Machine Learning Algorithms for Detecting and Analyzing Social Bots Using a Novel Dataset

Niyaz Jalal[1] and Kayhan Z. Ghafoor[1,2]

[1]Department of Software and Informatics Engineering, College of Engineering, Salahaddin University-Erbil, Erbil 44001, Iraq

[2]Department of Computer Science, Knowledge University, Erbil 44001, Iraq

Abstract—**Social media is internet-based technology and an electronic form of communication that facilitates sharing of ideas, documents, and personal information. Twitter is a microblogging platform and is the most effective social service for posting microblogs and likings, commenting, sharing, and communicating with others. The problem we are shedding light on in this paper is the misuse of bots on Twitter. The purpose of bots is to automate specific repetitive tasks instead of human interaction. However, bots are misused to influence people's minds by spreading rumors and conspiracy related to controversial topics. In this paper, we initiate a new benchmark created on a 1.5M Twitter profile. We train different supervised machine learning on our benchmark to detect bots on Twitter. In addition to increasing benchmark scalability, various autofeature selections are utilized to identify the most influential features and remove the less influential ones. Furthermore, over-/under-sampling is applied to reduce the imbalance effect on the benchmark. Finally, our benchmark compared with other state-of-the-art benchmarks and achieved a 6% higher area under the curve than other datasets in the case of generalization, improving the model performance by at least 2% by applying over-/under-sampling.**

***Index Terms*—Machine learning, Misinformation detection, Twitter bot detection, Twitter profile metadata.**

I. Introduction

In the past decade, the influence of social media has increased rapidly; this growth will be more tangible in the upcoming years. One of the famous social platforms is Twitter; millions of people, including influential figures, have Twitter accounts to interact with their audiences. Besides the advantages, Twitter is used to exploit and delude others in many events, such as the COVID-19 outbreak, the Russian invasion of Ukraine, and the USA presidential elections in 2016 and 2020, by spreading falsified conspiracies and manipulating public opinion (Khanday, Khan and Rabani, 2021; Shevtsov, et al., 2021).

These actions are usually performed by automated programs, so-called bots. Broadly speaking, bots aim to ease automation processes such as sending a friendly message or giving some instruction on social media. Yet, the automation capability of bots is unfavorably used for spreading spam, fake news, and hate speech (Davis, et al., 2016).

A botmaster manages social bot accounts, which controls many social bots to influence public opinion toward a specific ideology or purpose by spreading low credible information (Ferrara, et al., 2016). The social bot impact is so significant that some reports indicate that 9–15% of the active accounts on Twitter are social bots (Varol, et al., 2017). In some cases, the fake news reached 100K users, and false information had 70% more retweets than trustworthy news (Hanouna, et al., 2019; Orabi, et al., 2020a).

In this paper, the machine learning framework is proposed based on a novel dataset. We collected more than 1.5M Twitter accounts during the US presidential election in 3 months. Our dataset only includes the metadata (profile) features that contain a small number of features. The benefits of using profile features are increasing the model's scalability and decreasing training time. The main goal of this paper is to create a benchmark suitable for real-time bot detection by increasing the number of samples and reducing the number of features. At the same time, dataset generalization has increased by 6% average AUC compared to other datasets in this research area. Finally, we can conclude our main distribution as follows:

1. We collected 1.5 million users and created up-to-date 100K datasets; to the best of our knowledge, it is the biggest only metadata dataset.
2. We increase generalization, and our dataset achieves a 6% higher average accuracy among all datasets in the research area.
3. We improve the model performance by at least 2% by introducing over-/under-sampling algorithms with our dataset.
4. We interpret the prediction of our machine learning models with the help of Shapley Additive Explanations (SHAP)

ARO-The Scientific Journal of Koya University
Vol. X, No. 2 (2022), Article ID: ARO.11032. 11 pages
DOI: 10.14500/aro.11032
Received: 15 July 2022; Accepted: 15 August 2022
Regular research paper: Published: 10 September 2022
Corresponding author's email: niyaz.jalal@su.edu.krd

II. Related Work

Different types of bots target various audiences; for example, spam bots for spreading spam, fake follower for increasing followers of a particular account, COVID, or political bots for spreading conspiracy. Alom, Carminati and Ferrari (2018) and Shukla, Jagtap and Patil (2021) tried to detect spam bots on Twitter platform. Alom, Carminati and Ferrari (2018) have used metadata and graph-based features from the 42K dataset collected by a Social Honeypot. They used different ML models; the random forest (RF) achieved the best result compared to other models. Furthermore, Shukla, Jagtap and Patil (2021) have used a public dataset with 38K users and 19 features with different Ensemble models, feature engineering, feature encoding, and feature selection. The Ensemble model with ANN, RF, and AB achieved the best result and also best ensemble model outperformed other individual models by attaining 93% AUC.

Feng, et al. (2021); Khanday, Khan and Rabani, (2021); and Shevtsov, et al. (2021) focused on detecting bots that spread rumors. Khanday, Khan and Rabani (2021) have used Tweeter API to collect tweets related to the COVID19 outbreak. Different ML applied to detect bots that spread COVID rumors. The decision tree (DT) gave the best result with a 99% F1 score; the result showed that tweets generated by bots have a greater length than regular tweets. Shevtsov, et al. (2021) have collected tweets related to the 2020 US presidential election to detect political bots. The data were collected over 2 months with a total of 15.6M tweets and 3.2M users. Furthermore, Bot Sentinel and Botometer were used to label the dataset. Different ML, feature engineering, feature selection, and under-/over-sampling applied to the dataset; as a result, XGBoost achieved 92% as the best F1 score. Feng, et al. (2021) have collected tweets related to different hashtags with 34M tweets and 8M users to create a dataset with a variety of bots in it (called TwiBot-20). The dataset trained with the previous works models, the result shows the model accuracy declined with TwiBot-20. They blamed the absence of user diversity, limited user information, and data scarcity for this decline.

On the other hand, Yang, et al. (2019) and Hayawi, et al. (2022) tried to increase model scalability and reduce training time by only using metadata to train a model. Yang, et al. (2019) have focused on enhancing the generalization of detection models. They used 14 public datasets for training the models; in some cases, two or more datasets have combined. They manually selected 20 features from datasets, then used the RF for training. The RF trained with the entire dataset and then tested with other datasets. As a result of this approach, the generalization between the datasets is very low. In the second experiment, they selected seven datasets for training, and all possible combinations between them were trained (247 combinations) and tested the model with the remaining datasets using the RF classifier. This approach improved the generalization between datasets. Hayawi, et al. (2022) have used public only metadata datasets to train deep neural networks. Furthermore, text features have transformed with long short-term memory (LSTM) and GLoVE. The same practice as Yang, et al., 2019, was followed: Training the model on one dataset and evaluating it with other datasets. The final results show improvement in the model generalization.

