's anti-abuse mechanisms and mistakenly ban legitimate accounts [17], mention users in their tweets to redirect them to malicious sites [18], and even meddle in political elections [19], social discussions [20] or public opinion [21].

The associate editor coordinating the review of this manuscript and approving it for publication was Hocine Cherifi.

Despite Twitter's best efforts to mitigate the presence of botnets by issuing restrictions and bans [22], they are insufficient due to botnets' rapid creation and expansion, to the point where researchers inform these botnets are easier to construct and manage than to identify and report [23]. These situations have led to an uneven arms race where botmasters are getting the upper hand, as the main focus of researchers and the platform is the identification of bots on an individual basis rather than on a group level [24].

Botnet detection on Twitter is a relatively under-researched field [23]. Echeverria and Zhou [25] have discussed the need for complete and updated collections, as many of the botnets currently reported in the literature are outdated, incomplete, or unobtainable due to bans or the deletion of accounts involved. Also, Twitter has toughened its policy for sharing the public data of its users, which has complicated the exchange of information between research groups and has led to most of the new botnets reported in the literature being private [18].

Therefore, in this paper, we propose a solution divided into two phases based on Machine Learning (ML) that identifies with high certainty to which botnets a set of bots belongs to. We first pair the bots with each other to quantify their resemblance using their public metadata. Next, we train regression methods using this similarity measurement to predict if the pairings are from the same botnet. Then, we construct a distance matrix describing the separation between bots using these predictions. Finally, we utilize this matrix on different clustering methods to obtain the botnets formed by these bots. For comparison, we evaluate our performance results with those of five other works in the literature.

The main contributions of this work are listed as follows:

- A novel model for botnet detection, statistically outperforming five other approaches.
- A new feature representation and score based on the similarity of bots to identify the botnet they belong to.
- A new botnet database gathered from the literature and improved by updating its public metadata and extracting social metadata missing from the original publication.
- A new 2-D space representation of botnets and the proximity between them using a distance matrix.

The rest of the paper is organized as follows: In Section II, we explore the relevant concepts used in our research. Next, we review related works from the literature on bot and botnet detection on Twitter in Section III. After that, we describe in Section IV the methodology of our proposed approach for identifying botnets on Twitter, explaining its two phases and their associated steps in detail. Then, in Section V, we present and discuss the results of implementing the two phases of our solution and comparing our approach with other works from the literature. Finally, in Section VI, we give the final remarks on our research and explore future work.

II. PRELIMINARIES

This section presents the preliminaries for detecting botnets and quantifying the resemblance between bots. We define what a bot, a botnet, and Twitter's network are and the problem we will address in this work using those definitions. We also explain how we utilize the concepts of behaviors and attributes for a bot to describe our solution in Section IV.

A. PROBLEM DEFINITION

Researchers [26], [27] usually portray Twitter as a social graph $G = (V, E)$ where V contains a collection of *vertices* v representing nodes in the network, and each node is considered a Twitter account. E depicts a set of *edges* e that characterizes the relations between nodes, where each relation expresses a follower relationship (from target to source) or following relationship (from source to target) between Twitter accounts.

In Twitter, a bot $B = (Us, Tw, Sc)$ is an automated account containing the bot's user Us, tweet Tw, and social Sc metadata. A Botnet BNT is a set of bots B operated by a botmaster BMT and coordinated to accomplish complex tasks in the platform, such as manipulating users to click advertisements or unwanted media, constantly harass other users or public figures, manipulate trending topics or public opinion using hashtags, or artificially enhance statistics of other tweets or accounts [25], [28].

In this work, we consider the problem of botnet detection on Twitter as a multiclass classification problem where we search to which botnet BNT (class) a given bot B (member) belongs to (if any). These botnets may contain bots of various types, but they form a group because of their coordination towards a common goal. For instance, botnet classes could include political influencers, statistic enhancers, or content polluters. Our goal is to discover whether a bot belongs to one of these known classes or if it operates independently, which may suggest that it is part of an unknown botnet (class) that we have not yet encountered.

B. BOT BEHAVIOR

The behavior of a bot is how an automated account conducts itself on the platform to fulfill its goal [29]. Such conduct can be spamming [30], manipulating [6], or inflating statistics of legitimate users [25] and represents the core of many algorithms for bot detection on Twitter [31].

C. BOT ATTRIBUTES

The attributes of a bot refer to extracted metadata from an API endpoint that has been preprocessed on the dataset, describing their Twitter configuration, profile details, or preferred tools of interaction on the platform [32]. Researchers have used these attributes before to differentiate between types of bots and genuine users with great success [33], [34].

III. RELATED WORK

In this section, we review five works from the literature related to our work. We have limited the scope of this review

to articles that use datasets previously used as benchmarks or available upon request, employ a replicable methodology useful for botnet detection, and their feature characterizations are applicable in a similarity context. We summarize their findings and highlight their contributions, explaining their relevance in developing our solution.

A. MANUAL BOTNET IDENTIFICATION

Researchers have identified several botnets in the literature using an exploratory approach [35], [36]. It is a reliable and feasible method that involves the manual revision of thousands of accounts by researchers and experts [37], [38], [39].

In [17], Echeverría et al. proposed a framework named LOBO and a collection of twenty botnets to solve the biases and overfitting of training datasets in detection algorithms by training and evaluating such methods with all but one class (botnet). These botnets were obtained from the literature and their research, retrieving and updating their members' metadata. Their experiments using several classifiers and a feature model composed of user and tweet information proved that training and testing with only one botnet lead to an accurate prediction for that class, but including unknown botnets drops the prediction considerably.

For our work, we implemented a methodology similar to LOBO and left several botnets in our training to reduce bias and overfitting. Also, we considered this work for our performance evaluation as the authors validated their results employing several botnets.

Adewole et al. [40] extracted accounts from Twitter's API by searching tweets containing spam-related words. The authors proposed 11 features to group the Twitter accounts using K-Means clustering algorithm, resulting in 5 clusters of users. After inspecting these groups, they presented 10,280 spambots and 15,000 genuine users as their ground truth, and trained a Random Forest (RF) [41] classifier using an improved set of 23 features. Finally, the authors proposed to compare with others using their feature representations instead of their datasets to circumvent the problem of sharing restrictions on Twitter [42].

We followed the authors' proposal for comparison and used the feature representations of other works employing group-based metrics for botnet (multiclass) identification. Also, we used this work in our performance evaluation as the authors stated their solution outperformed four other works in the literature.

In [43], Breiman used a RF [41] model trained with 13 features to identify and extract around 140,000 Twitter users by exploring popular hashtags. Then, they applied K-Means to cluster these accounts by their proposed 13 features based on user activity. Finally, they obtained and manually labeled four groups: low-activity users, medium-activity users, verified users, and high-activity users. After applying their RF classifier on each account in these clusters, the authors discovered that users from the low and medium activity groups were bots, present in more than 50% of the popular hashtags.

For our work, we also discovered hidden botnets by grouping accounts using our feature model, but we decided to summarize them instead into a single metric. Therefore, we included this work in our comparison for their similarities to our solution.

B. AUTOMATIC BOTNET IDENTIFICATION

In recent years, researchers have used efficient reduced feature representations and clustering techniques to automate the detection of bots and botnets on Twitter [44], [45]. Where these approaches lack a generalized approach to solve the automated presence in the platform, they compensate it with compelling results under different experimentation settings.

Khalil et al. [44] used four botnet datasets from the public collection of Cresci et al. [38] to compare two clustering methods from the literature: DBSCAN [46] and K-Means [47]. They created six combinations for experimentation by mixing these botnet datasets and grouped each experiment's accounts into genuine users or social spambots using the clustering methods.

We followed the authors' approach by making combinations of our datasets of different sizes and objectives for training and testing to avoid hypersensitivity and over-specification towards a given botnet. We have included this work in our performance evaluation as they also employed different clustering methods and botnets for classification.

In [45], Fonseca et al. proposed a reduced set of five features based on user activity for automated detection to prove they can obtain good performance without using a complex model. They employed two botnet datasets from Cresci et al. 2017's public collection [38], splitting them into 90% training and 10% testing with 10-fold cross-validation. Then, the authors trained three binary classifiers and a one-class classifier to compare directly with Rodriguez-Ruiz's work [48]. At the end of the paper, the authors stated that their approach performed reasonably well with their reduced set and proved that it obtains better results than a random solution in the worst scenario.

We were motivated by the authors' statement that reducing a feature set will provide a good performance regardless of their missing features, and we applied this to our work. Also, we compared our results with their solution because we wanted to measure how much performance gain we obtained by incorporating features based on tweet and social activity.

C. BOTNET COLLECTION UPDATE

Researchers have published these discovered and synthetic botnet datasets with different sizes and objectives through the years [31]. However, due to the restrictions imposed by Twitter's privacy policy in 2018 [42], only a few botnets with limited metadata remain publicly available [18]. Consequently, researchers have been gathering collections of

botnet datasets from the literature with updated and expanded metadata for preservation while sharing them on request [49].

One of these botnet collections is also found in Echeverria et al.'s 2018 [17] work, where he published a collection of twenty botnets from the literature and previous work, with their user and tweet metadata renewed but without social information. We used these botnets throughout this paper to develop our solution. We improved this collection by gathering all the user, tweet, and social information of the bots inside the botnets using Twitter's API v2 [50] from March to November 2021. On average, we recovered 81.42% of the bot accounts from this collection, with the remaining 18.58% not being accessible. We believe this is a consequence of the three years gap (2018 to 2021) since Echeverria's work was published, which could have triggered the concealment of these accounts by botmasters or bans from Twitter.

D. CONCLUDING REMARKS

Echeverria et al. [17], Adewole et al. [40], and Barhate et al. [43] presented botnet collections using an exploratory approach, but they required human annotators and considerable time for identification, making them unfeasible to face their rapid proliferation. Also, while Khalil et al. [44] and Fonseca et al. [45] proposed compelling methods for automation, they lack a group focus that allows identification at a large scale. Hence, there is room to contribute to the literature on botnet detection on Twitter. Also, new methods for automatic detection should help identify new botnets in the wild without the drawbacks of exploratory approaches, which would significantly increase the ground truth currently available. Furthermore, reduced feature representations that exploit the similarity between bots employing their user, tweet, and social information to form groups still have untapped potential. Therefore, we detail the methodology of our solution with the implementation of these ideas next.

IV. OUR PROPOSED APPROACH

This section describes the methodology followed to implement our solution. We have divided it into two phases: the construction phase and the detection phase. For each one, we gathered a set of botnets, namely the training and detection dataset, to help us train and evaluate different regression methods and clustering algorithms.

We describe our methodology thoroughly in the following sections. We first discuss the botnets datasets used, and then explain in detail the construction phase in Section IV-A and the detection phase in Section IV-B.

We performed exploratory and analytical work on the twenty botnet datasets inside Echeverria et al.'s collection. We manually revised the bots within these botnets and identified several shared attributes, including similar user descriptions, creation dates, tweet counts, tweeting behaviors, hashtags, spam words, and URLs used. Additionally, we examined their number of social connections and reciprocity, their interactions with other accounts (mentions, retweets, or replies) through publicly available information, and their objective pursued in the platform. We randomly selected ten botnets to work instead of the twenty available to reduce half our computational requirements while having equally significant results.

Therefore we use $\mathcal{BNT}_{\text{Complete}}$ to denote the ten botnets, so that

$$\mathcal{BNT}_{\text{Complete}} = \{BNT_1, BNT_2, \ldots, BNT_{10}\} \quad (1)$$

comprises all the botnets extracted from the platform using Twitter API v2 [50] and randomly chosen from the twenty available. We also use $\mathcal{B}$ to denote the set of Twitter accounts classified as bots obtained after flattening[1] the set of botnets from $\mathcal{BNT}_{\text{Complete}}$, so that $\mathcal{B} = \{B_1, B_2, \ldots, B_n\}$. We present a brief description of each botnet in Table 1.

