Detecting Twitter bots using Unsupervised Machine Learning and Natural Language Processing

Meagan Vu
School of Engineering
Santa Clara University
Santa Clara, CA
mavu@scu.edu

Haozhe Zhang
Khoury College
Northeastern University
Boston, MA
zhang.haozhe1@northeastern.edu

Fabio Di Troia
Department of Computer Science
San Jose State University
San Jose, CA
fabio.ditroia@sjsu.edu

Younghee Park
College of Engineering
San Jose State University
San Jose, CA
younghee.park@sjsu.edu

Abstract— **The proliferation of bots on social media platforms has experienced an exponential growth in recent years. Although supervised learning methods have demonstrated effectiveness in controlled environments, their real-life applicability remains questionable due to inconsistent results. In this research, we propose a novel approach to detect computer-generated tweets using unsupervised machine learning techniques, specifically OCSVM and K-Means. Prior to applying these methods, the tweets are preprocessed through natural language processing techniques, including DistilBERT. The primary advantage of employing unsupervised learning is its applicability to real-life scenarios where labeled data is scarce. Through extensive evaluations on publicly available datasets, namely cresci-2015 and cresci-2017, we assess the reliability and efficacy of our proposed models in identifying computer-generated tweets. By leveraging unsupervised techniques, this study aims to improve the robustness of bot detection in real-world social media environments, where accurate labeling is often limited.**

Keywords: Unsupervised – Autoencoder – Natural Language Processing – BERT – Twitter Social Bots

1. INTRODUCTION

Social Media platforms allow people from across the globe to connect, share their ideas, and interact with one another with ease. Unfortunately, this also means that people with malicious intent have access to these platforms to push their ideals, promote political agendas, and disrupt the peace in online communities [5] [6]. People tend to implement the use of computer automated profiles, bots, for their personal gains.

Bots are computer automated accounts that do not require human intervention. Bots attempt to emulate human behavior, like speech and interaction patterns [20]. Online social networks, like Twitter, have the influence that spans across the globe. As of July 2023, Twitter has reported to have 353.9 million users [18], and twitter reports that 5% of these accounts are controlled by bots [19]. However, based on [19] findings, the content that these bots produce are anywhere between 20.8% and 29.2% of all tweets posted on the platform. Bots have the potential of being harmful to online communities. In the 2010 United States midterm election, it was observed that the use of bots was implemented to chastise the other candidate's campaign [21].

In this way, Twitter is a prominent target for these attacks; thus, it is necessary to find a way to counter these malicious bots. In this paper we focus on developing a way to use unsupervised machine learning to distinguish between bots and real users. It is rare for accounts to be provided with accurate labels or labels at all, and when they are, they are usually out of date. Bots are transforming at increasing rates to become indistinguishable from humans. We propose using unsupervised models to detect bot in order to get over this cavate.

In section 2, we discuss other related papers that cover their methodologies on bot detection. Section 3 explains our methodology of our experimentation. We discuss our datasets, how we preprocessed the samples and the parameter used for each unsupervised model. In section 4, we report our results for each model and analyze the outcome. Section 5 covers our plans for future works.

2. BACKGROUND

2.1 RELATED WORK

The topic of bot detection on Online Social Networks has been thoroughly reviewed and researched. With the evolution of the bots, many methods such as sentiment analysis [23], supervised [22] and unsupervised machine learning [24], and one class classification [25] have been implemented to distinguish them from real users.

Heidari and Jones [23] proposed a technique that used sentiment analysis and supervised machine learning models to predict the bot status of a Twitter account. They utilized BERT to implement sentiment analysis. Sentiment Analysis is a technique to analyze the emotion of the text sample. They were able to analyze if the tweets contained content that was positive, negative, or neutral. After they processed the text samples through sentiment analysis, they embedded the samples using Global Vectors (GloVe). This converted the text samples into data that the computer can understand. After preprocessing, Heidari and Jones ran the dataset into four supervised models, Feed-Forward Neural Network (FFNN), Random Forest, Support Vector Machine (SVM), and Logistic Regression.

In another bot detection study, Wu, Teng, and Cao [24] used an unsupervised approach. They extracted data from Twitter's API, most of which were unlabeled. Their dataset focused on the metadata of each user's account. Twitter provides basic metadata including follower counts, friend count, created at dates and more. They extracted features directly from Twitter's API, as well as created their own which included follower to following ratio, list number per day, tweets per day rate, and more. After extracting and creating their feature set, they implemented two unsupervised machine learning models to train and test on.