Kudugunta and Ferrara (2018), Rodríguez-Ruiz, et al. (2020), and Martin-Gutierrez, et al. (2021) introduced new detection models for detecting bots on Twitter. Kudugunta and Ferrara (2018) have used a neural network with LSTM and GloVE to detect bots. A total of 16 features from metadata and tweet content from the public dataset had used to train the model. Furthermore, the over-/under-sampling technique used to balance the dataset. The result is 96% AUC for a single tweet account and 99% AUC for account level. Martin-Gutierrez, et al. (2021) have focused on detecting multilingual bots using a deep learning model. The BERT, Flair, and RoBERT models were used separately to transform text-based features into a numeric vector. The model was trained with a 60K public dataset and achieved a 77% F1 score. Rodríguez-Ruiz, et al. (2020) have used a one-class classification. This classification algorithm determines whether the data belong to this class or not, which usually uses imbalanced data. They trained public datasets with binary and one-class classification models with different algorithms. The results indicate that one-class classification achieved a better result than binary classification.

Finally, this paper aims to create a benchmark suitable for real-time bot detection by increasing the number of samples using labeling API to make the labeling process faster and having more labeled data, decreasing the number of features by selecting the most influential features, and ignoring the less influential ones. Furthermore, oversampling and undersampling are used to overcome the over=/under-fitting problem.

III. Models

A. Machine Learning Models

Different machine learning algorithms have used to train the proposal dataset: Adaboost Classifier (AB), Bagging Classifier (BC), DT, Extra Tree Classifier (ET), Gaussian Naive Bayes (GN), K-nearest neighbor (KNN), logistic regression (LR), random forest (RF), support vector machines (SVM), and eXtreme Gradient Boosting (XGBoost). However, RF and ET have tested separately with 100 and 500 trees, and KNN with three and five neighbors. Except for XGBoost, the sklearn Python package was used for all models. Furthermore, 5-fold cross-validation with random shuffling splits the dataset into five parts; the training and validation occur in four portions, and the last portion tests the model performance. Accuracy, precision, recall, F1 score, and AUC (Huang and Ling, 2005) were used to evaluate model performance. (Fig. 1 shows the steps for the experiment).

B. Feature Selection Models

Feature selection is an essential part of any machine learning algorithm. Feature selection has many advantages, for example, removing irrelevant data and reducing data

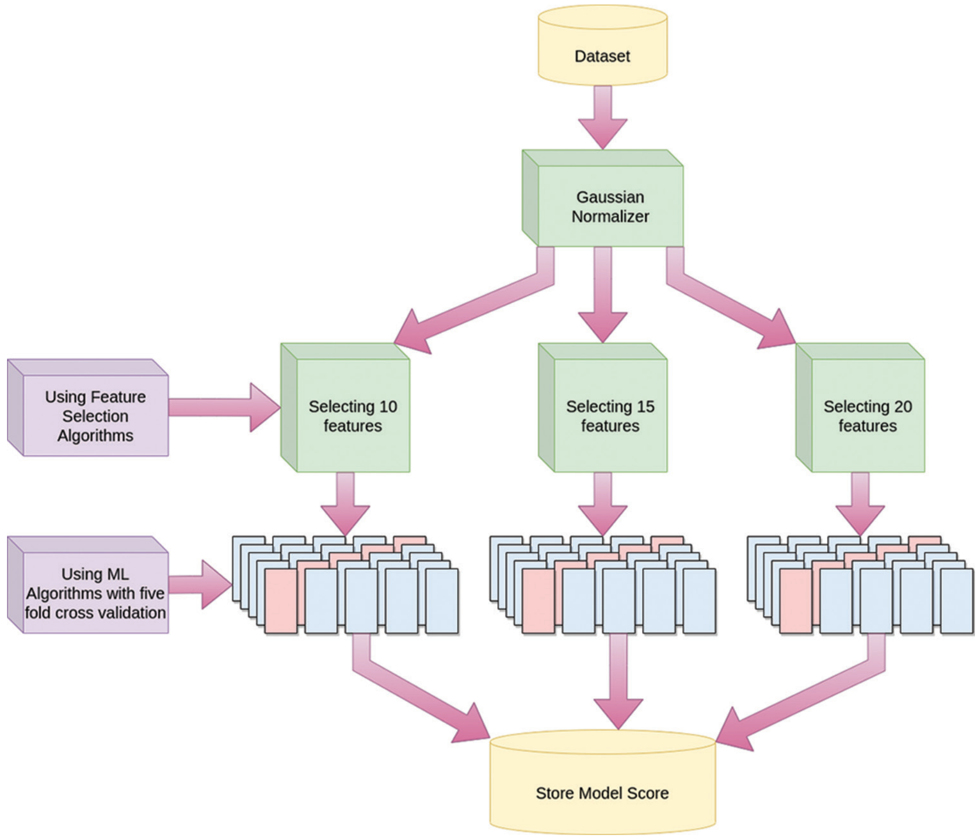

Fig. 1. The proposed experiments to detect the best model approaches.

dimensions, increasing model performances, reducing overfitting and underfitting, and speeding up training time (Shevtsov, et al., 2021).

There are two types of feature selection, manual selection and automatic selection. In our work, we perform both types of feature selections. Two types of features were excluded from original user metadata by manual selection. The first group features have different values for each Twitter account. The second group is the features with almost the same values for all Twitter accounts.

On the other hand, autoselection models use statistics correlation, ML algorithm, permutation importance, and coefficient scores to find the relationship between features; when features are more correlated, a higher score will assign to the candidate feature. In our work, we used recursive feature elimination (RFE) (Granitto, et al., 2006), univariate feature selection (UFS) (Jović, Brkić and Bogunović, 2015), and feature importance (FSI) (Altmann, et al., 2010). All three techniques are implemented in the sklearn Python package.

Our work used: RFE-DT, FSI-RF, FSI-XGboost, ANOVA F-value (Shaw and Mitchell-Olds, 1993), and information gain (Peng, Long and Ding, 2005); we tried to extract 10, 15, and 20 separately using each of the mentioned algorithms.

C. Over-/Under-Sampling Models

When one class has majority samples and others have a minority, oversampling and undersampling are usually used for balancing a dataset. There are a variety of techniques for balancing a dataset; in case of oversampling, the number of minority classes will increase, but under sampling reduces the sample numbers of majority classes (Shevtsov, et al., 2021).

Moreover, some techniques combine both over-/under-sampling. The synthetic minority oversampling technique (SMOTE) (Wang, et al., 2006) is the most popular technique to increase the number of minority classes. SMOTE uses KNN to generate N number of synthetic samples for each sample in the minority class. Furthermore, there are other modified versions of SMOTE, such as ADASYN, Borderline-SMOTE, and SVM-SMOTE.

On the other hand, edited nearest neighbor (ENN) (Wilson, 1972) is an undersampling algorithm that utilizes KNN to eliminate samples close to the boundary or misclassified. Tomek Links (Elhassan and Aljurf, 2016) is another undersampling model created based on condensed nearest neighbors (CNNs). The CNNs randomly chose samples to eliminate from the majority class, but in the case of Tomek Links, the KNN was used to find those samples with the lowest distance from the minority class, then remove them.