TABLE 1. The botnets from the literature used throughout this paper for developing our solution. Echeverria et al. republished them in 2018 [17] with updated user/tweet metadata. We improved them in 2021 by aggregating social metadata. The first column states the botnet name, the second where it was first published, the third the number of accounts available, and the last the coordinated objective of the botnet.

Dataset	Work	Accounts	Objective
FastFollowersZ (FSF)	Cresci *et al.* [37] (2015)	28	Inflate followers of an account.
TwitterTechnology (TWT)	Cresci *et al.* [37] (2015)	583	Inflate followers of an account.
Star Wars (SW)	Echeverria *et al.* [25] (2017)	117	Publish book quotes.
Traditional Spambots #3 (TS3)	Cresci *et al.* [38] (2017)	187	Spam open job positions.
Social Spambots #1 (SS1)	Cresci *et al.* [38] (2017)	464	Retweet a political candidate.
Social Spambots #3 (SS3)	Cresci *et al.* [38] (2017)	375	Spam links of products on sale.
InterTwitter (INT)	Cresci *et al.* [37] (2015)	60	Inflate followers of an account.
Traditional Spambots #1 (TS1)	Cresci *et al.* [25] (2017)	660	Spam phishing URLs.
Traditional Spambots #4 (TS4)	Echeverria *et al.* [25] (2017)	425	Spam different job offers.
Attack on Brian Krebs (ABK)	Echeverria *et al.* [17] (2018)	681	Force a ban on a target account.

We divided these ten botnets further into two groups: the training dataset containing six botnets and the detection dataset containing the remaining ones, forming a partition.

Training Dataset:

$\mathcal{BNT}_{\text{Training}}$ is given by

$$\mathcal{BNT}_{\text{Training}} = \{FSF, TWT, SW, TS3, SS1, SS3\} \quad (2)$$

We use $\mathcal{B}_{\text{Training}}$ to stand for all the bots in $\mathcal{BNT}_{\text{Training}}$ after flattening.

Detection Dataset:

$\mathcal{BNT}_{\text{Detection}}$ is defined as

$$\mathcal{BNT}_{\text{Detection}} = \{INT, TS1, TS4, ABK\} \quad (3)$$

We similarly use $\mathcal{B}_{\text{Detection}}$ to denote the set of all bots in $\mathcal{BNT}_{\text{Detection}}$.

[1] Let $\mathcal{S}$ be a set of sets S of objects p of type E. Then **Flatten**$(\mathcal{S}) = \{p \mid p \in S; S \in \mathcal{S}\}$.

A. CONSTRUCTION PHASE

To identify the botnets hidden in a set of bots, we first need to measure quantitatively the similarity between bots. Bots sharing attributes (e.g., time zone, language), objectives (e.g., harass users, promote a brand), and behaviors (e.g., tweeting patterns, user interactions) on the platform have a higher chance of belonging to the same botnet [29]. Thus, in this phase, we developed a regression model to predict the similarity between bots, assigning a score ranging from 0 to 1. We present the pipeline of this phase in Fig. 1. We divided this phase into five steps (each with its input and output), explained below.

1) DATA PREPROCESSING

As input, we received the set of bots $\mathcal{B}_{\text{Training}}$ and their associated metadata extracted from Twitter's API of each botnet BNT chosen for the training dataset $\mathcal{BNT}_{\text{Training}}$.

a: CLEANSING

This preprocessing task comprehends the deletion of corrupt data and filling of missing data of the training dataset. Extraction of information from Twitter's API is prone to failures and errors [18] such as null characters or blank fields. Therefore, we fixed them beforehand to prevent false or contradictory results from leading to wrong conclusions.

b: REDUCTION

This preprocessing task involves extracting and grouping relevant data into separate fields in the dataset for each bot. Since some accounts can have over 3,000 tweets, we specifically extracted the hashtags, URLs, interactions (such as mentions, replies, and retweets), and publishing mediums (e.g., Web, Android, iPhone) from the tweets. This approach allows us to avoid accessing all tweets each time we need the data, making the process more efficient.

c: ENRICHMENT

This preprocessing task consists in the transformation of data, such as scaling and normalizing variables. The unprocessed information from Twitter's API for the botnets may come in various formats and ranges, so we need to reshape it to a consistent scale for ease of use and understanding. Also, some ML algorithms, such as Linear Support Vector Regression [51], require data re-scaling to improve their produced models [52].

As a result, we obtained the training dataset $\mathcal{BNT}_{\text{TrainingProc}}$ with their corresponding set of bots $\mathcal{B}_{\text{TrainingProc}}$ ready to be used in the next steps of the construction phase.

2) BOT PAIRING AND SIMILARITY FEATURES EXTRACTION

Firstly, we paired every bot in $\mathcal{B}_{\text{TrainingProc}}$ so that

$$(B_i, B_j)|B_i, B_j \in \mathcal{B}_{\text{TrainingProc}} \wedge B_i \neq B_j \tag{4}$$

yields $C(n, 2)$ pairs or m tuples T where n is the cardinality of $\mathcal{B}_{\text{TrainingProc}}$. By extension, we define $\mathcal{T}_{\text{PairsTraining}}$ as the tuple collection of all pairs B_iB_j calculated from $\mathcal{B}_{\text{TrainingProc}}$.

For reference and future comparison with other works, the time complexity for these calculations is $\Theta(n^2)$, determined by the number of bot pairings needed or $C(n, 2)$, whose growth is proportional to polynomial time n^2. If we add a new bot, it would be linear, and only the direct pairings would have to be calculated. With these bot pairs, we aim to calculate the bots' resemblances by comparing their behaviors and attributes extracted from the platform, which are measured and transformed into feature representations.

We used a feature representation of 14 similarity features based on the bots' user, tweets, and social information or $SF_{B_iB_j}$ that characterize the resemblance between two bots B_i and B_j. In the past, others have used this concept of similarity measurement to detect botnets with significant success [29]. We employed the Jaccard similarity coefficient JSC [53] and normalized difference ND [52] to calculate these features depending on whether their associated attributes contain quantitative (e.g., number of tweets or followers) or qualitative (e.g., hashtags or URLs used) information. We divided these similarity features into the following categories: user activity, tweet activity, and social activity. We present each category next.

a: USER ACTIVITY

This category is about the metadata extracted from the users' account information. This information has proven valuable in bot detection [17], [29], so we incorporated it into our feature representation. Table 2 introduces the four similarity features based on user activity employed in our work. We use $\mathcal{SFUA}_{\text{B}_\text{i}\text{B}_\text{j}}$ to describe these four similarity features:

$$\mathcal{SFUA}_{\text{B}_\text{i}\text{B}_\text{j}} = \langle Age, Usage, Frequency, Profile \rangle \tag{5}$$

TABLE 2. The similarity features proposed based on user activity from the literature and our previous research. The first column states the feature's name, the second the work it comes from, the third the method used (Jaccard Similarity Coefficient or Normalized Difference) to calculate its value, and the last its description in natural language.

Name	Origin	Method	Similarity Based On
Age	Wright *et al.* [54]	ND	Account creation dates.
Usage	Echeverria *et al.* [17]	ND	Periods of activity.
Frequency	Ilias *et al.* [55]	ND	Frequency of tweeting.
Profile	Samper *et al.* [18]	JSC	Public information.

b: TWEET ACTIVITY

These features utilize the tweets published by an account to determine if they can be related to those tweets posted by other users. Kantepe and Ganiz [56] have found that behaviors, relationships, and sentiments can be revealed by analyzing bots' tweets alone, and that these patterns are significantly different from those in legitimate accounts. We present in Table 3 the list of four similarity features based

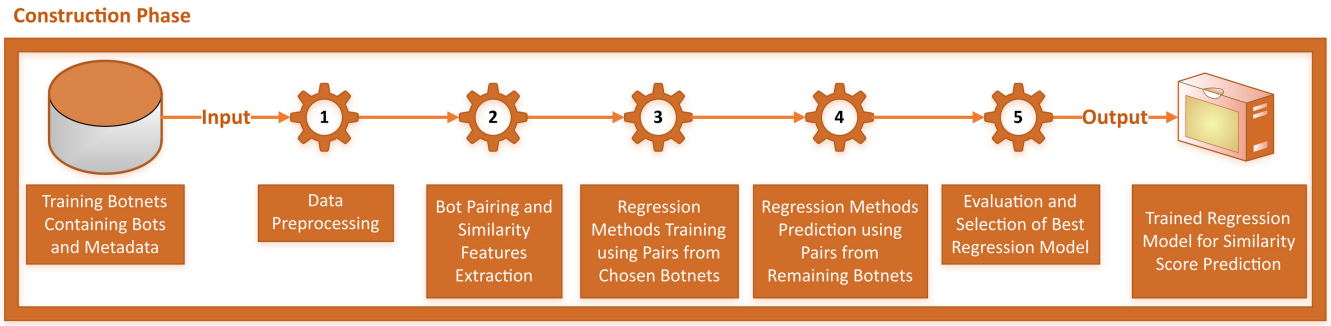


FIGURE 1. The construction phase of our approach. We received a set of bots from different botnets as input, and our goal is to process the bots' metadata to pair all of them with each other, extracting each pair's similarity features. We then trained different regression methods using the bot pairings from selected botnets and employed the resulting models to predict the similarity of the bot pairings from the remaining botnets. Finally, we evaluated and chose the best regression model as the output of this phase.

on tweet activity. We use $\mathcal{SFTA}_{\mathrm{B_iB_j}}$ to denote these four features:

$$\mathcal{SFTA}_{\mathrm{B_iB_j}} = \langle Hashtags, URLs, Interactions, Sources \rangle \tag{6}$$

TABLE 3. The similarity features proposed based on tweet activity from the literature and our previous research. The first column states the feature's name, the second the work it comes from, the third the method used (Jaccard Similarity Coefficient or Normalized Difference) to calculate its value, and the last its description in natural language.

Name	Origin	Method	Similarity Based On
Hashtags	Echeverria *et al.* [17]	JSC	Hashtags posted.
URLs	Lingam *et al.* [29]	JSC	URLs employed.
Interactions	Samper *et al.* [18]	JSC	Contact with others.
Sources	Samper *et al.* [18]	JSC	Publication medium.

c: SOCIAL ACTIVITY

The last category employs the metadata from the users' profile and tweet information related to social interaction. Dorri et al. [57] have reported better performance while incorporating features based on social interaction compared to other solutions using only user and tweet information. Therefore, we introduce the six similarity features in Table 4. We also use $\mathcal{SFSA}_{\mathrm{B_iB_j}}$ to describe these six features:

$$\mathcal{SFSA}_{\mathrm{B_iB_j}} = \langle Followers, Followings, \ldots, Reciprocity \rangle \tag{7}$$

For each pair of bots B_i and B_j, we calculated the fourteen similarity features of the feature representation SF and stored their values in their tuple T. At the end of T, we appended a value **SS** representing our similarity score:

$$\mathbf{SS}(B_i, B_j) = \begin{cases} 1, & \text{if } BNT(B_i) == BNT(B_j) \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

We made this scoring based on the idea that if two bots belong to the same botnet, their probability of being similar is high, so they receive the highest score. On the other hand,

TABLE 4. The similarity features proposed based on social activity from the literature and our previous research. The first column states the feature's name, the second the work it comes from, the third the method used (Jaccard Similarity Coefficient or Normalized Difference) to calculate its value, and the last its description in natural language.

Name	Origin	Method	Similarity Based On
Followers	Samper *et al.* [18]	JSC	Users following.
Followings	Samper *et al.* [18]	JSC	Users being followed.
Likes	Samper *et al.* [18]	JSC	Liked tweets.
Popularity	Dorri *et al.* [57]	ND	Follower/Following ratio.
Reputation	Dorri *et al.* [57]	ND	Network importance.
Reciprocity	Dorri *et al.* [57]	ND	Bidirectional relations.

if they do not belong to the same botnet, the probability that they are similar is low, thus receiving the lowest score since they will not share as many characteristics or have the same objective. This score will be of invaluable help in the regression methods' training to indicate to the algorithms which of the bot pairs' tuples and features' values form a botnet and which do not.

