Article

Enhancing Security in Social Networks through ML: Detecting and Mitigating Sybil Attacks

José Antonio Cárdenas-Haro [*] , Mohamed Salem , Abraham Netzahualcoyotl Aldaco-Gastélum , Roberto López-Avitia , Maurice Dawson

Posted Date: 8 August 2024

doi: 10.20944/preprints202408.0542.v1

Keywords: Machine Learning; Sybil attacks; Online Social Networks; Cybersecurity; Random Forest; Support Vector Machine; KNN

Article

Enhancing Security in Social Networks through ML: Detecting and Mitigating Sybil Attacks

J. Antonio Cárdenas-Haro [1,*], Mohamed Salem [1], Abraham Aldaco-Gastelum [2], Roberto López-Avitia [3] and Maurice Dawson [4]

[1] School of Computer Sciences, Western Illinois University, 1 University Cir, Macomb 61455, Illinois, USA; mo-salem@wiu.edu
[2] Department of Computer Science, Iowa State University, 2434 Osborn Dr, Ames, 50011, Iowa, USA; aaldaco@iastate.edu
[3] School of Electronic Engineering, Mexicali Institute of Technology, Instituto Tecnológico s/n, Mexicali, 21376, Baja California, México; ravitia@itmexicali.edu.mx
[4] Center for Cyber Security and Forensics Education. College of Computing, Illinois Institute of Technology, 10 West 35th Street, Chicago, 60616, Illinois, USA; mdawson2@iit.edu
* Correspondence: a-cardenas-haro@wiu.edu

Abstract: This study contributes to the Sybil nodes detecting algorithm in Online Social Networks (OSNs). As major communication platforms, online social networks get significantly guarded from malicious activity. A thorough literature review identified various detection and prevention Sybil attack algorithms. Additional exploration of distinct reputation systems and their practical application led to the study's discovery of machine learning algorithms, i.e., the KNN, Support Vector Machines, and Random Forest algorithms. The study details the data cleansing process for the employed dataset in its process for optimizing the computational demands required to train machine learning algorithms, achieved through dataset partitioning. Such a process led to explaining and analysis of conducting experiments and comparing their results. Such experiments demonstrate the algorithm's ability to detect Sybil nodes in OSNs (100% accuracy in SVM, 99.6% in Random Forest, and 97% in KNN algorithms); thus concluding by proposing future research opportunities.

Keywords: machine learning; sybil attacks; online social networks; cybersecurity; random forest; support vector machine; KNN

1. Introduction

1.1. Background and Motivation

In this current era, Wi-Fi and various Peer-to-Peer (P2P) networks are integral to daily lives. Safeguarding such networks with their sensitive data is paramount from various cyberattacks. Sybil attacks emerge as a prominent concern (Saxena et al. [1]), targeting and exploiting vulnerabilities of distributed and P2P networks and can infiltrate compromising networks. Thus, it poses serious risks to applications ranging from military operations to social networks and everyday activities like connected commuting [2–5]. The consequences of a network that becomes a victim of a Sybil attack go beyond immediate compromise: empowering the attackers to manipulate the network through fake identities, thus identifying newer vulnerabilities [1]. This study's comprehensive literature review addresses such growing challenges posed by Sybil attacks and prevention strategies, with its research aim to contribute a proposed novel machine learning techniques' driven Sybil detection algorithm [6].

1.2. Problem Statement and Research Question

Previous research ([7–9]) leveraged Machine Learning (ML) algorithms, Extreme Learning Machine (ELM) and Support Vector Machine (SVM) to detect Sybil attacks. In Online Social Networks (OSNs), ML-based detection methods are promising but remain under-researched. Hence, this study builds upon the works of Du, et al. [10] and other scholars, e.g., [11–14] to cite a few, to delve into Sybil attacks within OSNs. Sybil attacks manifest across various network types, e.g., Wireless Sensor Networks (WSNs), Mobile Ad-hoc Networks (MANETs), and Online Social Networks (OSNs) [12]. WSNs consist of interconnected nodes equipped with sensors for data processing and analysis. MANETs, featuring mobile nodes without a fixed infrastructure, find applications in vehicular networks. OSNs, encompassing platforms such as Facebook and Twitter, offer attackers opportunities to introduce malicious nodes and undermine the network's integrity [15]. Literature reports on the different patterns exhibited by Sybil attacks, such as direct and indirect attacks, stolen identities, or fabricated attacks [3,16]. In direct attacks, malicious nodes mimic legitimate ones, directly communicating with authentic nodes. Indirect attacks involve manipulated communications between legitimate nodes, confusing while altering information flow. Stolen identity or fabricated attacks involve malicious nodes claiming legitimate credentials and sometimes replacing authentic nodes undetected [17–19]. Amidst the escalating Sybil attacks within OSN, scholars urgently demand user identity protection from rampant fake identities (bots) [17,18]. Notably, networks can be exploited as propaganda tools if Sybil attacks persist; a paramount motivation of this study is to propose an innovative algorithm that this study names as "SybilSocNet," designed to identify and counteract Sybil nodes within OSNs to address this challenge. The proposed algorithm harnesses ML methodology, such as K-Nearest Neighbors (KNN), Support Vector Machine (SVM), and Random Forest, to optimize detection accuracy. The following research question initiates such a research aim: How can the SybilSocNet algorithm effectively identify and counteract Sybil entities and the Sybil attack within OSN using a combination of ML methodologies, including KNN, SVM, and Random Forest.

1.3. Paper's Significance and Contribution

As technology continues to integrate into every aspect of peoples' lives, it is crucial to safeguard the integrity of networks. This study underlines the significance of countering these threats by thoroughly examining Sybil's attacks and their implications. This study's proposed robust ML techniques-driven algorithm contributes to global, ongoing efforts to fortify network security confronted by the ever-evolving cyber challenges. As for the remainder of this article, Section 2 critiques the literature on cybersecurity, network security, Sybil Attacks, ML, social network security, and algorithmic and ML-based detection. Section 3 explains the research methodology applied in this study, followed by Section 4, presenting the analysis of collected secondary open-source data. Section 5 discusses the findings, as well as reports on implications and limitations, and concludes with some open questions for future research.

2. Theoretical Foundation And Sybil Attack Landscape

2.1. Sybil Attacks: Concepts, Core Characteristics, Varieties, and Impact on Network Integrity

Scholars are concerned about how cybercrime is escalating rapidly, with unprecedented incidents. Considering that modern technologies facilitate swift data transfer through various communication protocols, with the key communication player being wireless technologies, e.g., 5G and WLAN, security must be assured for such wireless technologies [20]. By compromising robust security measures, numerous essential applications like mobile phones, satellites, access points, and wireless internet may face vulnerabilities, such as data breaches via such devices [21]. Researchers are keen to counter the security attached to such technologies. Such wireless technology, due to its extensive usage across a wide spectrum of applications, is vulnerable, according to the Pew Research Center (2023), which highlights that about 97% of Americans own mobile phones, 85% smartphones, with the reported American population roughly 307 million [22]. According to Vincent [23], the count of mobile phones in 2010 was around 4.5 billion, a sizable target pool for attackers. As per NBC News [22], three-quarters of Americans possess laptops or desktops, thus sizable vulnerable targets. Thus, considering their data sensitivity, it calls for a paramount initiative to safeguard such a large pool of targets. Particularly when such targets use wireless sensors in their modern applications, causing security concerns to escalate further. One drawback of wireless communication systems is unrestricted data and media sharing over networks, susceptible to signal interception, allowing potential attackers to spoof crucial information for launching attacks. When wireless networks get exploited, exposing information necessary for successful attacks, i.e., an identity-based attack. Literature sheds light on two simple yet destructive identity-based attacks: spoofing and Sybil attacks. Spoofing attacks are where attackers monitor targeted networks, gather legitimate user information, e.g., IP or MAC addresses, and pose as legitimate users to gain access to unauthorized services and data. A Sybil attack is a malicious node network infiltration to form multiple fake nodes that claim legitimately to deceive neighboring nodes by forwarding inappropriate information. By spoofing and legitimate information, attackers gradually introduce newer fake nodes, with time assuming control, thus enabling unauthorized data access and launching various attacks, e.g., denial of service attacks and data manipulation [24]. A Sybil attack targets P2P networks to create and control multiple fake identities, allowing an attacker to launch deceptive activities. Through an attacker's multiple created and controlled identities, they can concurrently deploy multiple nodes, misleading the network into granting unauthorized data access; like a reviewer fabricating numerous product reviews on Amazon to fake positive customer feedback or creating multiple accounts on Twitter to influence users. Also, sensor networks suffer from Sybil attacks where attackers pose as legitimate nodes, cascading to deceptively obtaining precise user information through social engineering or network spoofing. Such private information fuels Sybil's attacks, eventually granting attackers undue network influence [25].

According to Douceur [26] Sybil nodes illicitly claim multiple identities or fake IDs. Figure 1 depicts a Sybil attack, where a yellow node denotes legit ones and red nodes represent Sybil nodes. Sybil misleads the victim network during an attack when posing as legit, consequently redirecting traffic. For instance, node (A) mimics (H) to node (L) and (J) to node (K) causing traffic heading to node (A), which either forwards, alters, or blocks messages. Such a tactic extends to other red nodes. Attackers exploit victim nodes to wreak havoc, breach data, or lurk and launch attacks, yielding diverse outcomes. Such actions lead to malfunctioning networks with halted functions and data breaches entailing spoofed traffic between legitimate nodes. Also, it leaves manipulated networks that now meet attackers' wishes, e.g., altered weather predictions that declare specific regions disaster zones [27].

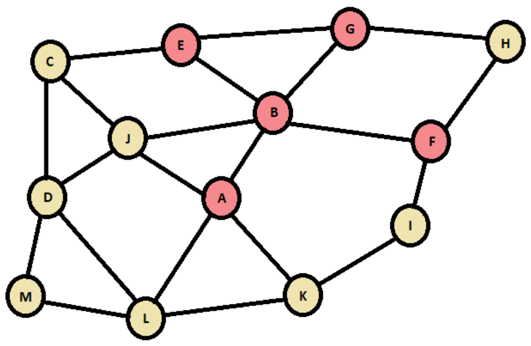


Figure 1. Sybil Attack on a P2P network.

2.2. Targeted Networks and Vulnerabilities

Sybil attacks target diverse networks of four types: (1) OSN platforms such as Twitter connect millions worldwide. An attack, as publicized by NBC [22], can impact masses and even nations. (2) Online store platforms such as Amazon can end up hosting fake product reviews, inflating a product's image to drive sales; tactics like such often involve payment for generating fake reviews. (3) Vehicular Ad-Hoc Networks (VANETs) are a subset of Mobile Ad-Hoc Networks (MANETs), where VANETs enable vehicular communication protocols like Vehicle-to-Vehicle (V2V), Vehicle-to-Roadside

(V2R), and Vehicle-to-Infrastructure (V2I) connections. VANETs Sybil attacks vary, including those that impersonate by fabricating MAC addresses or certificates when attackers infiltrate by breaching sensitive data or disrupting vehicle traffic. Moreover, (4) Wireless Sensor Networks (WSNs) include weather-related networks for environment tracking, disaster management, and industrial automation. Sybil attacks are apparent through nodes simulating sensors to transmit false data, overloading networks with malware, or deploying data spoofing or resource exhaustion tactics.