In our experiments, mentioned algorithms are applied individually to the dataset, and the results show little or no improvement in the model's performance. As a second experiment, both under-/over-sampling algorithms are combined; first, SMOTE and Tomek links are combined (SMOTETomek), then SMOTE and edited nearest neighbor are combined (SMOTEENN). Applying both under-/over-sampling algorithms to our datasets accelerate the model performance. We discuss their results in the performance evaluation section.

IV. Datasets

A. Data Collection

The lack of datasets is a barrier to social bot detection (Adewole, et al., 2017). Collecting data from social platforms are prohibited by most influential social media platforms; Twitter is one of a few platforms allowing collecting public data by providing API (Orabi, et al., 2020b). We collected data from Twitter using Twitter API, retrieving tweets related to the USA 2020 presidential election topics. The data collection process started 3 months before the election day (in July 28, 2020) and ended 16 days (November 19, 2020) after the election day. This topic is selected because it is a controversial topic on Twitter and makes all types of bot involvement very high. As shown in Table I, 19 hashtags used to collect data, and a total of 1.5M unique Twitter accounts were collected.

B. Profile Features

The retrieved data from Twitter API contain more than 1000 features, mainly divided into four categories, profile feature (user metadata), context feature, time-based feature, and interaction features (Yang, et al., 2019). All categories have been used to detect bots on Twitter platforms (Orabi, et al., 2020b). Still, in our work, we only focus on the profile feature, a small object containing all critical information related to an account. The advantage of using profile features is reducing training time without affecting model performance. The user profile object contains 44 features in total, such as name, description, picture's URL, colors, statistical, Boolean flags, language related, and other parameters. We divided the profile features into three groups: Newly derived, dropped, and direct used features. In the next sections, we describe each of them. It is also worth mentioning that the latest version of Twitter API removes some features from user profile object, but they are not essential features; the feature selection process eliminated them.

C. Dropped Features

The values of properties from profile features are different; some properties have different (unique) values for each account, and others are totally similar (include none values). In those cases, properties do not give any meaningful information to distinguish between humans and bots. After analyzing each property, the result shows that 23 features represent ineffective data, meaning either their values are different for each account or they have one or two unique values for all accounts regardless of account type (bot or human). id_str, translator_type, time_zone, contributors_enabled, following, profile_text_color, profile_background_color, profile_sidebar_border_color, profile_link_color, profile_sidebar_fill_color, lang, withheld_in_countries, is_translator, utc_offset, notifications, id, follow_request_sent, is_translation_enabled, protected, profile_background_image_url_https, profile_background_image_url, profile_image_url, and profile_image_url_https features are eliminated from our datasets.

Table I
List of Hashtags That Used for Collecting Data

Hashtag		
#democrat	#politics	#america
#mikepence	#vote	#guncontrol
#election	#bluewave	#getoutthevote
#republican	#gop	#president
#racistinchief	#joebiden	#trump
#biden	#trumpvoters	#donaldtrump
	#racistpresident	

D. Direct used Features

The majority of the previous works have used a set of features and they have proven their effectiveness (Yang, et al., 2019; Martin-Gutierrez, et al., 2021). Those features are either numeric or binary features. The numeric features obtain essential statistical information such as the number of followers. The binary features give the basic account information, for example, is account verified or not. Followers_count, friends_count, listed_count, statuses_count, favourites_count, has_extended_profile, default_profile, geo_enabled, and verified are some features that exist in a majority of previous works. Furthermore, we add default_profile_image, profile_use_background_image, and profile_background_tile features to our dataset by converting the Boolean values (true/false) into numerical representation (1/0) (Equation 6). Table II shows features: Name, types, and description. In our work, all features mentioned in this section are used directly without performing any feature engineering; simultaneously, we use the numeric feature to create new features.

E. New Derived Features

Derived features are created by performing a mathematical or logical operation on one or more original features. We made 21 new features based on calculating other features. The previous works proved that the extracted features include valuable information for the detection process (Hayawi, et al., 2022). We create three new feature sets for statistical, text, and binary features. In the case of statistical features, the features are divided by an account age (the account creation date minus the last tweet date in our dataset). Furthermore, we find the ratio between followers and friends. The second set is text features; the metadata object contains name, screen_name, and description as a text feature. We calculate the text length for name, screen_name, and description (Equation 1); the number of digits (Equation 2), frequency (Equation 5 when numerator is the n-g (Brown, et al., n.d.), and the denominator is the total unique characters.), and entropy (shows in Equation 3 when numerator is Shannon entropy (Shannon and Weaver, 1949) and denominator is feature length) for name and screen_name; similarity between name and screen_name

Table II
List of Features used in this Paper with Their data Type and Short Description

Feature	type	Description
listed_count	Numerical	Public lists that use members of
statuses_count	Numerical	Total number of tweets and retweet
friends_count	Numerical	Number of users the account following
favorites_count	Numerical	Total number of tweets liked by an account
followers_count	Numerical	Number of users following this account
default_profile	Binary	If profile is set
has_extended_profile	Binary	If profile is extend is true otherwise false
geo_enabled	Binary	Twitter has access to profile location
verified	Binary	If profile is verified
profile_background_tile	Binary	If profile is tiled is true otherwise false
profile_use_background_image	Binary	If profile has background image
default_profile_image	Binary	If profile image is default
profile_has_banner_url (DB)	Binary	If profile has banner URL
has_entities (DB)	Binary	If entities is set is true otherwise false
has_location (DB)	Binary	If location is set is true otherwise false
has_url (DB)	Binary	If URL is set is true otherwise false
account_age (DB)	Numerical	last_tweet_date - account_created_date
name_length (DT)	Numerical	Length of name feature
screen_name_length (DT)	Numerical	Length of screen_name feature
description_length (DT)	Numerical	Length of description feature
num_digits_in_name (DT)	Numerical	Number of digit in name feature
num_digits_in_screen_name (DT)	Numerical	Number of digit in screen_name feature
name_entropy (DT)	Numerical	Entropy of name feature
screen_name_entropy (DT)	Numerical	Entropy of screen_name feature
screen_name_freq (DT)	Numerical	Mean bigram frequency for screen name
name_freq (DT)	Numerical	Mean bigram frequency for name
name_similarity (DT)	Numerical	Similarity between screen_name and name
tweet_freq (DS)	Numerical	Statuses_count/account_age
followers_growth_rate (DS)	Numerical	followers_count/account_age
favourites_growth_rate (DS)	Numerical	favourites_count/account_age
listed_growth_rate (DS)	Numerical	listed_count/account_age
friends_growth_rate (DS)	Numerical	friends_count/account_age
followers_friends_ratio (DS)	Numerical	followers_count/friends_count

DB: Binary derived, DS: Statistically derived, DT: Text derived

(shows in Equation 4 when numerator is the number of matches, and the denominator is the length of both texts). The last set includes features that either contains a value or not (Equation 6); for example, the location feature is either set or not (empty). In our work, we call those features binary-derived features; we convert them to one if it has a value; otherwise, we convert them to zero. Table II shows statistically derived features (their names end with DS), text-derived features (their names end with DT), and binary-derived features (their names end with DB.)