So, for each pairing, we created a new extended tuple T with the bots' identifiers, feature representation, and similarity score, replacing the original in $\mathcal{T}_{\mathrm{PairsTraining}}$ and representing this step's result:

$$T_{B_iB_j} = \langle \mathbf{ID}_{B_i}, \mathbf{ID}_{B_j}, SF_{B_iB_j}, \mathbf{SS}_{B_iB_j} \rangle \tag{9}$$

3) REGRESSION METHODS TRAINING

We received $\mathcal{T}_{\mathrm{PairsTraining}}$ to train regression methods for predicting a given pair's similarity score based on the values for each feature on the tuple. We chose and implemented ten robust algorithms for our initial training: Multiple Linear Regression [58], Polynomial Regression [59], Linear Support Vector Regression [51], Decision Tree [60], [61], Random Forest [41], XGBoost [62], Adaboost [63], Stochastic Gradient Descent [64], TheilSen [65], and Deep Neural Networks [66]. Researchers have employed these regression methods with excellent results for bot detection on Twitter [26], [67], [68], and thus we consider them suitable and relevant to our research. We performed variable

scaling as a preprocessing task for the values of the similarity features inside the tuples, as some regression methods' objective functions are sensitive to standardization and normalization [69].

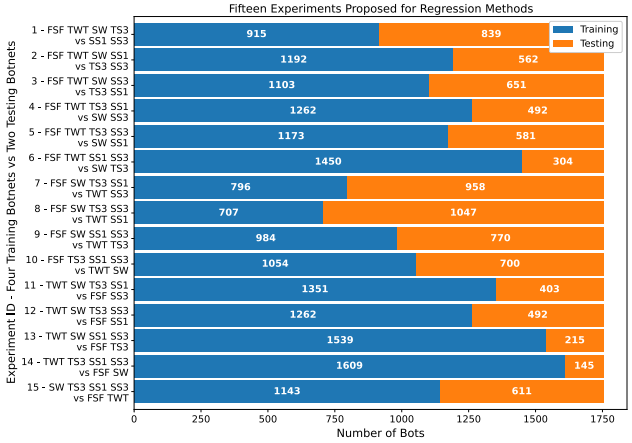


FIGURE 2. An overview of the fifteen experiments proposed for training. They derived from combining six botnets from Echeverria et al.'s 2018 [17] work with aggregated social metadata by us: FastFollowerZ (*FSF*), TwitterTechnology (*TWT*), Star Wars (*SW*), Traditional Spambots #3 (*TS*3), Social Spambots #1 (*SS*1), and Social Spambots #3 (*SS*3). The x-axis displays the distribution of the 1754 bots total and the y-axis the experiment's ID with its four botnets chosen for training and two for testing.

For the training setup, we decided to follow the methodology proposed by Echeverria et al. [17]: we only used the bot pairs' tuples from four botnets for model training or $\mathcal{BNT}_{\text{ModelTrain}} \subset \mathcal{BNT}_{\text{TrainingProc}}$ and $\mathcal{T}_{\text{ModelTrain}} \subset \mathcal{T}_{\text{PairsTraining}}$. Then, we used the remaining tuples from the other two botnets for testing or $\mathcal{BNT}_{\text{ModelTest}} \neq \mathcal{BNT}_{\text{ModelTrain}}$ and $\mathcal{T}_{\text{ModelTest}} \neq \mathcal{T}_{\text{ModelTrain}}$. This approach allowed us to evaluate the regression methods when only four of the six classes are known and thus reduce the possibility of overfitting and misleading results when using the same data for training and testing.

For the six botnets in the training dataset, we made $C(6, 4)$ possible combinations obtained by taking four botnets to train the ten regression algorithms and the remaining two to test them, giving us a total of fifteen experiments. We got ten trained regression models for each experiment as the output of this step. Each trained model has learned which set of tuples constitutes an actual botnet in its training using the similarity score and now predict the probability of two bots being in the same botnet by evaluating the pair's tuple. Finally, we use SS_{True} to specify the similarity scores corresponding to their true values of the tuples from the botnets and SS_{Pred} to indicate those scores predicted by regression methods for the tuples.

4) SIMILARITY SCORE PREDICTION

For each of the fifteen experiments, we used the ten regression models trained with the tuples of $\mathcal{T}_{\text{ModelTrain}}$ from the four botnets $\mathcal{BNT}_{\text{ModelTrain}}$ to predict the similarity score $\mathbf{SS} \in SS_{Pred}$ for each tuple in $\mathcal{T}_{\text{ModelTest}}$ obtained in the bot pairing of the remaining two botnets $\mathcal{BNT}_{\text{ModelTest}}$. We provide an overview of the experiments in Fig. 2. The resulting predictions are the results of this step and essential for the next step when comparing the regression models.

5) EVALUATION AND SELECTION

In the final step of the construction phase, we received the results from the regression models' predictions of the similarity score SS_{Pred} for the botnets from $\mathcal{BNT}_{\text{ModelTest}}$ to measure and select the model with the best performance. To assess the regression models' output, we use the area under the ROC curve (AUC), a standard numerical metric resilient to class imbalance problems [70], which is common in the literature on bot detection as there are more genuine accounts than automated ones in the wild [48]. Also, one has been using this evaluation metric for comparing their ML approaches with others [23], [37], [48] and thus considers it viable for botnet detection.

Since we know SS_{True}, we can calculate the AUC using SS_{Pred} of the ten regression models for $\mathcal{T}_{\text{ModelTest}}$ on each experiment. We then compared the ten AUC scores for each experiment and chose the regression method with the best overall performance or *BstRegMethod*. Lastly, we used the Wilcoxon signed-rank test [71] to differentiate between two regression methods' results when they are similar and thus difficult to identify which has better AUC performance.

After selecting *BstRegMethod* for our SS_{Pred}, we trained it with all of $\mathcal{T}_{\text{PairsTraining}} \in \mathcal{BNT}_{\text{TrainingProc}}$. This training enables the model to learn from all available data, enhancing its generalizability to predict whether a pair of bots belong to a botnet based on their similarity. This remains effective even when the model has not been specifically trained for those particular types of bots or botnets. We support this reasoning on the grounds that no matter how complex a bot can be, it will always cast a shroud of automation [38], which we believe can be leveraged to relate bots of different types and objectives and thus detect hidden groups. The resulting trained regression model or *TrainedRegMod* is the outcome of this phase.

B. DETECTION PHASE

In the literature, Lingam et al. [29] have clustered automated accounts using their likeness in various attributes and behaviors, achieving precision, recall, and f-measure of over 89%. Encouraged by their findings, in this phase consisting of five steps we discovered groups of bots using clustering methods with a distance matrix constructed from the similarity measurement between them. In Fig. 3 we introduce the pipeline of this phase.

1) DATA PREPROCESSING

Following the same methodology presented in Section IV-A1, we first obtained $\mathcal{B}_{\text{Detection}}$ and their metadata retrieved from Twitter after flattening $\mathcal{BNT}_{\text{Detection}}$. Then we performed

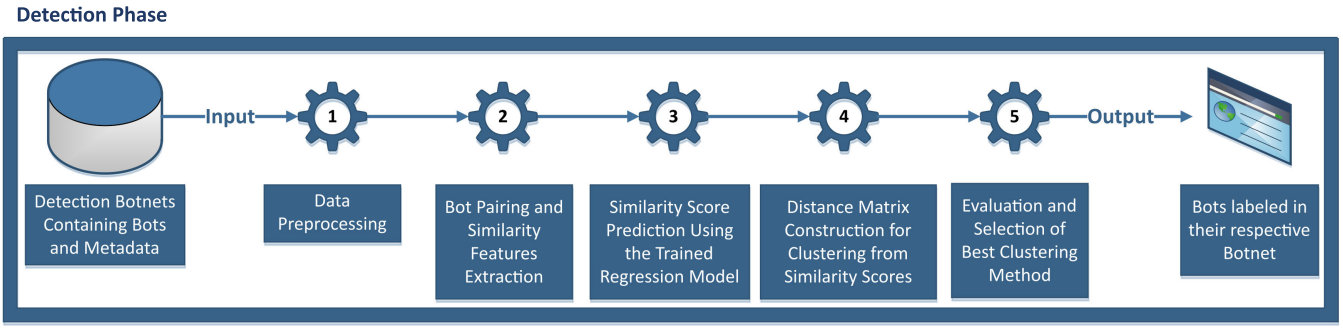


FIGURE 3. The detection phase of our approach. We received a set of bots from distinct botnets not used before as input and processed the bots' metadata to pair them with each other, extracting each pair's similarity features. We then predicted the similarity of each bot pair utilizing the previously trained regression model and constructed a distance matrix from its results. Finally, we evaluated different clustering methods using this matrix and chose the one with the best performance as the output of this phase.

the cleansing, reduction, and enrichment techniques for each bot and its user, tweet, and social information. The resulting $\mathcal{BNT}_{\text{DetectionProc}}$ and $\mathcal{B}_{\text{DetectionProc}}$ are used to search for hidden clusters in the following steps.

2) BOT PAIRING AND SIMILARITY FEATURES EXTRACTION

Similar to Section IV-A2, we paired every bot from $\mathcal{B}_{\text{DetectionProc}}$, obtaining m tuples T of the possible combinations. The time complexity for these calculations is also $\Theta(n^2)$. We calculated the SF for each pair, appending the true value of **SS** and storing them in $\mathcal{T}_{\text{PairsDetection}}$, the output of this step.

3) SIMILARITY SCORE PREDICTION

After obtaining $\mathcal{T}_{\text{PairsDetection}}$, we used *TrainedRegMod* to quantitatively predict whether the bots in the pairs' tuples belong to the same botnet. However, we did not send $SS_{True} = \{\mathbf{SS}_1, \mathbf{SS}_2, \ldots, \mathbf{SS}_m\}$ to the model. The botnets of $\mathcal{BNT}_{\text{DetectionProc}}$ and their bots from $\mathcal{B}_{\text{DetectionProc}}$ are unknown to *TrainedRegMod*, so it should predict SS_{Pred} based solely on its training using $\mathcal{BNT}_{\text{TrainingProc}}$.

To evaluate the performance of *TrainedRegMod* with the new botnets, we used SS_{Pred} to compare them with SS_{True} and calculate the AUC. We preferred the AUC because it is a measure relative to each class that is unvarying of our imbalanced tuple collection, where more than half of the pairings are between bots from different botnets. Other metrics, such as Accuracy, have the potential to mislead the performance evaluation as they only predict the majority class and still achieve a high score [72].

We used the resulting AUC to improve our solution through a feature selection process. We identified the feature with the least or negative contribution to *TrainedRegMod* and removed it from SF. We repeated this process, retraining the model until we reduced SF to a single similarity feature. We selected the best *TrainedRegMod* that achieved the highest AUC with the smallest SF. We used this improved trained regression model or *Smp-22* to predict SS_{Pred} for the rest of the phase. Finally, the resulting SS_{Pred} using *Smp-22* for each bot pairing is our expected output for this step.

4) DISTANCE MATRIX CONSTRUCTION

Starting with the bot pairs' similarity scores calculated in the previous step, we constructed a distance matrix to be used by clustering methods to group the set of bots based on their resemblance to each other.

First, we calculated the distance $\mathbf{Dist}_{B_iB_j}$ for each bot pair B_iB_j in the bot set $\mathcal{B}_{\text{DetectionProc}}$ based on the similarity score predicted $\mathbf{SS} \in SS_{Pred}$. The formula to calculate this distance is:

$$\mathbf{Dist}(B_i, B_j) = 1 - \mathbf{SS}_{B_iB_j} \tag{10}$$

Then with these distances, we created a matrix *DistMtx* that describes how far apart bots are between them in $\mathcal{B}_{\text{DetectionProc}}$, based on their similarities and how likely they are to form a botnet BNT. Therefore, the distance matrix is given next, where n is the cardinality of $\mathcal{B}_{\text{DetectionProc}}$:

$$DistMtx = [\mathbf{Dist}_{B_iB_j}]_{nxn} \tag{11}$$

Finally, the resulting *DistMtx* serves as the output of this step and subsequently as input for the clustering algorithms in the next step.

