**Journal of Advances in Mathematical & Computational Sciences**
An International Pan-African Multidisciplinary Journal of the SMART Research Group
International Centre for IT & Development (ICITD) USA
© Creative Research Publishers
Available online at https://www.isteams.net/ mathematics-computationaljournal.info
CrossREF Member Listing - https://www.crossref.org/06members/50go-live.html

# An Advanced Feature Engineering Approach for Sybil Attack Detection in Fog Computing Environment

[1]Shuaib Maryam, [1,2]Abdulhamid, Shafii Muhammad, [1]Ojeniyi, Joseph A., [3]Dauda, Umar Suleiman, [1]Ismaila Idris & [1]Dogonyaro, Noel Moses
[1]Department of Cyber Security Science Federal University of Technology Minna, Nigeria
[2]Dept of Information Technology (Cybersecurity Unit), Community College of Qatar, Lusail, Qatar
[3]Department of Electrical Engineering, Federal University of Technology Minna, Nigeria
**Emails:** maryambobi@gmail.com, shafii.abdulhamid@futminna.edu.ng, ojeniyija@futminna.edu.ng, usdauda@gmail.com, ismi.idris@futminna.edu.ng, moses.noel@futminna.edu.ng

## ABSTRACT

Sybil attacks gravely impair the integrity and dependability of fog computing environments, especially when operating in IoT networks. These attacks consist of malicious entities creating multiple identities to disrupt authentic operations of a network. Traditional detection mechanisms have been known to report very high false-positive rates along with latency issues. This paper is an introduction to an advanced feature engineering strategy focused on Sybil attack detection in fog computing environments. The proposed strategy, when experimenting with a balanced and engineered dataset, achieved an accuracy of 86%, which is an improvement over the result gotten from the original dataset. The proposed approach uses the Synthetic Minority Over-sampling Technique (SMOTE) to address class imbalance problems in sybil attack simulation datasets. The results illustrate the promise of advanced feature engineering on datasets to further protect fog computing infrastructure from Sybil attacks especially when integrated with Federated Learning.

**Keywords:** Feature Engineering, Fog Computing, Fog Security, Sybil Detection, Anomaly Detection, Sybil Attack simulation Dataset.

Shuaib, M., Abdulhamid, S.M., Ojeniyi, J.A., Dauda, U.S., Ismaila, I. & Dogonyaro, N.M. (2025): An Advanced Feature Engineering Approach for Sybil Attack Detection in Fog Computing Environment. Journal of Advances in Mathematical & Computational Science. Vol. 13, No. 2. Pp 15-34. Available online at www.isteams.net/mathematics-computationaljournal. dx.doi.org/10.22624/AIMS/MATHS/V13N2P2

## 1. INTRODUCTION

Fog Computing is a new model of computing that geographically extends the Cloud Computing services to the edge of the network and distributes computing architecture with a resource pool consisting of one or more ubiquitously connected heterogeneous devices at the edge of network and not entirely seamlessly backed by cloud services, to provide collaborative elastic computation, storage and other services either in remote locations or in large number of clients nearby through facilities or infrastructures referred to as fog devices (Albdour et al., 2020; Prabhu, 2019). The emergence of the cloud model is attached to the surge in internet computation, web expansion, and complexity growth due to the rise of new technologies and solutions (Yakubu et al., 2019). Fog gives increased support to cloud environment by performing certain local data analysis at the edge of the devices, facilitating networking, computing, infrastructure and storage support as backbone for end user computing while solving bandwidth, latency, and communications challenges associated with next generation networks (Priyadarshini & Barik, 2019).

Federated Learning (FL) is an emerging approach in distributed machine learning where multiple clients—such as mobile devices—work together to train a shared model without exposing their personal data. In a standard FL framework, a central server manages the global model and coordinates client participation. Instead of sharing raw data, each client sends model updates to the central server, which aggregates them to refine the global model. Since the data remains on local devices, FL ensures strong privacy protection for users and has been widely adopted in areas like edge computing, finance, and healthcare. However, the presence of malicious clients poses significant security challenges, hindering the real-world implementation of FL systems (Li et al., 2020)

Feature engineering plays a vital role in preparing data for machine learning models. It involves creating effective features from existing ones to enhance predictive accuracy. This process includes applying various transformation functions, like arithmetic or aggregation operations, to develop new features. Such transformations can help rescale features or convert complex non-linear relationships between features and the target variable into simpler linear ones, which are generally easier for models to learn (Nargesian et al., 2017).