F. Data Labeling

Data labeling is a process of giving actual labels to the data; labeling is a mandatory step for supervised machine learning (Derhab, et al., 2021). We use the data label for comparing with predicted results by ML models to determine how well the model performed. As mentioned earlier, the total number of users in our dataset is more than 1.5M. Two possible ways to know

$$Feature_{len} = \begin{cases} len\ (feature),\ len\ (feature) \geq 0 \\ 0,\ None \end{cases}$$

$$Feature_{digit\_count} = \begin{cases} digit\ (feature),\ len\ (feature) \geq 0 \\ 0,\ empty \end{cases}$$

$$Feature_{entropy} = \frac{-\sum_{i=1}^{k} p(i) * \log_2 p(i)}{len\ (feature)}$$

$$Name_{similarity} = \frac{2M}{N}$$

$$Feature_{freq} = \frac{\sum_{i=1}^{k} C(b_i)}{K}$$

$$Feature_{convert}(x) = \begin{cases} 1,\ (x = True\, or\, x = Vaue\, exist) \\ 0,\ (x = False\, or\, x = Value\, not\, exist) \end{cases}$$

the actual data labels are manually checking or using previously trained models. In our case, manual checking is impossible due to dataset size; therefore, we choose Botometer API to classify our dataset. The Botometer has two versions: BotometerLite, which only uses metadata to classify; Botometer-V4, which predicts based on more than 1000 features, including tweets, metadata, and other features. In our case, we use BotometerLite because it follows the same strategy as our work by only focusing on metadata features for labeling data. The BotometerLite allows 20K samples to be labeled every day. In our case, it took around 75 days to classify 1.5M samples. The BotometerLite score sample is between 0 and 1. In our case, the API could not classify approximately 16.9% of our data, and 83.1% remaining samples are classified, as shown in Table III. To create our dataset, we classify any account score equal to or greater than 0.6 as bot accounts, so 42K accounts are classified as bots. Regarding a human account, we randomly selected 58K accounts that scored <0.1. However, all the samples bigger than 0.1 and <0.6 are ignored to have a big gap between bot and human. In conclusion, we created a dataset with 100K samples classified based on BotometerLite. Table III shows the start and end of the labeling process and the total accounts that were classified based on the specific range.

V. Performance Evaluation

A. Performance Evaluation of Machine Learning Models

We tested our dataset with 10 different models (Fig. 1 shows the steps for the experiment); however, we used two versions of KNN, RF, and BC and 5-fold cross-validation to reduce the chance of overfitting in the dataset. However, different algorithms were tested to normalize our dataset, such as scaling, standardizing, transforming, and normalizing. Transforming (Gaussian normalization) usually achieves the best result. Furthermore, it is worth mentioning that the dataset without normalization was tested but achieved a very low result. Therefore, we use Gaussian normalization in all experiments in this paper. Five selection models are utilized; the feature selection extracts 10, 15, and 20 features separately. We trained different models (using 13 ML algorithms). Table IV shows the max, min, and mean result based on an accurate measurement. As shown in Table IV, RF with 20 features and ET with 15 features achieved the best accuracy result. However, surprisingly, increasing the number of features has a very slim effect on performance and an enormous impact on training time. For example, in the case of RF with 100 trees, the difference between the model with 20 features and 10 features is just 0.04.

Nevertheless, in the case of AB, KNN with five neighbors, and RF with 500 trees, the model performed better when trained with 10 features rather than 15 or 20. Moreover, the mean columns show that SVM and BC overall performance achieve the best result. Another interesting finding is that the increasing number of trees or neighbors has little impact on the performance. For instance, in the case of ET, the model achieved precisely the same result, RF performance declined slightly, and KNN performance increased somewhat. Table V shows the result of the same experience as Table IV, but besides average (mean) accuracy, it also reveals the average of recall, precision, F1 score, and AUC.

B. Feature Selection Evaluation

As mentioned earlier, we trained different models based on five feature selection algorithms – the performance of all algorithms was excellent in 15–20 features. REF-DT obtains the best performance, a little bit higher than other algorithms. However, for average best performance, information gain gets a higher result. However, in 10 features, the performance of the ANOVA F-value declined by more than 10%, but other algorithms performed relatively high. Nevertheless, when we consider time, XGBoost is achieving the best; for instance, in the case of 20 features, it is 3 times faster than RFE-DT with a 0.45 decline in performance; the same is true for 10–15 features with a little difference. Surprisingly, the appropriate time for 15 features is higher than for 10–20 feature models. We thought that the time for the 15 models should be between 10 and 20. We achieved the time results using the sklearn Python package, which is a very respectful package among the Python community; yet, we are not sure about the exact reasons. Table VI shows the mean (average), maximum, and minimum accuracy for different feature selection models; each value is calculated based on 13 ML models (Table IV). It also shows FitTime (the time for fitting the model for each train set split) and ScoreTime (the time for fitting the model on the test set).

However, regarding the most influential features for the 10 features approach, our algorithms selected favorites_count, friends_growth_rate, friends_count, followers_count, num_digits_in_screen_name, tweet_freq, favorites_growth_rate, followers_growth_rate, followers_friends_ratio, and description_length as the most 10 influential features. The same experiment was repeated for the 15-feature approach; besides, features in the 10-feature list, also listed_count, statuses_count, Name_similarity, account_age, and Name_entropy have added to complete the 15-feature list. Finally,

Table III
Labeling Process for BotometerLite

Score range	BotometerLite
Start date	January 27, 2022
End date	April 13, 2022
0–0.09	310,026
0.1–0.19	321,790
0.2–0.29	236,290
0.3–0.39	175,183
0.4–0.49	112,001
0.5–0.59	50,313
0.6–0.69	19,837
0.7–0.79	12,367
0.8–0.89	6995
0.9–1	2134
Error	253,064
Total	1,500,000

Table IV
Mean (average), Maximum, and Minimum Accuracy Values for Different ML Algorithms

ML models	20			15			10		
	Max	Min	Mean	Max	Min	Mean	Max	Min	Mean
AB	98.00	73.55	94.32	97.98	73.43	94.05	98.01	73.45	92.06
BC-100	98.61	73.52	95.22	98.61	73.52	95.03	98.57	73.46	92.26
DT	97.67	73.60	94.19	97.69	73.43	93.94	97.65	73.52	91.16
ET-100	98.66	73.46	94.67	98.64	73.52	94.48	98.64	73.39	91.93
ET-500	98.66	73.51	94.72	98.67	73.50	94.59	98.64	73.52	92.00
GN	93.09	67.83	80.87	92.83	67.50	80.09	92.34	67.01	78.59
KNN-3	97.94	72.24	89.65	98.25	73.44	91.23	98.29	60.84	89.82
KNN-5	98.05	67.03	89.89	98.32	73.44	91.25	98.37	73.40	90.80
LR	93.88	69.86	85.70	93.52	62.49	84.64	93.38	66.84	83.07
RF-100	98.67	73.62	95.21	98.65	73.43	94.97	98.63	73.50	92.21
RF-500	98.00	73.55	94.32	97.98	73.43	94.05	98.01	73.45	92.06
SVM	98.61	73.52	95.22	98.61	73.52	95.03	98.57	73.46	92.26
XGboost	97.67	73.60	94.19	97.69	73.43	93.94	97.65	73.52	91.16