5) EVALUATION AND SELECTION

In this last step we received the distance matrix constructed from the tuples of the bot pairs formed using the detection dataset. We had two goals for completing this step: to assess whether the proposed training botnets and feature representation provide relevant information to the regression model for this prediction and to compare our performance with other works from the literature. We measured both goals by evaluating the groups generated by the clustering algorithms from $\mathcal{B}_{\text{DetectionProc}}$.

For the clustering algorithms, we chose and implemented four from the literature: K-Medoids [73], HDBSCAN [74], Agglomerative [75], and Spectral [76]. In the literature, researchers have used clustering for grouping suspicious accounts from Twitter with remarkable performance [33], [40], [43]. We have chosen these algorithms as they allow us to represent bots as objects separated from each other by a certain distance measurement [77], [78]. In addition, we sped

up their calculation and clustering process by providing a distance matrix and the number of clusters beforehand.

For the detection setup, we conducted two experiments: the first using the default settings of the clustering algorithms, and the second with the best configuration obtained by tuning the hyper-parameters for each clustering algorithm. To evaluate the performance of our approach, we used the following external clustering validation indices (CVIs): Normalized Mutual Information (NMI) [79], Adjusted Rand Index (ARI) [80], and Fowlkes-Mallows Index (FMI) [81]. Since we already know the botnets to which the bots belong and their true clustering labels $\mathbf{CL} \in CL_{True}$, we can employ external clustering evaluation techniques to assess whether the clustering labels $\mathbf{CL} \in CL_{Pred}$ assigned to the bots from $\mathcal{B}_{\text{DetectionProc}}$ by the clustering method match with the real ones. We selected these indices because one have used them previously for botnet detection on Twitter [29], they handle similar ranges to compare them with each other, and their results are not affected by chance or the type of clustering used [82].

Since we know the classes from CL_{True} and algorithms' predictions CL_{Pred}, we can treat our problem as multiclass, where the number of clusters formed will correspond to the number of classes used. With this consideration, we calculated the AUC metric by adapting it for multiple classes using a one-versus-rest strategy (OvR), where we compared the classes with each other simultaneously and averaged the score obtained [83]. Utilizing the AUC score as an additional evaluation metric allowed us to assess the performance of each clustering algorithm in predicting each botnet (class) and corroborate the results obtained by the CVIs. We chose the clustering method with the highest CVIs and AUC performance obtained in both experiments as the outcome of this phase and expected it to be the best in our comparison with other works.

After determining our performance with these metrics in both experiments, we compared our solution with others works from the literature. Dimitriadis et al. [34] have stated that direct comparison with other approaches is not feasible as different materials and methods are used by one for training and testing. Also, recent bot and botnet datasets cannot be shared publicly between researchers due to Twitter's privacy policy [18], reducing the viability of comparisons using the same data. However, Adewole et al. [40] proposed to extract the features introduced in other works to provide a reasonable and unbiased comparison. We followed this methodology to evaluate our solution with five approaches from the literature of bot detection and botnet detection on Twitter: Echeverria et al.'s [17] 2018 work, Khalil et al.'s [44] 2020 work, Adewole et al.'s [40] 2020 work, Fonseca et al.'s [45] 2020 work, and Barhate et al.'s [43] 2021 work. We have chosen these works because they meet many of our criteria for selection: they use datasets similar to ours, compare and outperform other approaches, follow a methodology comparable to ours, and their feature representation is adaptable for measuring similarity between bots.

For this comparison, we replaced our feature representation presented in Section IV-A2 with those of the five works. Although some of these representations and features do not measure similarity directly, we adapted them to match our methodology while preserving their original idea and objective.

Consequently, this feature representation replacement required changes in our original outline of the construction phase. We received $\mathcal{B}_{\text{Training}}$ from $\mathcal{BNT}_{\text{Training}}$, conducted the data preprocessing proposed in Section IV-A1, and obtained $\mathcal{BNT}_{\text{TrainingProc}}$ with $\mathcal{B}_{\text{TrainingProc}}$ as before. Then, we paired all bots with each other from the training botnets. Next, we chose one of the five feature representations from the literature or *SFLit* for implementation and extracted its associated similarity features by forming the pairings' tuples $\mathcal{T}_{\text{PairsTrainingLit}}$ of the selected representation. Finally, we trained *BstRegMethod* using $\mathcal{T}_{\text{PairsTrainingLit}}$ and generated a different regression model *TrRegModLit* for each work. For the detection phase, we worked without relevant changes, as we only used the trained regression model for the similarity score prediction. We ended up having our improved trained regression model *Smp-22* and those from the other works *Ech-18*, *Khl-20*, *Adw-20*, *Fsc-20*, and *Bht-21* to create their distance matrix and evaluate them with the same CVIs for each of the four clustering algorithms. We present in Fig. 4 the pipeline of the construction phase of our solution with the changes that incorporate the feature representation of the other works.

Finally, we expected the results of the clustering algorithms using our feature representation and regression model (especially the one chosen as the output of our detection phase) to be significantly higher than those of the other approaches in most of the CVIs employed. We believed that similarity features based on social interaction gave us a significant advantage over other works for detecting botnets on Twitter.

V. EXPERIMENTAL RESULTS

We show in this section the outcomes of performing our phases and the experiments proposed in the previous section. We also discuss their meaning and evaluate the success of our approach when detecting botnets against other works.

A. CONSTRUCTION PHASE IMPLEMENTATION

We received the training dataset $\mathcal{BNT}_{\text{Training}}$ as input containing 1,754 bots $\mathcal{B}_{\text{Training}}$ with their user, tweet, and social information from the six botnets stored within. We started by preprocessing the metadata from the bots ($\mathcal{BNT}_{\text{TrainingProc}}$ and $\mathcal{B}_{\text{TrainingProc}}$) and forming all their pairings, obtaining 1,537,381 pairs or $C(1754, 2)$. We then extracted the fourteen similarity features of our representation *SF* for each pair and created the tuples $\mathcal{T}_{\text{PairsTraining}}$ with their associated similarity score $\mathbf{SS} \in SS_{True}$. From the total pairings generated using the training dataset, we obtained 371,749 tuples of bot pairs belonging to the same botnet, or $\mathbf{SS} = 1$. The remaining

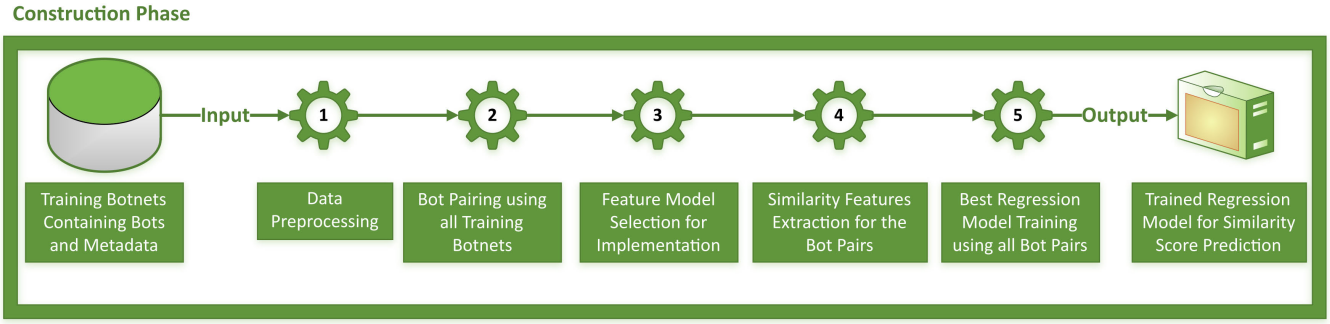


FIGURE 4. The changes to the construction phase for comparison. We received a set of bots from different botnets as input and processed their metadata to pair them with each other. We selected the feature representation from the other works and extracted its similarity features from the bot pairs. We trained a new regression model using all pairings to predict the similarity score between bots. We used this new model in the detection phase for the distance matrix construction without requiring significant changes.

1,165,632 tuples are from bot pairs belonging to different botnets, or $\mathbf{SS} = 0$.

1) REGRESSION METHODS EXPERIMENTS

We performed the fifteen experiments proposed to train the ten regression methods using a combination of four botnets' tuples serving as training $\mathcal{T}_{\text{ModelTrain}}$ and the remaining two botnets' tuples $\mathcal{T}_{\text{ModelTest}}$ for testing. Our training using the regression methods' default settings yielded compelling results, presented in Fig. 5. Linear-based methods overall performance is slightly higher than tree-based methods. From our results, we concluded that, in general, linear models are better suited for predicting the similarity score $\mathbf{SS} \in SS_{Pred}$ using the proposed feature representation.

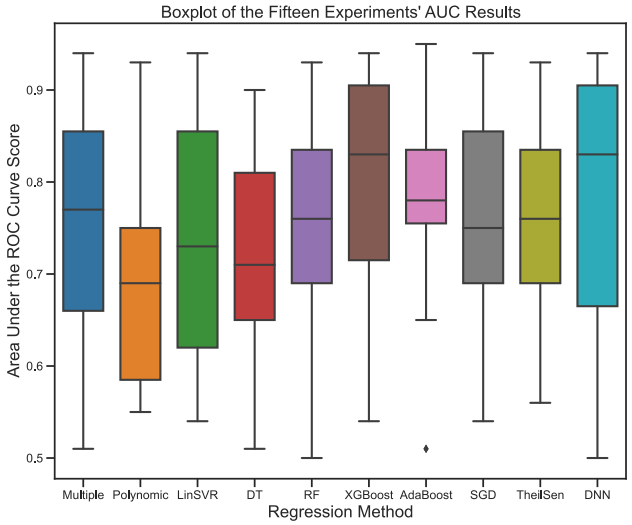


FIGURE 5. The AUC performance's boxplot of the ten regression methods in the fifteen experiments using their default settings: Multiple Linear Regression, Polynomial Regression, Linear Support Vector Regression, Decision Tree, Random Forest, XGBoost, Adaboost, Stochastic Gradient Descent, TheilSen, and Deep Neural Network. Tree-based methods obtained slightly worse results than their linear counterparts. Also, Deep Neural Network and XGBoost achieved the best results, with their median significantly above the others.

From the boxplot of Fig. 5, we identified two regression methods that obtained the best AUC performance of the ten: Deep Neural Network (DNN) and XGBoost. We made this assessment based on the medians of both boxes, which are the highest of all the ten methods. Also, their AUC for each experiment was among the highest. It is worth noting that for testing, older botnets (*FSF* and *TWT*, both from 2015) are harder to predict than newer ones (*SW*, *TS*3, *SS*1, and *SS*3, from 2017). When we employed these older botnets for testing, they decreased the performance of the ten regression methods for the last two experiments. However, when we used these old botnets as training botnets, they improved the scores significantly, most notably in the second experiment. Therefore, we highlight the importance of including old and new botnets in the training since both can provide characteristics and behaviors to identify unknown botnets.

2) COMPARISON OF BEST METHODS

We improved the AUC results for XGBoost and DNN using the fifteen experiments to select the best as our regression model *BstRegMethod*. We started with the DNN, where we tested different layouts, with the configuration of 6 hidden layers yielding the best scores. We used a dropout and a L2 regularizer to reduce the risks of overfitting and increase the network's performance [84]. After comparing all the available weight initialization techniques, He Uniform variance scaling initializer [85] converged faster for our problem. We employed the ReLU activation function in the input and hidden layers for its simplicity, speed, and linear-like behavior [86]. Also, we selected the sigmoid activation function [87] for the output layer to obtain the similarity score's prediction.

Next, we optimized the XGBoost AUC results. Starting with the type of model to use for the booster [88], we obtained better scores with a linear gradient booster rather than a tree one. This occurrence matches our previous results from the 15 experiments, where linear-based methods performed slightly better than those based on trees for the similarity score prediction. For the learning task [88], we used logistic

regression as it obtained a higher performance than binary classification or squared loss.