This research aims to develop an automated feature engineering algorithm capable of extracting interpretable features from network traffic and augment minority samples to enhance the identification of Sybil attack patterns in fog computing environments using SMOTE algorithm.

## 2. REVIEW OF RELATED WORKS

The effectiveness of using feature engineering in enhancing performance of datasets is seen in numerous research works.

Hollmann et al., (2023) presents CAAFE- Context-aware automated feature extraction from tabular datasets using large language models (LLMs). CAAFE repackages semantically meaningful features iteratively based on the description of the datasets, outputting both Python for feature construction and justification of value. The approach resulted in improved performance on 11 out of 14 datasets, resulting in an increase in mean ROC AUC from 0.798 to 0.822.

Machine learning algorithms, along with feature engineering techniques were used by (Sihombing, 2024) to predict property prices. Feature importance analysis identified key factors influencing property prices, offering valuable insights for property appraisals and investment decisions for valuators and property investment decisions.

Sultana et al., (2024) identified that Vehicular Ad Hoc Networks (VANETs) face significant security threats, particularly from Sybil attacks, which compromise the trustworthiness of information by enabling attackers to create multiple fake identities. The proposed approach analyzes RSSI time series as vehicular speech to compare similarities among received series, allowing for independent detection without centralized support. Extensive simulations and real-world experiments demonstrate that the Voiceprint method effectively detects Sybil attacks while considering cost, complexity, and performance, enabling detection on Service Channels (SCH) to reduce observation time. Existing RSSI-based methods often rely on absolute positioning or statistical testing, which can be limited in effectiveness, highlighting the need for a lightweight, fully distributed detection mechanism that does not depend on predefined radio propagation models and advised that future research could focus on enhancing the Voiceprint method with additional features through feature engineering to improve detection accuracy and exploring its application in various vehicular environments and conditions.

## 3. MATERIALS AND METHODS

In carrying out this research three steps were involved: Dataset Preparation, Pre-Processing and Application of Two machine learning classifiers Support Vector Machine (SVM) and Random Forest (RF)) to evaluate the performance in sybil attack detection.

### *A. Dataset Preparation, Pre-Processing and Algorithm Application*

The Sybil Attack Simulation dataset (Muhammad Zunnurain Hussain, 2024) gotten from the IEEE Dataport was used. The dataset has 6 features (ID, Timestamp, User_ID, Action_Type, Is_Sybil, IP_Address) in 50,000 records.

### *B. Feature Engineering*

Feature engineering constitutes a critical phase in the development of robust Sybil attack detection systems, serving as the bridge between raw data and effective machine learning models. Given the sophisticated and evolving nature of Sybil attacks, which exploit multiple fake identities to manipulate networked systems, it is imperative to extract meaningful and discriminative features that capture both behavioural nuances and network-level anomalies. This stage involves the systematic transformation of raw user activity logs and network metadata into quantifiable metrics that reflect underlying patterns indicative of malicious behaviour.

By grounding the detection framework in carefully crafted features, this stage lays the foundation for subsequent modelling efforts, enabling classifiers such as Support Vector Machines and Random Forests to leverage rich, multidimensional representations of user behaviour and network interactions. Four principal features were engineered for this purpose, each designed to capture distinct facets of user activity and network characteristics relevant to Sybil attack identification. The additional features produced are discussed in detail:

#### i. Frequency of User Actions

The first feature focuses on the frequency of user actions, which serves as a fundamental indicator of user behaviour patterns. This feature quantifies how often a particular user performs actions within the system, mathematically represented as:

Given a Dataset $D$ with user actions, the frequency of user actions $F_{UA}$ for each user $U$ can be calculated as:

$$F_{UA}(U) = \sum_{i=1}^{N} \mathbb{I}(a_i \in U) \quad (3.1)$$

Where $N$ is the total number of actions, $a_i$ is the $i^{th}$ action, and $\mathbb{I}$ is the indicator function that returns 1 if the action belongs to user $U$, otherwise 0.

By aggregating the total number of actions, this metric captures the intensity and volume of user engagement. In the context of Sybil attack detection, abnormal activity frequencies-such as unusually high or low numbers of actions-can be indicative of malicious behavior, since Sybil accounts often generate excessive or patterned activities to manipulate the system. This feature thus establishes a baseline for normal user activity, enabling the identification of outliers that may correspond to Sybil entities.

ii. IP Address Frequency

The second feature examines the frequency distribution of IP addresses associated with user activities, expressed Mathematically as:

$$F_{IP}(IP) = \sum_{i=1}^{M} \mathbb{I}(iP_i = IP) \quad (3.2)$$

Where $M$ is the total number of IP addresses, $iP_i$ is the $i^{th}$ IP address and $\mathbb{I}$ is the indicator function that returns 1 if $iP_i$ matches $IP$ otherwise 0.