Table V
Mean (average) for Different ML Algorithms with Various Performance Measurements

ML models	20					15					10				
	A	F1	R	P	AUC	A	F1	R	P	AUC	A	F1	R	P	AUC
AB	94.32	96.98	92.56	90.75	95.37	94.05	96.61	91.99	89.90	95.56	92.06	94.70	89.03	85.88	94.64
BC-100	95.22	97.02	93.67	92.43	95.83	95.03	97.27	93.67	92.75	95.11	92.26	94.45	89.76	88.21	93.05
DT	94.19	94.76	92.55	91.79	94.08	93.94	94.36	92.04	91.17	94.18	91.16	91.15	88.27	86.81	91.75
ET-100	94.67	96.95	92.90	91.18	95.74	94.48	97.18	92.98	91.62	94.87	91.93	93.77	88.86	86.63	93.73
ET-500	94.72	97.07	93.06	91.53	95.56	94.59	96.92	92.91	91.48	95.37	92.00	94.03	89.36	87.62	92.96
GN	80.87	90.20	73.18	68.39	85.66	80.09	90.15	72.10	68.00	85.18	78.59	88.54	67.33	60.02	86.66
KNN-3	89.65	92.34	87.26	85.61	89.51	91.23	93.18	88.30	85.86	92.80	89.82	91.98	87.43	86.15	90.78
KNN-5	89.89	93.42	87.59	85.61	90.26	91.25	93.84	88.42	86.03	92.71	90.80	92.74	87.68	84.99	92.68
LR	85.70	91.58	81.41	78.40	86.09	84.64	91.23	78.57	75.54	86.82	83.07	88.96	76.69	71.94	85.53
RF-100	95.21	97.16	93.80	92.82	95.56	94.97	96.81	93.23	91.95	95.86	92.21	94.37	89.41	87.52	93.51
RF-500	94.74	96.76	93.02	91.65	95.36	95.06	96.99	93.53	92.50	95.47	92.13	94.46	89.55	88.07	92.91
SVM	90.54	94.42	87.93	85.77	91.18	90.02	93.90	87.07	84.66	91.08	87.99	91.78	83.86	80.40	90.28
XGboost	95.16	97.14	93.45	91.83	96.25	95.06	96.91	93.30	91.92	96.07	92.84	95.03	89.97	87.14	95.34

A: Accuracy, F1: F1 score, R: Recall, P: Precision

Table VI
Mean (Average), Maximum, and Minimum Accuracy Values for Different Feature Selection Models

Selection models	20					15					10				
	MAX	MIN	MEAN	FT	ST	MEAN	MAX	MIN	FT	ST	MIN	MEAN	MAX	FT	ST
ANOVA F-value	97.04	76.96	91.09	228.58	4.98	96.61	75.30	91.07	308.07	6.93	86.46	75.09	82.26	123.15	7.52
Information gain	98.62	73.52	93.05	340.72	4.13	98.52	73.52	92.98	461.53	5.72	98.48	60.84	91.75	315.85	5.10
RFE-DT	98.67	72.24	92.44	488.23	5.03	98.71	73.47	92.30	731.81	5.90	98.64	69.76	91.87	535.74	4.87
RF	98.46	67.83	91.95	296.79	5.08	98.47	67.50	91.42	415.34	5.46	98.47	67.39	91.87	276.53	5.02
XGboost	98.22	67.03	91.49	123.83	4.35	98.15	62.49	91.11	454.45	5.17	98.21	66.84	91.62	123.26	4.41

FT: Fit Time, ST: Score time

Table VII
Means for Different Performance Measurements with Over-/Under-sampling Models Tested with Various ML Models

ML model	None					SMOTETomek					SMOTEENN				
	A	F1	R	P	AUC	A	F1	R	P	AUC	A	F1	R	P	AUC
AB	97.89	99.61	97.45	96.85	98.05	98.27	99.77	98.26	97.89	98.63	99.70	99.99	99.70	99.65	99.75
BC-100	98.55	99.63	98.25	97.94	98.57	98.94	99.80	98.94	98.85	99.04	99.81	99.98	99.81	99.83	99.80
DT	97.60	97.53	97.11	97.15	97.08	98.29	98.29	98.29	98.33	98.25	99.71	99.71	99.71	99.72	99.71
ET-100	98.61	99.72	98.32	97.94	98.70	99.03	99.87	99.03	98.90	99.17	99.87	100	99.87	99.86	99.89
ET-500	98.63	99.75	98.34	97.98	98.71	99.03	99.89	99.03	98.91	99.16	99.88	100	99.88	99.89	99.88
GN	89.55	97.67	86.28	79.01	95.02	88.72	97.87	87.72	80.60	96.23	90.03	98.48	89.29	82.87	96.79
KNN-3	98.32	98.91	97.97	97.18	98.77	98.72	99.38	98.72	98.30	99.13	99.78	99.91	99.78	99.71	99.84
KNN-5	98.36	99.13	98.00	97.11	98.92	98.76	99.53	98.75	98.27	99.24	99.74	99.93	99.74	99.60	99.87
LR	91.27	94.43	89.10	85.79	92.68	90.94	94.77	90.72	88.57	92.98	92.94	95.81	92.83	91.18	94.55
RF-100	98.62	99.70	98.34	98.07	98.60	98.95	99.85	98.95	98.87	99.03	99.86	100	99.86	99.91	99.82
RF-500	98.63	99.73	98.34	98.10	98.59	98.98	99.87	98.98	98.90	99.06	99.84	100	99.84	99.89	99.79
SVM	98.45	99.65	98.13	97.64	98.62	98.76	99.77	98.76	98.42	99.10	99.68	99.97	99.68	99.56	99.79
XGboost	98.62	99.74	98.33	98.03	98.63	98.98	99.88	98.98	98.87	99.10	99.89	100	P. 89	99.89	99.89

A: Accuracy, F1: F1 score, R: Recall, P: Precision

Table VIII
Five Best AUC-ROC Achieved ML Models

Selection model	ROC AUC mean
SVM	75.75
ET-500	75.03
KNN-5	74.73
ET-100	74.66
KNN-3	74.40
SVM	75.75
ET-500	75.03

for the 20-feature approach, geo_enabled, Screen_name_freq, Name_freq, screen_name_length, and name_length features have been chosen with features from the 15-feature set to complete the 20-feature list. In addition, we draw the SHAP plot (Lundberg, Erion, and Lee, 2018) for the dataset using a RF model. The X-axis represents a SHAP value for each feature and the Y-axis shows features name order by most to least influential. In the SHAP plot, a positive value means pushing to one (bot-like), and a negative value means driving to zero (human-like). Color is another indication in the SHAP