Then, we compared the two regression methods' results with their best configurations for the fifteen experiments. The boxplot diagram with the AUC results revealed that both performances are similar, with XGBoost having a slightly better median. Also, XGBoost converged faster than DNN in all our experiments. These occurrences coincide with the findings of Giannakas et al. [89] and Husna et al. [90], where XGBoost obtained a better performance against Neural Networks for regression problems.

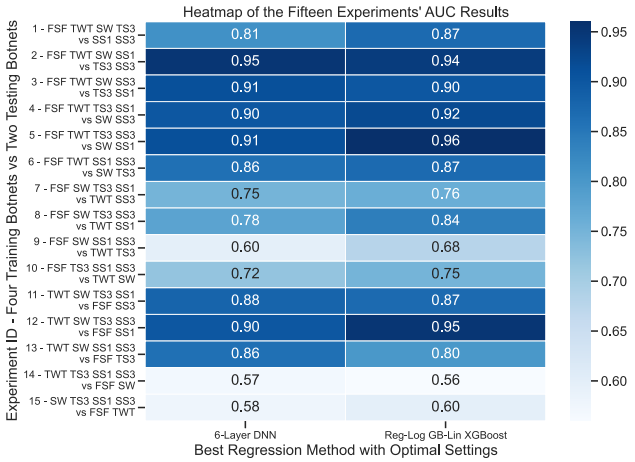


FIGURE 6. The heatmap of the AUC scores for the best regression methods with their optimal settings. The DNN employs six hidden layers with a single output layer for prediction, and XGBoost uses a linear gradient booster (GB-Lin) with a learning objective based on logistic regression (Reg-Log). The DNN has lower scores than XGBoost, only winning on five experiments.

3) WILCOXON SIGNED-RANK TEST

The heatmap of the AUC scores in Fig. 6 confirmed our findings in the boxplot diagram: both methods' results share similarities, but XGBoost outperformed DNN in ten of the fifteen experiments by a small margin. However, being these results not as conclusive as our comparison between the ten regression methods, we used the Wilcoxon signed-rank test to compare the AUC scores of DNN against XGBoost and determine if their differences are statistically significant.

Our null hypothesis h_0 for the Wilcoxon signed-rank test is that there is no difference between populations. The sum of positive ranks (in favor of DNN) is 27 (calculated by subtracting the DNN results from the XGBoost results) and 93 for the negatives (favoring XGBoost), so the W value is 27 as it is the minimum. Critical value C for our test is 25 ($n = 15$ with a two-tailed significance level $\alpha = 0.05$). Given that $\mathcal{W} > \mathcal{C}$, we cannot reject h_0, meaning the difference found between the populations is not statistically significant. However, we highlight that the sum of the negative rankings (93) in favor of XGBoost is significantly higher than that of the positive ones favoring DNN (27) by more than three times.

Given that the boxplot and heatmap diagrams showed that DNN performed slightly worse, the Wilcoxon signed-rank test's sum of rankings for XGBoost was the highest, and the DNN model converged slower in all the experiments, we decided XGBoost is the best suited for the similarity score prediction. We used the resulting trained XGBoost Regression model *TrainedRegMod* in the next phase's implementation for the distance matrix construction that leads to the clustering of bots into their corresponding botnets.

B. DETECTION PHASE IMPLEMENTATION

We received the detection dataset $\mathcal{BNT}_{\text{Detection}}$ of 1,826 bots $\mathcal{B}_{\text{Detection}}$ with their associated metadata from the four botnets stored within as input. We preprocessed the metadata of the bots (from $\mathcal{BNT}_{\text{DetectionProc}}$ and $\mathcal{B}_{\text{DetectionProc}}$) and constructed their pairings, resulting in 1,666,225 pairs or $C(1826, 2)$. Next, we extracted SF for each bot pair to create $\mathcal{T}_{\text{PairsDetection}}$ and $\mathbf{SS} \in SS_{True}$. We obtained 540,880 tuples with $\mathbf{SS} = 1$ and 1,125,345 tuples with $\mathbf{SS} = 0$.

1) FEATURE SELECTION PROCESS

We predicted the similarity score $\mathbf{SS} \in SS_{Pred}$ for each tuple inside $\mathcal{T}_{\text{PairsDetection}}$ using *TrainedRegMod* and compared them with SS_{True} to calculate our regression model's AUC performance. As stated before, we preferred AUC over other metrics because it provides a more comprehensive evaluation of our model's performance by considering both the true positive rate and the false positive rate, independent of class distribution. We performed our feature selection process using this performance evaluation to improve SF. We started by eliminating popularity, reputation, and reciprocity (social-based) because they contributed negatively to the model. The next three removed were usage, frequency, and profile (user-based) with a contribution near zero. Age (user-based) with sources, interactions, and hashtags (tweet-based) followed suit having a small contribution to the model. Then, we eliminated followers, followings, and likes (social-based) playing a significant role in the performance of our model, as their removal reduced the AUC by 0.27. At the end of our selection process, we had only URLs (tweet-based) left with an AUC of 0.51. We detail the results of this feature selection in Fig. 7.

After analyzing our outcomes, we concluded that we obtained the best results with eight features, as removing any of those eight from the model also diminished our AUC performance, which is not preferable. Therefore, we obtained the model *Smp-22* as a result with an AUC improved by 0.01 and SF reduced from fourteen to only eight similarity features: age (user-based), sources, interactions, hashtags, and URLs (tweet-based), and followers, followings, and likes (social-based). The feature with the most contribution for the top half of the selection process (from fourteen to eight) was the similarity in the bots' liked tweets, and for the bottom half (from seven to one feature), the similarity of employed URLs. Analyzing the performance of the features proposed in

our previous work [18], we discovered that only profile was eliminated for the top half selection. Interactions, sources, followers, followings, and likes contributed significantly to staying in the reduced feature model. Furthermore, the features based on social activity were among the highest contributors by being eliminated last in the bottom half selection, only surpassed by URLs from tweet activity.

Making a comparison between categories, the user-based one was the least contributing. Three-fourths of its features were eliminated for the reduced feature model, and none of them increased our performance notably, with about 0.001 in AUC performance on average. In the middle are the features from the social-based category, which contributed 0.046 on average. Half of them diminished slightly the model's performance slightly (reputation, popularity, and reciprocity), but the other three were among the highest contributors to performance (followers, followings, and likes). Lastly, the tweet-based category improved the model by 0.135 on average, the highest of the three. Also, all their features made it to the reduced feature model due to their significant contribution.

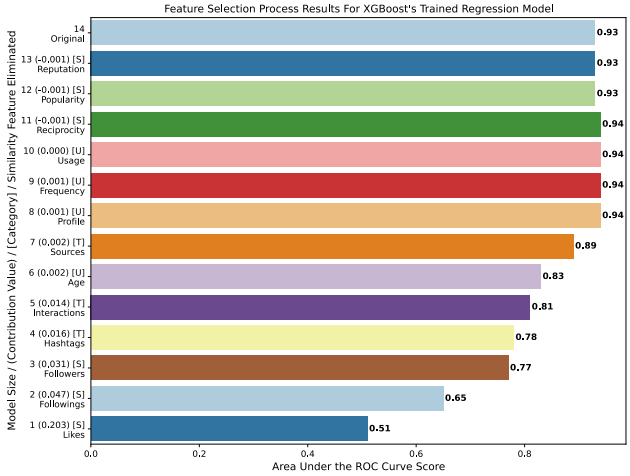


FIGURE 7. The results of the feature selection process for our trained regression model. The x-axis displays the AUC performance of each model, and the y-axis the number of features used with the contribution value and category (user [U], tweet [T], and social [S]) of the last feature eliminated along with its name. The model started with fourteen and ended with one: URLs from the tweet-based category with 0.51. We selected the eight-features model because it improves the AUC performance while reducing our original model.

2) EXPERIMENTS' CONFIGURATION AND HYPER-PARAMETERS TUNING

We used *Smp-22* to predict SS_{Pred} for all the pairs. We created a distance matrix *DistMtx* of size $1,826 \times 1,826$ with these predictions and used it as input for the four clustering algorithms chosen. For the default settings experiment, we employed the standard configuration of the clustering algorithms from the libraries [91], [92], [93], only specifying the number of clusters searched, which in this work are the four botnets from $\mathcal{BNT}_{\text{DetectionProc}}$. Regarding the best configuration experiment, we opted for a manual exploration approach to reduce the computational and memory cost of implementing a full grid search, as some of the hyper-parameters were numeric, making the model evaluation grow exponentially [94]. Also, using a manual approach lets us efficiently explore the hyper-parameter combinations and space, as we could adapt our tweaking based on the results of previous iterations and discard those producing redundant outcomes [95].

The hyper-parameters tuned for K-Medoids clustering were the distance metric we used, the medoid initialization method, and the algorithm used as the solver. For HDBSCAN clustering, we varied the minimum size to be considered a cluster and the density of such clusters. The tuning for Agglomerative clustering was the minimum distance between clusters to be merged (linkage), and for Spectral clustering, the strategy followed for assigning labels in the spectral domain.

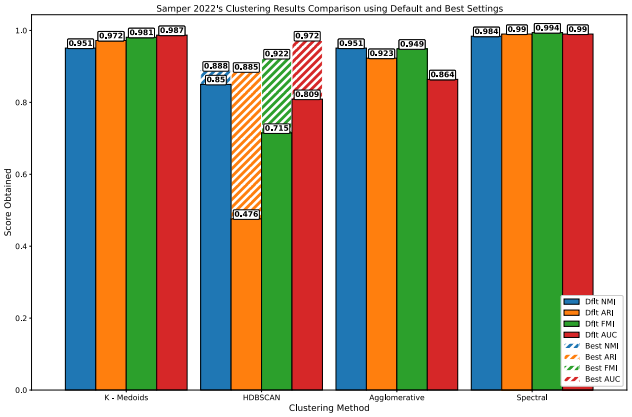


FIGURE 8. The NMI, ARI, FMI, and AUC performances of K-Medoids, HDBSCAN, Agglomerative, and Spectral clustering algorithms. The colored bars present the results of the algorithms with their default settings, and the striped bars show the ones obtained using their best configuration. K-Medoids, Agglomerative, and Spectral clustering achieved the same results for the four metrics in both settings, and only HDBSCAN requires hyper-parameter tuning to achieve its best performance.

3) CLUSTERING EVALUATION

From the experiments' results, we determined that the performance of K-Medoids, Agglomerative, and Spectral is the same in both settings, and only HDBSCAN performs worse at its default settings in the four metrics. This outcome means most clustering algorithms chosen using our *DistMtx* do not require hyper-parameter tweaking to yield their best performance. However, for HDBSCAN, the results are different. Being a density-based hierarchical algorithm, McInnes et al. [74] have stated it is sensitive to its initial parameters, and thus poor performance is expected in its default configuration. After varying its hyper-parameters, we obtained scores in the four metrics similar to the other three algorithms. We present the full results of the four clustering algorithms in Fig. 8.

We determined from the obtained CVIs and AUC scores that Spectral clustering was the highest of the four, choosing it as the best clustering method and output of the detection phase.

4) MULTI-DIMENSIONAL SCALING

We employed Multi-Dimensional Scaling or MDS [96] to create a new visualization of the clustering results. We used our *DistMtx* as input for MDS and painted the resulting nodes (bots) according to their predicted clustering label (botnet) from CL_{Pred}. Using the true clustering labels CL_{True} as the ground truth, we marked with a black cross those CL_{Pred} that do not match the botnet they belong to.

