Intelligent Automation & Soft Computing

Tech Science Press

DOI: 10.32604/iasc.2023.041645

ARTICLE

Real-Time Spammers Detection Based on Metadata Features with Machine Learning

Adnan Ali[1], Jinlong Li[1], Huanhuan Chen[1], Uzair Aslam Bhatti[2] and Asad Khan[3,*]

[1]School of Computer Science and Technology, University of Science and Technology of China, Hefei, 230026, China

[2]School of Information and Communication Engineering, Hainan University, Haikou, 570228, China

[3]Metaverse Research Institute, School of Computer Science and Cyber Engineering, Guangzhou University, Guangzhou, 510006, China

*Corresponding Author: Asad Khan. Email: asad@gzhu.edu.cn

Received: 30 April 2023 Accepted: 10 July 2023 Published: 27 February 2024

ABSTRACT

Spammer detection is to identify and block malicious activities performing users. Such users should be identified and terminated from social media to keep the social media process organic and to maintain the integrity of online social spaces. Previous research aimed to find spammers based on hybrid approaches of graph mining, posted content, and metadata, using small and manually labeled datasets. However, such hybrid approaches are unscalable, not robust, particular dataset dependent, and require numerous parameters, complex graphs, and natural language processing (NLP) resources to make decisions, which makes spammer detection impractical for real-time detection. For example, graph mining requires neighbors' information, posted content-based approaches require multiple tweets from user profiles, then NLP resources to make decisions that are not applicable in a real-time environment. To fill the gap, *firstly*, we propose a **RE**al-time **M**etadata based **S**pammer detection (REMS) model based on only *metadata* features to identify spammers, which takes the least number of parameters and provides adequate results. REMS is a scalable and robust model that uses only 19 metadata features of Twitter users to induce 73.81% F1-Score classification accuracy using a balanced training dataset (50% spam and 50% genuine users). The 19 features are 8 original and 11 derived features from the original features of Twitter users, identified with extensive experiments and analysis. *Secondly*, we present the largest and most diverse dataset of published research, comprising 211 K spam users and 1 million genuine users. The diversity of the dataset can be measured as it comprises users who posted 2.1 million Tweets on seven topics (100 hashtags) from 6 different geographical locations. The REMS's superior classification performance with multiple machine and deep learning methods indicates that only metadata features have the potential to identify spammers rather than focusing on volatile posted content and complex graph structures. Dataset and REMS's codes are available on GitHub (www.github.com/mhadnanali/REMS).

KEYWORDS

Spam detection; online social networks; metadata; machine learning

1 Introduction

Spam is fabricated content and comprises hashtags, phone numbers, images, malicious URLs, fake healthcare advice, bulk messages, wrong political facts, ransomware, stock market, and money scams

[1]. Another shape of spam is misinformation, including fake news, rumors, fake stories, deep fakes, political conspiracies, and personal trolls to destroy a person's or organization's reputation [2]. Spam can be spread via social media, calls, and emails [3]. Traditional spamming was via emails, while in the modern era, social media is in the top spot because of the recent boost in social media usage [4,5]. Spam spreads faster on social media than emails because of its advanced content-based recommendation algorithms. In contrast, email, SMS, and calls can be only for targeted persons. Authors in paper [6] stated that Twitter spam is considered particularly severe and advanced, with a 0.13% click-through rate compared to email spam, up to 0.0006%.

Spam motivations include scamming people, phishing, degrading information quality, spreading fake news, cyberbullying, spying, and giving bogus reviews to targeted or mass audiences [1,6,7]. A spammer is a user who spreads spam. Spammers use Twitter for financial gains and send unsolicited messages that consist of bulk messages, telephone numbers, malignant tiny web addresses, popular hashtags, falsified commercials, click fraud, images with concealed URLs, healthcare suggestions, malware, share-market spams, fake news, and political paid trolls [8]. Spamming costs resources like bandwidth consumption, user watch and focus time, speed and computing power wastage, polarizing users' sentiments, and slowing the spread of authentic information [2]. One of the reasons users use social media is leisurely and fast access to the right content [9,10]. At the same time, resources like time are limited and valuable and need to be saved by identifying spam.

We chose Twitter for spammer detection because of its massive size and diverse content. In the Twitter context, a spammer can be a human or a bot, as Twitter officially allows bots on Twitter [11]. Twitter is a social media platform where 556 million total and 239 million monetizable daily active users post more than 500 million tweets per day and generate massive user-generated content [12]. Twitter's Application Programming Interface (API) enables programmatic access to Twitter content with multiple access levels for research and business applications [13]. Such immense content and easy access make Twitter a valuable platform for research and business, where more than ten thousand research articles have been published in the last decade [12]. Researchers analyze Twitter content for politics [14,15], activism [16], rumor detection [17], fake news detection [18,19], information credibility [20], topic modeling [21], and sentimental analysis [22,23]. Many businesses, public figures, government representatives, celebrities, influencers, and news agencies around the globe use their verified official Twitter accounts [24] to directly communicate with their customers/followers, making Twitter a valuable tool for customer interaction, communication, and reviews [22,23].

Meanwhile, such a valuable platform also became an unwanted host for anomalies [25], where users are also exploiting content for spreading fake news [18,19,26], and hate speech [27]. Some users started using social media for unfair means, where spam and fake users are being used as a tool [28]. Fake trends are being started, promoted, and manipulated by fake accounts [29]. More followers of a user make him a popular, influential candidate of trustworthiness [20] and may lead to higher social impact [14], which motivates people to have inactive bots as followers. In paper [30], legitimate politicians were found to have bots in their followers. Twitter allows bots on its platform, and creating a new account is easy, which introduces the potential to create spam and fake accounts. Twitter rules and regulations are effective; Twitter monitors and suspends users who post against the platform policy. However, Twitter is a private organization and does not provide the reason for suspension, while its spam detection policy changes over time [31]. With such massive potential, fake or spam user detection [32–34] has been one of the active research areas in recent years, where several studies proposed to detect fake users. However, such studies mainly present a quantitative analysis, lack the proper ground truth [33], and have limited scope, the dataset focus is not broad enough [35].

However, previous work has four research gaps. *First*, the spam detection decision-making is made on posted content [6,36]. *Second*, datasets are small and lack the diversity of users [31,37,38]. *Third*, datasets are labeled manually and prone to have subjectiveness [37,39,40]. At *last*, it takes user features, posted content, and graph properties to decide if a user is a spammer or not [18,31,41], which makes the decision-making process slow and not robust for real-time detection. We present our arguments against each of the gaps. *First*, the posted content is volatile and can be deleted anytime; language-dependent, and not all languages' natural language processing techniques are as strong as English. *Second*, the dataset should be diverse enough to consider users discussing multiple topics from multiple countries. *Third*, Twitter spam detection should be taken as ground truth rather than manual labeling, as humans have different thought processes, cultures, and religious associations. A tweet offensive for one group can be funny for other groups, making the manual annotation doubtful. *At last*, it is not robust to scrap a user profile and get hundreds of tweets or create graphs to find if a person is a spammer, considering that Twitter has 556 million users, and each user can follow multiple people and can post many tweets. Considering the gaps, below are the contributions of this study:

- We propose to use only metadata features for real-time spammer detection as metadata features are rich in information, do not require making complex graphs, are language independent, can be processed fast, are opposed to *posted content*, and do not need resource costly natural language processing. To achieve the metadata-based model learning, we identify 8 and derive 11 (total 19) features from the metadata of the Twitter user object.
- We present a spammer detection largest dataset compared to previous work. The dataset comprises 1 million existing and 211 K deleted users. Deleted users labeled as spam and existing users as genuine users. To cover the diversity, the dataset is based on our previous research [42], which has 1.2 million users fetched from 2.1 million tweets on 100 hashtags covering 7 topics from 5 countries and globally. We monitored these users for 1 year, where 211 K users, making 16% of the total, were deleted in the mentioned period.
- We propose a real-time spammer detection framework called **RE**al-time **M**etadata based **S**pammer detection (REMS). REMS utilizes only metadata features to predict whether the user is a spammer or genuine. The ground truth of the model is users' existence on Twitter as we use deleted users as ground truth, considering that deleted user accounts were spammers.