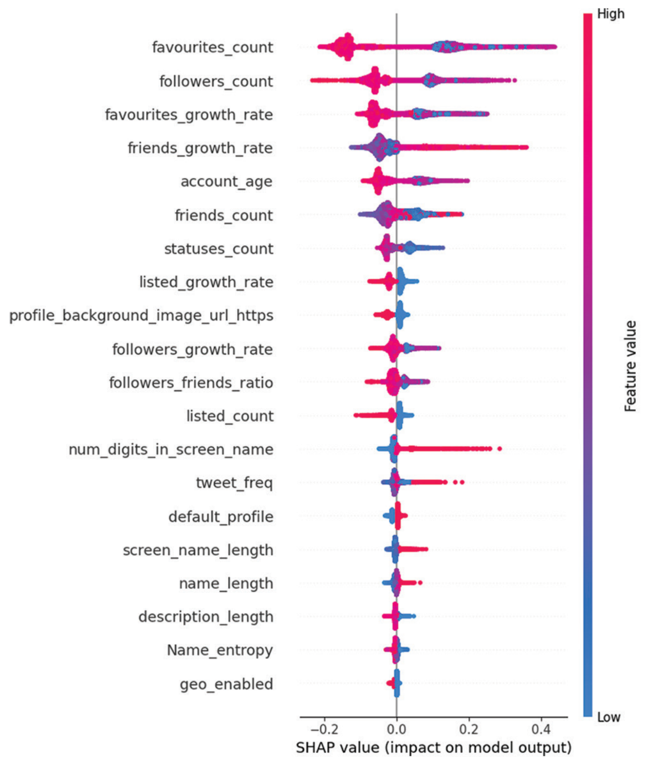

Fig. 2. The SHAP graph for our dataset. The graph shows the 20 features with the highest impact on the dataset. Using RF and Euclidean normalization.

plot, as shown on the right side of the Y-axis; when the value of features is large, the color moves to red, but when the value of the features is small, the color moves to blue. The plot shows that the original statistic features and statistically derived features are highly effective and ranked at the top of the graph. Furthermore, some features drive the model in one direction; for example, in the case of the num_digits_in_screen_name feature, when the number of digits is increased, the result moves toward one (bot-like), and in the case of the listed_count feature, when the number rises the model drive toward zero (human-like). Fig. 2 shows the result of the SHAP plot, the order of the features may be slightly different than what we mentioned in the text because the SHAP plot only depends on one algorithm, but our experience is based on five algorithms.

C. Oversampling and Undersampling Evaluation

We use the synthetic minority oversampling technique, Tomek Link (SMOTETomek), and Edited Nearest Neighbor (SMOTEENN) to reshape the dataset size. The original dataset includes 58K of humans and 42K of bots; SMOTEENN reshapes the dataset to 54K for each human and bot, and SMOTETomek reshapes the dataset to 57K for each human and bot. Our experiment (Fig. 3) used Gaussian normalization and 10 best features based on feature selection experiment. In addition, an original dataset was used without any changes. As shown in Table VII, reshaping the dataset increases performance measurements positively. When SMOTETomek and SMOTEENN were applied, all ML

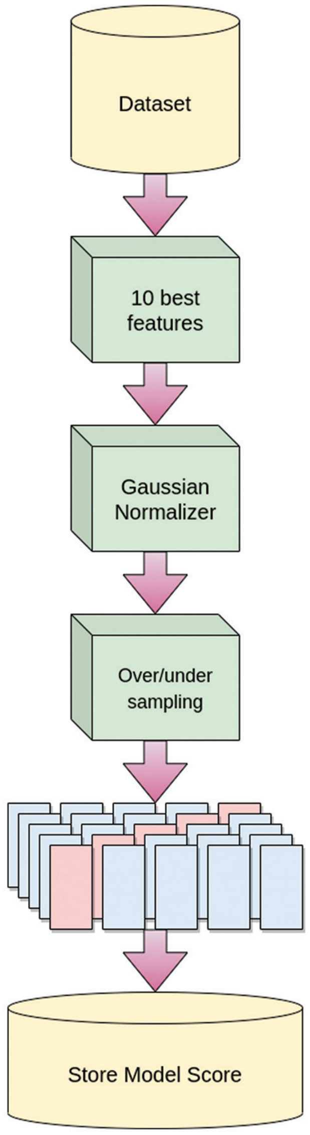

Fig. 3. The proposed experiments to determine the impact of over-/under-sampling algorithms on bot detection performance.

models' measurement performance increased (GN accuracy decreased with SMOTETomek as an exception). However, when we compare SMOTEENN and SMOTETomek, SMOTEENN performed much better than SMOTETomek. Furthermore, we used some oversampling algorithms such as SMOTE, ADASYN, and Borderline-SMOTE to reshape a dataset. Those algorithms increase the samples of lower class (oversampling) without performing any undersampling. The results are inferior, and usually, the original dataset performs better (their results are not mentioned in Table VII).

D. Generalization

Creating new types of bot with a different specialty are easy; because of that, many Botmasters develop new types of a bot that can avoid detection by the previous models. Therefore, a model's ability to distinguish between humans and bots outside their datasets (unseen data) is essential for any machine learning model, and this characteristic is

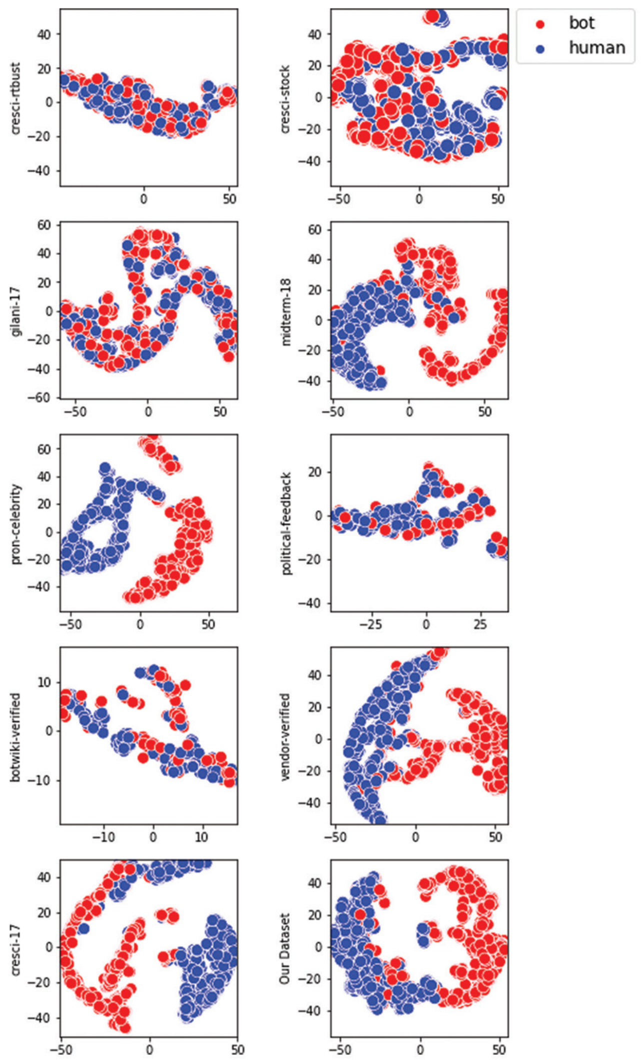

Fig. 4. The distribution of bots and humans for all the datasets using a TSNE plot, 1000 samples for each class.