This feature captures the network-level access patterns by counting how often a specific IP address is used across the dataset. Legitimate users tend to access the system from a limited and relatively stable set of IP addresses, reflecting consistent geographic or network origins. In contrast, Sybil attackers often employ multiple IP addresses, including proxies or VPNs, to obscure their identity and evade detection. By analyzing the frequency and variability of IP addresses, this feature helps in identifying suspicious access patterns that are characteristic of Sybil attacks, such as rapid switching between IPs or use of IP addresses known to be associated with malicious activity.

iii. Interaction Patterns (Action Type Ratio)

The third feature characterizes user behavior through the ratio of different action types performed by a user, formulated as:

The ratio of a specific action type T for each user $U$ can be calculated as:

$$F_{AT}(U,T) = \frac{\sum_{i=1}^{N} \mathbb{I}(a_i = T \wedge a_i \in U)}{\sum_{i=1}^{N} \mathbb{I}(a_i \in U)} \quad (3.3)$$

Where $N$ is the total number of actions, $a_i$ is the $i^{th}$ actions, and $\mathbb{I}$ is the indicator function that returns 1 if the action matches type $T$ and belongs to user $U$ otherwise 0.

This ratio measures the proportion of a specific action type T$T$ relative to the total actions performed by user U$U$. Legitimate users typically exhibit diverse and organic distributions of action types that reflect genuine interests and interactions within the system. Conversely, Sybil accounts often display skewed or constrained action type distributions, as they are programmed to perform repetitive or narrowly focused actions to achieve their malicious goals. By capturing these behavioral signatures, this feature enhances the detection framework's ability to distinguish between authentic and synthetic user profiles based on the semantic composition of their activities.

iv. Network Features

The fourth feature integrates network-centric metrics by quantifying the diversity of IP addresses and actions associated with each user. Specifically, it measures the cardinality of unique IP addresses and the number of distinct actions.
Unique IPs per User:

$$U_{IPs}(U) = |\{ip_i : ip_i \in U\}| \tag{3.4}$$

Where |·| denotes the cardinality of the set of unique IP addresses associated with user $U$.
Unique Actions per User:

$$U_{Acts}(U) = |\{a_i : a_i \in U\}| \tag{3.5}$$

Where |·| denotes the cardinality of the set of unique actions associated with user $U$.

These features enhance the dataset by incorporating user behavior patterns, IP address usage, and network interaction metrics, which can be crucial for detecting sybil attacks. These set-based features provide insights into the relational and contextual properties of user accounts within the network. Authentic users generally demonstrate a balance between consistency and diversity in their network footprints, accessing the system from a limited yet varied set of IPs and engaging in multiple types of actions.

In contrast, Sybil accounts may exhibit either overly narrow or excessively broad network characteristics, such as using a single IP for many accounts or an unusually high number of IPs to mask identity. By combining behavioral and network diversity metrics, this feature enriches the detection model's capacity to identify Sybil attacks through comprehensive profiling of user activity and network behavior.

*C. Data Augmentation*

Synthetic Minority Over-sampling Technique (SMOTE) is an advanced statistical method designed to address class imbalance problems in machine learning datasets. SMOTE works by generating synthetic samples for the minority class rather than simply duplicating existing instances, which distinguishes it from traditional oversampling methods.

The core mechanism of SMOTE is captured in the mathematical model presented given as:

$$x_{synthetic} = x + \lambda \cdot \left(x_{neighbor} - x\right) \quad (3.6)$$

where:

a. x represents an existing minority class sample
b. x_neighbor is one of the k-nearest neighbors of x within the minority class
c. λ (lambda) is a random factor in the range 1
d. x_synthetic is the newly generated synthetic instance

This formula creates new samples along the line segments connecting a minority instance to its neighbors in feature space. The randomization factor λ ensures diversity among the synthetic samples.