Further paper is arranged as Section 2 reviews the literature. Section 3 introduces the methodology and formal definitions. Experiments and results are presented in Section 4, and finally, the conclusion and future implications are discussed in Section 5.

2 Literature Review

Spammer detection literature can be classified into content-oriented and metadata feature-oriented subsets. Below we discuss both domains.

2.1 Content Oriented

Content-oriented spam detection is to identify spam from the content of a Tweet. It includes investigating messages' content to find things frequently appear in spam, like particular phrases, words, or types of attachments. Such literature assumes that if a person post content that is not as per regular standards, it would be labeled as spam [38,43]. However, the identification process can be based on Tweet Text or its features. Such techniques also rely on the fact that spammers are trying to promote the same type of content by posting similar tweets to promote any product or campaign [36]. URLs are the most popular identifier in this category.

Due to Twitter's character count restriction, shrinking services pre-process lengthy URLs to include more descriptive words in the tweet [39]. Zheng et al. [43] created a support vector machine (SVM) based spammer detection model with the labeled dataset to detect spammers, while the dataset comprised 30,116 users and 16 million messages. They manually classify users into spammers and non-spammers. The authors [38] detected fake accounts using the content of tweets with natural language processing techniques and metadata features but use just 1000 users (500 fake and 500 legitimate) for testing.

The framework EGSLA [33] used graph-based deep learning algorithms to detect fake users based on content-based features like retweets, tweet length, a fraction of URLs, and average time. They assume that fake users' every tweet includes links (URLs), which is a shallow assumption to proceed with, and it has a small dataset of 800 users. A spammer detection paper [44] is a combination of metadata-based and textual properties to detect spam users. It considers only 16700 users, while suspended and deleted users are labeled spammers. The study's accuracy is 94.27%; however, there are issues with the approach to getting the results. The authors create two samples of datasets while both samples are imbalanced; where in sample 1, there are around 4000 users, not spammers, and around 700 are spams which is almost a 5 to 1 ratio. Authors quote that "better results could be obtained by reducing the ratio between legitimate and spam accounts, especially to avoid imbalanced data," which we argue is false as balanced data will decrease the accuracy. Furthermore, authors marked more users as spam based on their subjective observations.

2.2 Metadata Based

Literature also refers to them as feature-based methods. Such methods make decisions based on user-level behaviors; these behaviors can be profile descriptions, usernames, etc. Suspicious or exceptional similarity-based users called coordinated users' behavior is analyzed in information spreading where results show that coordinated users depict different patterns than non-coordinated users. Authors find that coordinated accounts spread information initially and tend to share the message in early states of diffusion [45]. A total of 3351 fake users' dataset is collected in the study [37], and authors perform the classification to detect fake users. Authors [37] used metadata features and properties of the account like profile name if the profile has an image etc. The study [40] created a small dataset of 1000 users and 16 features, but three people label the dataset manually, which is not a logically solid ground truth. An identity theft-based fake user detection chrome extension is made in [46] with a 6500 user profiles dataset where each user has seven features, but the paper failed to provide deep details of their framework. Authors in the study [47] considered their own Twitter circle of 1103 connections with 25 features as a dataset and get them manually label from three people, which is not a robust and practical approach for future work.

A recent paper [31] performed experiments on 46 K user accounts identified using the Twitter API where deleted accounts are marked as spam and existing ones are marked as genuine accounts. The authors use 34 account features, and although the paper is not very old, some of the used features in the study are depreciated [48] (e.g., profile_link_color). The study also performs graph analytics by creating followers, following graphs, and finding Degree Centrality, Shortest-Path Betweenness Centrality, Eigenvector Centrality, and PageRank Centrality. Furthermore, graph representation learning and NLP-based methods like tweet similarity, lexical diversity, and co-occurring word distribution are also applied. Although paper results are 97% on F1-Score, the paper is a combination of everything, and its solution is suitable for better accuracy but is not robust.

Furthermore, performing this many actions would be computationally expensive, especially creating a follower graph where some Twitter users have more than 100 million followers; it becomes near to impossible. It is to be noticed that the dataset is not balanced for training. An assumption is that as Twitter allows a limited number of characters, there are more chances that spam tweets have a link to another web page [6]. Twitter spam detection falls in the Tweet and account levels [10]. Comparing results accuracy is not ideal, especially without common ground truth. Different datasets are downloaded in different time periods, and Twitter keeps updating its spam detection policy over time [31].

Overall, published research lacks the proper ground truth, primarily represents the quantitative analysis, and the available datasets are small. Either author shares their findings in a limited scope or combines everything to get the best results, making the over-fitted models for specific datasets. Such datasets and studies are helpful for the limited and specific domain but fail to provide proper ground and base to grow upon for future research. We aim to provide a real-time and robust solution that can be used as a benchmark for future research.

3 Methodology

This section presents the working structure of the REMS (Real-time metadata-based spammer detection), where Fig. 1 illustrates the model diagram. The framework comprises 5 components: *Downloading*, *Labeling*, *Feature extraction*, *Classification*, and *Prediction*, which are explained below.

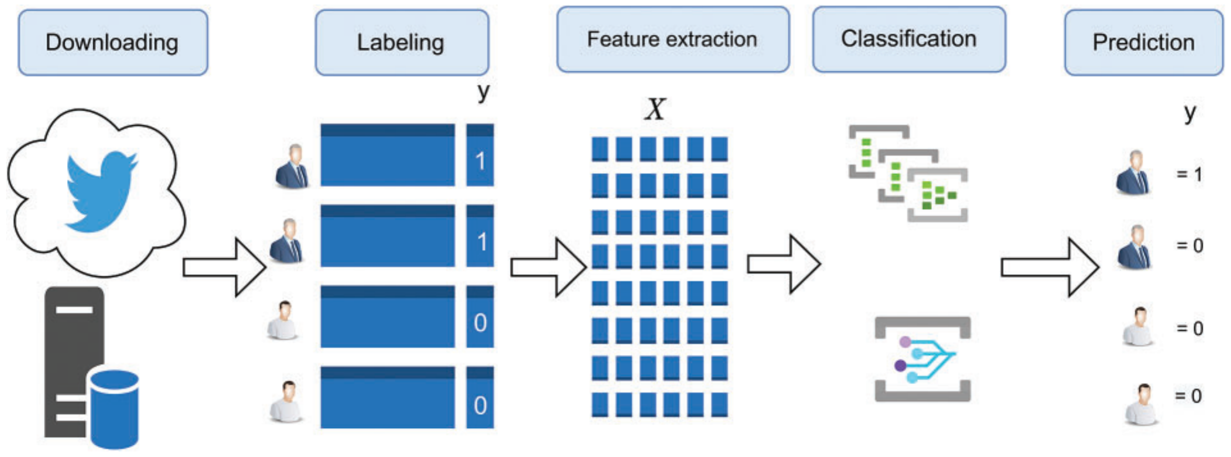

Figure 1: Model diagram of REMS, illustrating five components: Downloading, labeling, feature extraction, classification, and prediction

3.1 Downloading

Downloading refers to the user's dataset downloading from Twitter. The paper [42] dataset is used as the base dataset, where a function $\delta(\cdot)$ takes a primary hashtag H_p as input and downloads the tweets objects and merges them in a file T.

$$T = \delta(H_p) \tag{1}$$

where T is a file comprised of tweets. A user can post multiple tweets, so we discard duplicates and extract unique users set U from T. U is a set of users $U = \{u_1, u_2, u_3, \ldots, u_i\}$ who posted tweets in file T, therefore $U \subseteq T$.