These models were K-Means and Agglomerative clustering algorithm.

Our methodology is based on the findings of other's previous works. We found that there was a lack of experimentation on the tweets alone, many studies such as [23] use tweets as another way to extract more features to describe the accounts. We plan to use BERT, similarly to [23], but to analyze the tweets themselves. We also wanted to explore the implementation of unsupervised models like [24] as we see the value of not utilizing labels. Similar to [23], [24] report the bot status of the accounts rather than the tweets. In addition, [24] also does not process users' tweets, rather they used their metadata. Our plan is to expand on the findings from [23] and [24] but put the emphasis on the tweets rather than the overall accounts.

2.2 Machine Learning Techniques

In this section, we discuss the natural language processing method we used as well as the different types of unsupervised machine learning models we implemented.

2.2.1 Natural Language Processing

Machines do not have the ability to understand and analyze natural human language like humans can. Thus, computers need the help of natural language processing (NLP). Natural Language Processing analyzes human languages using theoretical, statistical, and computational techniques [14]. One application for NLP models is converting non-numeric data into a format which computers can understand. One very popular NLP model is called Bidirectional Encoder Representations Transformer (BERT).

BERT. BERT is a pre-trained bidirectional neural network that considers both context both before and after the word to help with text embedding [7]. BERT can achieve this by training their model using "masked language model" (MLM) [13], which picks a token to mask at random and the machine attempts to predict the hidden word. This forces the machine to use both the left and right context, resulting in the ability to have a deep directional transformer [4].

DistilBERT. A popular adaptation of BERT is DistilBERT [8], a compact version of BERT. DistilBERT can replicate the results of BERT while using less computational resources and time, making DistilBERT more efficient. DistilBERT requires less parameters compared to BERT's several hundred million, and it does this with half the number of layers [8]. Compared to its predecessor, DistilBERT is scaled down by 40% while maintaining 97% of BERT's language understanding abilities. In addition to the reduction in size, DistilBERT is 60% faster that BERT [8].

2.2.2 Unsupervised Machine Learning

Bot detection via supervised methods have been heavily explored, but many have not been able to provide high accuracy scores when tested in real time. This is because labels are rarely provided in real life. Due to this, unsupervised model can provide more realistic results, as labels are not necessary for classification [3].

K-Means. K-Means is an unsupervised machine learning clustering algorithm that plot samples into a n-dimensional space [9]. From there, the machine will section the dataset into k-number of clusters, where k is predefined. The first step of K-Means is randomly selecting k points to represent the initial center of the clusters [15]. The algorithm for creating the clusters goes as follows:

1. Measure the distance between given point and each cluster's centroid.
2. Assign the point to a cluster based on which cluster gave the shortest Euclidian distance.
3. After assigning the point, calculate the new center of the cluster.
4. Repeat this process for each point in the dataset.

The measurement used to determine the distance between a point and a centroid is the Euclidian distance. The Euclidian distance formula goes as follows:

$$d(x_i, y_i) = \left[\sum_{i=1}^{n} (x_i - y_i)^2\right]^{\frac{1}{2}}$$

where x is the vector of the individual point and y is the vector of the cluster's centroid. Once the machine is finished assigning all points to their clusters, it will find the sum of the squared error for each cluster. This calculation is called the criterion function and it is defined as:

$$E = \sum_{i=1}^{k} \sum_{x \in C_i} |x - x_i|^2$$

where x is the individual data point and x_i is the average of the cluster. The model will repeat this process for a specified number of times. The clusters that minimize this criterion function will then be the ultimate clustering format that the machine uses for future classification.

OCSVM. Schölkopf created the One Class Support Vector Machine (OCSVM) model based off the supervised learning algorithm Support Vector Machine (SVM) [27]. OCSVM is an unsupervised machine learning model that is commonly implemented in anomaly detection use [10]. Anomaly detection is a technique that is used to detect rare or unusual patterns within the dataset. These samples usually deviate from the rest of the of the dataset and are called, anomalies.