The SMOTE algorithm follows these steps:

1. For each minority class sample, identify its k-nearest neighbors from the same class
2. Randomly select one of these neighbors
3. Calculate the feature-space difference vector between the sample and its selected neighbor
4. Multiply this difference vector by a random number λ between 0 and 1
5. Add this weighted difference to the original sample to create a new synthetic sample
6. Repeat until the desired balance between classes is achieved

The k parameter (typically 5) controls locality sensitivity - lower values create synthetic samples closer to existing minority samples, while higher values allow for more generalization.

4. RESULTS

The sample of the original dataset used and the result from feature engineering and augmentation are presented in Table 4.1 – 4.3. The comparison of performance in terms of Accuracy, Precision, Recall and F1 – score is summarised here.

Table 4.1: Original Sybil Attack Dataset

| ID | Timestamp | User_ID | Action_Type | Is_Sybil | IP_Address |
|---|---|---|---|---|---|
| 1 | '01/01/2023 00:00' | 7219 | 1 | 'FALSE' | 3232275849 |
| 2 | '01/01/2023 00:01' | 856 | 3 | 'FALSE' | 3232279682 |
| 3 | '01/01/2023 00:02' | 5353 | 3 | 'FALSE' | 3232282638 |
| 4 | '01/01/2023 00:03' | 5155 | 4 | 'FALSE' | 3232286889 |
| 5 | '01/01/2023 00:04' | 5692 | 1 | 'FALSE' | 3232267203 |
| 6 | '01/01/2023 00:05' | 6219 | 5 | 'FALSE' | 3232236226 |
| 7 | '01/01/2023 00:06' | 464 | 4 | 'FALSE' | 3232290372 |
| 8 | '01/01/2023 00:07' | 4396 | 5 | 'FALSE' | 3232236261 |
| 9 | '01/01/2023 00:08' | 5539 | 2 | 'FALSE' | 3232242539 |
| 10 | '01/01/2023 00:09' | 8261 | 1 | 'FALSE' | 3232275577 |
| 11 | '01/01/2023 00:10' | 1680 | 4 | 'FALSE' | 3232293113 |
| 12 | '01/01/2023 00:11' | 765 | 4 | 'FALSE' | 3232257273 |
| 13 | '01/01/2023 00:12' | 6898 | 2 | 'TRUE' | 3232300381 |
| 14 | '01/01/2023 00:13' | 2424 | 2 | 'FALSE' | 3232287518 |
| 15 | '01/01/2023 00:14' | 5274 | 2 | 'FALSE' | 3232243373 |
| 16 | '01/01/2023 00:15' | 5015 | 4 | 'FALSE' | 3232286572 |
| 17 | '01/01/2023 00:16' | 6372 | 3 | 'FALSE' | 3232294018 |
| 18 | '01/01/2023 00:17' | 1179 | 5 | 'FALSE' | 3232295283 |
| 19 | '01/01/2023 00:18' | 4523 | 5 | 'FALSE' | 3232284360 |
| 20 | '01/01/2023 00:19' | 3366 | 2 | 'FALSE' | 3232248622 |
| 21 | '01/01/2023 00:20' | 6348 | 3 | 'TRUE' | 3232274152 |
| 22 | '01/01/2023 00:21' | 8603 | 3 | 'FALSE' | 3232288166 |
| 23 | '01/01/2023 00:22' | 9208 | 2 | 'FALSE' | 3232297681 |
| 24 | '01/01/2023 00:23' | 2548 | 1 | 'FALSE' | 3232271272 |
| 25 | '01/01/2023 00:24' | 7793 | 4 | 'FALSE' | 3232255450 |
| 26 | '01/01/2023 00:25' | 2039 | 1 | 'FALSE' | 3232279520 |
| 27 | '01/01/2023 00:26' | 2736 | 3 | 'FALSE' | 3232239990 |
| 28 | '01/01/2023 00:27' | 9102 | 2 | 'FALSE' | 3232284224 |
| 29 | '01/01/2023 00:28' | 9923 | 4 | 'FALSE' | 3232292819 |
| 30 | '01/01/2023 00:29' | 190 | 5 | 'FALSE' | 3232251171 |

### 4.1 Feature engineered dataset

The dataset shown in Table 4.2 represents a sample of engineered features designed to enhance Sybil attack detection by capturing user behavior and network characteristics. Each row corresponds to a unique user interaction instance characterized by several attributes: User_ID, Action_Type, IP_Address, User_Action Count, IP_Frequency, Action_Type_ratio, User_Unique_IP, User_Unique_Action, and a binary Label indicating whether the instance is benign (0) or Sybil (1).