3.2 Labeling

Labeling is the process of labeling deleted users U_d with 0, presented as $U_d = \{u \in U_d : f(u) = 0\}$, and to label existing users U_e as 1, presented as $U_e = \{u \in U_e : f(u) = 1\}$. Fig. 1 illustrates the labeling with $y \in \{0, 1\}^{N\times 1}$ where 0 indicates that the user u_i is deleted spam, and 1 indicates the user exists on Twitter and is a genuine user. Set U is comprised of two types of users:

$$U = U_d \cup U_e \tag{2}$$

where deleted spam users are U_d and existing genuine users are U_e and $U_d \cap U_e = \{\}$.

3.3 Feature Extraction

Feature extraction is to extract features from Twitter user objects and create a feature matrix. A total of 19 features are extracted from users' u_i object and feature matrix X is created, where $X \in R^{N\times F}$ collects all users' features, N is the number of users, and F is the number of features. Users are rows of feature matrix X and presented as X_i, while $X_i = U$ and $N = |X_i| = |U|$. Features are columns of X and presented as $X_j = \{x_1, x_2, x_3, \ldots, x_i\}$, where $F = \left|X_j\right| = 19$. Furthermore, $F = F_o \cup F_d$ where F_o are original features, and F_d are derived features, explained in the below subsections.

3.3.1 Original Features

Original features refer to the metadata properties originally existing in Twitter objects. Let $F_o = \{x_1, x_2, x_3, \ldots, x_8\}$ refers to original features where $|F_o| = 8$ are explained below.

1. Followers (x_1): Number of users following the particular user u_i. We indicate it as feature one and present it as x_1. There is no limit imposed on the number of followers by Twitter, and a user can get as many followers as possible.
2. Friends (x_2): The number of users followed by a particular user u_i and indicated as x_2. In contrast to x_1, which has no limit per any timespan, having friends have a limit per day.
3. List (x_3): The number of lists a user u_i has joined in his time on Twitter.
4. Account life (x_4): Since a user u_i joined Twitter, time is presented as x_4 in days.
5. Favorites count (x_5): The number of tweets the user u_i has liked in his account's lifetime.
6. Verified (x_6): This feature depicts whether Twitter verifies a user; verified users have a blue, grey, or gold tick on their profiles and enjoy different benefits than non-verified users.
7. Tweets (x_7): The number of tweets and retweets users posted is presented as x_7.
8. Protected (x_8): Twitter facilities privacy by giving the option to users, if they want to limit their content and if they want to control who follows them. We present this feature as x_8.

3.3.2 Derived Features

Features derived from original features. Let $F_d = \{X_8, x_9, x_{10}, \ldots, x_{19}\}$ refers to original features where $|F_d| = 11$ are explained below:

1. Followers per day (x_9): Having followers strongly indicates that a person is more famous among other users. However, if it is an absolute value, it becomes ambiguous. We include the time property with the number of followers to overcome the issue. It is the ratio between followers x_1 and account life x_4 to depict the temporal behavior and present as: $x_9 = \dfrac{x_1}{x_4}$
2. Tweets per day (x_{10}): The average number of tweets per day posted by the user u_i since joining Twitter is presented as: $\mathrm{x}_{10} = \dfrac{\mathrm{x}_7}{\mathrm{x}_4}$

3. Favorites per tweet (x_{11}): A user can like as many tweets as he wants. There is no limit to it. However, a ratio between liking and posting tweets depicts whether a user posts more tweets or likes more. It is presented as: $\mathrm{x}_{11} = \frac{\mathrm{x}_5}{\mathrm{x}_7}$
4. Friends per day (x_{12}): Number of people, a user u_i followed per day. Following people have a limit, and a person can follow up to 5000 people from his account until he has enough followers. The number of days normalizes if the user network is growing faster or normally than peers. It is introduced as: $x_{12} = \frac{x_2}{x_4}$
5. Followers per friend (x_{13}): We define a feature with the ratio between followers and friends. To depict if a user follows more people or if his content is good enough to gain more followers. The equation for followers per friend is depicted as $x_{13} = \frac{x_1}{x_2}$, while a higher x_{13} value will depict the person's importance.
6. Friend per list (x_{14}): To analyze that, a user makes more friends or joins more lists, we introduce this feature. It illustrates whether a person focuses on the following topics or users. The feature is presented as follows: $x_{14} = \frac{x_2}{x_3}$
7. Followers per tweet (x_{15}): This feature shows the value of posted content in getting new followers. It is calculated as "How many followers does a user have per tweet?" which will depict the posted content's significance. The feature is presented as follows: $x_{15} = \frac{x_1}{x_7}$ where x_1 represents followers and feature x_7 represents the tweets.
8. Friends per Tweet (x_{16}): This feature depicts if a user u_i is making more friends or posting more tweets. It is introduced as: $x_{16} = \frac{x_2}{x_7}$, where x_2 represents the friends and x_7 are tweets.
9. Favorite per follower (x_{17}): Sometimes, liking other contents attract users to visit and follow the person and increases the chance of building a network. So, we introduce this feature. Formally presented in the equation $x_{17} = \frac{x_5}{x_1}$, where x_1 represents followers and feature x_4 represents the number of favorites.
10. Favorite per friends (x_{18}): Favorites ratio with friends is presented in the equation $x_{18} = \frac{x_5}{x_2}$
11. Lists per day (x_{19}): Twitter does not employ any limit on following the lists, which enables a user to keep following the limits as much as he wants and may lead to spam behavior. This feature describes the ratio by explaining how many lists a user has followed per day. The feature is presented as follows: $x_{19} = \frac{x_3}{x_4}$

The outcome of this component is feature matrix X, which along with y from *labeling*, is used for *Classification*.

3.4 Classification

Classification is to arrange and group items according to their features and attributes. In machine learning, classification is to train a model to assign labels to the input data. This study's dataset has only two classes; hence, it is a binary classification aiming to classify spam and genuine users. This work uses six state-of-the-art classification methods *Decision trees, Random forest, Extreme gradient boosting, Support vector machine, K-nearest neighbors*, and *Deep neural networks*.

3.5 Prediction

The last module of the framework, where unknown data (20% of total data) is input to the model, and the model predicts that the user is spam or genuine. Formally, $\hat{y}$ is used to depict prediction where $\hat{y} \in \{0, 1\}$ illustrated in Fig. 1.

4 Results

4.1 Experimental Tools and Setup

We use NetworkX [49] and PyTorch Geometric [50] for scalability purposes for dataset construction. Each user is a graph node, but for these experiments, there are no edges between nodes. By connecting nodes (if required in future work) dataset will be a graph, and both mentioned libraries support graph-based deep learning.