OCSVM used for anomaly detection is trained on "normal" samples and creates a decision boundary around these datapoints. No labels are necessary for training as it is assumed that the model is being trained on all good samples. The samples within this boundary are called inliers. In order to create a non-linear boundary between the inliers and outliers OCSVM uses a kernel function. Functions often used for the kernel parameter are the Radial Basis Function (RBF) or Gaussian kernel.

OCSVM uses a decision function to classify unseen information as an inlier or an outlier. The decision function classifies a sample as an inlier or outlier based on the score it assigns the sample. The decision function goes as followed:

$$f(x) = sin(w^{*} * \Phi(x) - \rho)$$

where $\Phi(x)$ is the formula of how the kernel function maps the input data into a higher dimension. w and ρ are the weights and thresholds respectively.

The model also implements an optimization step that attempts to separate the normal samples from the origin of the higher dimensional space. To achieve this, the optimization step tries to maximize the distance between the points and the decision boundary, similar to SVM. During the optimization process, the weight vector w and the threshold ρ are altered until this maximized distance is found.

After optimization, the model analyzes new input data and classifies the sample as an inlier or an outlier. This is decided is by the decision function mentioned previously. The decision goes as followed:

$$c(n) = \begin{cases} f(x) \geq 0 \; inlier \\ f(x) < 0 \; outlier \end{cases}$$

Isolation Forest. Isolation Forest are sets of binary decision trees or iTrees that isolate data points [11]. Like Random Forest [16], the final classification is determined by the results of each decision tree it went through. Isolation is commonly used for anomaly detection.

The way that each isolation tree is trained goes as followed:

1. Select a random subset of features from the dataset.
2. For each feature within the subset of features, pick a partitioning value. The tree splits based on this partitioning value. If a sample's value for this feature is less than partitioning value than the sample will travel to the left side of this node.
3. This partitioning process is repeated until the maximum tree depth is reached, which is predetermined, or until a certain amount of datapoints fall before a threshold.

This process is repeated for each iTree, picking a new feature from the randomly selected feature set to train on. The number of trees within the Isolation Forest is predetermined prior to running the model.

The number of partitions it takes for a point to be isolated is call the path length. For a singular sample, the path lengths of each iTree are averaged and based off the average length, the sample is given an anomaly score [17]. The formula for the anomaly score goes as follows:

$$s(x, n) = 2^{\frac{-E(h(x))}{c(n)}}$$

where $E(h(x))$ is the average depth that data point, x, traverse through for all iTrees in the Isolation Forest, and $c(n)$ is the average depth of an unsuccessful search in a Binary Search Tree. The formula for $c(n)$ is:

$$c(n) = 2H(n-1) - (\frac{2(n-1)}{n})$$

where $H(i)$ is characterized as: $\ln(i) + 0.5772156649$. Based on all the datapoints average path length, the computer automatically assigns a threshold for determining whether the sample is an anomaly or not. If the average path length to isolate the one data point is significantly shorter than the threshold value, that data point is flagged as an anomaly.

Autoencoder. An Autoencoder is a neural network used in unsupervised machine learning. Autoencoders are comprised of two components: encoders and decoder. The function of an encoder is to project the input data, which resides in a higher dimensional space, and projects the data into a lower dimensional space. The input vector is fed into a hidden layer containing m nodes and the activation function in each node is represented as the following formula [12]:

$$h_i = f_\theta(x) = s(\sum_{j=1}^{n} W_{ij}{}^{input} x_j + b_i{}^{input})$$

where W^{input} is the encoder's weight matrix, b^{input} is the bias vector, and x is the input vector to the encoder.

The decoder's responsibility is to reverse engineer the output of the encoder back to the data's original form. The formula of the decoder goes as follows [12]:

$$x_i{}' = g_{\theta'}(h) = s(\sum_{j=1}^{n} W_{ij}{}^{hidden} h_j + b_i{}^{hidden})$$

The way that autoencoder detect anomalies is by their reconstruction error. The reconstruction error is the difference between the original and reconstructed data. The formula [12] for the reconstruction error goes as followed, where x_i and $x_i{}'$ are the original data and reconstructed data, respectively, and d describes the dimension that the vector originates from.

$$\varepsilon(x_i, x'_i) = \sum_{j=1}^{d} (x_i - x'_i)^2$$

This reconstruction error should be consistent for most "normal" data samples as it was trained to reconstruct similar samples. The determining number that distinguishes normal reconstruction error from suspicious reconstruction error is called the threshold [12]. If a data sample's reconstruction error surpasses this threshold, the sample will be flagged as an anomaly. As all datasets are different, the threshold where the data is considered an anomaly will also be unique to the given dataset.