Table 4.2: Newly Generated Sybil Attack Data with New Features

| User_ID | Action_Type | IP_Address | User_Action Count | IP_Frequency | Action Type_ratio | User Unique_IP | User_Unique Action | Label |
|---|---|---|---|---|---|---|---|---|
| 7219 | 1 | 3232275849 | 2 | 2 | 0.5 | 2 | 2 | 0 |
| 856 | 3 | 3232279682 | 6 | 2 | 0.333333333 | 6 | 4 | 0 |
| 5353 | 3 | 3232282638 | 6 | 1 | 0.166666667 | 6 | 4 | 0 |
| 5155 | 4 | 3232286889 | 8 | 3 | 0.625 | 8 | 3 | 0 |
| 5692 | 1 | 3232267203 | 4 | 3 | 0.25 | 4 | 4 | 0 |
| 6219 | 5 | 3232236226 | 8 | 1 | 0.25 | 8 | 4 | 0 |
| 464 | 4 | 3232290372 | 7 | 1 | 0.142857143 | 7 | 4 | 0 |
| 4396 | 5 | 3232236261 | 8 | 1 | 0.25 | 8 | 4 | 0 |
| 5539 | 2 | 3232242539 | 8 | 2 | 0.125 | 8 | 5 | 0 |
| 8261 | 1 | 3232275577 | 2 | 1 | 0.5 | 2 | 2 | 0 |
| 1680 | 4 | 3232293113 | 8 | 1 | 0.25 | 8 | 3 | 0 |
| 765 | 4 | 3232257273 | 5 | 1 | 0 | 5 | 3 | 0 |
| 6898 | 2 | 3232300381 | 5 | 1 | 0 | 5 | 2 | 1 |
| 2424 | 2 | 3232287518 | 5 | 1 | 0 | 5 | 3 | 0 |
| 5274 | 2 | 3232243373 | 10 | 1 | 0 | 10 | 4 | 0 |
| 5015 | 4 | 3232286572 | 10 | 1 | 0.2 | 10 | 4 | 0 |
| 6372 | 3 | 3232294018 | 10 | 1 | 0.3 | 10 | 5 | 0 |
| 1179 | 5 | 3232295283 | 10 | 1 | 0.3 | 10 | 5 | 0 |
| 4523 | 5 | 3232284360 | 9 | 3 | 0.333333333 | 9 | 4 | 0 |
| 3366 | 2 | 3232248622 | 5 | 3 | 0 | 5 | 2 | 0 |
| 6348 | 3 | 3232274152 | 6 | 2 | 0.5 | 6 | 3 | 1 |
| 8603 | 3 | 3232288166 | 5 | 1 | 0 | 5 | 3 | 0 |
| 9208 | 2 | 3232297681 | 8 | 2 | 0.125 | 8 | 5 | 0 |
| 2548 | 1 | 3232271272 | 4 | 5 | 0.25 | 4 | 4 | 0 |
| 7793 | 4 | 3232255450 | 8 | 1 | 0.125 | 8 | 4 | 0 |
| 2039 | 1 | 3232279520 | 5 | 1 | 0.4 | 5 | 3 | 0 |
| 2736 | 3 | 3232239990 | 8 | 2 | 0.125 | 8 | 4 | 0 |
| 9102 | 2 | 3232284224 | 7 | 1 | 0.285714286 | 7 | 4 | 0 |
| 9923 | 4 | 3232292819 | 6 | 1 | 0.166666667 | 6 | 4 | 0 |
| 190 | 5 | 3232251171 | 4 | 1 | 0 | 4 | 2 | 0 |

### *4.1.1 Feature descriptions*

i. User_Action Count: This feature quantifies the total number of actions performed by a user, reflecting the user's activity level. For example, User_ID 5353 has performed 6 actions, while User_ID 5274 has 10 actions.

ii. IP_Frequency: This measures how often a particular IP address appears in the dataset, indicating the commonality of the IP usage. For instance, IP 3232275849 has a frequency of 2, suggesting limited reuse, whereas IP 3232271272 appears 5 times, indicating a more frequent access point.

iii. Action_Type_ratio: This ratio represents the proportion of a specific action type relative to the total actions performed by the user. Values range from 0 (no occurrence of that action type) to 0.5 or higher, showing the dominance of certain action types in user behavior. For example, User_ID 7219 has an action type ratio of 0.5, indicating half of their actions are of the specified type.

iv. User_Unique_IP: This denotes the number of unique IP addresses associated with a user, capturing the diversity of network access points. Values vary, with some users like 5539 having 5 unique IPs, while others like 8261 have only 2.

v. User_Unique_Action: This indicates the count of distinct action types performed by the user, reflecting behavioral diversity. Most users show diversity between 2 and 5 unique action types.

vi. Label: The binary target variable classifies instances as benign (0) or Sybil (1). In this sample, Sybil instances are rare, e.g., User_IDs 6898 and 6348 are labeled as Sybil.