We train data on five metrics with six machine learning and one deep learning method, where scikit-learn [51] implementations are used for training and testing. There are 1 million existing and 211 K deleted users, but for fair evaluation, we select random 211 K users from existing users to make it equal for each class. Furthermore, 80% is taken as training data and 20% as testing data, while results are an average of 20 runs.

4.2 Accuracy Metrics

We are performing binary classification with two classes, True and False. Genuine users are labeled True, and spam users are labeled False. All of the classification algorithms are tested on the given five metrics.

- True Negative (TN): If the actual label is negative and the model predicts it as negative, it is a true prediction of negative and presented as TN.
- True Positive (TP): The real prediction of positive. If the real label is true and the model also predicts it as true, it is called the True positive and presented as TP.
- False Negative (FN): False indicates a false prediction. It can be said as a false prediction of the negative and presented as FN.
- False Positive (FP): It indicates that the model predicts positive. However, the actual label is negative. In an ideal scenario, TN and TP values should be higher, and FN and FP values should be lower.

4.2.1 Accuracy

Accuracy refers to classification accuracy as the ratio of correct predictions against total predictions. Per the above definition, TP and TN are true predictions. Hence equation of Accuracy is:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

4.2.2 Precision

It is the ratio between true positive predictions and total positive predictions. For a good classifier, precision should be higher as in the denominator; there is false positive FP; if its value increase, the precision value will decrease, which is not good for a good model.

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

4.2.3 Recall

It is the ratio of true positives and the sum of true positives and false negatives. For a good classifier, the recall value should be high; a lower value will depict false negatives are more.

$$Recall = \frac{TP}{TP + FN} \tag{5}$$

4.2.4 F1-Score

It is *Harmonic Mean* between precision and recall. It is the most used classification metric [52] and is better than accuracy. It is calculated as:

$$\mathrm{F1-score} = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{6}$$

4.2.5 Jaccard Score

It depicts how similar are two sets (real labels and predicted labels) are and is calculated as follows:

$$J\left(y, \hat{y}\right) = \frac{\left|y \cap \hat{y}\right|}{\left|y \cup \hat{y}\right|} \tag{7}$$

where y and $\hat{y}$ refer to true and predicted labels.

4.2.6 Time

We argue that time is an essential factor in results and should also be considered for results representation. We present the average time of training and testing against each variant. Time is in seconds and an average of 20 executions.

4.3 Accuracy Algorithms and Results

We use the six classification methods below to verify the accuracy of metadata features, our dataset, and its usability. All of the below methods are trained in a supervised manner.

4.3.1 Decision Trees

Decision trees solve the problem by splitting the data based on features where decisions are at leaves. We perform decision tree experiments with three criterion, *gini, entropy,* and *log loss,* with tree depths 1 to 20. Data train and test splitting for these experiments were the same all the time. Experiments are performed twenty times at each depth, and the average is presented in Table 1. The table indicates that with the criterion gini, decision trees generate the best results and converge faster than other criterion variants. Only precision has a lower value in gini than entropy, indicating that false positives are higher in this case than entropy but have lower false positive predictions in log loss.

Table 1: Classification accuracy of algorithms where finest values are bold, where JS is Jaccard Score

Algorithm	Depth	Accuracy	F1-Score	Precision	Recall	JS	Time (s)
DT_Entropy	5	67.50	74.38	61.27	94.63	59.21	5.22
DT_Log loss	5	67.47	74.43	61.20	94.96	59.28	3.72

(Continued)

Table 1 (continued)

Algorithm	Depth	Accuracy	F1-Score	Precision	Recall	JS	Time (s)
DT_GINI	6	67.51	74.49	61.21	**95.12**	59.35	3.64
KNN	81	68.62	71.54	65.47	78.87	55.70	56.55
RF_Entropy	9	69.20	74.21	63.89	88.50	58.99	109.6
RF_Log loss	10	**69.36**	74.21	64.12	88.07	58.99	131.1
RF_GINI	9	69.28	74.27	63.94	88.58	59.07	88.15
XGB_Linear	nil	65.32	63.15	67.42	59.40	46.15	**2.45**
XGB_Gb tree	5	67.91	74.64	61.73	94.38	59.54	13.37
XGB_Dart	5	67.91	**74.64**	61.73	94.38	**59.54**	28.50
SVM_RBF	nil	66.52	69.98	63.18	78.42	53.82	16254
SVM_Linear	nil	62.68	52.82	72.36	41.58	35.89	120066
SVM_Poly	nil	64.74	61.78	67.30	57.84	44.69	17605
SVM_Sigmoid	nil	58.83	58.85	58.65	59.30	41.69	12063
MLP	nil	68.51	72.98	63.87	85.15	57.45	4884

Fig. 2 illustrates all six metrics results on three loss functions. There are 20 experiments run for each depth. First, Fig. 2a presents the criterion entropy results. Recall has the highest value at depth two, but as depth increases, its value goes down, and Time has a mirror effect to recall where it increases with the depth. The F1-Score and Accuracy metrics do not have high fluctuations but show minor changes in depth. However, Jaccard Score and precision show a significant difference after depth 10; the former's value decrease after 10, and the latter values increase. Second, Fig. 2b presents the results for criterion log loss. However, compared to entropy, both figures show almost the same behavior with minor differences in time. At last, Fig. 2c illustrates the results for the criterion gini, which has the highest value of F1-Score as per Table 1. At depth 7, its results are a bit odd, while other than that, its behavior is almost the same as the other two variants. Overall, it can be seen that increasing the depth of the tree increases the classification time, but increasing the depth always does not improve the performance.

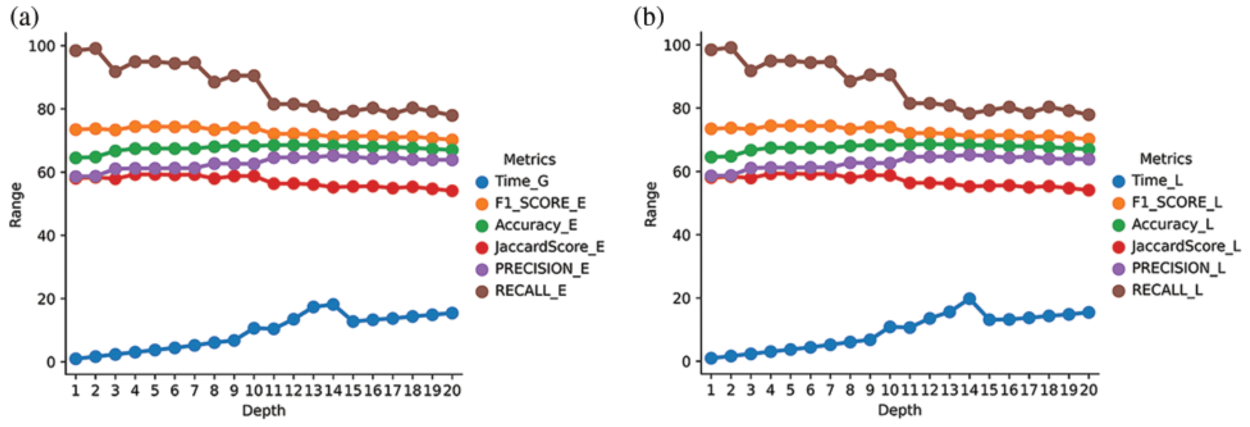

Figure 2: (Continued)