3. Methodology

In this section, we will discuss our data collection as well as what preprocessing methods we used. Following this, we will explain the parameters and training split we used for our unsupervised machine learning models.

3.1 *Dataset*

Twitter allows people to access metadata and other data, like tweets, through their API. Unfortunately, due to new restrictions, Twitter limits the retrieval rates causing collection to be slow and costly [26]. Thus, we resorted to using pre-made datasets from previous years.

We used datasets cresci-2015 [2] and cresci-2017 [1], where we obtained over nine million samples. Both datasets gave us features such as "created_at", "user_id", "retweet_count" and more. We were most interested in how tweets preformed in an unsupervised model, so we only extracted the "text" feature. Each dataset had a set of bots and a set of humans along with labels and in total we had over 9 million samples, which were comprised of 3.9 million bot tweets and 5.5 million human tweets.

Dataset	Bots Tweets	Humans Tweets
cresci-2015	196,027	2,631,730
cresci-2017	3,798,254	2,839,362
Total	3,994,281	5,471,092

Figure 1. Datasets Bot and Human Tweet Counts

3.2 *Data Preprocessing*

While preprocessing the data we ran into an issue due to limited computational resources. We ran the experiment on Google Colab which offers 12.7 GB of RAM, Intel Xeon CPU 2.20GHz, and 1 CPU core. This reduced the number of samples we were able to process. To accurately represent all aspects of our dataset, we decided to include approximately 800,000 bot samples (400,000 to train the model and 400,000 to test the model) and 400,000 humans samples.

Within the smaller dataset, there were a handful of null values. There are multiple methods to deal with these null values, such as replacing the null values with the average value, inputting the most common value, or removing the sample completely. Since we are only including text samples, we could not average the values. We also found that replacing null values with the most common value did not apply to our dataset. Thus, we decided on deleting null samples.

This resulted in our final dataset having 1,195,282 samples. The method that we chose to preprocess our text samples was DistilBERT. DistilBERT represents the text embedding in *n* by 768 arrays, where *n* is the number of tokens in the text sample. The most important token from this embedding is the classification (CLS) token, which is the first token of the embedded text. To simplify and reduce noise in the input data, we extracted the CLS token and used that to represent our data in the models. This results in a 1x768 size array that we then merged into a final dataset that we could feed into our unsupervised machine learning model.

3.3 *Training and Testing Classifiers*

The models we chose to implement were K-Means, One Class Support Vector Machine, and Isolation Forest.

OCSVM, Isolation Forest, and Autoencoder were implemented to preform anomaly detection thus we trained those models on 400,000 of the bot tweet samples. We then combined 400,000 bot tweets and 400,000 human tweets to generate the test accuracy score.

We used grid search on OCSVM and Isolation Forest to find the "kernel" parameter and "n_estimator" parameter respectively. We found that having 100 estimators for Isolation Forest worked best. For OCSVM we found that "rbf" was the best hyperparameter. Because there were only bots in the training sample of both models, we had a small "nu" value and "contaminator" value, both of 0.01, for OCSVM and Isolation Forest respectively.

For Autoencoders, we used the following hyperparameters:

Hyperparameter	Value
input_shape	(770,)
intermediate_dim	256
latent_space_dim	10
dropout_rate	0.3
learning_rate	2×10^{-5}
batch_size	32
epochs	5

Figure 2. Autoencoder Hyperparameters

Once we trained our neural network on the training bot samples, the model gave us a threshold of 0.22 to distinguish between the acceptable and unacceptable reconstruction error. If the reconstruction error for the sample is larger than 0.22 then it will be flagged as an anomaly.

4. Results

In Figure 3, 4, and 5 we report the results from each unsupervised model.