### 4.1.2 Statistical Analysis of Generated Features

i. Descriptive Statistics:

a. User_Action Count: The mean action count is approximately 6.3, with a range from 2 to 10, indicating moderate variability in user activity levels.
b. IP_Frequency: Most IP addresses have low frequency (1 or 2), with a few outliers reaching up to 5, suggesting most users access the system from relatively few IPs.
c. Action_Type_ratio: The values are skewed towards lower ratios (many zeros), with some users exhibiting ratios as high as 0.75, showing that some users focus heavily on specific action types.
d. User_Unique_IP: The average number of unique IPs per user is around 3.5, indicating moderate network diversity.
e. User_Unique_Action: The average number of unique actions per user is about 3.5, reflecting a reasonable spread of behavior types.

ii. Correlation Analysis:

a. Positive correlation is expected between User_Action Count and User_Unique_Action, as more active users tend to perform a wider variety of actions.
b. IP_Frequency may negatively correlate with User_Unique_IP, since higher IP frequency implies repeated use of fewer IPs.
c. The Action_Type_ratio could inversely correlate with User_Unique_Action, as users focusing on fewer action types will have higher ratios for those actions.

iii. Class Distribution and Feature Differences:

a. The dataset is imbalanced with a majority of benign labels (0) and a few Sybil labels (1).
b. Preliminary observation suggests Sybil users (e.g., User_ID 6898, 6348) tend to have lower action type ratios and fewer unique IPs, which aligns with the hypothesis that Sybil accounts exhibit less diverse and more repetitive behavior.
c. For example, User_ID 6898 has an Action_Type_ratio of 0 and only 2 unique IPs, which may be indicative of Sybil behavior.

iv. Feature Distributions by Label:

a. User_Action Count: Benign users show a wide range, while Sybil users cluster around moderate counts.
b. IP_Frequency: Sybil users tend to have IPs with lower frequency, possibly due to IP hopping to avoid detection.
c. Action_Type_ratio: Sybil users often have lower ratios, indicating less focused or more uniform action patterns.
d. User_Unique_IP and User_Unique_Action: Sybil users generally have fewer unique IPs and actions, consistent with constrained behavior..

Balanced Dataset

The oversampling results are presented in Table 4.3. showing the class distribution before and after applying the SMOTE method. The data was balanced from 9,940 to 40,060 samples for the minority class samples. Table 4.4. shows the samples of the generated data.

Table 4.3: Balanced Data Statistics

| Label | Unbalanced Data | Balanced Data |
|---|---|---|
| Non-sybil | 40,060 | 40,060 |
| Sybil | 9,940 | 40,060 |
| TOTAL | 50,000 | 80,120 |