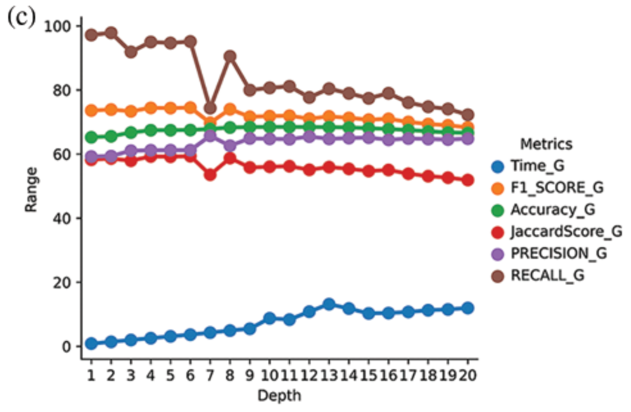

Figure 2: Decision tree results on all six metrics, (a) entropy, (b) log loss, (c) gini

4.3.2 K-Nearest Neighbour

A swift but the non-parametric model with reasonable accuracy, where labeling decisions are made on the bases of neighbors, where the K value indicates how many neighbors to choose. K-nearest neighbors (KNN) is a non-parametric supervised instance-based learning algorithm used for both regression and classification. Non-parametric means there is no assumption for underlying data distribution. KNN tries to predict the correct class for the test data by calculating the distance between the test data and all the training points. K-Nearest Neighbour performance is good on our dataset. However, finding the optimal K value is a hefty task. First, we find the optimal K value as Fig. 3a illustrates that increasing the K value improves the performance until, at one point, improvement is marginal. Too small K value is bad, while too large value causes over-fitting. Table 1 presents the best results among 70 to 90 K value, with the best results at $K = 81$. From 70 to 90, each K value's training and testing is performed 20 times on the same data.

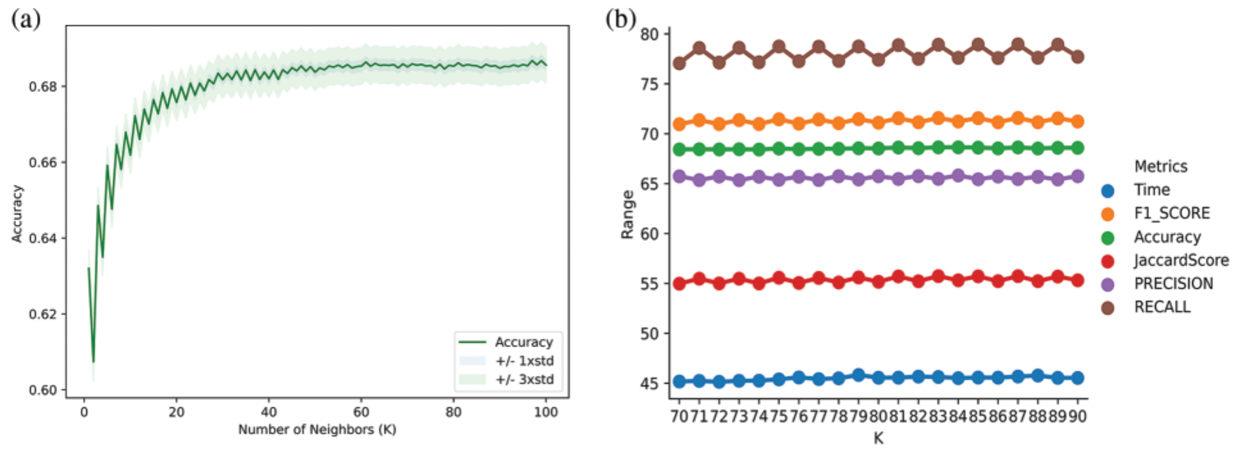

Figure 3: (a) K value line graph between 1 to 100 and (b) KNN results against accuracy metrics

Fig. 3a illustrates the KNN results against all metrics while K value is between 70 to 90. Fig. 3b presents that K value has minor effects on time, making KNN time independent As the figure indicates that after crossing a specific limit of K, accuracy does not change much, which can be verified from the figure. Recall is the only metric with a bit more fiction than other metrics. There is a common behavior in both Figs. 3a and 3b that odd K value gives better results against accuracy, F1-Score, recall, and Jaccard score. However, this effect is vice versa for the precision metric. It indicates that false positive FP values are higher at odd K values than even K values.

4.3.3 Random Forest

Random Forest is a robust machine-learning algorithm that can be used for various tasks, including regression and classification. It is an ensemble method, meaning that a random forest model comprises many small decision trees, called estimators, which each produce their own predictions. The random forest model combines the predictions of the estimators to produce a more accurate prediction. Following the decision trees setting, we use three criterion gini, entropy, and log loss with depths 1 to 20 for the random forest. Each depth is trained and tested 20 times under the same but randomly selected dataset. Oppose to decision trees, using different criterion makes no difference in random forest, and Table 1 indicates that random forest yields the exact same results at each depth, making the depth option useless in terms of accuracy metrics except time.

Fig. 4 presents the random forest results on all metrics while everything is the same except for time. Using all three criterion depicts the same behavior in terms of time as well, where increasing the depth is time costly. This section concludes that random forest is exempt from the depth and depicts the same results under different combinations of settings; however, like the decision tree gini is the best in terms of time and should be considered in our dataset.

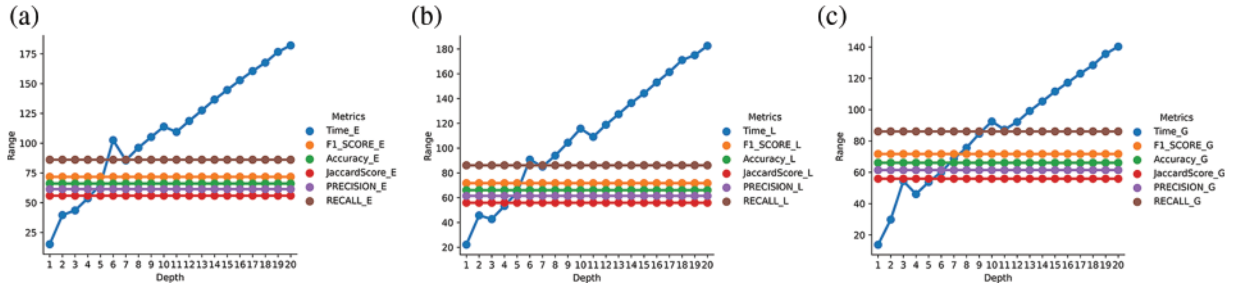

Figure 4: Random forest results on all six metrics. (a) Entropy, (b) log loss, (c) gini

4.3.4 Support Vector Machine

Support vector machine works as the data is transferred into embedded space where a hyperplane is defined as a decision boundary to separate classes. SVMs are particularly useful when the data has many features and/or when there is a clear margin of separation in the data. Hyperplane in an N-dimensional space that distinctly classifies the data points. The dimension of the hyperplane depends upon the number of features.

Table 1 presents the classification results using four kernel variants Radial basis function (RBF), Sigmoid, Polynomial, and Linear. Among all four kernels, RBF gives the best results on four out of six metrics. However, its precision is not as good as the polynomial kernel. If we notice the time and compare it to other algorithms applied in this study, SVM is the slowest among all, and its accuracy is also not up to mark. For inter-kernel comparison on time, the polynomial is slow and takes ten times more time than the sigmoid. SVM does not have any depth as it is not tree based classifier; that is why there are no figures for this section. We do not suggest using SVM on our dataset because of time even its linear classifier is too slow than other variants, and overall, it does not offer very good classification accuracy.