2.3. Unleashing Defense Strategies: Safeguarding Against Sybil Attacks

Various techniques protect against Sybil attacks by mitigating their likelihood. However, no efficient solution can eradicate such a threat [11,28–30]. Rahbari et al. [4] suggested a cryptography-based approach with overheads; while Almesaeed et al. [13] proposed controllable overheads. From the view of the authors of this study, an optimal approach should be able to balance security and performance, as also elaborated by Rahbari et al. [4]. It is essential to assess widely recognized techniques for mitigating Sybil attacks, categorized by protection, detection, and mitigation. Such frequently cited strategies/techniques/approaches can get categorized into six types: (1) Certifications that are recognized mitigation approaches involving certifying nodes through authorized identification [18]. Douceur and John exemplify VeriSign as a certification system that foils Sybil attacks, though challenges emerge without central certification. (2) is resource testing in known resource networks like sensor arrays [31], where nodes' physical resources validate legitimacy through commonly known resource allocation statistics [3]. Consider a network of temperature sensors where all sensors possess identical computational power, memory, and battery capacity. In this mitigation approach, nodes' physical resources get examined to ascertain their alignment with the attributes of legitimate nodes. While nodes execute tasks, reputation systems assess if the node is legit or Sybil. Lee et al. [32] used the resource testing approach to detect Sybil nodes in wireless networks. Their system would determine whether a node is Sybil based on how each node responds to a request. Newsome et al. (2004) showed how attackers use a single physical device to introduce multiple nodes into the network. Additionally, (3) incentive-based detection, where Margolin et al. [9] propose an economic detection protocol for Sybil attacks, financially incentivizing an attacker for two or more controlled identities. (4) Location/position and timing Verification is based on detecting a Sybil attack based on its location. To demonstrate legitimacy, every node must be in a specific location. It detects the Sybil attackers by discovering if multiple nodes are in an exact location or moving in a coordinated or synchronous pattern [14]. (5) Random key distribution where before nodes get deployed, each node gets a unique identifying key (Cuocolo, et al. 2019), a much more suitable technique for WSNs as nodes typically possess limited computational and energy resources. Eschenauer et al. [33] and Dhamodharan and Vayanaperumal [34] proposed a key-management scheme for distributed sensor networks. (6) Privilege Attenuation, where Fong [8] counters Sybil attacks in Facebook-style Social Network Systems (FSNSs) with privilege attenuation, an approach aligned with Denning's Principle of Privilege Attenuation, to limit and prevent Sybil attacks such as access restrictions among friends to reduce Sybil attack risks in FSNSs. Sybil detection algorithms get deployed to detect if a Sybil attack is attacking. They determine nodes' legitimacy by differencing malicious from legitimate nodes. Several scholars [2,28,29,35] explored techniques for preventing and mitigating Sybil attacks. Quevedo et al. [36] suggest a SyDVELM mechanism employing the Extreme Learning Machine (ELM) algorithm to detect Sybil attacks in VANETs. Yu et al. [37] proposed a SybilLimit protocol to counter Sybil attacks in OSNs. Patel et al. [12] examined Sybil detection algorithms for wireless sensor networks where nodes function as sensors. Often, WSNs get deployed for monitoring purposes, e.g., military applications, and their unattended nature renders them susceptible to Sybil attacks and cyber threats, such as a network of sensors measuring air speed and temperature to predict future weather. In this scenario, the sensor network's communication gets compromised when a new sensor gets introduced under the guise of legitimacy but is, in fact, malicious. Such deceptive nodes can discriminate against fake results, distort weather forecasts, and propagate misleading data to other nodes. Furthermore, malicious nodes might persistently introduce seemingly legitimate nodes by mimicking the behavior of genuine ones. In both scenarios, successful infiltration by a node can profoundly disrupt the network's functionality. In critical infrastructures, e.g., traffic systems, such a disruption can result in city-wide chaos, causing severe accidents. Similarly, a swarm of compromised military drones could result in significant casualties. To safeguard against such threats, constant network monitoring for Sybil attacks is crucial. Hence, the study aims to comprehend Sybil attack detection techniques within P2P networks.

2.4. Sybil Detection Algorithms

In WSNs, Dhamodharan et al. [34] formulated a Sybil detection algorithm: message authentication with a Random Password Generation (RPG) algorithm to identify and neutralize Sybil attacks. RPG generates routing tables for network nodes, comparing node details during message transmission to identify unauthorized nodes. Compare and Match-Position Verification Method (CAM-PVM) detects Sybil attacks. Data is rerouted through verified nodes, omitting known Sybil nodes. However, due to high computational demands, Dhamodharan et al. [34] advocate prevention prioritization over detection. CAM-PVM gets selectively employed when data-sending nodes suspect a recipient, ensuring efficiency.

Regarding Sybil attacks detection in Mobile Ad Hoc Networks (MANETs), Abbas et al. [38] introduced a detection scheme for versatile MANETs, widely applied across various fields. MANETs self-organize into versatile typologies driven by application needs, with wireless nodes that connect/disconnect dynamically based on bandwidth. However, such wireless connectivity reduces capacities compared to wired connections. Security is challenging due to autonomy and the absence of centralized authority, rendering MANETs vulnerable to diverse cyber threats. All nodes in MANETs function as hosts/routers, enhancing their configurations while compromising their security. While wireless connectivity boosts data transfer rates, latency gets introduced due to constrained resources like battery-powered small units. Despite these complexities, the unique properties of MANETs offer diverse applications.

Newsome et al. [39] examined security threats in sensor networks. Due to the cost-efficiency requirement for mass production, sensor nodes often lack advanced hardware, computational power, and storage space, a limitation that hinders the use of robust cryptography algorithms, leaving networks vulnerable to threats like Sybil attacks. For instance, in a weather prediction ad-hoc network, a Sybil node could manipulate forecasts to predict extreme conditions, or in a military application's MANET, a Sybil node breaching the network might endanger lives.

Abbas et al. [38] proposed that the received Signal Strength (RSS)-based localization algorithm is a solution for Sybil attacks. An algorithm leveraged by Chen et al. [40] is for the identification of Sybil attacks within sensor networks. Chen et al., recognized its key role as a promising strategy for detecting Sybil attacks in MANETs. They benchmarked it as a measure of the signal's strength received from a specific node at a given location. The RSS value varies depending on the signal's source node location. According to Abbas et al. [38], one way to distinguish between Sybil and legitimate nodes is by analyzing the nodes' entry and exit behaviors within the network. RSS is the measurement of signal strength received by a receptor, typically gauged at the receiver's antenna. Several factors influence RSS, e.g., transmitter-receiver distance, transmission medium, and receiver signal power. RSS applications are crucial in detecting and estimating the geographical location of a transmitter. Abbas et al. [38] examined the network nodes as they join and exit a network assessing proximity and signal strength, to distinguish legitimate from Sybil nodes. When a newly Sybil node enters a network, it is expected to exhibit a high RSS. Conversely, legitimate nodes tend to join a network upon receiving a signal which results in a lower RSS value during network entry. Such discrepancy arises from the fact that Sybil nodes get spawned by nodes already in the network.

In contrast, legit nodes, drawn by proximity, join the network promptly with weaker signals, learning to a lower RSS value. The detection mechanism involves storing communication histories between neighboring nodes in individual nodes such that each node maintains a table that encompasses the entire communication history. Such a table becomes the foundation for each node's assessment of whether its neighbor node is a Sybil. Such a table includes the signal strength measurements for surrounding nodes. This information enables the algorithm to classify the investigated node. Data accumulated in such tables occurs during the direct node-to-node communication or even when an indirect node is engaged. Abbas et al. concludes two experiments that discern distinctive behaviors between Sybil and legit nodes. In experiment 1, the docs are on distinguishing between the behaviors of Sybil and legit nodes, and gets achieved by monitoring a node's RSS over time. A legit node typically enters another node's range gradually, strengthening its RSS as it gets closer to intended connections. Conversely, as it moves farther away, the RSS weakens. Figure 2 depicts the RSS progression values over time for both Sybil and legit nodes. For legit nodes, naturally entering and exiting a network, the plot behavior generates a bell-shape RSS curve (Figure 2A) versus a Sybil node which exhibits a higher RSS compared to a legit node as depicted in Figure 2B, resulting in a constant line followed by a negatively sloping curve; signifying that the node's entry with a consistent high RSS values diminishing as the node departs the network.

As per the second experiment, the pace by which nodes enter the network significantly influence the establishment of the threshold RSS value. It becomes crucial to ascertain the extent to which a node can traverse a specific node's range before being detected. To investigate the relationship between node speed, penetration distance, and RSS value, Abbas et al. [38] experimented on node X and Y where Y's movement got initiated away from X at varying speeds, while recording RSS values. Their findings reveal that a higher rate of movement allows nodes to penetrate into the range of other nodes while undetected. Such insights led the system threshold to get established, assuming that no node could surpass a predetermined maximum speed.

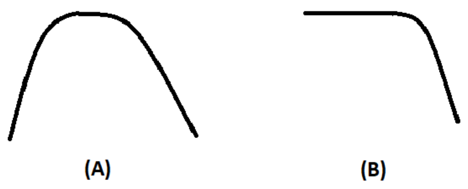
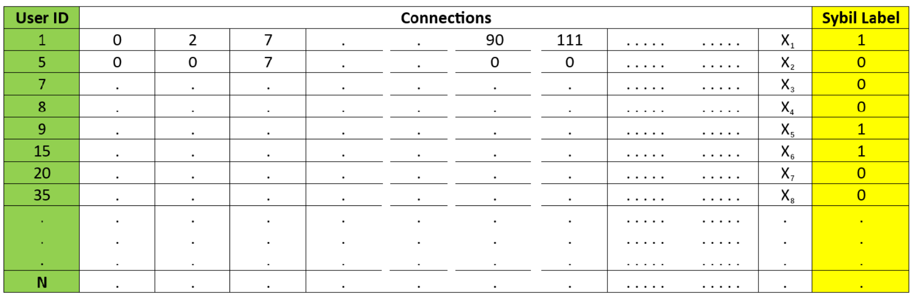


Figure 2. RSS vs Time for Sybil and Legit nodes

3. Methodology: Proposed SybilSocNet Algorithm

This study employed ML algorithms to develop predictive models that are capable of identifying Sybil nodes within social networks. Considering the widespread popularity of such platforms, e.g., Twitter and Facebook, they are the basis for millions to convene globally. Hence, Sybil attacks exhibit within such networks by penetrating fake user accounts, so individuals can falsely assume identities. Such fraudulent accounts get employed to influence the public or manipulate network dynamics, involving propagating specific ideologies through orchestrated campaigns [41]. The modus operandi involves a Sybil attacker who controls multiple Sybil nodes, while orchestrating fraudulent reports on the social network admin, where such reports falsely implicate legitimate accounts as fraudulent or compromised. Subsequently, the admin proceeds with automated actions that restrict the legit users' network access or even ban them. Cases of cybercrimes were observed where social media got exploited to sway public opinion during elections, convincing votes for specific political parties. For instance, a hacker in Russia commissioned approximately 25,000 fake accounts, generating about 440,000 tweets that influenced public sentiment after a country's parliamentary election [42]. To construct a supervised ML model skilled at discerning Sybil nodes in social networks and to distinguish Sybil nodes from legit ones, it is vital to acquire a dataset comprising labeled tweets or posts categorized as Sybil or legit. Such a dataset becomes the foundation for employing diverse unsupervised ML algorithms to train a model to differentiate malicious users, posts, or tweets from authentic ones. The next step is data cleaning, to explain the study's ML algorithm by utilizing an adopted dataset. A dataset that got reformatted via Python code to form a matrix suitable for input for this study's ML algorithms. Multiple ML algorithms got trained and their outputs got tested on the validation data, to determine the most effective model for this study's problem. Figure 7 offers an overview of the deployed dataset. Subsequently, this study constructs a matrix (Figure 3) to train the SVM, KNN, and Random Forest ML algorithms: where the first column, green, signifies user ID; the remaining columns, white, denote the other users following those from the first column. The last column, yellow, labels users (from the first column) as '1' if it is detected as Sybil, but marked as '0' if it is considered a legit user. Such a matrix encompasses all users and their connections. To further clarify, the first row depicts data for user 1, followed by 2, 7, 90, and 111, labeled as Sybil users. The second row corresponds to user 5, followed by user 7 which corresponds to legit users. The matrix continues encompassing all users in the dataset [43] along with their connections. In the second last column (Table's right side - Figure 3), the "X" signifies either the node with the highest index or zero if not being followed, denoted by X_i, where $X_i \in \{0, M\}$, and where M is the value of the highest index value among the sorted nodes before being included in the matrix.



User ID	Connections										Sybil Label
1	0	2	7	.	.	90	111	.....	.....	X_1	1
5	0	0	7	.	.	0	0	.....	.....	X_2	0
7	.	.	.	.	.	.	.	.....	.....	X_3	0
8	.	.	.	.	.	.	.	.....	.....	X_4	0
9	.	.	.	.	.	.	.	.....	.....	X_5	1
15	.	.	.	.	.	.	.	.....	.....	X_6	1
20	.	.	.	.	.	.	.	.....	.....	X_7	0
35	.	.	.	.	.	.	.	.....	.....	X_8	0
.	.	.	.	.	.	.	.	.....	.....	.	.
.	.	.	.	.	.	.	.	.....	.....	.	.
.	.	.	.	.	.	.	.	.....	.....	.	.
N	.	.	.	.	.	.	.	.....	.....	.	.