called generalization. We use nine datasets out of 11 from Yang, et al., 2019, work to test our models capability for generalization. The dataset's names are Cresci-Rtbust, Cresci-stock, Gilani-17, midterm-18, pron-celebrity, political-feedback, Botwiki-verified, vendor-verified, and Cresci-17. (Fig. 4 shows the distribution of each dataset). However, Varol-Icwsm and Caverlee datasets are excluded because their public version only contains id (unique id for an account) and the remaining features are missing.

We trained the model with one of the datasets (including ours) and tested the model with the remaining datasets separately. The 10 best score features are used in feature selection experiment, SMOTEENN for balancing dataset, and Gaussian normalization. (Fig. 5 shows the steps for the experiment). Table VIII shows five ML models achieved the highest ROC AUC when the models were trained on one dataset and tested with the remaining datasets.

The SVM model achieves the best AUC average, but when compared to the result of SVM and ET-500, the generalization of ET-500 is higher than SVM. As a result, Fig. 6 is a complete comparison of all datasets based on the ET-500

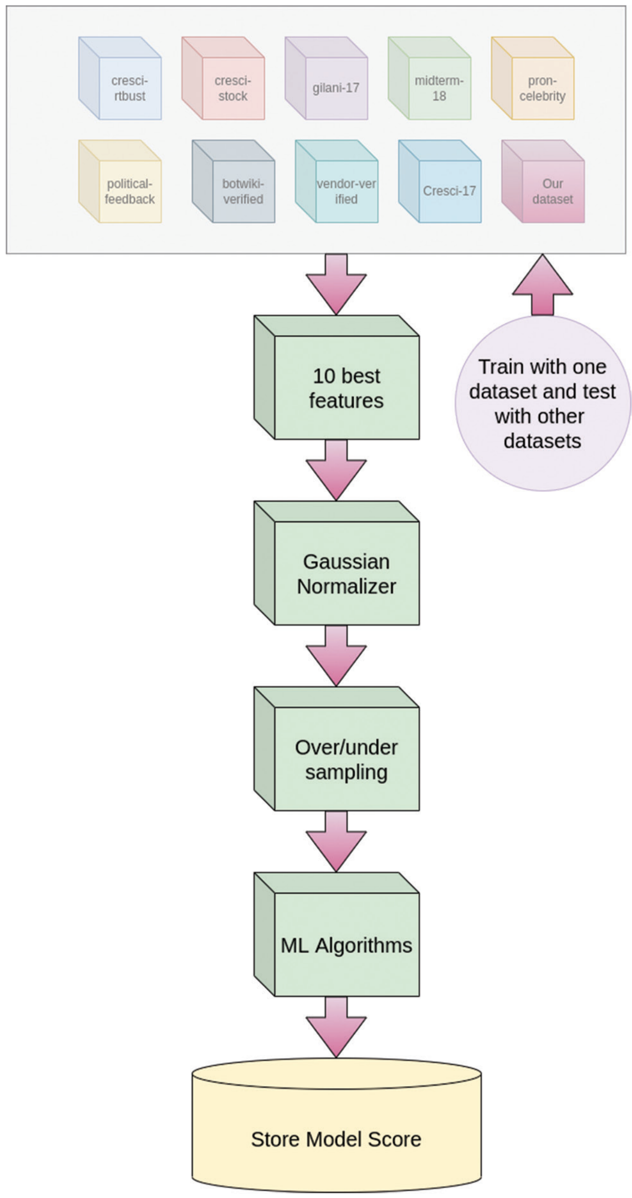

Fig. 5. The generalization experiments using one dataset for training and other datasets for testing.

model. As shown in Fig. 6, our dataset achieved the best average accuracy among all datasets, 6% higher than others. Moreover, it achieved the best result for three datasets; the other two achieved good results, but the remaining datasets performed poorly. An average of 75% is not satisfying for a realistic bot detection model.

Nevertheless, two critical points explain why the results are inadequate. First, our models are trained with 10 features from metadata which is insufficient for a real-life situation to increase the performance. Other features such as tweet content need to be considered. Second, there are different types of bots with different parameters; for example, fake followers and spammer bots have quite unlike specialties. Because of that, we thought in those cases, when the model performs poorly, the test dataset contains bots that do not exist in a trained dataset or vice versa.

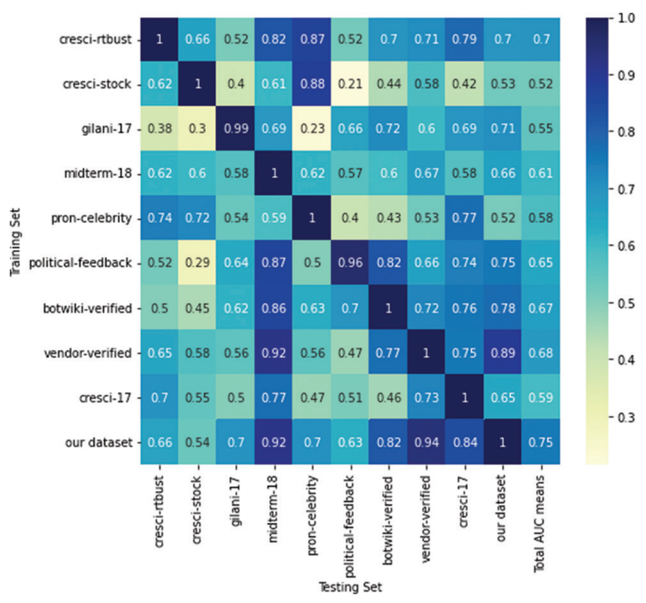

Fig. 6. AUC scores of ET-500 classifiers trained on one dataset and tested on others; the final columns also show each dataset's mean AUC.

As a result, we believe a dataset cannot achieve high performance for all other datasets unless to make the bot detection multi-class classification instead of a binary one. Instead of having one class for bots, we should create different bot classes such as spammer, fake follower, and others. Furthermore, it is worth mentioning that in our experiment, the result of the comparison between other datasets is much lower (besides our dataset) than the ones mentioned in Yang, et al., 2019, paper. We thought that it was because of a decrease in the number of features; the same experiment was repeated with the 20 features mentioned in Yang, et al., 2019, paper, which increased the average result by almost 2%, but still considerably less than the number in the original paper.