4.3.5 XGBoost

Extreme Gradient Boosting in short "XGBoost" is an optimized distributed gradient boosting algorithm for decision trees designed for efficient and scalable training of machine learning models. XGBoost has three types of boosters Linear, GbTree, and dart, while the first is a linear function and the last two are tree-based models. Table 1 presents the classification accuracy of XGBoost using these boosters while the learning rate for all experiments is 0.001. The table illustrates that linear booster is the worst performing in terms of accuracy metrics, but its only positive point is, it is fast. Being fast makes sense as its decision-making is less complex than tree classifiers. However, the results are the same on tree-based boosters except for time.

Fig. 5 presents the XGB results with GB tree and dart. We do not present linear booster results as the presented graphs are drawn based on depth, while the linear function does not have depth. The figure illustrates that results are the same for both boosters except when dart consumes more time than gb tree. The time chart in Fig. 5 shows that increasing the depth of the tree increases the time, and model accuracy decreases after the peak point. Increasing the depth increases the complexity and creates more chances of over-fitting and memory consumption.

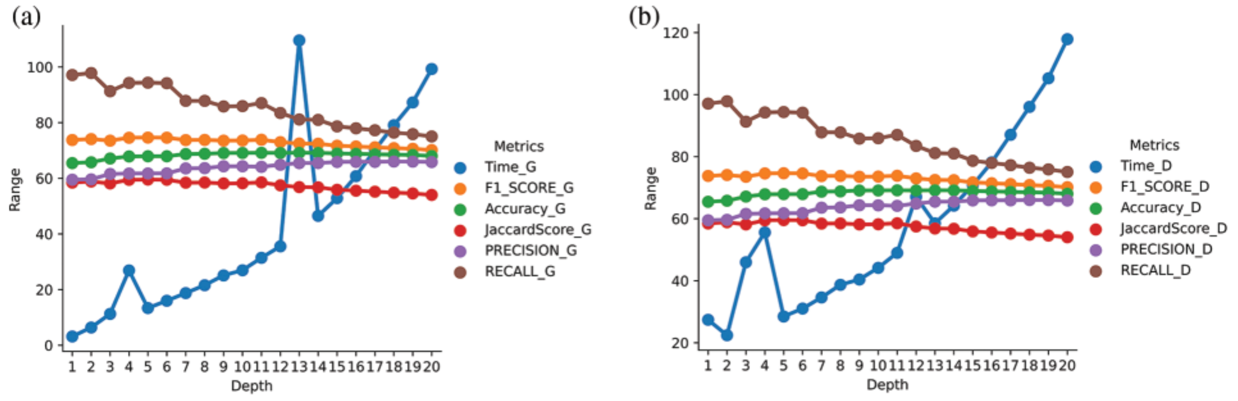

Figure 5: Decision tree results on all six metrics. (a) GB tree, (b) dart

4.3.6 Multilayer Perceptron

The multilayer perceptron (MLP) is a feed-forward artificial neural network consisting of at least three layers of nodes. Inspired by the biological brain, these nodes are interconnected and transform any input dimension to the desired dimension. To obtain these results, we ran experiments 20 times with the 20–80 ratios of data splitting, where 20% is the testing dataset. Specifications of the learning model are three linear layers and a Relu activation function with a 0.001 learning rate. Each linear layer has 900, 450, and 64 neurons for decision-making, while experiments are executed for 1000 epochs with testing after every 50 epochs, and the best results are presented in Table 1.

Overall, Table 1 presents the classification results based on all classification algorithms and matrices used in this study. There are 15 variants of 6 algorithms tested on six metrics. Overall results depict that the decision tree and its variants are fast algorithms, and their F1-Score is also one of the tops in the table. Furthermore, the Jaccard score of decision trees is also good and the second highest in the table. However, their accuracy is low. Random forest covers this gap and offers better accuracy by marginal differences on F1-Score but on the cost of time. Random forests have the best accuracy among all algorithms. XGB is the fastest in linear evaluation, but its other results are among the lowest.

XGB tree varieties have the best F1-Score, and execution time is not very high compared to decision trees, but it is lower than random forests.

SVM is the slowest among all, and the results are also not good; we do not suggest using it. At last, deep learning-based results are not very good, but we believe these results can be useful as base results and can be improved with state-of-the-art deep learning methods. We leave this for future work. Finally, we also present the confusion matrix of six classification methods in Fig. 6. The figure shows that the highest accuracy is on the true negative (TN), which concludes that algorithms are very accurate in predicting deleted users. However, true positives accuracy needs improvements, and adding some content-based features will also make a difference which is not the scope of this work.

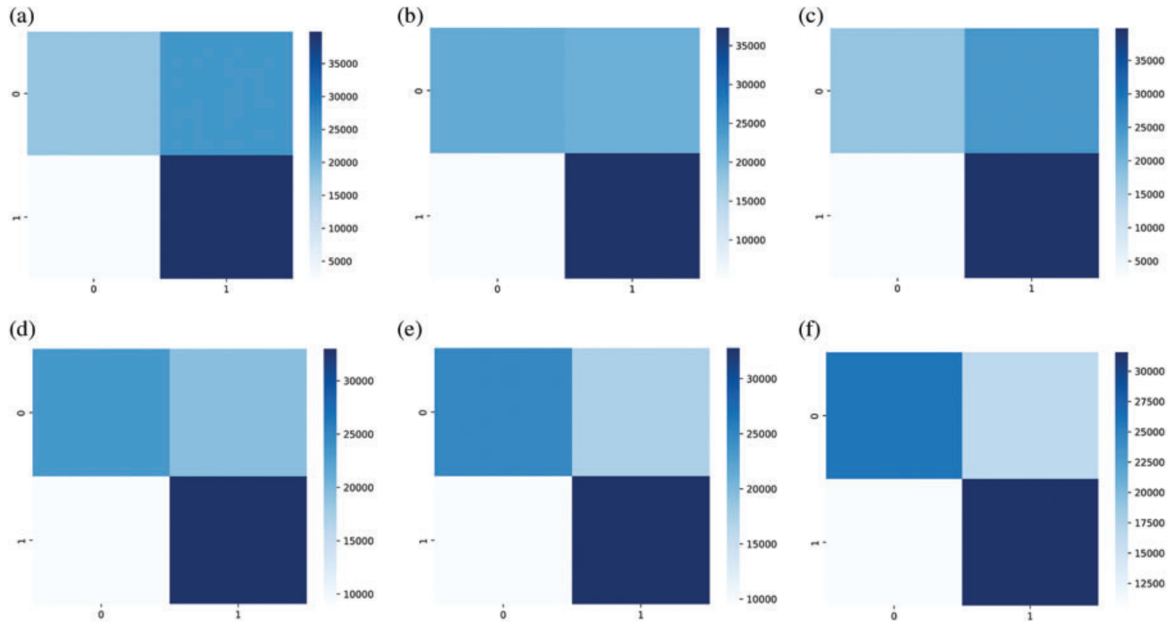

Figure 6: Confusion matrices of (a) DT, (b) RF, (c) XGB, (d) SVM, (e) KNN, (f) DNN, where 0, 0 is TP, 0, 1 is FP, 1, 0 is FN, and 1, 1 is TN

4.4 Feature Importance

Fig. 7 illustrates all nineteen features' importance on four accuracy metrics using five machine learning algorithms. All four sub-figures depict that all nineteen features are equally essential and show no major effects on results except x_4 on KNN. These results indicate that all features play a vital role in accuracy rather than depend on any single feature or a subset of features.