Figure 3. Generated big matrix

Next, our algorithm pseudocode gets portrayed. This study's algorithm utilizes a dataset generating 71 smaller data blocks. The algorithm reads the dataset's first line, subsequently skipping 70 nodes, capturing each new node after jumping and adding it to a small matrix. Such a process repeats itself until reaching the end, thus creating the first small matrix. Such iteration gets repeated for the second line, resulting in a second small matrix, thus continuing until 71 small matrices get generated. The Algorithm 1 depicted below outlines the pseudocode for this initial algorithm part. Once the 71 data blocks get formed, a matrix gets created for each, similar to the one depicted in Figure 3. Algorithm 1 provides pseudocode for generating these matrices.

```
Input  : TwitterDataset
Output : BlocksOfData
$BlocksOfData \leftarrow$ Empty array to store blocks of data;
$nodeIndex \leftarrow 0$;
for $i \leftarrow 1$ to 71 do
    while Not at end of TwitterDataset do
        $currentNode \leftarrow$ read node at $nodeIndex$;
        $BlocksOfData \leftarrow$ append $currentNode$ to $BlocksOfData$;
        $nodeIndex \leftarrow nodeIndex + 70$;
        if $nodeIndex$ is at the end of $BlocksOfData$ then
            break;
        end
    end
    Save SmallBlockOfData to a file;
    $nodeIndex \leftarrow i + 1$;
end
return $BlocksOfData$;
```

Algorithm 1: Pseudocode For Splitting The Data.

3.1. Applied ML Terminologies And Methodologies By This Study

3.1.1. Machine Learning

ML enables learning like the human brain without explicit programming. For instance, training a computer to distinguish between cats and dogs using labeled pictures constitutes ML. Or, training computers to recognize individual voices. Many AI-driven companies use ML emulating human intelligence by learning from the environment [44] with widespread applications spanning sectors, from software to medical and military domains such as image detection, face recognition, and voice assistants like Siri [45] or Alexa [46]. Home security cameras employ ML for movement and face detection. Radiology employs it for diagnosing tumors. This study investigates various ML methods, supervised and unsupervised learning, to detail algorithmic approach and findings.

3.1.2. Supervised Learning

ML algorithms get classified on training methods, with one type being supervised learning. Where algorithms learn by mapping inputs to outputs via a labeled dataset containing input-output pairs [47], i.e., the algorithm gets trained on instances with known labels. Supervised learning effectiveness arises from having correct target outputs while the algorithm continually refines its output by calculating a value from a loss function, a value that guides adjustments to the model's weights for achieving the desired output. For instance, when predicting car prices via features' data, e.g., previous prices, model number, brand, mileage, and accident history, an algorithm predicts the car value via inputs like model, make, and mileage. Supervised learning, with broad applications like image recognition, Natural Language Processing (NLP), and speech recognition, gets employed in this study in supervised ML to distinguish Sybil from legit social media Twitter accounts via two key types: classification and regression. A classification problem concerns categorizing data into specified classes, e.g., classifying animal images into cats or dogs, or segregating individuals by age, either 20-30 or greater than 30. Such exemplary classification ML algorithm tasks determine the group where an instance belongs. Common classification algorithms are decision trees, support vector machines, and random forests and get trained on labeled data to classify new data but the prerequisite is to first choose the right algorithm relying on factors like dataset size, training methods, and desired accuracy. Sahami et al. [48] tackles classification problems via a naive Bayes classifier to distinguish spam from legit emails through automated filters for classifying inbox emails, to enhance user experience by excluding unwanted content. Unlike classification, where ML algorithm predicts a predetermined class, regression predicts a value, e.g., car selling price or a woman's due date. Unlike supervised learning, unsupervised learning omits labeled data for training, instead analyzing unlabeled data, revealing patterns and relationships, applicable in clustering and anomaly detection [49]. The K-nearest neighbors (KNN) algorithm, a flexible supervised method, learns from labeled data to predict labels for unlabeled instances. It memorizes patterns during training, then utilizes them for predicting new instance labels, serving both classification and regression tasks based on its training.To label a new instance, KNN compares memorized instances and assigns labels using similarities [50]. Next, this study illustrates real-world KNN functions for predicting housing price trends for investors' decision-making. Initially, when employing the KNN algorithm as a classifier, classes get distinguished and the model's accuracy gets assessed via labeled instances by splitting the dataset for training and testing using various techniques, e.g., cross-validation, including re-substitution Validation, K-fold cross-validation, and repeated K-fold cross-validation, each with its strengths and applications. Next, KNN algorithm determines the number of neighbors (K) for predicting class, influencing accuracy. Optimal K selection balances pattern detection and sensitivity. Next, compute distances between the instance and its neighbors. The closest K neighbors' classes decide the predicted class using distance metrics (e.g., Euclidean, Manhattan, Minkowski). Refer to formula (1) for these methods [51]. Another distance measuring method is cosine similarity, depicted in formula (2), commonly used in text retrieval, as discussed by Lu [11]. Additionally, the KNN equation employs diverse distance formulas from Batchelor [14] to introduce the Minkowski distance formula (3), the Chi formula (Formula 4), and the correlation equation (Formula (5) by Michalski and Stepp [52], which constitute distinct distance metrics for KNN's neighbor distance calculation.

$$dist(A,B) = \sqrt{\frac{\sum_{i=1}^{m}(x_i - y_i)^2}{m}} \tag{1}$$

$$sim(A,B) = \frac{\overrightarrow{A} \cdot \overrightarrow{B}}{\left|\overrightarrow{A}\right|\left|\overrightarrow{B}\right|} \tag{2}$$

$$dist\_Minkowsky(A,B) = \left(\sum_{i=1}^{m} |x_i - y_i|^r\right)^{1/r} \tag{3}$$

$$dist\_Chi\text{-}square(A,B) = \sum_{i=1}^{m} \frac{1}{sum_i}\left(\frac{x_i}{size_Q} - \frac{y_i}{size_I}\right)^2 \tag{4}$$

$$dist\_correlation(A,B) = \frac{\sum_{i=1}^{m}(x_i - \mu_i)(y_i - \mu_i)}{\sqrt{\sum_{i=1}^{m}(x_i - \mu_i)^2 \sum_{i=1}^{m}(y_i - \mu_i)^2}} \tag{5}$$

Each equation is used to calculate distances between the instance under study and other instances in the dataset, leading to varying outputs and accuracy measurements. These aspects are explored in ongoing research projects. Medjahed et al. [53] investigate breast cancer diagnoses using KNN algorithms and their dependency on diverse distance metrics. Iswanto et al. [28] conduct comparable studies on the impact of these distance equations in stroke disease detection. After calculating distances between the studied instance and all other dataset instances, the algorithm picks K nearest neighbors, i.e., K determining the count of neighbors. For instance, with K set to 100, it chooses the closest 100 neighbors with such neighbors bearing the smallest distances to the instance. However, this step can get computationally intensive, especially for large datasets, requiring distance calculations and comparison operations. Once the K nearest neighbors gets identified, the algorithm predicts the instance's class based on their classes, a step called "Voting," offering Weighted and Majority options. Weighted Voting assigns varying weights to classes, while Majority Voting treats all classes equally, so adaptability accommodates developers aiming to emphasize specific class attributes.

3.1.3. Support Vector Machine Algorithm (SVM)

SVM, a supervised learning algorithm, operates on labeled datasets, aiming to learn a hyperplane that separates data points into distinct classes [54]. Its training optimizes the hyperplane's position for effective classification of new points [55]. When assessing a new point's position relative to the hyperplane, SVM assigns it to an appropriate class. In Figure 4, the hyperplane visually separates red dots from green stars, with dashed lines indicating the distance between the hyperplane and the nearest support vectors [23,56]. Such support vectors' positioning during training determines the hyperplane's position and margin size [37]. In this testing phase, the hyperplane functions as the class boundary [57], classifying new instances. SVM's hyperplane is crucial for classification and regression tasks, predicting class or value for new instances, relying upon the new instance's position with respect to the hyperplane [58,59].

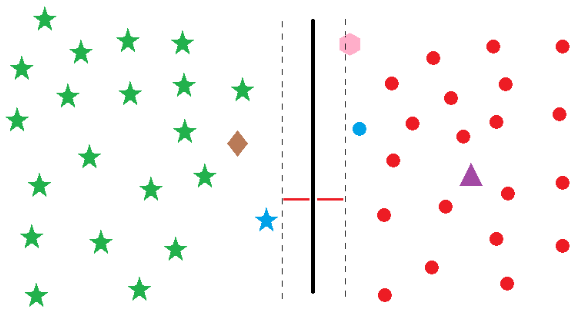


Figure 4. SVM hyperplane.

Based on the hyperplane distance, the instance receives a specific class or value based on problem type. For instance, in a classification problem, the trained SVM algorithm predicts the class of a new instance. After training and hyperplane positioning, the algorithm assigns an instance to a class. If the brown rhombus falls to the hyperplane's left side, associated with green stars, the algorithm predicts it as a duck (assuming green stars are ducks and red dots are lions). Similarly, the purple triangle and pink hexagon on the hyperplane's right side belong to the lion class. Consequently, the algorithm predicts them as lions. Even the pink hexagon, on the right margin, gets classified as a lion as it is on the right-side, regardless of being on the margin. Margins are only deployed during training to optimize hyperplane positioning and are unutilized during testing or applications. SVM optimizes the hyperplane in training, via margins to solve the optimization problem, hence, margins play no role in classifying new instances during testing. Hyperplanes take diverse shapes based on dataset and feature count. With one feature, they are points, e.g., grading students as pass/fail. Two features yield lines or shapes, three form planes, and M features create an M-dimensional hyperplane. Formulas (1)–(5) and Figure 4 depict such variations, showcasing blue and red circles representing classes; with the hyperplane depicted as a brown line or circle, refer to Figure 5, which also depicts a three-dimensional hyperplane, distinct from the line and circle, where a hyperplane correlates with a dataset bearing three features, X, Y, and Z axes when working with N features to allow the hyperplane to gain N dimensions; a concept easier to grasp when observing the one to two, and then finally to the three dimensions.

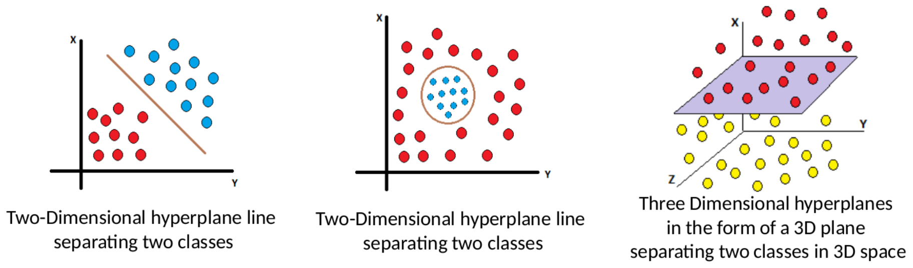

X
Y
Two-Dimensional hyperplane line separating two classes
X
Y
Two-Dimensional hyperplane line separating two classes
X
Y
Z
Three Dimensional hyperplanes in the form of a 3D plane separating two classes in 3D space

Figure 5. Two and three dimensional hyperplane line

Implementing a SVM involves five steps:

1. **Data Prep:** As SVM is supervised, labeled data gets deployed for training. Instances must be associated with specific classes where data collection takes various routes like downloading datasets from platforms like Kaggle [60] or customizing existing ones.
2. **Feature Scaling:** SVM gives weight to features based on values where scaling ensures fairness among features. While some features hold more magnitude, maintaining equitable treatment prevents bias. Scaling methods like standardization and normalization are often employed.
3. **Kernel Selection:** SVM utilizes the kernel trick, mapping data to higher dimensions for easier classification. Kernels are of various types, suited to different data and applications. Optimal kernel choice impacts computational efficiency.
4. **Training and Optimization:** occurs where after the kernel gets selected, training involves determining the hyperplane position. This necessitates solving an optimization problem, minimizing the cost function while considering margin and regularization factors. Different algorithms, e.g., SMO and Quadratic Programming can get deployed.
5. **Testing:** SVM can handle classification and regression. In classification, algorithm predictions are compared to known instances. For regression, SVM predicts values based on the point's distance from the hyperplane.