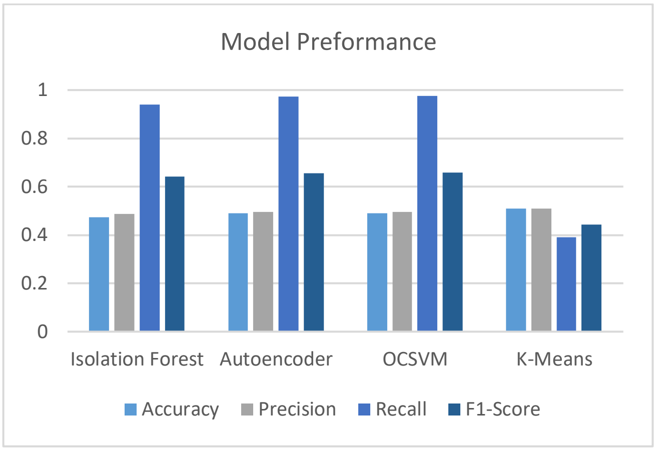

Figure 3. Unsupervised Models Accuracy Graph

	Accuracy	Precision	Recall	F1-Score
Isolation Forest	47.45%	48.70%	94.00%	64.16%
Autoencoder	49.02%	49.52 %	97.34%	65.65%
OCSVM	49.09%	49.56%	97.55%	65.72%
K-Means	50.90%	50.90%	39.14%	44.26%

Figure 4. Unsupervised Models Metrics Report

We used the models OCSVM, Isolation Forest, and Autoencoders to preform anomaly detection. For OCSVM, we found that our accuracy for correctly classifying human tweets was extremely low. As shown in Figure 4, 397,147 of 399,321 (99.5%) human tweets were classified as bot tweets, while only 2,174 (0.5%) were correctly labeled as human generated. OCSVM model was significantly better at classifying bots as it labeled 97.55% of bots correct. This resulted in an accuracy score of 49.09%.

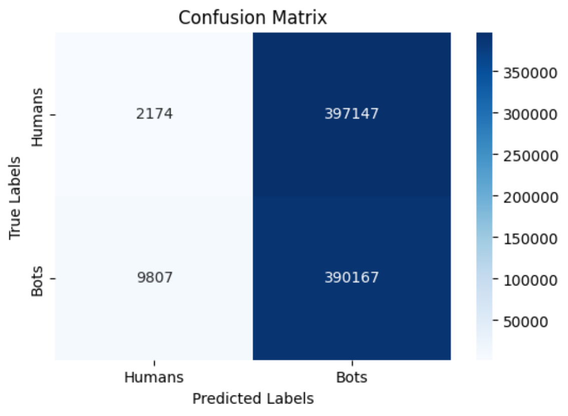

Figure 5. Confusion Matrix for Human Tweets in OCSVM

We obtained similar results from our Isolation Forest and Autoencoders models, which gave us a 47.45% accuracy score and 49.02 % respectively. For Isolation Forest 99.17% of the human tweet test samples were incorrectly classified as computer generated, while 94% of bot tweets were correctly categorized. This could imply that the two classes are so similar to each other that the machine is unable to tell them apart both in isolation path length and reconstruction for Isolation Forest and Autoencoder respectfully.

As a reminder, all three models were trained only on bot tweet samples. These results indicate to us that the machines are unable to distinguish between embedded bot tweets and embedded human tweets. For OCSVM, the model succeeds in creating a hyperplane that defines a clear decision boundary around the bot tweets, but the majority of the human tweets are also mapped within the boundary. This same logic applies for Isolation Forest and Autoencoders. The models do not see much variation between the two classes. This causing them to correctly identify bots but

Our K-Means model attempted to separate our plotted data samples into two distinct sections, bot tweets and human tweets. Similarly to OCSVM, the issue is that the bot embedded samples and the human embedded samples have a lot of overlap, making it difficult to accurately separate the data into two clear sections. Using K-Means' categorization, we were able to get an accuracy of 50.90%.