VI. Conclusion

This paper proposes novel benchmarks (datasets) with 100K samples based on 1.5M metadata collected from Twitter API. We collected data for more than 100 days to capture various bot types related to controversial topics like the US presidential election. Moreover, we use state-of-the-art online API to obtain the ground truth labels for the benchmark. The dataset includes 100K samples, and to the best of our knowledge, it is the largest only metadata dataset in this research area. Furthermore, this paper applied various autofeature selections and over-/under-sampling to the benchmark to increase the benchmark's generalization and scalability, reduce training time, and prevent over-/under-fitting while achieving very accurate results based on five-cross validation. As a result, our dataset achieved better AUC compared to other datasets by 6% in the case of generalization. Furthermore, applying the SMOTEENN technique achieved 2% higher results than the original dataset. In the future work, we intend to increase dataset generalization by including additional features rather than metadata features. Since we thought that bot detection on Twitter is a multiclass nature problem, we planned to create a multiclass benchmark instead of a binary one. Dataset and experiment codes are available on this GitHub link to support overcoming the lack of datasets for the future research.

References

Adewole, K.S., Anuar, N.B., Kamsin, A., Varathan, K.D. and Razak, S.A., 2017. Malicious accounts: Dark of the social networks. *Journal of Network and Computer Applications*, 79, pp.41-67.

Alom, Z., Carminati, B. and Ferrari, E., 2018. Detecting spam accounts on Twitter. In: *Proceedings of the 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2018*. Institute of Electrical and Electronics Engineers Inc., Piscataway, New Jersey. pp.1191-1198.

Altmann, A., Toloşi, L., Sander, O. and Lengauer, T., 2010. Permutation importance: A corrected feature importance measure. *Bioinformatics*, 26(10), pp.1340-1347.

Brown, P.F., de Souza, P.V., Mercer, R.L., Della Pietra, V.J. and Lai, J.C., n.d. *Class-Based n-gram Models of Natural Languag*e. Computational linguistics, 18, pp.467–480.

Davis, C.A., Varol, O., Ferrara, E., Flammini, A. and Menczer, F., 2016. BotOrNot: A System to Evaluate Social Bots. In: *Proceedings of the 25th International Conference Companion on World Wide Web*. pp.14-16.

Derhab, A., Alawwad, R., Dehwah, K., Tariq, N., Khan, F.A. and Al-Muhtadi, J., 2021. Tweet-based bot detection using big data analytics. *IEEE Access*, 9, pp.65988-66005.

Elhassan, T. and Aljurf, M., 2016. Classification of imbalance data using tomek link (t-link) combined with random under-sampling (rus) as a data reduction method. *Global J Technol Optim*, 1, pp.1-11.

Feng, S., Wan, H., Wang, N., Li, J. and Luo, M., 2021. TwiBot-20: A comprehensive twitter bot detection benchmark. *arXiv*, 2021, p.13088.

Ferrara, E., Varol, O., Davis, C., Menczer, F. and Flammini, A., 2016. The rise of social bots. *Communications of the ACM*, 59, pp.96-104.

Granitto, P.M., Furlanello, C., Biasioli, F. and Gasperi, F., 2006. Recursive feature elimination with random forest for PTR-MS analysis of agroindustrial products. *Chemometrics and Intelligent Laboratory Systems*, 83, pp.83-90.

Hanouna, S., Neu, O., Pardo, S., Tsur, O. and Zahavi, H., 2019. Sharp power in social media: Patterns from datasets across electoral campaigns. *Australian and New Zealand Journal of European Studies*, 11, pp.95-111.

Hayawi, K., Mathew, S., Venugopal, N., Masud, M.M. and Ho, P.H., 2022. DeeProBot: A hybrid deep neural network model for social bot detection based on user profile data. *Social Network Analysis and Mining*, 12, p.43.

Huang, J. and Ling, C.X., 2005. Using AUC and accuracy in evaluating learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 17, pp.299-310.

Jović, A., Brkić, K. and Bogunović, N., 2015. A review of feature selection methods with applications. In: 2015 *38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE. pp.1200-1205.

Khanday, A.M.U., Khan, Q.R. and Rabani, S.T., 2021. Identifying propaganda from online social networks during COVID-19 using machine learning techniques. *International Journal of Information Technology (Singapore)*, 13, pp.115-122.

Kudugunta, S. and Ferrara, E., 2018. Deep neural networks for bot detection. *Information Sciences*, 467, pp.312-322.

Martin-Gutierrez, D., Hernandez-Penaloza, G., Hernandez, A.B., Lozano-Diez, A. and Alvarez, F., 2021. A deep learning approach for robust detection of bots in twitter using transformers. *IEEE Access*, 9, pp.54591-54601.

Orabi, M., Mouheb, D., Al Aghbari, Z. and Kamel, I., 2020a. Detection of bots in social media: A systematic review. *Information Processing and Management*, 57, p.102250.

Orabi, M., Mouheb, D., Al Aghbari, Z. and Kamel, I., 2020b. Detection of bots in social media: A systematic review. *Information Processing and Management*, 57, p.102250.

Peng, H., Long, F. and Ding, C., 2005. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27, pp.1226-1238.

Rodríguez-Ruiz, J., Mata-Sánchez, J.I., Monroy, R., Loyola-González, O. and López-Cuevas, A., 2020. A one-class classification approach for bot detection on Twitter. *Computers and Security*, 91, 101715.

Shannon, C.E. and Weaver, W., 1949. *The Mathematical Theory of Communication*. The University of Illinois Press, Urbana, IL.

Shaw, R.G. and Mitchell-Olds, T., 1993. ANOVA for unbalanced data: An overview. *Ecology*, 74, pp.1638-1645.

Shevtsov, A., Tzagkarakis, C., Antonakaki, D. and Ioannidis, S., 2021. Identification of Twitter Bots Based on an Explainable Machine Learning Framework: The US 2020 Elections Case Study. *Proceedings of the International AAAI Conference on Web and Social Media*.

Shukla, H., Jagtap, N. and Patil, B., 2021. Enhanced twitter bot detection using ensemble machine learning. In: *Proceedings of the 6th International Conference on Inventive Computation Technologies, ICICT 2021*. Institute of Electrical and Electronics Engineers Inc., Piscataway, New Jersey. pp.930-936.

Varol, O., Ferrara, E., Davis, C.A., Menczer, F. and Flammini, A., 2017. Online Human-bot Interactions: Detection, Estimation, and Characterization. In: *Proceedings of the 11th International Conference on Web and Social Media, ICWSM 2017*, pp.280-289.

Wang, J., Xu, M., Wang, H. and Zhang, J., 2006. Classification of Imbalanced Data by Using the SMOTE Algorithm and Locally Linear Embedding. In: *2006 8th International Conference on Signal Processing*. IEEE.

Wilson, D.L., 1972. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, 3, pp.408-421.

Yang, K.C., Varol, O., Hui, P.M. and Menczer, F., 2019. Scalable and generalizable social bot detection through data selection. *arXiv*, 2019, p. 09179.