Table 4.4: Sample of Generated Balanced Data

| User_ID | Action_Type | IP_Address | User Action Count | IP Frequency | Action Type_ratio | User Unique_IP | User_Unique Action | Label |
|---|---|---|---|---|---|---|---|---|
| 6898 | 2 | 3232300381 | 5 | 1 | 0 | 5 | 2 | 1 |
| 6348 | 3 | 3232274152 | 6 | 2 | 0.5 | 6 | 3 | 1 |
| 2991 | 4 | 3232256947 | 8 | 1 | 0.25 | 8 | 5 | 1 |
| 4625 | 4 | 3232275770 | 6 | 1 | 0.166666667 | 6 | 5 | 1 |
| 8728 | 1 | 3232278294 | 10 | 2 | 0.4 | 10 | 5 | 1 |
| 3058 | 4 | 3232290612 | 3 | 1 | 0 | 3 | 1 | 1 |
| 162 | 3 | 3232242848 | 5 | 1 | 0.2 | 5 | 3 | 1 |
| 7575 | 4 | 3232285902 | 4 | 3 | 0 | 4 | 3 | 1 |
| 4824 | 4 | 3232235527 | 2 | 1 | 0.5 | 2 | 2 | 1 |
| 65 | 5 | 3232299196 | 7 | 1 | 0 | 7 | 4 | 1 |
| 5426 | 4 | 3232280550 | 6 | 1 | 0.333333333 | 6 | 4 | 1 |
| 2019 | 3 | 3232300914 | 5 | 2 | 0.4 | 5 | 3 | 1 |
| 390 | 5 | 3232295428 | 8 | 3 | 0.125 | 8 | 5 | 1 |
| 3539 | 2 | 3232298359 | 10 | 2 | 0.2 | 10 | 4 | 1 |
| 3083 | 4 | 3232243838 | 4 | 1 | 0.25 | 4 | 3 | 1 |
| 7219 | 1 | 3232275849 | 2 | 2 | 0.5 | 2 | 2 | 0 |
| 856 | 3 | 3232279682 | 6 | 2 | 0.333333333 | 6 | 4 | 0 |
| 5353 | 3 | 3232282638 | 6 | 1 | 0.166666667 | 6 | 4 | 0 |
| 5155 | 4 | 3232286889 | 8 | 3 | 0.625 | 8 | 3 | 0 |
| 5692 | 1 | 3232267203 | 4 | 3 | 0.25 | 4 | 4 | 0 |
| 6219 | 5 | 3232236226 | 8 | 1 | 0.25 | 8 | 4 | 0 |
| 464 | 4 | 3232290372 | 7 | 1 | 0.142857143 | 7 | 4 | 0 |
| 4396 | 5 | 3232236261 | 8 | 1 | 0.25 | 8 | 4 | 0 |
| 5539 | 2 | 3232242539 | 8 | 2 | 0.125 | 8 | 5 | 0 |
| 8261 | 1 | 3232275577 | 2 | 1 | 0.5 | 2 | 2 | 0 |
| 1680 | 4 | 3232293113 | 8 | 1 | 0.25 | 8 | 3 | 0 |
| 765 | 4 | 3232257273 | 5 | 1 | 0 | 5 | 3 | 0 |
| 2424 | 2 | 3232287518 | 5 | 1 | 0 | 5 | 3 | 0 |
| 5274 | 2 | 3232243373 | 10 | 1 | 0 | 10 | 4 | 0 |
| 5015 | 4 | 3232286572 | 10 | 1 | 0.2 | 10 | 4 | 0 |

## 4.2 Federated Learning Model Results

The Federated Learning Approach using two machine learning models (Random Forest and support Vector Machines) were trained and evaluated for three cases, which are;

i. Case 1 (Baseline): Raw features, imbalanced data
ii. Case 2: Engineered features, imbalanced data
iii. Case 3: Engineered features, balanced data

This is to evaluate the effect of feature engineering and data balancing on the sybil attack detection performance.

### 4.2.1. Confusion Matrix

Table 4.5 presents a comparative analysis of the confusion matrix results for Support Vector Machine (SVM) and Random Forest (RF) classifiers across three experimental cases. Each case represents a different combination of feature sets and data balance, allowing us to study how model performance evolves under varying conditions. The confusion matrix breaks down the results into True Positives (TP), False Negatives (FN), False Positives (FP), and True Negatives (TN), giving a detailed view of classification accuracy and error distribution.

Table 4.5: Confusion Matrix

| Model | Dataset | Case | TP | FN | FP | TN |
|---|---|---|---|---|---|---|
| SVM | Normal | Case1 | 0 | 2982 | 0 | 12018 |
| RF | Normal | Case1 | 118 | 2864 | 461 | 11557 |
| SVM | Engineered | Case2 | 3 | 2979 | 22 | 11996 |
| RF | Engineered | Case2 | 28 | 2954 | 169 | 11849 |
| SVM | Engineered & Balanced | Case3 | 7887 | 4131 | 2577 | 9441 |
| RF | Engineered & Balanced | Case3 | 9049 | 2969 | 194 | 11824 |

#### i. Case 1: Baseline – Raw Features, Imbalanced Data

In this case, models were trained using raw features without addressing the imbalance in the dataset. The SVM model failed to detect any true positives for the NORMAL class (TP = 0), classifying all positive samples incorrectly (FN = 2982), though it did correctly identify 12,018 true negatives. The RF classifier performed slightly better, detecting 118 true positives, but still had a high number of false negatives (2864) and 461 false positives, indicating difficulty in classifying the minority class under imbalanced conditions as shown in Figures 4.1 and 4.2.