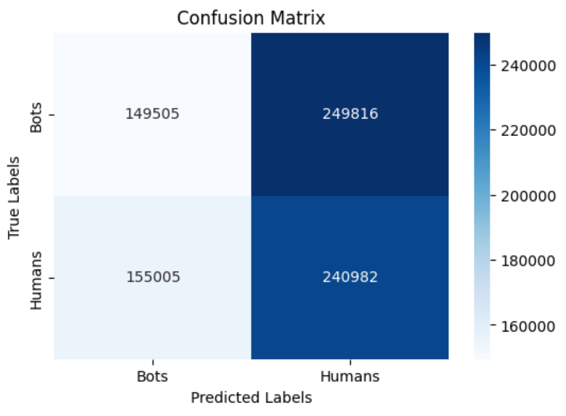

Figure 6. Confusion Matrix for K-Means

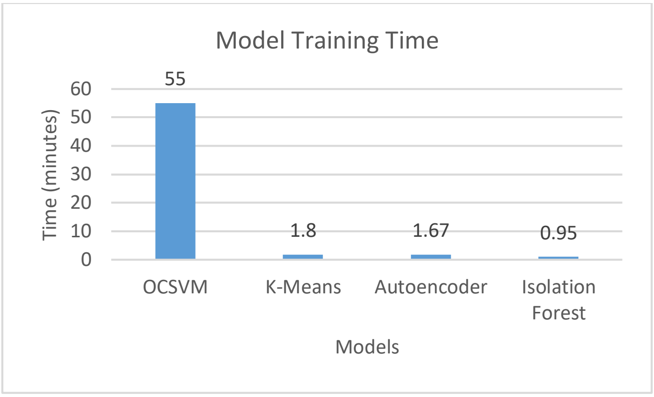

Figure 7. Confusion Matrix for K-Means

In Figure 7, we see that most of the models took approximately one minute to train the model. The only model that took significantly longer was OCSVM which require 55 minutes to train the model on around 400,000 bot samples.

5. Conclusion and Future Works

We experimented with natural language processing and unsupervised machine learning to detect twitter bots. From our experimentation we found that the methodology of using DistilBERT as the sole preprocessing method is not sufficient. We infer that the NLP models do not see variation in the different class tweets, thus it will embed both classes in similar manners. Using NLP's to embed tweets does not distinguish bot classes and human classes enough for our unsupervised machines to see any underlying differences. Before dismissing tweet bot status classification, we must do further research on other preprocessing methods that will better distinguish bot tweets from human tweets.

In the future, we would want to embellish in more feature engineering to gain more context for the tweets. One method that is very popular when using NLP and machine learning to detect twitter bots is using a method called semantic analysis. We would also like to implement more unsupervised learning models. We recognize that our data preprocessing technique is most likely the reason why our models are preforming so poorly. Hence, we would like to improve our preprocessing methods before implementing more advance machine learning model such as neural networks. In addition to this, we would like to implement a real time bot detection simulation using the resource of Twitter API. The theory behind unsupervised machine learning methods for up-to-date bot detection still stands, but we must first find an effective way to preprocess the data to create two distinguishable classes.

References

[1] Cresci, S., Di Pietro, R., Petrocchi, M., Spognardi, A., Tesconi, M. (2017). The paradigm shift of social spambots: Evidence, theories, and tools for the arms race. In Proceedings of the 26th International Conference on World Wide Web Companion (pp. 963-972).

[2] Cresci, S., Di Pietro, R., Petrocchi, M., Spognardi, A., & Tesconi, M. (2015). Fame for sale: efficient detection of fake Twitter followers. Decision Support Systems, 80, 56-71.

[3] Chen, R.S. Tanash, R. Stoll and D. Subramanian, "Hunting Malicious Bots on Twitter: An Unsupervised Approach," in Social Informatics. SocInfo 2017. Lecture Notes in Computer Science, vol. 10540, 2017,

pp. [page numbers]. [Online]. Available: https://doi.org/10.1007/978-3-319-67256-4_40.

[4] H. Choi, J. Kim, S. Joe, and Y. Gwon, "Evaluation of BERT and ALBERT Sentence Embedding Performance on Downstream NLP Tasks," in *2020 25th International Conference on Pattern Recognition (ICPR)*, Milan, Italy, 2021, pp. 5482-5487, doi: 10.1109/ICPR48806.2021.9412102.

[5] A. Thieltges, O. Papakyriakopoulos, J.C.M. Serrano, and S. Hegelich, "Effects of social bots in the iran-debate on twitter," arXiv preprint arXiv:1805.10105, 2018.

[6] A. Shevtsov, C. Tzagkarakis, D. Antonakaki, and S. Ioannidis, "Identification of Twitter Bots Based on an Explainable Machine Learning Framework: The US 2020 Elections Case Study," in *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 16, no. 1, 2022, pp. 956-967, https://doi.org/10.1609/icwsm.v16i1.19349.