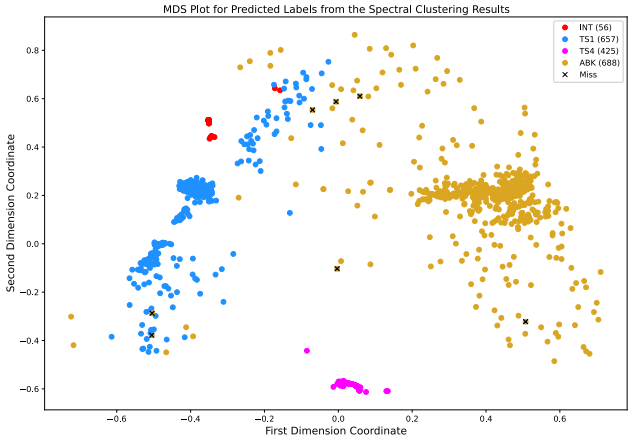


FIGURE 9. The MDS plot of bots from the detection dataset using the distance matrix created from our similarity score and colored according to the predicted labels by the Spectral clustering algorithm. Each color represents a botnet with its size on the side and a black cross indicating a mistake in the prediction. The high evaluation metrics' results coincide with the minimum presence of black crosses throughout the figure.

The resulting figures confirm our findings when evaluating the clustering algorithms: on the one hand, lower values in the four metrics used translate into many black crosses in the plot, the product of erroneous predictions. On the other hand, higher values in the metrics used translate into a figure with few black crosses, as the clustering algorithm predicted the botnets correctly. We can also distinguish three of the four botnets (*TS*1, *TS*4, *ABK*) from the detection dataset $\mathcal{BNT}_{\text{Detection}}$ in the plots. Each has its recognizable area, with *TS*4 being the most cohesive and separated from the others. The last one (*INT*) does not stand out due to its size (56 members), but it is visible in the northwest area of the plot. Finally, these findings allow us to verify the performance of our similarity score since we built these graphs from the distance matrix and thus let us identify the main clusters composing the botnets. For space and convenience, we only display the resulting plot of the algorithm with the best performance, Spectral clustering (Fig. 9). We will publish the other figures for the remaining clustering algorithms in a repository once the paper is accepted.

TABLE 5. The feature representations chosen for comparison, gathered from different works in the literature and our improved approach after the feature selection process. The first column states the original work of the feature representation, the second the year it was published, the third to fifth the number of features based on user, tweet, and social activity, and the last the total features.

Original Work	Year	User	Tweet	Social	Total
Echeverria et al. [17]	2018	10	19	0	29
Khalil et al. [44]	2020	4	4	0	8
Adewole et al. [40]	2020	8	14	0	22
Fonseca et al. [45]	2020	5	0	0	5
Barhate et al. [43]	2021	13	0	0	13
Our Representation	2022	1	4	3	8

5) COMPARISON WITH OTHER WORKS

We implemented each of the five feature representations *SFLit* from the literature and calculated their pairings' tuples $\mathcal{T}_{\text{PairsTrainingLit}}$ to train their regression model *BstRegMethod*. We present a summary of the feature representations proposed for this comparison in Table 5.

We obtained the regression models from the literature *Ech-18*, *Khl-20*, *Adw-20*, *Fsc-20*, and *Bht-21* to compare them with our *Smp-22* following the experiment descriptions from Section V-B2. We present the results of these evaluations in Table 6.

For Spectral clustering, *Smp-22* obtained the best results in all metrics by a large margin using their default and best settings, which we expected from our highest performance method in the detection phase. Also, K-Medoids clustering obtained similar scores, outperforming the other five models by a statistically significant margin in both settings.

Ech-18 obtained the second best only in Spectral clustering for the four metrics; *Adw-20* surpassed it for K-Medoids clustering in all metrics. However, their scores in the evaluation metrics were still far from those obtained by our model. *Khl-20*, *Fsc-20*, and *Bht-21*'s performance were disappointing in the experiments with default settings for both clustering algorithms, but *Khl-20* improved considerably using the best configuration, so we consider it the fourth best in general. *Khl-20* and *Bht-21* obtained the lowest scores, with the latter being the worst overall in both experiments.

The results of the Agglomerative clustering present an interesting case: while *Smp-22* was the best in both configurations by a considerable margin, the second best changes in every metric of the experiment using best settings, and for default settings, the scores of the five literature regression models are low and near each other for ARI, FMI, and AUC. We associate this occurrence with the linkage used by default in the algorithm, which minimizes the variance between clusters. Data points (bots) created from the distance matrices of the literature's regression models are not separated enough from each other, so the default configuration of the algorithm ends up generalizing the clusters, giving low and similar results.

The outcome of the two experiments using HDBSCAN Clustering presents contrasting results. Using the default configuration, *Smp-22* only won in NMI with a significant margin; *Adw-20* surpassed it in every other metric.

TABLE 6. The CVIs and AUC results of each model for comparison, employing their default and best settings. For the default configuration, our model obtained the best scores in all metrics for K-Medoids, Agglomerative, and Spectral, and won in NMI by a significant margin for HDBSCAN. For the best, our model obtained the highest scores in all metrics for all the clustering methods. The first column states the name of the clustering algorithm, the second the feature model evaluated, and from the third to the tenth the CVIs and AUC score with default and best settings, with the best marked in bold.

Method	Model	NMI-Dflt	NMI-Best	ARI-Dflt	ARI-Best	FMI-Dflt	FMI-Best	AUC-Dflt	AUC-Best
K-Medoids	*Ech-18*	0.232	0.553	0.171	0.477	0.424	0.631	0.632	0.724
	Khl-20	0.206	0.412	0.119	0.351	0.376	0.540	0.671	0.697
	Adw-20	0.202	0.562	0.218	0.507	0.450	0.655	0.629	0.744
	Fsc-20	0.385	0.415	0.327	0.368	0.523	0.551	0.686	0.686
	Bht-21	0.407	0.407	0.290	0.318	0.527	0.556	0.686	0.715
	Smp-22	**0.951**	**0.951**	**0.972**	**0.972**	**0.981**	**0.981**	**0.987**	**0.987**
HDBSCAN	*Ech-18*	0.598	0.654	0.584	0.657	0.720	0.763	0.799	0.896
	Khl-20	0.408	0.595	0.175	0.578	0.541	0.707	0.624	0.854
	Adw-20	0.672	0.672	**0.699**	0.699	**0.795**	0.795	**0.819**	0.819
	Fsc-20	0.383	0.479	0.304	0.443	0.563	0.620	0.679	0.807
	Bht-21	0.386	0.442	0.191	0.252	0.543	0.572	0.624	0.673
	Smp-22	**0.850**	**0.888**	0.476	**0.885**	0.715	**0.922**	0.809	**0.972**
Agglomerative	*Ech-18*	0.397	0.424	0.031	0.308	0.566	0.593	0.573	0.703
	Khl-20	0.330	0.437	0.028	0.328	0.564	0.564	0.576	0.716
	Adw-20	0.640	0.640	0.034	0.228	0.578	0.578	0.575	0.682
	Fsc-20	0.249	0.249	0.024	0.024	0.548	0.548	0.585	0.585
	Bht-21	0.126	0.461	0.001	0.234	0.556	0.584	0.501	0.696
	Smp-22	**0.951**	**0.951**	**0.923**	**0.923**	**0.949**	**0.949**	**0.864**	**0.864**
Spectral	*Ech-18*	0.682	0.682	0.650	0.650	0.764	0.764	0.777	0.777
	Khl-20	0.580	0.580	0.496	0.496	0.647	0.647	0.712	0.712
	Adw-20	0.576	0.576	0.489	0.489	0.641	0.641	0.703	0.703
	Fsc-20	0.549	0.549	0.459	0.459	0.620	0.620	0.686	0.686
	Bht-21	0.434	0.434	0.325	0.325	0.557	0.557	0.687	0.687
	Smp-22	**0.984**	**0.984**	**0.990**	**0.990**	**0.994**	**0.994**	**0.990**	**0.990**

For the scores with the best configurations, *Smp-22* won in all metrics, with *Adw-20* being a distant second. We relate this situation to the sensibility of the algorithm to hyper-parameter configuration [74]. While *Adw-20* reached its best performance with the default settings, *Smp-22* required some tuning to obtain its best scores, them being significantly higher than *Adw-20*. We detail the outcome of HDBSCAN clustering for both experiments in Fig. 10. The figures for the K-Medoids, Agglomerative, and Spectral clustering results will be available in a repository once the paper is accepted.

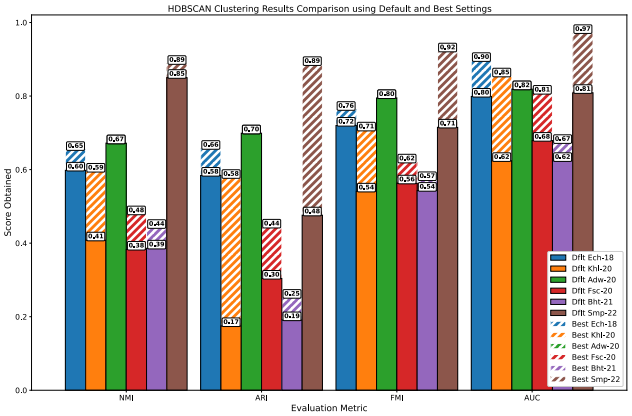


FIGURE 10. The performance of the HDBSCAN clustering algorithm in NMI, ARI, FMI, and AUC for the comparison of the six regression models: *Ech-18*, *Khl-20*, *Adw-20*, *Fsc-20*, *Bht-21*, and our approach *Smp-22*. The colored bars present the scores with the default settings, and the striped bars the ones obtained using the best configurations. *Smp-22* only won in one metric in the first and all metrics in the second. *Adw-20* model had the overall best score for default settings, followed closely by *Ech-18* model. The remaining models performed similarly, with *Bht-21* model being the worst overall.

C. DISCUSSION

From the experiments' results, we conclude that our proposed regression model achieved outstanding performance in three of the four clustering algorithms in both settings, with the method chosen in the detection phase (Spectral clustering) being the overall best. Our solution reached its best scores for most algorithms without any additional configuration, providing reliability and certainty for botnet identification in the wild, where hyper-parameter tweaking is unreliable or tricky. Concerning time complexity in a real-life implementation, it initially requires polynomial time for the bot pairings calculations from the construction phase. However, the detection phase is flexible enough without much re-configuration to identify botnets from a starting set of bots and incorporate new bots as we discover them without needing to calculate all the bot pairings again. It only requires the calculation of the new pairings associated with these recently discovered automated accounts to assign them to a botnet. Also, our solution recognizes botnets of different sizes, objectives, and years disregarding the clustering algorithm used without losing performance, making our model more generalizable and better prepared for botnets not encountered before. Although it was not the best in HDBSCAN's default configuration, it had a remarkable performance competing for the top places, allowing us to trust in the utility of our chosen features and the model's versatility to adapt to adverse situations.

VI. CONCLUSION AND FUTURE WORK

In this paper, we have presented a novel solution for the problem of botnet identification on Twitter. Our solution leverages social information and the similarities between accounts to create clusters of bots sharing attributes, connections, and objectives.

The implementation of our solution showed that it obtained its best performance in most clustering algorithms without requiring additional configuration. Also, in our comparison with five other works from the literature, our mechanism outperformed them by a statistically significant margin, achieving the best overall scores in every evaluation metric. Based on these results, we concluded that incorporating social features into our model allowed us a better separation of objects in the distance matrix, which translated into better performance for the default configurations of most of the clustering algorithms.

The encouraging results obtained with unknown botnets, using both proposed configurations (default and best), allows us to confirm the reduction of bias towards the botnets used for training. Furthermore, we have been able to guarantee the capacity of our solution for effective operation without relying on a specific algorithm or parameter, as well as our performance not being affected by groups (botnets) of varying characteristics (e.g., size, objective).

We believe this work can become a turning point for developing future detection mechanisms so researchers consider incorporating accounts' social relations more frequently. Although this relationship information can be costly in time and computer processing compared to the user and tweet information, the benefits can ensure an advantage in this uneven arms race against botmasters and their harmful intentions.

For future work, we plan to gather newer botnets from Twitter to evaluate their evolutionary changes, aimed at avoiding detection over the years, might affect our solution's performance. Additionally, we would like to adapt our mechanism to work directly on the platform and assess what type of suspicious groups we can discover in the wild.