3.1.4. Random Forest Algorithm

Random forest, an ensemble supervised method, combines small models to form a larger model for problem-solving. Such an algorithm operates in stages: small models tackle subproblems, their solutions contribute to the final prediction through a voting system. The large model is a collection of trees, with each tree handling a subset of features with the five steps characterizing the random forest algorithm:

1. **Dataset preparation:** like prior algorithms requires labeled data for training with dataset consists of features linked to corresponding classes, e.g., car quality correlates with attributes like make, model, and price.
2. **Decision tree:** creation and grouping of it via a central tree consisting of smaller decision trees that predict outcomes via dataset's distinct portions without all features getting related to target outputs to form a diverse set of trees, akin to people uniquely solving problems. The ensemble voting aggregates their solutions to determine the final prediction.
3. **Trees' feature selection:** Different randomly chosen features get assigned to each tree, to curb overfitting.
4. **Bootstrapping:** this involves newly created dataset's data samples, generating variations for each tree to avoid uniformity. Bootstrapping introduces randomness, yielding a diverse array of perspectives on the problem, crucial to preventing overfitting (Figure 6).
5. **Output Prediction:** where the algorithm predicts output, either class for classification or value for regression. All decision trees contribute predictions, and a voting mechanism selects the majority class for classification or the average output for regression.

The random forest algorithm offers adjustable parameters, including tree count, bootstrap dataset percentages, and decision tree node count, catering to the application's complexity [61].

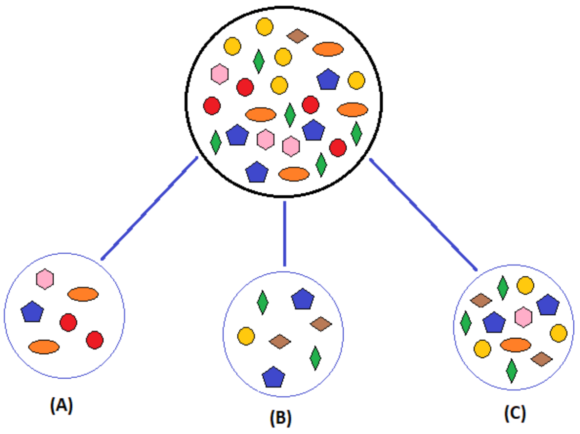
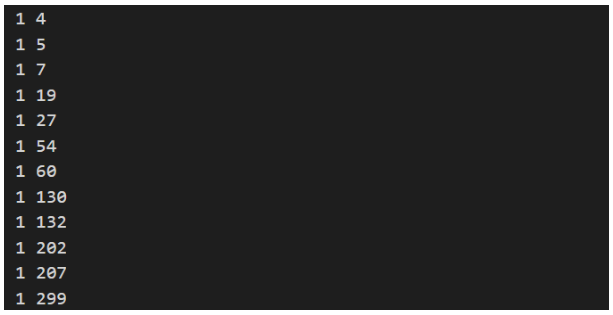


Figure 6. Bootstrapping generates sample subsets from the main dataset.

4. Data Analysis

4.1. Steps, Results, and Validation

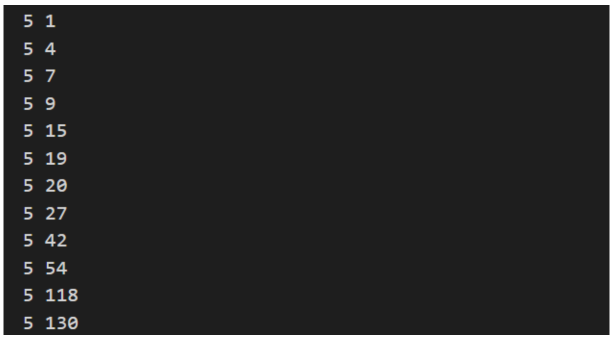
This section reviews the steps and results of this empirical study's experiments using the dataset by Jethava and Rao [62]; previously utilized by Patel and Mistry [12]. First data got cleansed and reformatted so it can be used to train different ML algorithms to detect Sybil nodes. Twitter users' dataset consists of followers and followees, i.e., indicates what accounts each account follows. Figure 7 signifies part of the Jethava and Rao [62] dataset, indicating that user ID 1 is following 4, 5, 7, etc., up to the end of the dataset.



```
1 4
1 5
1 7
1 19
1 27
1 54
1 60
1 130
1 132
1 202
1 207
1 299
```

Figure 7. Portion of adopted dataset

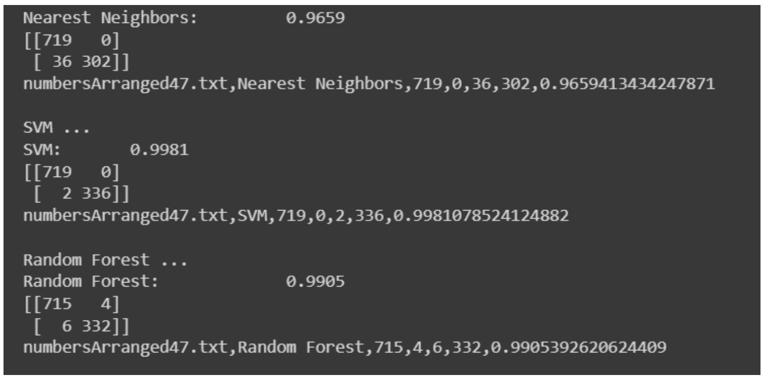
Figure 8 depicts user 5's connections 1, 4, 7, etc. A separate dataset segment consists of a two-row file containing Sybil and legitimate node IDs. This study has carefully processed and restructured the dataset for input into its ML algorithms accompanied by the matrix generation process using pseudocode, portrayed earlier in this article; Such entails crafting a comprehensive matrix where rows represented user IDs, followed by followers' IDs in subsequent columns, and the last column indicated user legitimacy (0 for non-Sybil, 1 for Sybil), derived from '269640_test.txt' dataset row 2 (Jethava and Rao [62]).



```
5 1
5 4
5 7
5 9
5 15
5 19
5 20
5 27
5 42
5 54
5 118
5 130
```

Figure 8. User ID number 5 and the user ID numbers that it follows

This study generates a single matrix for training ML algorithms, followed by experiments to optimize accuracy. Due to the dataset's extensive size, there arose the need for significant computational resources, e.g., supercomputers, but lacking access to one, circumstances got adapted to by splitting data into seventy one samples to generate corresponding smaller matrices for training and testing. Such an approach eased computational demands, crucial due to the initial matrix's massive 117 GB size. Each of the seventy one matrices got approximately 299.62 MB, totaling to 20.77 GB. Despite such a size reduction, the original data remains intact, and as a result organized into independent matrices, thus enhancing manageability and computational feasibility. Next comes the ML algorithm phase in this study, where ML algorithms get deployed to employ KNN, SVM, and Random Forest algorithms and the respective training to differentiate Sybil nodes from legit nodes on the previously mentioned seventy one matrices. In the next section this study reports the distinct accuracy outcomes of each algorithm. While training, a challenge got encountered when training the SVM on four matrices when extensive training times got elapsed, though KNN and Random Forest trained successfully. Hence, this study opted to analyze algorithm results on sixty seven matrices omitting the four problematic ones. All ML algorithms utilize the aforementioned seventy one matrices as input for training. Segmented data blocks serve for training and testing, yielding high accuracy. The algorithms produce a uniform output format, enabling direct comparison with their outputs represented by a 2×2 matrix (Figure 9).



```
Nearest Neighbors:         0.9659
[[719   0]
 [ 36 302]]
numbersArranged47.txt,Nearest Neighbors,719,0,36,302,0.9659413434247871

SVM ...
SVM:        0.9981
[[719   0]
 [  2 336]]
numbersArranged47.txt,SVM,719,0,2,336,0.9981078524124882

Random Forest ...
Random Forest:             0.9905
[[715   4]
 [  6 332]]
numbersArranged47.txt,Random Forest,715,4,6,332,0.9905392620624409
```

Figure 9. Output from three ML algorithms on matrix number 47

As per Figure 9, a common 2×2 output matrix is observed from the ML models, and can be interpreted as follows:

1. **Element I_{00} :** indicates true positives for non-Sybil instances, a value representing legit nodes correctly detected by the algorithm as such.
2. **Element I_{01} :** represents false positives for Sybil instances, signifying nodes falsely labeled as Sybil by the algorithm. Ideally, this value is minimal or zero, preventing the mislabeling of legit nodes as Sybil.
3. **Element I_{10} :** reflects undetected Sybil nodes by the algorithm while the general aim was to keep this count minimum to ensure Sybil detection.

4. **Element I_{11} :** stands for Sybil instances' true positives, indicating correctly identified Sybil nodes.

E.g., consider the SVM algorithm's output matrix (Figure 9), where $I_{00} = 719$ represents the correctly detected legit nodes, $I_{01} = 0$ means no false positives for Sybil nodes, which is desirable, $I_{10} = 2$ indicates that two Sybil nodes went undetected, labeled as legit. And, $I_{11} = 336$ specify the correct and accurate detection of Sybil nodes. The algorithm's accuracy in this case is 99.81%, calculated by adding I_{00} and I_{11} and dividing it by the sum of all the elements in the matrix. Accuracy gets computed in every algorithm after processing each matrix. This study encountered issues obtaining SVM outputs for some of the matrices; this is not uncommon [2,7,28,53]. In our study we employed a linear kernel for the training phase, other kernel types might yield less or no issues for SVM computing.

4.2. Data Overview

This study compared the performance of various ML algorithms in terms of accuracy. Table 1 summarizes the results exposing the mean value and standard deviation for each algorithm we examined. Table 1 reveals that the SVM algorithm achieves the highest accuracy, trailed by the Random Forest algorithm. KNN ranks last in accuracy. In terms of standard deviation, the Random Forest Algorithm demonstrates the lowest value, with KNN following in second place, and the SVM algorithm exhibiting the highest standard deviation. The data presented is the output accuracy for the K-Nearest Neighbor, Random Forest, and SVM algorithms. The Tables for these algorithm's data are portrayed in Appendix A, Appendix B, and Appendix C respectively.

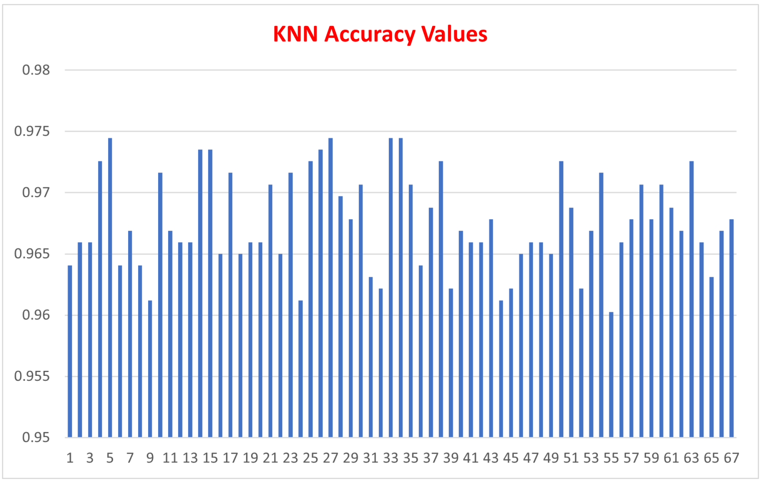


Figure 10. KNN Accuracy values

Figures 10 and 11 depict the KNN accuracy results for the different matrices. As depicted, the accuracy has a maximum of approximately 97% and a minimum of around 96%.

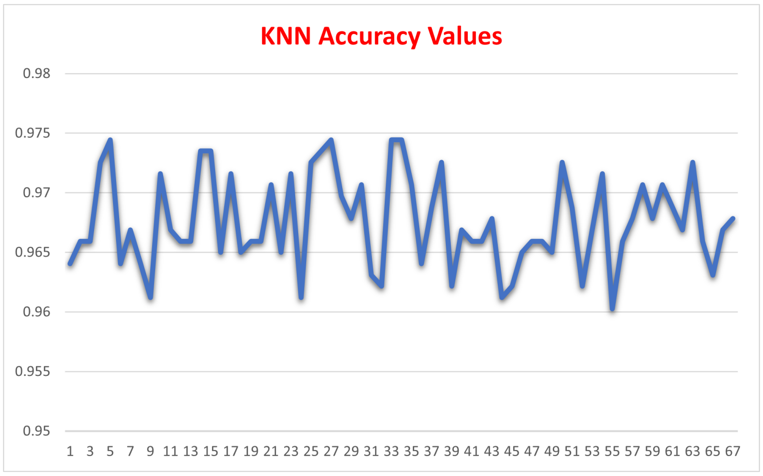


Figure 11. Line chat for KNN accuracy values with trends

Figures 12 and 13 depicts the accuracy for the Random Forest algorithm and the SVM algorithm, respectively.