[7] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," arXiv preprint arXiv:1810.04805, 2018.

[8] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter," arXiv preprint arXiv:1910.01108, 2019.

[9] S. Na, Xumin L., and G. Yong, "Research on k-means Clustering Algorithm: An Improved k-means Clustering Algorithm," in *2010 Third International Symposium on Intelligent Information Technology and Security Informatics*, Jian, China, 2010, pp. 63-67, doi: 10.1109/IITSI.2010.74.

[10] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural computation*, vol. 13, no. 7, 2001, pp. 1443-1471.

[11] F. T. Liu, K. M. Ting, and Z. -H. Zhou, "Isolation Forest," in *2008 Eighth IEEE International Conference on Data Mining*, Pisa, Italy, 2008, pp. 413-422, doi: 10.1109/ICDM.2008.17.

[12] Z. Chen, C. K. Yeo, B. S. Lee, and C. T. Lau, "Autoencoder-based network anomaly detection," in *2018 Wireless Telecommunications Symposium (WTS)*, Phoenix, AZ, USA, 2018, pp. 1-5, doi: 10.1109/WTS.2018.8363930.

[13] Wilson L Taylor. 1953. Cloze procedure: A new tool for measuring readability. *Journalism Bulletin*, 30(4):415–433.

[14] K.R. Chowdhary, "Natural Language Processing," in *Fundamentals of Artificial Intelligence*. Springer, New Delhi, 2020. https://doi.org/10.1007/978-81-322-3972-7_19

[15] H. B. Lee and J. B. Macqueen, "A K-Means Cluster Analysis Computer Program With Cross-Tabulations and Next-Nearest-Neighbor Analysis," *Educational and Psychological Measurement*, vol. 40, no. 1, 1980, pp. 133–138. https://doi.org/10.1177/001316448004000118

[16] S. J. Rigatti, "Random forest," *Journal of Insurance Medicine*, vol. 47, no. 1, 2017, pp. 31-39.

[17] S. Hariri, M. C. Kind, and R. J. Brunner, "Extended Isolation Forest," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 4, pp. 1479-1489, 1 April 2021, doi: 10.1109/TKDE.2019.2947676.

[18] "How Many Users Does Twitter Have?" *BankMyCell*. [Online]. Available: https://www.bankmycell.com/blog/how-many-users-does-twitter-have. Accessed: August 8, 2023.

[19] "Twitter Bots: Research, News and Analysis," *SimilarWeb Blog*. [Online]. Available: https://www.similarweb.com/blog/insights/social-media-news/twitter-bot-research-news/. [Accessed: August 8, 2023].

[20] Emilio Ferrara, Onur Varol, Clayton Davis, Filippo Menczer, and Alessandro Flammini, "The rise of social bots," *Communications of the ACM*, vol. 59, no. 7, July 2016, pp. 96-104. https://doi.org/10.1145/2818717.

[21] J. Ratkiewicz, M. Conover, M. Meiss, B. Goncalves, A. Flammini, and F. Menczer, "Detecting and Tracking Political Abuse in Social Media," in *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 5, no. 1, Aug. 2021, pp. 297-304. DOI:https://doi.org/10.1609/icwsm.v5i1.14127.

[22] P. Pham, L. T. Nguyen, B. Vo, and U. Yun, "Bot2Vec: A general approach of intra-community oriented representation learning for bot detection in different types of social networks," *Information Systems*, vol. 103, 2022, 101771.

[23] M. Heidari and J. H. Jones Jr, "Bert model for social media bot detection."

[24] J. Wu, E. Teng, and Z. Cao, "Twitter Bot Detection Through Unsupervised Machine Learning," in *2022 IEEE International Conference on Big Data (Big Data)*, Osaka, Japan, 2022, pp. 5833-5839, doi: 10.1109/BigData55660.2022.10020983.

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

[26] S. Ghosh, G. Korlam, and N. Ganguly, "The Effects of Restrictions on Number of Connections in {OSNs}: A {Case-Study} on Twitter," in *3rd Workshop on Online Social Networks (WOSN 2010)*.

[27] S. Yue, P. Li, and P. Hao, "SVM classification: Its contents and challenges," *Applied Mathematics-A Journal of Chinese Universities*, vol. 18, 2003, pp. 332-342.