ACKNOWLEDGMENT

The authors would like to thank the members of the Advanced Artificial Intelligence group at Tecnologico de Monterrey for providing feedback on this research, and the National Council of Humanities, Science, and Technology of Mexico (CONAHCyT) for financing this work through the scholarship grant 540975.

REFERENCES

[1] R. Alharthi, A. Alhothali, and K. Moria, "Detecting and characterizing Arab spammers campaigns in Twitter," *Proc. Comput. Sci.*, vol. 163, pp. 248–256, Jan. 2019.

[2] T. Elmas, "Analyzing activity and suspension patterns of Twitter bots attacking Turkish Twitter trends by a longitudinal dataset," in *Proc. Companion ACM Web Conf.*, Apr. 2023, pp. 1404–1412.

[3] Y. Zhang, J. Ma, and F. Fang, "How social bots can influence public opinion more effectively: Right connection strategy," *Phys. A, Stat. Mech. Appl.*, vol. 633, Jan. 2024, Art. no. 129386.

[4] O. E. Taylor and P. S. Ezekiel, "A smart system for detecting behavioural botnet attacks using random forest classifier with principal component analysis," *Eur. J. Artif. Intell. Mach. Learn.*, vol. 1, no. 2, pp. 11–16, Mar. 2022.

[5] P. R. Biju and O. Gayathri, "Self-breeding fake news: Bots and artificial intelligence perpetuate social polarization in India's conflict zones," *Int. J. Inf., Diversity, Inclusion (IJIDI)*, vol. 7, no. 1, pp. 1–25, Apr. 2023.

[6] Y. Lee, M. Ozer, S. R. Corman, and H. Davulcu, "Detecting and measuring the polarization effects of adversarial botnets on Twitter," in *Proc. 38th ACM/SIGAPP Symp. Appl. Comput.* New York, NY, USA: Association for Computing Machinery, Jun. 2023, pp. 1641–1649.

[7] O. Beatson, R. Gibson, M. C. Cunill, and M. Elliot, "Automation on Twitter: Measuring the effectiveness of approaches to bot detection," *Social Sci. Comput. Rev.*, vol. 41, no. 1, pp. 181–200, Feb. 2023.

[8] M. Fukuda, K. Nakajima, and K. Shudo, "Estimating the bot population on Twitter via random walk based sampling," *IEEE Access*, vol. 10, pp. 17201–17211, 2022.

[9] O. Loyola-González, R. Monroy, M. A. Medina-Pérez, B. Cervantes, and J. E. Grimaldo-Tijerina, "An approach based on contrast patterns for bot detection on web log files," in *Advances in Soft Computing*. Cham, Switzerland: Springer, Oct. 2018, pp. 276–285.

[10] I. Lin, O. Loyola-González, R. Monroy, and M. A. Medina-Pérez, "A review of fuzzy and pattern-based approaches for class imbalance problems," *Appl. Sci.*, vol. 11, no. 14, p. 6310, Jul. 2021.

[11] O. Loyola-González, M. A. Medina-Pérez, and K.-K.-R. Choo, "A review of supervised classification based on contrast patterns: Applications, trends, and challenges," *J. Grid Comput.*, vol. 18, no. 4, pp. 797–845, Dec. 2020.

[12] O. Loyola-González, J. F. C. O. Martínez-Trinidad, J. A. Carrasco-Ochoa, and M. García-Borroto, "Cost-sensitive pattern-based classification for class imbalance problems," *IEEE Access*, vol. 7, pp. 60411–60427, 2019.

[13] R. Shukla, A. Sinha, and A. Chaudhary, "TweezBot: An AI-driven online media bot identification algorithm for Twitter social networks," *Electronics*, vol. 11, no. 5, p. 743, Feb. 2022.

[14] S. Lopez-Joya, J. A. Diaz-Garcia, M. D. Ruiz, and M. J. Martin-Bautista, "Bot detection in Twitter: An overview," in *Flexible Query Answering Systems*. Cham, Switzerland: Springer, Sep. 2023, pp. 131–144.

[15] W. Shahid, Y. Li, D. Staples, G. Amin, S. Hakak, and A. Ghorbani, "Are you a cyborg, bot or human—A survey on detecting fake news spreaders," *IEEE Access*, vol. 10, pp. 27069–27083, 2022.

[16] M. R. Sethurajan and K. Natarajan, "Twitter bot detection to deter bogus news using machine learning algorithms," in *Proc. Int. Conf. Commun. Comput. Technol. (ICCCT)*. Singapore: Springer, Sep. 2023, pp. 799–815.

[17] J. Echeverria, E. De Cristofaro, N. Kourtellis, I. Leontiadis, G. Stringhini, and S. Zhou, "LOBO: Evaluation of generalization deficiencies in Twitter bot classifiers," in *Proc. 34th Annu. Comput. Secur. Appl. Conf.*, vol. 6, Dec. 2018, pp. 137–146.

[18] L. D. Samper-Escalante, O. Loyola-González, R. Monroy, and M. A. Medina-Pérez, "Bot datasets on Twitter: Analysis and challenges," *Appl. Sci.*, vol. 11, no. 9, p. 4105, Apr. 2021.

[19] R. Cantini, F. Marozzo, D. Talia, and P. Trunfio, "Analyzing political polarization on social media by deleting bot spamming," *Big Data Cognit. Comput.*, vol. 6, no. 1, p. 3, Jan. 2022.

[20] C. Ruiz-Núñez, S. Segado-Fernández, B. Jiménez-Gómez, P. J. J. Hidalgo, C. S. R. Magdalena, M. D. C. Á. Pollo, A. Santillán-Garcia, and I. Herrera-Peco, "Bots' activity on COVID-19 pro and anti-vaccination networks: Analysis of spanish-written messages on Twitter," *Vaccines*, vol. 10, no. 8, p. 1240, Aug. 2022.

[21] Y. Tian and Y. Xie, "Artificial cheerleading in IEO: Marketing campaign or pump and dump scheme," *Inf. Process. Manage.*, vol. 61, no. 1, Jan. 2024, Art. no. 103537.

[22] Y. Roth. (May 2020). *Bot or Not? The Facts About Platform Manipulation on Twitter*. Accessed: Jan. 23, 2024. [Online]. Available: https://blog.twitter.com/en_us/topics/company/2020/bot-or-not.html

[23] C. Besel, J. Echeverria, and S. Zhou, "Full cycle analysis of a large-scale botnet attack on Twitter," in *Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining (ASONAM)*, Aug. 2018, pp. 170–177.

[24] M. Mazza, S. Cresci, M. Avvenuti, W. Quattrociocchi, and M. Tesconi, "RTbust: Exploiting temporal patterns for botnet detection on Twitter," in *Proc. 10th ACM Conf. Web Sci.*, Jun. 2019, pp. 183–192.

[25] J. Echeverria and S. Zhou, "Discovery, retrieval, and analysis of the 'Star Wars' botnet in Twitter," in *Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining*, Jul. 2017, pp. 1–8.

[26] E. Alothali, M. Salih, K. Hayawi, and H. Alashwal, "Bot-MGAT: A transfer learning model based on a multi-view graph attention network to detect social bots," *Appl. Sci.*, vol. 12, no. 16, p. 8117, Aug. 2022.

[27] Y. Qiao, X. Luo, J. Ma, M. Zhang, and C. Li, "Twitter user geolocation based on heterogeneous relationship modeling and representation learning," *Inf. Sci.*, vol. 647, Nov. 2023, Art. no. 119427.

[28] T. Alshammari, "Using coloured acyclic nets to detect fake accounts on social media," in *Proc. 8th Int. Conf. Comput., Softw. Model. (ICCSM)*, vol. 2424, Jul. 2024, pp. 24–30.

[29] G. Lingam, R. R. Rout, D. Somayajulu, and S. K. Das, "Social botnet community detection: A novel approach based on behavioral similarity in Twitter network using deep learning," in *Proc. 15th ACM Asia Conf. Comput. Commun. Secur.*, Oct. 2020, pp. 708–718.

[30] F. K. Alarfaj, H. Ahmad, H. U. Khan, A. M. Alomair, N. Almusallam, and M. Ahmed, "Twitter bot detection using diverse content features and applying machine learning algorithms," *Sustainability*, vol. 15, no. 8, p. 6662, Apr. 2023.

[31] S. Cresci, "A decade of social bot detection," *Commun. ACM*, vol. 63, no. 10, pp. 72–83, Sep. 2020.

[32] N. Saxena, A. Sinha, T. Bansal, and A. Wadhwa, "A statistical approach for reducing misinformation propagation on Twitter social media," *Inf. Process. Manage.*, vol. 60, no. 4, Jul. 2023, Art. no. 103360.

[33] M. Fazil and M. Abulaish, "A socialbots analysis-driven graph-based approach for identifying coordinated campaigns in Twitter," *J. Intell. Fuzzy Syst.*, vol. 38, no. 3, pp. 2961–2977, Mar. 2020.

[34] I. Dimitriadis, K. Georgiou, and A. Vakali, "Social botomics: A systematic ensemble ML approach for explainable and multi-class bot detection," *Appl. Sci.*, vol. 11, no. 21, p. 9857, Oct. 2021.

[35] C. Anderson, J. Conwell, and T. Saleh, "Investigating cyber attacks using domain and DNS data," *Netw. Secur.*, vol. 2021, no. 3, pp. 6–8, Mar. 2021.

[36] A. Affinito, S. Zinno, G. Stanco, A. Botta, and G. Ventre, "The evolution of Mirai botnet scans over a six-year period," *J. Inf. Secur. Appl.*, vol. 79, Dec. 2023, Art. no. 103629.

[37] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, "Fame for sale: Efficient detection of fake Twitter followers," *Decis. Support Syst.*, vol. 80, pp. 56–71, Dec. 2015.

[38] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, "The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race," in *Proc. 26th Int. Conf. World Wide Web Companion*, 2017, pp. 963–972.

[39] S. Kaddoura and S. Henno, "Dataset of Arabic spam and ham tweets," *Data Brief*, vol. 52, Feb. 2024, Art. no. 109904.

[40] K. S. Adewole, T. Han, W. Wu, H. Song, and A. K. Sangaiah, "Twitter spam account detection based on clustering and classification methods," *J. Supercomput.*, vol. 76, no. 7, pp. 4802–4837, Jul. 2020.

[41] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, pp. 5–32, Oct. 2001.

[42] Twitter Public Policy. (Jan. 2018). *Update on Twitter's Review of the 2016 Us Election*. Accessed: Jan 23, 2024. [Online]. Available: https://blog.twitter.com/official/en_us/topics/company/2018/2016-election-update.html

[43] S. Barhate, R. Mangla, D. Panjwani, S. Gatkal, and F. Kazi, "Twitter bot detection and their influence in hashtag manipulation," in *Proc. IEEE 17th India Council Int. Conf. (INDICON)*, Dec. 2020, pp. 1–7.

[44] H. Khalil, M. U. S. Khan, and M. Ali, "Feature selection for unsupervised bot detection," in *Proc. 3rd Int. Conf. Comput., Math. Eng. Technol. (iCoMET)*, Jan. 2020, pp. 1–7.

[45] J. V. Fonseca Abreu, C. Ghedini Ralha, and J. J. Costa Gondim, "Twitter bot detection with reduced feature set," in *Proc. IEEE Int. Conf. Intell. Secur. Informat. (ISI)*, Nov. 2020, pp. 1–6.

[46] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "DBSCAN revisited, revisited: Why and how you should (still) use DBSCAN," *ACM Trans. Database Syst.*, vol. 42, no. 3, pp. 1–21, Sep. 2017.

[47] D. Sculley, "Web-scale k-means clustering," in *Proc. 19th Int. Conf. World Wide Web*. New York, NY, USA: Association for Computing Machinery, Apr. 2010, pp. 1177–1178.