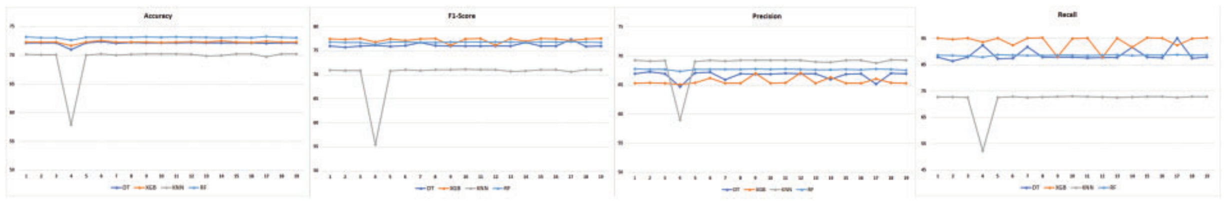

Figure 7: Effect of each feature on the accuracy, F1-score, precision, and recall with machine learning algorithms

5 Conclusion and Future Work

This study presents a metadata features-based spammer detection model which aims to provide a robust and real-time solution while taking only 8 metadata-based features as input. We monitored the 1.2 million users over 13 months and created 211 K deleted and 1 million existing users dataset, the largest dataset for spammer detection to date. We derive 11 features from 8 metadata features and combine them in the spammer detection model for decision-making. We classify these users with six machine and deep learning classification methods. Our superior results depict that only metadata features are enough to identify spammers, and complex processes like natural language processing and graph mining can be omitted for real-time and robust detection.

In the future, we aim to use this dataset as a benchmark for spammer detection as the dataset does not contain any volatile properties, e.g., tweet content or description. We will create a graph-based dataset from this dataset where users will be nodes, and one user mentioning other users will create an edge between users. We will apply the graph neural networks on that dataset and share our findings, analysis, and comparison between these variants. The presented dataset is the largest available and can be used as a base dataset to create more datasets. For example, the presented dataset comprises the users of seven topics which presents the opportunity to analyze which topic users have more chances to be spammers, and it can be considered in future studies.

Acknowledgement: The first author gratefully acknowledges the joint Ph.D. fellowship provided by the Chinese Academy of Sciences (CAS) and The World Academic of Sciences (TWAS) as the first author is sponsored by CAS-TWAS President's Fellowship for Ph.D. at the School of Computer Science and Technology, University of Science and Technology China. On behalf of all the authors, we are grateful for the reviewers' excellent suggestions to improve the quality of this work. Finally, we appreciate Twitter for providing free access to data and diagrams.net for free access to their wonderful graphics drawing software.

Funding Statement: This Research is supported by the Guangzhou Government Project (Grant No. 62216235) and the National Natural Science Foundation of China (Grant Nos. 61573328, 622260-1).

Author Contributions: The authors confirm their contribution to the paper as follows: study conception and design: Adnan Ali, Jinlong Li, Huanhuan Chen; data collection: Adnan Ali; analysis and interpretation of results: Adnan Ali, Asad Khan, Uzair Aslam Bhatti; draft manuscript preparation: Adnan Ali, Jinlong Li. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The dataset is available for research purposes. The dataset link and framework codes are uploaded to Github.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] S. Rao, A. K. Verma and T. Bhatia, "A review on social spam detection: Challenges, open issues, and future directions," *Expert Systems with Applications*, vol. 186, pp. 115742, 2021.

[2] M. Westerlund, "The emergence of deepfake technology: A review," *Technology Innovation Management Review*, vol. 9, pp. 40–53, 2019.

[3] A. Mewada and R. K. Dewang, “A comprehensive survey of various methods in opinion spam detection,” *Multimedia Tools and Applications*, vol. 82, pp. 13199–13239, 2022.

[4] S. Dixon, “Twitter global mDAU 2022,” Statista. 2022. [Online]. Available: https://www.statista.com/statistics/970920/monetizable-daily-active-twitter-users-worldwide/

[5] M. Ali, H. Mushtaq, M. B. Rasheed, A. Baqir and T. Alquthami, “Mining software architecture knowledge: Classifying stack overflow posts using machine learning,” *Concurrency and Computation: Practice and Experience*, vol. 33, no. 16, pp. e6277, 2021.

[6] T. Wu, S. Wen, Y. Xiang and W. Zhou, “Twitter spam detection: Survey of new approaches and comparative study,” *Computers & Security*, vol. 76, pp. 265–284, 2018.

[7] A. Ali, H. Rafique, T. Arshad, M. A. Alqarni, S. H. Chauhdary *et al.,* “A fractal-based authentication technique using sierpinski triangles in smart devices,” *Sensors*, vol. 19, no. 3, pp. 678, 2019.

[8] H. Shen, X. Liu and X. Zhang, “Boosting social spam detection via attention mechanisms on Twitter,” *Electronics*, vol. 11, pp. 1129, 2022.

[9] A. Mubashir, B. Anees, H. H. R. Sherazi, H. Asad, A. H. Alshehri *et al.,* “Machine learning based psychotic behaviors prediction from facebook status updates,” *Computers, Materials & Continua*, vol. 72, no. 2, pp. 2411–2427, 2022.

[10] R. Alharthi, A. Alhothali and K. Moria, “A real-time deep-learning approach for filtering Arabic low-quality content and accounts on Twitter,” *Information Systems*, vol. 99, pp. 101740, 2021.

[11] S. Li, J. Yang, G. Liang, T. Li and K. Zhao, “SybilFlyover: Heterogeneous graph-based fake account detection model on social networks,” *Knowledge-Based Systems*, vol. 258, pp. 110038, 2022.

[12] D. Antonakaki, P. Fragopoulou and S. Ioannidis, “A survey of Twitter research: Data model, graph structure, sentiment analysis and attacks,” *Expert Systems with Applications*, vol. 164, pp. 114006, 2021.

[13] T. Documentation, “Tweet Object,” Twitter. 2023. [Online]. Available: https://developer.twitter.com/en/docs/twitter-api/v1/data-dictionary/object-model/tweet

[14] T. Laor, “My social network: Group differences in frequency of use, active use, and interactive use on Facebook, Instagram and Twitter,” *Technology in Society*, vol. 68, pp. 101922, 2022.

[15] F. B. Keller, D. Schoch, S. Stier and J. Yang, “Political astroturfing on Twitter: How to coordinate a disinformation campaign,” *Political Communication*, vol. 37, no. 2, pp. 256–280, 2020.

[16] S. N. Puente, S. D. Maceiras and D. F. Romero, “Twitter activism and ethical witnessing: Possibilities and challenges of feminist politics against gender-based violence,” *Social Science Computer Review*, vol. 39, no. 2, pp. 295–311, 2021.

[17] Z. Luo, Q. Li and J. Zheng, “Deep feature fusion for rumor detection on Twitter,” *IEEE Access*, vol. 9, pp. 126065–126074, 2021.

[18] H. Yuan, J. Zheng, Q. Ye, Y. Qian and Y. Zhang, “Improving fake news detection with domain-adversarial and graph-attention neural network,” *Decision Support System*, vol. 151, pp. 113633, 2021.