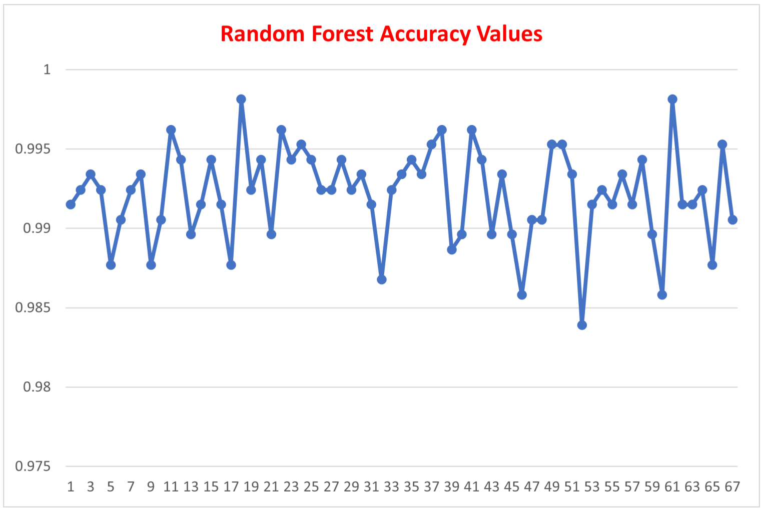


Figure 12. Random Forest accuracy values

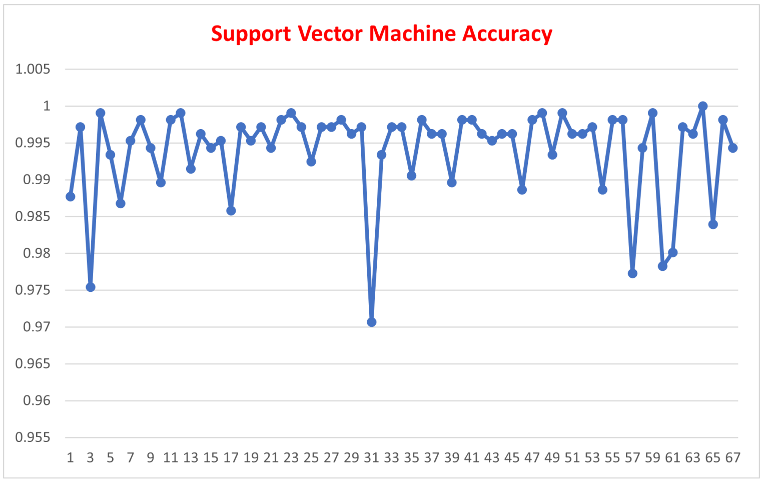


Figure 13. SVM accuracy values line chart

Table 1. Mean Values and Standard Deviation Comparison.

	K Nearest Neighbor	Support Vector Machine	Random Forest
Mean Value	96.759	99.394	99.213
Std. Dev.	0.0038	0.0061	0.0029

When comparing the performance of the different ML algorithms regarding accuracy, Table 1 depicts results revealing the mean value and standard deviation for each examined ML algorithm, highlighting that the SVM algorithm has the highest accuracy, followed by the Random Forest algorithm. KNN ranks last in terms of accuracy. Regarding standard deviation, the Random Forest Algorithm bears the lowest standard deviation; KNN ranks second, followed by the SVM algorithm, ranking the highest standard deviation. Figure 14 and 15 depict the different performance outcomes of all the examined algorithms.

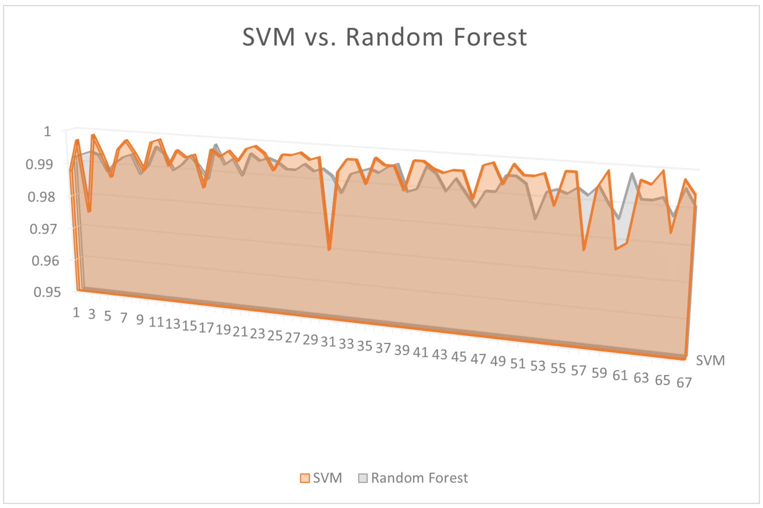


Figure 14. Accuracy values for SVM and Random Forest

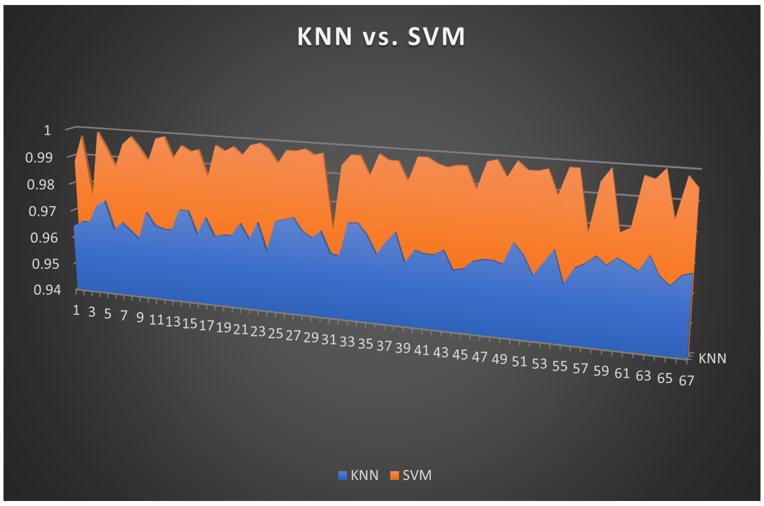


Figure 15. Accuracy values for KNN and Support Vector Machine

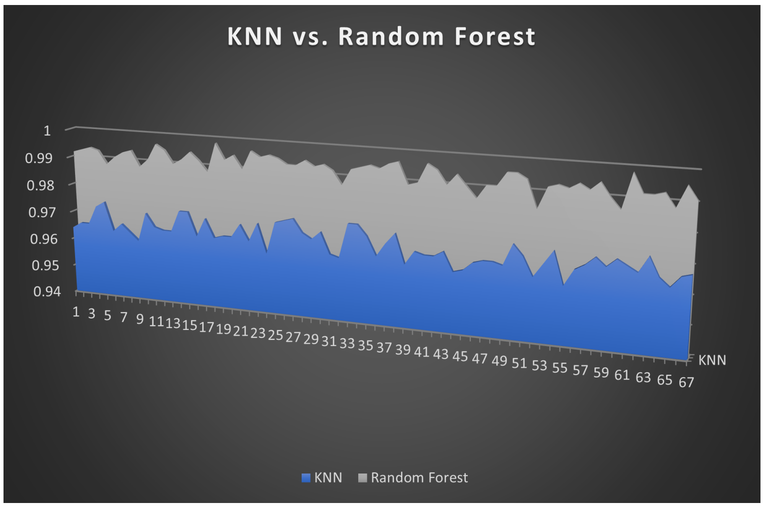


Figure 16. Accuracy values for KNN and Random Forest

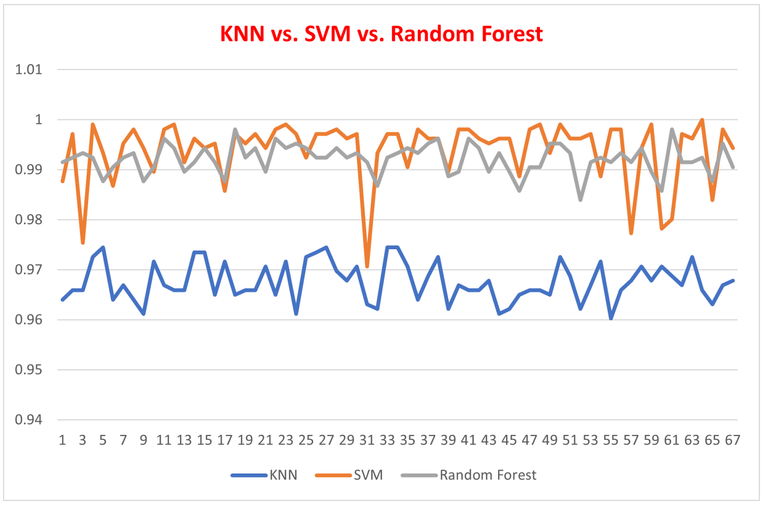


Figure 17. Accuracy values for KNN, SVM, and Random Forest

Furthermore, the KNN algorithm had high accuracy levels; however, these accuracy levels are relatively lower than those of the Support Vector Machine algorithm and the random forest algorithm. The constant number K that was chosen can be investigated alongside the dataset's size. As depicted in Figure 18, KNN has a normal distribution with accuracies ranging between 96% and 97.5%, with 96.6 having the highest frequency of occurring. While training the three ML algorithms, the SVM algorithm needed a relatively long time compared with the Random Forest and the KNN algorithm. It is important to notice that the SVM algorithm was taking too long to train on matrices [7,9,28,53]. we decided to skip these matrices. The average time the SVM took to train on other matrices was about 2 minutes. The normal distribution graph for the Support Vector Machine algorithm shown in Figure 19 indicates that the algorithm has a high probability of producing an accuracy value between 99.5% and 100% with a relatively low probability of having an accuracy between 97% and 99%.

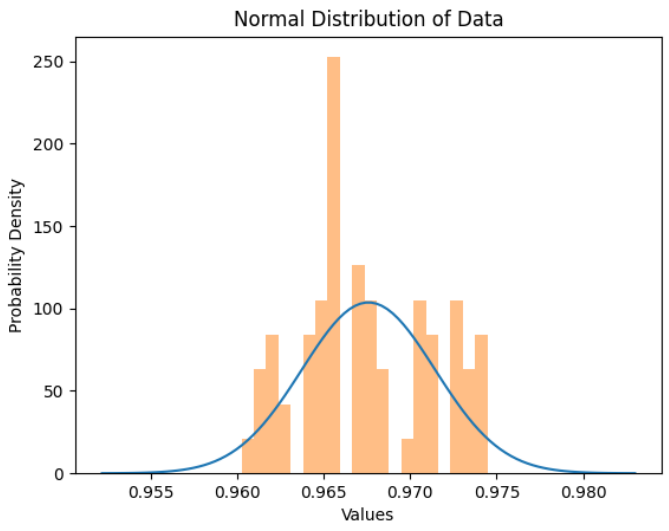


Figure 18. Normal distribution graph for KNN algorithm

During SCM training the three ML algorithms, the SVM algorithm needed a relatively long time compared with the Random Forest and the KNN algorithm. It is to be noted that the SVM algorithm also took too long to train on matrices for Medjahed, et al. [53]; Iswanto et al. [28]; Margolin and Levine [9]; and Misra et al. [7]; and others. After several days of not finishing with the computation, we decided to skip these matrices. The average time the SVM took to train on other matrices was about 2 minutes. The normal distribution graph for the Support Vector Machine algorithm shown in figure 19 indicates that the algorithm has a high probability of producing an accuracy value between 99.5% and 100% with a relatively low probability of having an accuracy between 97% and 99%. Random Forest Random forest had high accuracy rates with the lowest stan- dard deviation value compared to the other machine learning algorithms. Also, the time required to train the random forest algorithm was relatively short com- pared to the time required to train the Support Vector Machine algorithm. The random forest algorithm has a normal distribution with an accuracy span rang- ing between 98.4% and 99.6% values, with a high probability of accuracy between 99.0% and 99.5%. The related Results of Mounica et al. [42] achieved an accuracy of 93%; their approach was only for wireless networks. Nevertheless, our algorithm works for wired and wireless networks with higher accuracy. Lu et al. [11] achieved an AUC value of 0.93 in their proposed Sybil detection method. Patel and Mistry [12] have an AUC value of 0.82 for the large Twitter dataset. Our accuracy results range from 96% (KNN) to 99% (SVM).