[48] J. Rodríguez-Ruiz, J. I. Mata-Sánchez, R. Monroy, O. Loyola-González, and A. López-Cuevas, "A one-class classification approach for bot detection on Twitter," *Comput. Secur.*, vol. 91, Apr. 2020, Art. no. 101715.

[49] S. Feng, H. Wan, N. Wang, J. Li, and M. Luo, "TwiBot-20: A comprehensive Twitter bot detection benchmark," in *Proc. 30th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2021, pp. 4485–4494.

[50] T-Developer. (Jan. 2024). *Data Dictionary: The Set of Features That Can Be Extracted From the Twitter API Regarding a User's Public Information*. Accessed: Jan 17, 2024. [Online]. Available: https://developer.twitter.com/en/docs/twitter-api/data-dictionary

[51] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995.

[52] S. H. Javaheri, M. M. Sepehri, and B. Teimourpour, "Chapter 6—Response modeling in direct marketing: A data mining-based approach for target selection," in *Data Mining Applications With R*. Cambridge, MA, USA: Academic Press, Jan. 2014, pp. 153–180.

[53] Neo4j. (2021). *Jaccard Similarity*. Accessed: Nov. 4, 2021. [Online]. Available: https://neo4j.com/docs/graph-data-science/current/alpha-algorithms/jaccard/

[54] J. Wright and O. Anise, "Don't@me: Hunting Twitter bots at scale," in *Proc. Black Hat (BlackHat)*, Aug. 2018, pp. 1–43.

[55] L. Ilias and I. Roussaki, "Detecting malicious activity in Twitter using deep learning techniques," *Appl. Soft Comput.*, vol. 107, Aug. 2021, Art. no. 107360.

[56] M. Kantepe and M. C. Ganiz, "Preprocessing framework for Twitter bot detection," in *Proc. Int. Conf. Comput. Sci. Eng. (UBMK)*, Oct. 2017, pp. 630–634.

[57] A. Dorri, M. Abadi, and M. Dadfarnia, "SocialBotHunter: Botnet detection in Twitter-like social networking services using semi-supervised collective classification," in *Proc. IEEE 16th Int. Conf Dependable, Autonomic Secure Comput., 16th Int. Conf Pervasive Intell. Comput., 4th Int. Conf Big Data Intell. Comput. Cyber Sci. Technol. Congr. (DASC/PiCom/DataCom/CyberSciTech)*, Aug. 2018, pp. 496–503.

[58] X. Yan and X. G. Su, *Linear Regression Analysis: Theory and Computing*. Singapore: World Scientific, Jun. 2009.

[59] S. M. Stigler, "Gergonne's 1815 paper on the design and analysis of polynomial regression experiments," *Historia Mathematica*, vol. 1, no. 4, pp. 431–439, Nov. 1974.

[60] J. Fürnkranz, "Decision tree," in *Encyclopedia of Machine Learning*. Berlin, Germany: Springer, Jan. 2010, pp. 263–267.

[61] O. Loyola-González, "Black-box vs. white-box: Understanding their advantages and weaknesses from a practical point of view," *IEEE Access*, vol. 7, pp. 154096–154113, 2019.

[62] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 785–794.

[63] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Mach. Learn.*, vol. 37, no. 3, pp. 297–336, Dec. 1999.

[64] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proc. 19th Int. Conf. Comput. Statist. (COMPSTAT)*. Heidelberg, Germany: Springer, Aug. 2010, pp. 177–187.

[65] P. K. Sen, "Estimates of the regression coefficient based on Kendall's tau," *J. Amer. Stat. Assoc.*, vol. 63, no. 324, p. 1379, Dec. 1968.

[66] A. Krogh, "What are artificial neural networks?" *Nature Biotechnol.*, vol. 26, no. 2, pp. 195–197, Feb. 2008.

[67] D. Dukic, D. Keca, and D. Stipic, "Are you human? Detecting bots on Twitter using BERT," in *Proc. IEEE 7th Int. Conf. Data Sci. Adv. Analytics (DSAA)*, Oct. 2020, pp. 631–636.

[68] R. Kaur, S. Singh, and H. Kumar, "An intrinsic authorship verification technique for compromised account detection in social networks," *Soft Comput.*, vol. 25, no. 6, pp. 4345–4366, Mar. 2021.

[69] S. Lakshmanan. (May 2019). *How, When, and Why Should You Normalize/Standardize/Rescale Your Data?* Accessed: May 5, 2022. [Online]. Available: https://towardsai.net/p/data-science/how-when-and-why-should-you-normalize-standardize-rescale-your-data

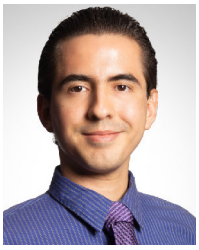
[70] O. Loyola-González, M. A. Medina-Pérez, J. F. Martínez-Trinidad, J. A. Carrasco-Ochoa, R. Monroy, and M. García-Borroto, "PBC4cip: A new contrast pattern-based classifier for class imbalance problems," *Knowl.-Based Syst.*, vol. 115, pp. 100–109, Jan. 2017.

[71] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bull.*, vol. 1, no. 6, p. 80, Dec. 1945.

[72] F. Harrel, "Chapter 10—Binary logistic regression," in *Regression Modeling Strategies*. Cham, Switzerland: Springer, Aug. 2015, p. 219.

[73] L. Kaufman and P. Rousseeuw, "Chapter 2—Partitioning around medoids (program PAM)," in *Finding Groups in Data*. Hoboken, NJ, USA: Wiley, Mar. 1990, pp. 68–125.

[74] L. McInnes, J. Healy, and S. Astels, "Hdbscan: Hierarchical density based clustering," *J. Open Source Softw.*, vol. 2, no. 11, p. 205, Mar. 2017.
[75] M. R. Anderberg, "Chapter 6—Hierarchical clustering methods," in *Cluster Analysis for Applications*. Cambridge, MA, USA: Academic Press, Jan. 1973, pp. 131–155.
[76] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, Aug. 2000.
[77] X. Jin and J. Han, "Chapter 12—K-medoids clustering," in *Encyclopedia of Machine Learning*. Berlin, Germany: Springer, Nov. 2010, p. 564.
[78] X. Jin and J. Han, "Chapter 12—K-way spectral clustering," in *Encyclopedia of Machine Learning*. Berlin, Germany: Springer, Nov. 2010, p. 565.
[79] T. Kvalseth, "On normalized mutual information: Measure derivations and properties," *Entropy*, vol. 19, no. 11, p. 631, Nov. 2017.
[80] J. M. Santos and M. Embrechts, "On the use of the adjusted Rand index as a metric for evaluating supervised classification," in *Proc. Int. Conf. Artif. Neural Netw. (ICANN)*. Berlin, Germany: Springer, Sep. 2009, pp. 175–184.
[81] E. B. Fowlkes and C. L. Mallows, "A method for comparing two hierarchical clusterings," *J. Amer. Stat. Assoc.*, vol. 78, no. 383, pp. 553–569, Sep. 1983.
[82] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Nov. 2011.
[83] D. J. Hand and R. J. Till, "A simple generalization of the area under the ROC curve for multiple class classification problems," *Mach. Learn.*, vol. 45, pp. 171–186, Nov. 2001.
[84] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, Jun. 2014.
[85] TensorFlow Core v2.8.0. (Apr. 2022). *Tf.Keras.Initializers.HeUniform*. Accessed: May 5, 2022. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/initializers/HeUniform
[86] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. 14th Int. Conf. Artif. Intell. Statist.*, 2011, pp. 315–323.
[87] TensorFlow Core v2.8.0. (Feb. 2022). *Tf.Keras.Activations.Sigmoid*. TensorFlow. Accessed: May 5, 2022. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/activations/sigmoid
[88] X. Developers. (Jan. 2021). *XGBoost Parameters*. XGBoost Documentation. Accessed: Apr. 28, 2022. [Online]. Available: https://xgboost.readthedocs.io/en/stable/parameter.html
[89] F. Giannakas, C. Troussas, A. Krouska, C. Sgouropoulou, and I. Voyiatzis, "Xgboost and deep neural network comparison: The case of teams' performance," in *Intelligent Tutoring Systems*. Berlin, Germany: Springer, Jul. 2021, pp. 343–349.
[90] N. A. Husna, A. Bustamam, A. Yanuar, D. Sarwinda, and O. Hermansyah, "The comparison of machine learning methods for prediction study of type 2 diabetes mellitus's drug design," in *Proc. Symp. Biomathematics*, Sep. 2020, p. 30010.
[91] L. McInnes, J. Healy, and S. Astels. (Jan. 2016). *The Hdbscan Clustering Library*. Accessed: Jan. 18, 2024. [Online]. Available: https://hdbscan.readthedocs.io/en/latest/index.html
[92] Scikit-Learn-Extra Developers. (Jan. 2019). *Sklearn_Extra.Cluster.KMedoids*. Accessed: Jan. 18, 2024. [Online]. Available: https://scikit-learn-extra.readthedocs.io/en/stable/index.html
[93] A. Amor and L. Liu. (Jan. 2024). *Clustering*. Accessed: Jan. 18, 2024. [Online]. Available: https://scikit-learn.org/stable/modules/clustering.html
[94] C. B. Do, C. S. Foo, and A. Y. Ng, "Efficient multiple hyperparameter learning for log-linear models," in *Proc. Adv. Neural Inf. Process. Syst.* NY, USA: Curran Associates, Dec. 2007, pp. 377–384.
[95] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, Feb. 2012.
[96] A. Mead, "Review of the development of multidimensional scaling methods," *J. Roy. Stat. Soc.*, vol. 41, pp. 27–39, Jan. 1992.



LUIS DANIEL SAMPER-ESCALANTE received the M.Sc. degree (summa cum laude) in Intelligent Systems from the Tecnologico de Monterrey, in 2017, where he is currently pursuing the Ph.D. degree in Computer Science. He has been publishing articles in national and international conferences and indexed and refereed journals, since 2013. His research interests include botnet detection on Twitter, wireless sensor networks, swarm intelligence, graph mining, and provenance.

OCTAVIO LOYOLA-GONZÁLEZ received the Ph.D. degree in Computer Science from the National Institute for Astrophysics, Optics, and Electronics, Mexico. He has won several awards from different institutions due to his research work on applied projects; consequently, he is a Member of the National System of Researchers in Mexico (Rank 1). He worked as a Distinguished Professor and Researcher at Tecnologico de Monterrey, Campus Puebla, for undergraduate and graduate programs of Computer Sciences. Also he was Managing Director of a consulting company, having branches in USA, Spain, Mexico, and U.K.; where he was leading several teams of data scientists for Artificial Intelligence. Currently, he is Executive Manager at NTT DATA, where he leads Advanced Analytics (AI & Gen AI) for sectors such as automotive, manufacturing, real estate, infrastructure, and services. From these research and applied projects, he has published several books and papers in well-known journals.

RAÚL MONROY received the Ph.D. degree in Artificial Intelligence (AI) from Edinburgh University, in 1998, under the supervision of Alan Bundy. He is currently a Professor of AI with Tecnologico de Monterrey, a member of CONAHCYT's Mexican Research System, currently rank three (top), a member of the Mexican Academy of Sciences, and a founding member of the Mexican Academy of Computing. He is the author of more than 120 publications. His research interests include the design and development of novel machine learning models, which he often applies in the domain of cyber security.

MIGUEL ANGEL MEDINA-PÉREZ received the Ph.D. degree in Computer Science from the National Institute of Astrophysics, Optics, and Electronics, Mexico, in 2014. He holds a top-tier ranking in the Mexican Research System. He has made considerable contributions to the field with numerous publications in prestigious journals, such as *Information Fusion*, IEEE TRANSACTIONS ON AFFECTIVE COMPUTING, *Pattern Recognition*, *Knowledge-Based Systems*, *Information Sciences*, and *Expert Systems with Applications*. His open-source code has more than 143 thousand downloads. Additionally, he has developed AI agents for large-scale scenarios (more than three million users), increasing web revenue by two million euros. His primary research interests include artificial intelligence, data visualization, machine learning, fingerprint recognition, and palmprint recognition.