[19] Z. Khanam, B. N. Alwasel, H. Sirafi and M. Rashid, “Fake news detection using machine learning approaches,” *IOP Conference Series: Materials Science and Engineering*, vol. 1099, no. 1, pp. 12040, 2021.

[20] C. Castillo, M. Mendoza and B. Poblete, “Information credibility on Twitter,” in *Proc. of the 20th Int. Conf. on World Wide Web, in WWW’11*, New York, NY, USA, pp. 675–684, 2011.

[21] M. Ali, A. Baqir, G. Psaila and S. Malik, “Towards the discovery of influencers to follow in micro-blogs (Twitter) by detecting topics in posted messages (Tweets),” *Applied Sciences*, vol. 10, no. 16, 2020.

[22] A. Onan, “Sentiment analysis on product reviews based on weighted word embeddings and deep neural networks,” *Concurrency and Computation: Practice and Experience*, vol. 33, no. 23, pp. e5909, 2021.

[23] A. Hasan, S. Moin, A. Karim and S. Shamshirband, “Machine learning-based sentiment analysis for Twitter accounts,” *Mathematical and Computational Applications*, vol. 23, no. 11, 2018.

[24] T. Documentation, “Verified Accounts,” Twitter. 2023. [Online]. Available: https://twitter.com/verified/following

[25] M. Munir, S. A. Siddiqui, A. Dengel and S. Ahmed, “DeepAnt: A deep learning approach for unsupervised anomaly detection in time series,” *IEEE Access*, vol. 7, pp. 1991–2005, 2019.

[26] J. Zeng, Y. Zhang and X. Ma, "Fake news detection for epidemic emergencies via deep correlations between text and images," *Sustainable Cities and Society*, vol. 66, pp. 102652, 2021.
[27] E. Grolman, H. Binyamini, A. Shabtai, Y. Elovici, I. Morikawa *et al.,* "HateVersarial: Adversarial attack against hate speech detection algorithms on Twitter," in *Proc. of the 30th ACM Conf. on User Modeling, Adaptation and Personalization*, New York, NY, USA, pp. 143–152, 2022.
[28] T. Elmas, R. Overdorf, A. F. Özkalay and K. Aberer, "Ephemeral astroturfing attacks: The case of fake Twitter trends," in *The 2021 IEEE European Symp. on Security and Privacy*, Vienna, Austria, pp. 403–422, 2021.
[29] Y. Zhang, X. Ruan, H. Wang, H. Wang and S. He, "Twitter trends manipulation: A first look inside the security of Twitter trending," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 11, pp. 144–156, 2017.
[30] B. C. Silva and S. O. Proksch, "Fake it 'til you make it: A natural experiment to identify European politicians' benefit from Twitter bots," *American Political Science Review*, vol. 115, no. 11, pp. 316–322, 2021.
[31] M. Chakraborty, S. Das and R. Mamidi, "Detection of fake users in Twitter using network representation and NLP," in *2022 14th Int. Conf. on Communication Systems & Networks (COMSNETS)*, Bangalore, India, pp. 754–758, 2022.
[32] A. Gupta and R. Kaushal, "Towards detecting fake user accounts in Facebook," in *2017 ISEA Asia Security and Privacy (ISEASP)*, Surat, India, pp. 1–6, 2017.
[33] M. BalaAnand, N. Karthikeyan, S. Karthik, R. Varatharajan, G. Manogaran *et al.,* "An enhanced graph-based semi-supervised learning algorithm to detect fake users on Twitter," *The Journal of Supercomputing*, vol. 75, no. 9, pp. 6085–6105, 2019.
[34] K. Shu, X. Zhou, S. Wang, R. Zafarani and H. Liu, "The role of user profiles for fake news detection," in *Proc. of the 2019 IEEE/ACM Int. Conf. on Advances in Social Networks Analysis and Mining*, New York, NY, USA, pp. 436–439, 2020.
[35] P. Harrigan, T. M. Daly, K. Coussement, J. A. Lee, G. N. Soutar *et al.,* "Identifying influencers on social media," *International Journal of Information Management*, vol. 56, pp. 102246, 2021.
[36] X. Zhang, Z. Li, S. Zhu and W. Liang, "Detecting spam and promoting campaigns in Twitter," *ACM Transactions on the Web*, vol. 10, no. 1, pp. 1–28, 2016.
[37] S. Cresci, R. di Pietro, M. Petrocchi, A. Spognardi and M. Tesconi, "Fame for sale: Efficient detection of fake Twitter followers," *Decision Support System*, vol. 80, pp. 56–71, 2015.
[38] M. M. Swe and N. Nyein Myo, "Fake accounts detection on Twitter using blacklist," in *2018 IEEE/ACIS 17th Int. Conf. on Computer and Information Science (ICIS)*, Singapore, pp. 562–566, 2018.
[39] K. Kawintiranon, L. Singh and C. Budak, "Traditional and context-specific spam detection in low resource settings," *Machine Learning*, vol. 111, no. 7, pp. 2515–2536, 2022.
[40] B. Erşahin, Ö. Aktaş, D. Kılınç and C. Akyol, "Twitter fake account detection," in *2017 Int. Conf. on Computer Science and Engineering (UBMK)*, Antalya, Turkey, pp. 388–392, 2017.
[41] L. Jain, R. Katarya and S. Sachdeva, "Opinion leader detection using whale optimization algorithm in online social network," *Expert Systems with Applications*, vol. 142, pp. 113016, 2020.
[42] A. Ali, J. Li, H. Chen and A. K. Bashir, "Temporal pattern mining from user-generated content," *Digital Communications and Networks*, vol. 8, no. 6, pp. 1027–1039, 2022.
[43] X. Zheng, Z. Zeng, Z. Chen, Y. Yu and C. Rong, "Detecting spammers on social networks," *Neurocomputing*, vol. 159, pp. 27–34, 2015.
[44] A. S. Alhassun and M. A. Rassam, "A combined text-based and metadata-based deep-learning framework for the detection of spam accounts on the social media platform Twitter," *Processes*, vol. 10, no. 3, 2022.
[45] M. Cinelli, S. Cresci, W. Quattrociocchi, M. Tesconi and P. Zola, "Coordinated inauthentic behavior and information spreading on Twitter," *Decision Support System*, vol. 160, pp. 113819, 2022.
[46] S. R. Sahoo and B. B. Gupta, "Real-time detection of fake account in Twitter using machine-learning approach," in *Advances in Computational Intelligence and Communication Technology*, Singapore, pp. 149–159, 2021.

[47] K. K. Bharti and S. Pandey, “Fake account detection in twitter using logistic regression with particle swarm optimization,” *Soft Computing*, vol. 25, no. 16, pp. 11333–11345, 2021.

[48] Tweet object | Docs | Twitter Developer, 2023. https://developer.twitter.com/en/docs/twitter-api/v1/data-dictionary/object-model/tweet

[49] A. A. Hagberg, D. A. Schult and P. J. Swart, “Exploring network structure, dynamics and function using network X,” in *Proc. of the 7th Python in Science Conf.*, Pasadena, CA, USA, pp. 11–15, 2008.

[50] M. Fey and J. E. Lenssen, “Fast graph representation learning with PyTorch geometric,” arXiv:1903.02428, 2019.

[51] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion *et al.,* “Scikit-learn: Machine learning in python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[52] A. Ali and J. Li, “Features based adaptive augmentation for graph contrastive learning,” arXiv:2207.01792, 2022.