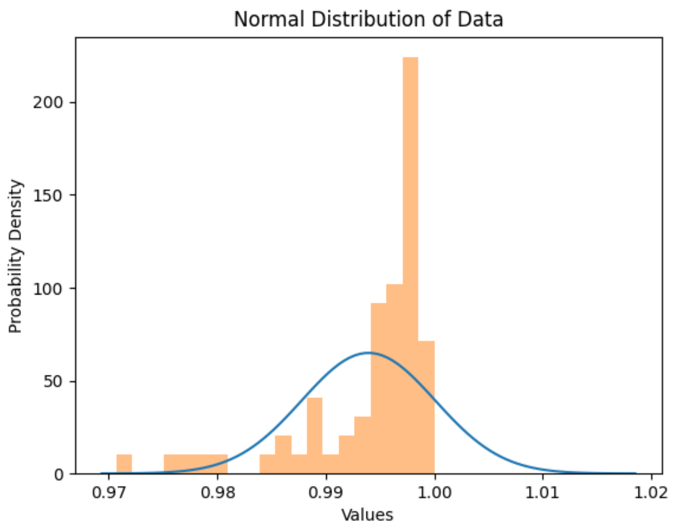


Figure 19. Normal distribution graph for Support Vector Machine algorithm

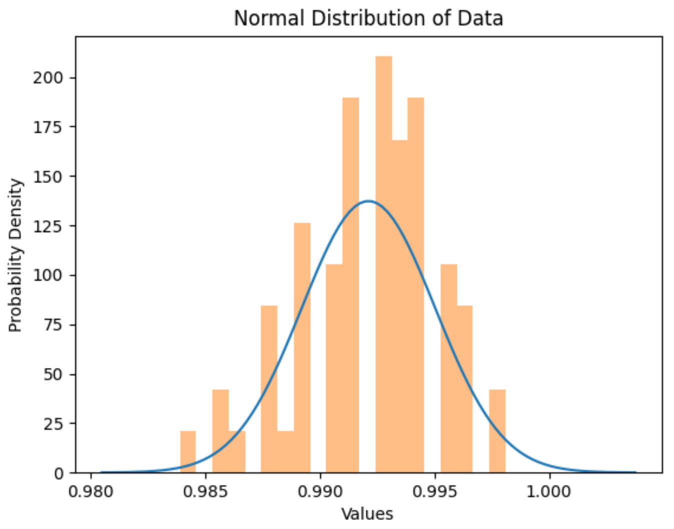


Figure 20. Normal distribution graph for Random Forest algorithm

5. Conclusions, Limitations, and Future Recommendations

5.1. Conclusions

This study began with a literature review on Sybil attacks and their targeting of different types of networks. The review delved into the protection and prevention algorithms employed to secure networks vulnerable to Sybil attacks. It also included an overview of reputation systems and an outline of our research methodologies. The study concludes by detailing the steps taken, analyzing results, explaining research limitations, and proposing future research directions. In this study, we developed an algorithm to detect Sybil nodes in an online social network, specifically Twitter, using the number of direct connections as a criterion. This algorithm can function as part of a reputation system within online social networks to identify Sybil nodes. The algorithms yield high detection accuracy rates, ranging between 96% and 97% for the KNN algorithm, 97% and 100% for the SVM algorithm, and 98.4% to 99.6% for the Random Forest algorithm. Notably, these algorithms exhibit reliability with minimal standard deviations, all under 0.0061%, with the Random Forest algorithm showing the lowest deviation at 0.0029%. These outcomes are attainable using standard workstation or desktop computers.

The algorithm developed in this study depends solely on connection data and their directions. In contrast, the approach by Patel and Mistry [12] necessitates more parameters to ascertain whether a node is a Sybil or not. Our achieved accuracy results are notably high. Furthermore, the scalability of this algorithm is noteworthy. We efficiently divided large datasets into smaller matrices, enhancing computation speed and enabling parallel execution. This matrix division also significantly reduces the total size of matrices used for training, contributing to improved performance and stability.

5.2. Limitations

One notable study's limitation pertains to the inability to train the SVM algorithm on the aforementioned four matrices. Additionally, due to the lack of access to a supercomputer, the scholars of this study faced constraints in conducting broader testing.

5.3. Future Research Opportunities

In the future, there are several avenues that offer opportunities for investigation and exploration. These are the possible propositions that can direct future research directions for readers' consideration: First, the study's algorithms can get tested on a supercomputer using the complete, undivided matrix is of interest. Such an exploration aims to assess how results might be influenced when all data is consolidated into a single, large matrix. Second, future research endeavors could explore the extension or reconfiguration of data to make it applicable to other ML algorithms is an intriguing prospect. Third, by examining the application of this algorithm on various Online Social Networks (OSNs) and Wireless Sensor Networks (WSNs) could yield valuable insights. And finally, additionally, investigating the possibility of variable reduction can reduce matrix size and minimize the algorithm's memory footprint is a worthwhile pursuit.

Data Availability

The code and data utilized in this research are available upon reasonable request. Please contact the corresponding author for access to the resources mentioned with a brief description of your intended use. We aim to respond to requests within a reasonable timeframe and may require additional information to ensure responsible data sharing practices. We are committed to promoting transparency and reproducibility in our research practices, and therefore, we encourage interested parties to reach out for access to the materials supporting this study. We believe in promoting open research and data sharing to facilitate reproducibility and advancement in the field.

Funding

Not applicable.

Appendix A

The following is the output from the machine learning algorithms. Each line consists of the file's name (i.e., the matrix number), followed by the type of the machine learning algorithm being trained and tested. The algorithm's name is followed by the four numbers inside each matrix of results and then by the accuracy value for each algorithm. The following is a detailed example explaining the first line in the data:

numbersArranged0.txt,Nearest Neighbors,699,0,38,320,0.96404919

- **numbersArranged0.txt :** The file name of the first matrix number zero.
- **Nearest Neighbors :** The type of machine learning algorithm being used, in this case, is KNN.
- **699 :** The number of true positives for Sybil nodes.
- **0 :** The number of false positives.
- **38 :** The number of false negatives.
- **320 :** The number of true positives for Legit nodes.
- **0.9640491958372753 :** This is the accuracy of KNN on the first matrix.

```
numbersArranged0.txt,Nearest Neighbors,699,0,38,320,0.9640491958372753
numbersArranged1.txt,Nearest Neighbors,699,0,36,322,0.9659413434247871
numbersArranged2.txt,Nearest Neighbors,699,0,36,322,0.965941434247871
numbersArranged3.txt,Nearest Neighbors,699,0,29,329,0.9725638599810785
```

doi:10.20944/preprints202408.0542.v1

```
numbersArranged4.txt,Nearest Neighbors,699,0,27,331,0.9744560075685903
numbersArranged5.txt,Nearest Neighbors,699,0,38,320,0.9640491958372753
numbersArranged6.txt,Nearest Neighbors,700,0,35,322,0.9668874172185431
numbersArranged7.txt,Nearest Neighbors,700,0,38,319,0.9640491958372753
numbersArranged8.txt,Nearest Neighbors,698,0,41,318,0.9612109744560076
numbersArranged9.txt,Nearest Neighbors,698,0,30,329,0.9716177861873226
numbersArranged10.txt,Nearest Neighbors,700,0,35,322,0.9668874172185431
numbersArranged11.txt,Nearest Neighbors,699,0,36,322,0.9659413434247871
numbersArranged12.txt,Nearest Neighbors,698,0,36,323,0.9659413434247871
numbersArranged13.txt,Nearest Neighbors,700,0,28,329,0.9735099337748344
numbersArranged14.txt,Nearest Neighbors,700,0,28,329,0.9735099337748344
numbersArranged15.txt,Nearest Neighbors,699,0,37,321,0.9649952696310312
numbersArranged16.txt,Nearest Neighbors,696,0,30,331,0.9716177861873226
numbersArranged17.txt,Nearest Neighbors,700,0,37,320,0.9649952696310312
numbersArranged18.txt,Nearest Neighbors,697,0,36,324,0.9659413434247871
numbersArranged19.txt,Nearest Neighbors,699,0,36,322,0.9659413434247871
numbersArranged20.txt,Nearest Neighbors,695,0,31,331,0.9706717123935666
numbersArranged21.txt,Nearest Neighbors,700,0,37,320,0.9649952696310312
numbersArranged22.txt,Nearest Neighbors,700,0,30,327,0.9716177861873226
numbersArranged23.txt,Nearest Neighbors,698,0,41,318,0.9612109744560076
numbersArranged24.txt,Nearest Neighbors,699,0,29,329,0.9725638599810785
numbersArranged25.txt,Nearest Neighbors,700,0,28,329,0.9735099337748344
numbersArranged26.txt,Nearest Neighbors,699,0,27,331,0.9744560075685903
numbersArranged27.txt,Nearest Neighbors,699,0,32,326,0.9697256385998108
numbersArranged28.txt,Nearest Neighbors,699,0,34,324,0.967833491012299
numbersArranged29.txt,Nearest Neighbors,698,0,31,328,0.9706717123935666
numbersArranged30.txt,Nearest Neighbors,697,0,39,321,0.9631031220435194
numbersArranged31.txt,Nearest Neighbors,697,0,40,320,0.9621570482497634
numbersArranged32.txt,Nearest Neighbors,699,0,27,331,0.9744560075685903
numbersArranged33.txt,Nearest Neighbors,699,0,27,331,0.9744560075685903
numbersArranged34.txt,Nearest Neighbors,697,0,31,329,0.9706717123935666
numbersArranged35.txt,Nearest Neighbors,699,0,38,320,0.9640491958372753
numbersArranged36.txt,Nearest Neighbors,700,0,33,324,0.9687795648060549
numbersArranged37.txt,Nearest Neighbors,700,0,29,328,0.9725638599810785
numbersArranged38.txt,Nearest Neighbors,695,0,40,322,0.9621570482497634
numbersArranged40.txt,Nearest Neighbors,699,0,35,323,0.9668874172185431
numbersArranged41.txt,Nearest Neighbors,699,0,36,322,0.9659413434247871
numbersArranged42.txt,Nearest Neighbors,699,0,36,322,0.9659413434247871
numbersArranged43.txt,Nearest Neighbors,700,0,34,323,0.967833491012299
numbersArranged44.txt,Nearest Neighbors,696,0,41,320,0.9612109744560076
numbersArranged45.txt,Nearest Neighbors,697,0,40,320,0.9621570482497634
numbersArranged46.txt,Nearest Neighbors,720,0,37,300,0.9649952696310312
numbersArranged47.txt,Nearest Neighbors,719,0,36,302,0.9659413434247871
numbersArranged48.txt,Nearest Neighbors,720,0,36,301,0.9659413434247871
numbersArranged49.txt,Nearest Neighbors,720,0,37,300,0.9649952696310312
numbersArranged50.txt,Nearest Neighbors,720,0,29,308,0.9725638599810785
numbersArranged51.txt,Nearest Neighbors,719,0,33,305,0.9687795648060549
numbersArranged52.txt,Nearest Neighbors,719,0,40,298,0.9621570482497634
numbersArranged53.txt,Nearest Neighbors,718,0,35,304,0.9668874172185431
numbersArranged56.txt,Nearest Neighbors,719,0,30,308,0.9716177861873226
numbersArranged58.txt,Nearest Neighbors,719,0,42,296,0.9602649006622517
numbersArranged59.txt,Nearest Neighbors,720,0,36,301,0.9659413434247871
numbersArranged60.txt,Nearest Neighbors,718,0,34,305,0.967833491012299
numbersArranged61.txt,Nearest Neighbors,719,0,31,307,0.9706717123935666
numbersArranged62.txt,Nearest Neighbors,720,0,34,303,0.967833491012299
numbersArranged63.txt,Nearest Neighbors,715,0,31,311,0.9706717123935666
numbersArranged64.txt,Nearest Neighbors,719,0,33,305,0.9687795648060549
numbersArranged65.txt,Nearest Neighbors,719,0,35,303,0.9668874172185431
numbersArranged66.txt,Nearest Neighbors,719,0,29,309,0.9725638599810785
numbersArranged67.txt,Nearest Neighbors,721,0,36,300,0.9659413434247871
numbersArranged68.txt,Nearest Neighbors,716,0,39,302,0.9631031220435194
numbersArranged69.txt,Nearest Neighbors,720,0,35,302,0.9668874172185431
numbersArranged70.txt,Nearest Neighbors,721,0,34,302,0.967833491012299
```

Appendix B

```
        numbersArranged0.txt,Random Forest,692,7,2,356,0.9914853358561968
numbersArranged1.txt,Random Forest,694,5,3,355,0.9924314096499527
numbersArranged2.txt,Random Forest,697,2,5,353,0.9933774834437086
numbersArranged3.txt,Random Forest,695,4,4,354,0.9924314096499527
numbersArranged4.txt,Random Forest,693,6,7,351,0.9877010406811731
numbersArranged5.txt,Random Forest,691,8,2,356,0.9905392620624409
numbersArranged6.txt,Random Forest,693,7,1,356,0.9924314096499527
numbersArranged7.txt,Random Forest,695,5,2,355,0.9933774834437086
numbersArranged8.txt,Random Forest,691,7,6,353,0.9877010406811731
numbersArranged9.txt,Random Forest,696,2,8,351,0.9905392620624409
numbersArranged10.txt,Random Forest,697,3,1,356,0.9962157048249763
numbersArranged11.txt,Random Forest,695,4,2,356,0.9943235572374646
numbersArranged12.txt,Random Forest,695,3,8,351,0.9895931882686849
numbersArranged13.txt,Random Forest,695,5,4,353,0.9914853358561968
numbersArranged14.txt,Random Forest,696,4,2,355,0.9943235572374646
numbersArranged15.txt,Random Forest,693,6,3,355,0.9914853358561968
numbersArranged16.txt,Random Forest,691,5,8,353,0.9877010406811731
numbersArranged17.txt,Random Forest,699,1,1,356,0.9981078524124882
numbersArranged18.txt,Random Forest,695,2,6,354,0.9924314096499527
numbersArranged19.txt,Random Forest,698,1,5,353,0.9943235572374646
numbersArranged20.txt,Random Forest,692,3,8,354,0.9895931882686849
numbersArranged21.txt,Random Forest,697,3,1,356,0.9962157048249763
numbersArranged22.txt,Random Forest,697,3,3,354,0.9943235572374646
numbersArranged23.txt,Random Forest,698,0,5,354,0.9952696310312205
numbersArranged24.txt,Random Forest,695,4,2,356,0.9943235572374646
numbersArranged25.txt,Random Forest,697,3,5,352,0.9924314096499527
numbersArranged26.txt,Random Forest,698,1,7,351,0.9924314096499527
```

```
numbersArranged27.txt,Random Forest,697,2,4,354,0.9943235572374646
numbersArranged28.txt,Random Forest,696,3,5,353,0.9924314096499527
numbersArranged29.txt,Random Forest,696,2,5,354,0.9933774834437086
numbersArranged30.txt,Random Forest,693,4,5,355,0.9914853358561968
numbersArranged31.txt,Random Forest,691,6,8,352,0.9867549668874173
numbersArranged32.txt,Random Forest,695,4,4,354,0.9924314096499527
numbersArranged33.txt,Random Forest,695,4,3,355,0.9933774834437086
numbersArranged34.txt,Random Forest,695,2,4,356,0.9943235572374646
numbersArranged35.txt,Random Forest,696,3,4,354,0.9933774834437086
numbersArranged36.txt,Random Forest,698,2,3,354,0.9952696310312205
numbersArranged37.txt,Random Forest,698,2,2,355,0.9962157048249763
numbersArranged38.txt,Random Forest,690,5,7,355,0.988647114474929
numbersArranged40.txt,Random Forest,692,7,4,354,0.9895931882686849
numbersArranged41.txt,Random Forest,697,2,2,356,0.9962157048249763
numbersArranged42.txt,Random Forest,696,3,3,355,0.9943235572374646
numbersArranged43.txt,Random Forest,694,6,5,352,0.9895931882686849
numbersArranged44.txt,Random Forest,694,2,5,356,0.9933774834437086
numbersArranged45.txt,Random Forest,692,5,6,354,0.9895931882686849
numbersArranged46.txt,Random Forest,713,7,8,329,0.9858088930936613
numbersArranged47.txt,Random Forest,715,4,6,332,0.9905392620624409
numbersArranged48.txt,Random Forest,713,7,3,334,0.9905392620624409
numbersArranged49.txt,Random Forest,719,1,4,333,0.9952696310312205
numbersArranged50.txt,Random Forest,719,1,4,333,0.9952696310312205
numbersArranged51.txt,Random Forest,716,3,4,334,0.9933774834437086
numbersArranged52.txt,Random Forest,711,8,9,329,0.9839167455061495
numbersArranged53.txt,Random Forest,715,3,6,333,0.9914853358561968
numbersArranged56.txt,Random Forest,716,3,5,333,0.9924314096499527
numbersArranged58.txt,Random Forest,713,6,3,335,0.9914853358561968
numbersArranged59.txt,Random Forest,717,3,4,333,0.9933774834437086
numbersArranged60.txt,Random Forest,714,4,5,334,0.9914853358561968
numbersArranged61.txt,Random Forest,715,4,2,336,0.9943235572374646
numbersArranged62.txt,Random Forest,717,3,8,329,0.9895931882686849
numbersArranged63.txt,Random Forest,708,7,8,334,0.9858088930936613
numbersArranged64.txt,Random Forest,719,0,2,336,0.9981078524124882
numbersArranged65.txt,Random Forest,717,2,7,331,0.9914853358561968
numbersArranged66.txt,Random Forest,717,2,7,331,0.9914853358561968
numbersArranged67.txt,Random Forest,717,4,4,332,0.9924314096499527
numbersArranged68.txt,Random Forest,710,6,7,334,0.9877010406811731
numbersArranged69.txt,Random Forest,717,3,2,335,0.9952696310312205
numbersArranged70.txt,Random Forest,718,3,7,329,0.9905392620624409
```

Appendix C

```
        numbersArranged0.txt,SVM,689,10,3,355,0.9877010406811731
numbersArranged1.txt,SVM,699,0,3,355,0.9971617786187322
numbersArranged2.txt,SVM,674,25,1,357,0.9754020813623463
numbersArranged3.txt,SVM,699,0,1,357,0.9990539262062441
numbersArranged4.txt,SVM,693,6,1,357,0.9933774834437086
numbersArranged5.txt,SVM,686,13,1,357,0.9867549668874173
numbersArranged6.txt,SVM,697,3,2,355,0.9952696310312205
numbersArranged7.txt,SVM,699,1,1,356,0.9981078524124882
numbersArranged8.txt,SVM,696,2,4,355,0.9943235572374646
numbersArranged9.txt,SVM,692,6,5,354,0.9895931882686849
numbersArranged10.txt,SVM,700,0,2,355,0.9981078524124882
numbersArranged11.txt,SVM,699,0,1,357,0.9990539262062441
numbersArranged12.txt,SVM,694,4,5,354,0.9914853358561968
numbersArranged13.txt,SVM,699,1,3,354,0.9962157048249763
numbersArranged14.txt,SVM,697,3,3,354,0.9943235572374646
numbersArranged15.txt,SVM,696,3,2,356,0.9952696310312205
numbersArranged16.txt,SVM,689,7,8,353,0.9858088930936613
numbersArranged17.txt,SVM,699,1,2,355,0.9971617786187322
numbersArranged18.txt,SVM,697,0,5,355,0.9952696310312205
numbersArranged19.txt,SVM,699,0,3,355,0.9971617786187322
numbersArranged20.txt,SVM,695,0,6,356,0.9943235572374646
numbersArranged21.txt,SVM,698,2,0,357,0.9981078524124882
numbersArranged22.txt,SVM,700,0,1,356,0.9990539262062441
numbersArranged23.txt,SVM,698,0,3,356,0.9971617786187322
numbersArranged24.txt,SVM,694,5,3,355,0.9924314096499527
numbersArranged25.txt,SVM,699,1,2,355,0.9971617786187322
numbersArranged26.txt,SVM,699,0,3,355,0.9971617786187322
numbersArranged27.txt,SVM,699,0,2,356,0.9981078524124882
numbersArranged28.txt,SVM,698,1,3,355,0.9962157048249763
numbersArranged29.txt,SVM,698,0,3,356,0.9971617786187322
numbersArranged30.txt,SVM,678,19,12,348,0.9706717123935666
numbersArranged31.txt,SVM,694,3,4,356,0.9933774834437086
numbersArranged32.txt,SVM,698,1,2,356,0.9971617786187322
numbersArranged33.txt,SVM,699,0,3,355,0.9971617786187322
numbersArranged34.txt,SVM,694,3,7,353,0.9905392620624409
numbersArranged35.txt,SVM,699,0,2,356,0.9981078524124882
numbersArranged36.txt,SVM,698,2,2,355,0.9962157048249763
numbersArranged37.txt,SVM,698,2,2,355,0.9962157048249763
numbersArranged38.txt,SVM,693,2,9,353,0.9895931882686849
numbersArranged40.txt,SVM,699,0,2,356,0.9981078524124882
numbersArranged41.txt,SVM,699,0,2,356,0.9981078524124882
numbersArranged42.txt,SVM,698,1,3,355,0.9962157048249763
numbersArranged43.txt,SVM,699,1,4,353,0.9952696310312205
numbersArranged44.txt,SVM,696,0,4,357,0.9962157048249763
numbersArranged45.txt,SVM,697,0,4,356,0.9962157048249763
numbersArranged46.txt,SVM,712,8,4,333,0.988647114474929
numbersArranged47.txt,SVM,719,0,2,336,0.9981078524124882
numbersArranged48.txt,SVM,720,0,1,336,0.9990539262062441
numbersArranged49.txt,SVM,716,4,3,334,0.9933774834437086
numbersArranged50.txt,SVM,720,0,1,336,0.9990539262062441
numbersArranged51.txt,SVM,717,2,2,336,0.9962157048249763
numbersArranged52.txt,SVM,717,2,2,336,0.9962157048249763
numbersArranged53.txt,SVM,718,0,3,336,0.9971617786187322
numbersArranged56.txt,SVM,714,5,7,331,0.988647114474929
```

```
numbersArranged58.txt,SVM,719,0,2,336,0.9981078524124882
numbersArranged59.txt,SVM,719,1,1,336,0.9981078524124882
numbersArranged60.txt,SVM,705,13,11,328,0.9772942289498581
numbersArranged61.txt,SVM,715,4,2,336,0.9943235572374646
numbersArranged62.txt,SVM,720,0,1,336,0.9990539262062441
numbersArranged63.txt,SVM,700,15,8,334,0.978240302743614
numbersArranged64.txt,SVM,700,19,2,336,0.9801324503311258
numbersArranged65.txt,SVM,718,1,2,336,0.9971617786187322
numbersArranged66.txt,SVM,719,0,4,334,0.9962157048249763
numbersArranged67.txt,SVM,721,0,0,336,1.0
numbersArranged68.txt,SVM,707,9,8,333,0.9839167455061495
numbersArranged69.txt,SVM,719,1,1,336,0.9981078524124882
numbersArranged70.txt,SVM,718,3,3,333,0.9943235572374646
```

References

1. Saxena, G.D.; Dinesh, G.; David, D.S.; Tiwari, M.; Tiwari, T.; Monisha, M.; Chauhan, A. Addressing The Distinct Security Vulnerabilities Typically Emerge On The Mobile Ad-Hoc Network Layer. *NeuroQuantology* **2023**, *21*, 169–178.
2. Manju, V. Sybil attack prevention in wireless sensor network. *International Journal of Computer Networking, Wireless and Mobile Communications (IJCNWMC)* **2014**, *4*, 125–132.
3. Mahesh, B. Machine learning algorithms-a review. *International Journal of Science and Research (IJSR).[Internet]* **2020**, *9*, 381–386.
4. Rahbari, M.; Jamali, M.A.J. Efficient detection of Sybil attack based on cryptography in VANET. *arXiv preprint arXiv:1112.2257* **2011**.
5. Chang, W.; Wu, J. A survey of sybil attacks in networks. *Sensor Networks for Sustainable Development* **2012**.
6. Balachandran, N.; Sanyal, S. A review of techniques to mitigate sybil attacks. *arXiv preprint arXiv:1207.2617* **2012**.
7. Misra, S.; Tayeen, A.S.M.; Xu, W. SybilExposer: An effective scheme to detect Sybil communities in online social networks. 2016 IEEE International Conference on Communications (ICC). IEEE, 2016, pp. 1–6.
8. Fong, P.W. Preventing Sybil attacks by privilege attenuation: A design principle for social network systems. 2011 IEEE Symposium on Security and Privacy. IEEE, 2011, pp. 263–278.
9. Margolin, N.B.; Levine, B.N. Informant: Detecting sybils using incentives. International Conference on Financial Cryptography and Data Security. Springer, 2007, pp. 192–207.
10. Du, W.; Deng, J.; Han, Y.S.; Varshney, P.K.; Katz, J.; Khalili, A. A pairwise key predistribution scheme for wireless sensor networks. *ACM Transactions on Information and System Security (TISSEC)* **2005**, *8*, 228–258.
11. Lu, H.; Gong, D.; Li, Z.; Liu, F.; Liu, F. SybilHP: Sybil Detection in Directed Social Networks with Adaptive Homophily Prediction. *Applied Sciences* **2023**, *13*, 5341.
12. Patel, S.T.; Mistry, N.H. A review: Sybil attack detection techniques in WSN. 2017 4th International conference on electronics and communication systems (ICECS). IEEE, 2017, pp. 184–188.
13. Almesaeed, R.; Al-Salem, E. Sybil attack detection scheme based on channel profile and power regulations in wireless sensor networks. *Wireless Networks* **2022**, *28*, 1361–1374.
14. Batchelor, B.G. *Pattern recognition: Ideas in practice*; Springer Science & Business Media, 2012.
15. Kafetzis, D.; Vassilaras, S.; Vardoulias, G.; Koutsopoulos, I. Software-defined networking meets software-defined radio in mobile ad hoc networks: State of the art and future directions. *IEEE Access* **2022**, *10*, 9989–10014.
16. Cui, Z.; Fei, X.; Zhang, S.; Cai, X.; Cao, Y.; Zhang, W.; Chen, J. A hybrid blockchain-based identity authentication scheme for multi-WSN. *IEEE Transactions on Services Computing* **2020**, *13*, 241–251.
17. Saraswathi, R.V.; Sree, L.P.; Anuradha, K. Support vector based regression model to detect Sybil attacks in WSN. *International Journal of Advanced Trends in Computer Science and Engineering* **2020**, *9*.
18. Choy, G.; Khalilzadeh, O.; Michalski, M.; Do, S.; Samir, A.E.; Pianykh, O.S.; Geis, J.R.; Pandharipande, P.V.; Brink, J.A.; Dreyer, K.J. Current applications and future impact of machine learning in radiology. *Radiology* **2018**, *288*, 318–328.
19. Tong, F.; Zhang, Z.; Zhu, Z.; Zhang, Y.; Chen, C. A novel scheme based on coarse-grained localization and fine-grained isolation for defending against Sybil attack in low power and lossy networks. *Asian Journal of Control* **2023**.

20. Nayyar, A.; Rameshwar, R.; Solanki, A. Internet of Things (IoT) and the digital business environment: A standpoint inclusive cyber space, cyber crimes, and cybersecurity. *The evolution of business in the cyber age* **2020**, *10*, 9780429276484–6.
21. Alsafery, W.; Rana, O.; Perera, C. Sensing within smart buildings: A survey. *ACM Computing Surveys* **2023**, *55*, 1–35.
22. NBC News. https://www.nbcnews.com/tech/social-media/, 2023. [NBC News Home.].
23. Vincent, J. Emotion and the mobile phone. *Cultures of participation: Media practices, politics and literacy* **2011**, pp. 95–109.
24. Xiao, L.; Greenstein, L.J.; Mandayam, N.B.; Trappe, W. Channel-based detection of sybil attacks in wireless networks. *IEEE Transactions on Information Forensics and Security* **2009**, *4*, 492–503.
25. Samuel, S.J.; Dhivya, B. An efficient technique to detect and prevent Sybil attacks in social network applications. 2015 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT). IEEE, 2015, pp. 1–3.
26. Douceur, J.R. The sybil attack. International workshop on peer-to-peer systems. Springer, 2002, pp. 251–260.
27. Arif, M.; Wang, G.; Bhuiyan, M.Z.A.; Wang, T.; Chen, J. A survey on security attacks in VANETs: Communication, applications and challenges. *Vehicular Communications* **2019**, *19*, 100179.
28. Iswanto, I.; Tulus, T.; Sihombing, P. Comparison of distance models on K-Nearest Neighbor algorithm in stroke disease detection. *Applied Technology and Computing Science Journal* **2021**, *4*, 63–68.
29. Helmi, Z.; Adriman, R.; Arif, T.Y.; Walidainy, H.; Fitria, M.; others. Sybil Attack Prediction on Vehicle Network Using Deep Learning. *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)* **2022**, *6*, 499–504.
30. Ben-Hur, A.; Ong, C.S.; Sonnenburg, S.; Schölkopf, B.; Rätsch, G. Support vector machines and kernels for computational biology. *PLoS computational biology* **2008**, *4*, e1000173.
31. Hu, L.Y.; Huang, M.W.; Ke, S.W.; Tsai, C.F. The distance function effect on k-nearest neighbor classification for medical datasets. *SpringerPlus* **2016**, *5*, 1–9.
32. Lee, G.; Lim, J.; Kim, D.k.; Yang, S.; Yoon, M. An approach to mitigating sybil attack in wireless networks using zigBee. 2008 10th International Conference on Advanced Communication Technology. IEEE, 2008, Vol. 2, pp. 1005–1009.
33. Eschenauer, L.; Gligor, V.D. A key-management scheme for distributed sensor networks. Proceedings of the 9th ACM Conference on Computer and Communications Security, 2002, pp. 41–47.
34. Dhamodharan, U.S.R.K.; Vayanaperumal, R.; others. Detecting and preventing sybil attacks in wireless sensor networks using message authentication and passing method. *The Scientific World Journal* **2015**, *2015*.
35. Ammari, A.; Bensalem, A.; others. Fault tolerance and VANET (Vehicular Ad-Hoc Network). PhD thesis, UNIVERSITY of M'SILA, 2022.
36. Quevedo, C.H.; Quevedo, A.M.; Campos, G.A.; Gomes, R.L.; Celestino, J.; Serhrouchni, A. An intelligent mechanism for sybil attacks detection in vanets. ICC 2020-2020 IEEE International Conference on Communications (ICC). IEEE, 2020, pp. 1–6.
37. Yu, H.; Gibbons, P.B.; Kaminsky, M.; Xiao, F. Sybillimit: A near-optimal social network defense against sybil attacks. 2008 IEEE Symposium on Security and Privacy (sp 2008). IEEE, 2008, pp. 3–17.
38. Abbas, S.; Merabti, M.; Llewellyn-Jones, D.; Kifayat, K. Lightweight sybil attack detection in manets. *IEEE systems journal* **2012**, *7*, 236–248.
39. Newsome, J.; Shi, E.; Song, D.; Perrig, A. The sybil attack in sensor networks: Analysis & defenses. Proceedings of the 3rd international symposium on Information processing in sensor networks, 2004, pp. 259–268.
40. Chen, Y.; Yang, J.; Trappe, W.; Martin, R.P. Detecting and localizing identity-based attacks in wireless and sensor networks. *IEEE Transactions on Vehicular Technology* **2010**, *59*, 2418–2434.
41. Shetty, N.P.; Muniyal, B.; Anand, A.; Kumar, S. An enhanced sybil guard to detect bots in online social networks. *Journal of Cyber Security and Mobility* **2022**, pp. 105–126.
42. Mounica, M.; Vijayasaraswathi, R.; Vasavi, R. RETRACTED: Detecting Sybil Attack In Wireless Sensor Networks Using Machine Learning Algorithms. IOP Conference Series: Materials Science and Engineering. IOP Publishing, 2021, Vol. 1042, p. 012029.
43. Twitter follower-followee graph . https://figshare.com/articles/dataset/Twitter_follower-followee_graph_labeled_with_benign_Sybil/20057300, 2022. [Labeled with benign/Sybil.].

44. Demirbas, M.; Song, Y. An RSSI-based scheme for sybil attack detection in wireless sensor networks. 2006 International symposium on a world of wireless, mobile and multimedia networks (WoWMoM'06). ieee, 2006, pp. 5–pp.
45. Machine LEarning Research. https://machinelearning.apple.com/research/hey-siri, 2023. [Apple's siri voice recognition software.].
46. Alexa. https://developer.amazon.com/, 2023. [Amazon's alexa voice recognition software.].
47. Kak, S. A three-stage quantum cryptography protocol. *Foundations of Physics Letters* **2006**, *19*, 293–296.
48. Sahami, M.; Dumais, S.; Heckerman, D.; Horvitz, E. A Bayesian approach to filtering junk e-mail. Learning for Text Categorization: Papers from the 1998 workshop. Citeseer, 1998, Vol. 62, pp. 98–105.
49. Schiappa, M.C.; Rawat, Y.S.; Shah, M. Self-supervised learning for videos: A survey. *ACM Computing Surveys* **2023**, *55*, 1–37.
50. Zamsuri, A.; Defit, S.; Nurcahyo, G.W. Classification of Multiple Emotions in Indonesian Text Using The K-Nearest Neighbor Method. *Journal of Applied Engineering and Technological Science (JAETS)* **2023**, *4*, 1012–1021.
51. Gupta, M.; Judge, P.; Ammar, M. A reputation system for peer-to-peer networks. Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video, 2003, pp. 144–152.
52. Michalski, R.S.; Stepp, R.E.; Diday, E. A recent advance in data analysis: Clustering objects into classes characterized by conjunctive concepts. In *Progress in pattern recognition*; Elsevier, 1981; pp. 33–56.
53. Medjahed, S.A.; Saadi, T.A.; Benyettou, A. Breast cancer diagnosis by using k-nearest neighbor with different distances and classification rules. *International Journal of Computer Applications* **2013**, *62*.
54. Swamynathan, G.; Almeroth, K.C.; Zhao, B.Y. The design of a reliable reputation system. *Electronic Commerce Research* **2010**, *10*, 239–270.
55. Valarmathi, M.; Meenakowshalya, A.; Bharathi, A. Robust Sybil attack detection mechanism for Social Networks-a survey. 2016 3rd International conference on advanced computing and communication systems (ICACCS). IEEE, 2016, Vol. 1, pp. 1–5.
56. Vasudeva, A.; Sood, M. Survey on sybil attack defense mechanisms in wireless ad hoc networks. *Journal of Network and Computer Applications* **2018**, *120*, 78–118.
57. Yu, H.; Kaminsky, M.; Gibbons, P.B.; Flaxman, A. Sybilguard: Defending against sybil attacks via social networks. Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications, 2006, pp. 267–278.
58. Yuan, D.; Miao, Y.; Gong, N.Z.; Yang, Z.; Li, Q.; Song, D.; Wang, Q.; Liang, X. Detecting fake accounts in online social networks at the time of registrations. Proceedings of the 2019 ACM SIGSAC conference on computer and communications security, 2019, pp. 1423–1438.
59. Zhang, K.; Liang, X.; Lu, R.; Shen, X. Sybil attacks and their defenses in the internet of things. *IEEE Internet of Things Journal* **2014**, *1*, 372–383.
60. Kaggle. https://www.kaggle.com/, 2023. [Level up with the largest AI & ML community.].
61. Jain, N.; Jana, P.K. LRF: A logically randomized forest algorithm for classification and regression problems. *Expert Systems with Applications* **2023**, *213*, 119225.
62. Jethava, G.; Rao, U.P. User behavior-based and graph-based hybrid approach for detection of sybil attack in online social networks. *Computers and Electrical Engineering* **2022**, *99*, 107753.