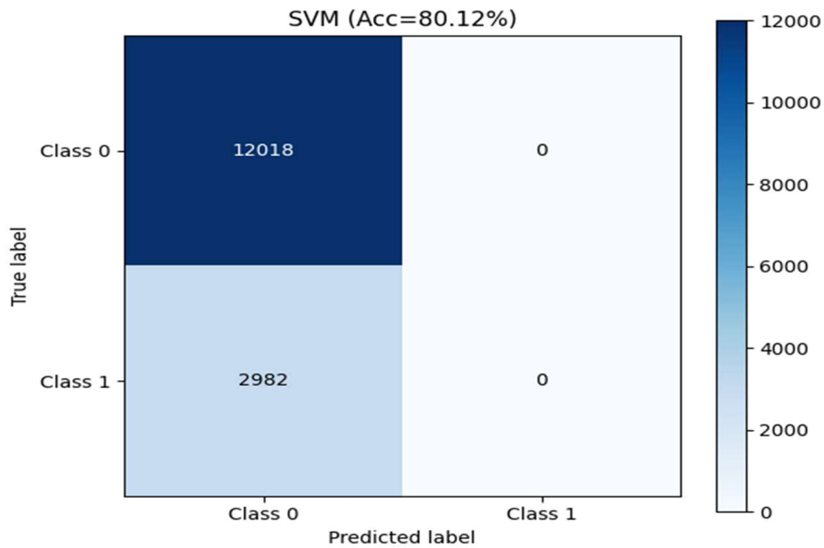


Figure 4.1: CFM for RF using the original dataset

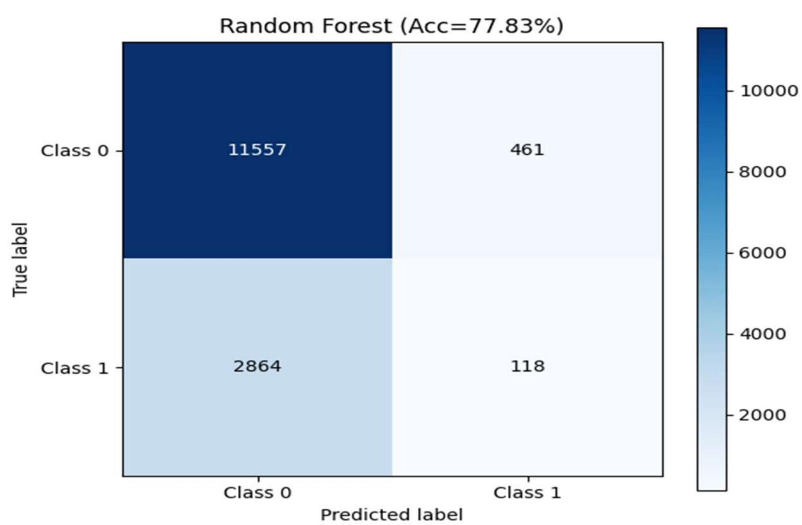


Figure 4.2: CFM for SVM using the Original Dataset

ii. Case 2: Engineered Features, Imbalanced Data

Feature engineering led to improved performance. For SVM, there was a modest gain, with 3 true positives and 22 false positives. RF showed further improvement, identifying 28 true positives with 169 false positives. However, both classifiers continued to struggle with high false negative counts (2979 for SVM, 2954 for RF), showing that while feature engineering helps, class imbalance still adversely affects classification, particularly recall as shown in Figures 4.3 and 4.4.

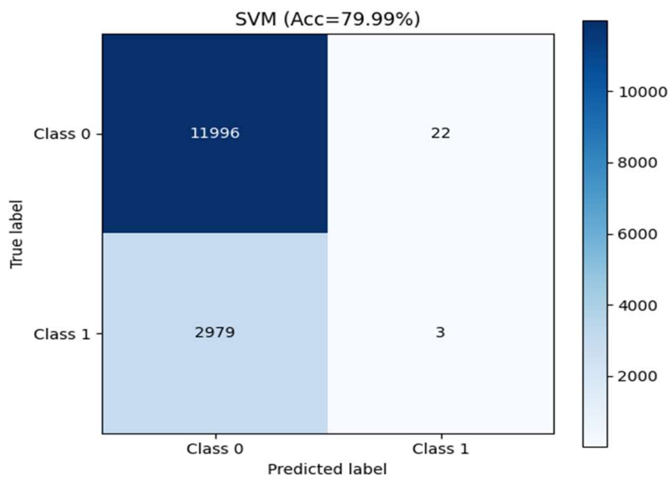


Figure 4.3: CFM for SVM using the Engineered Dataset

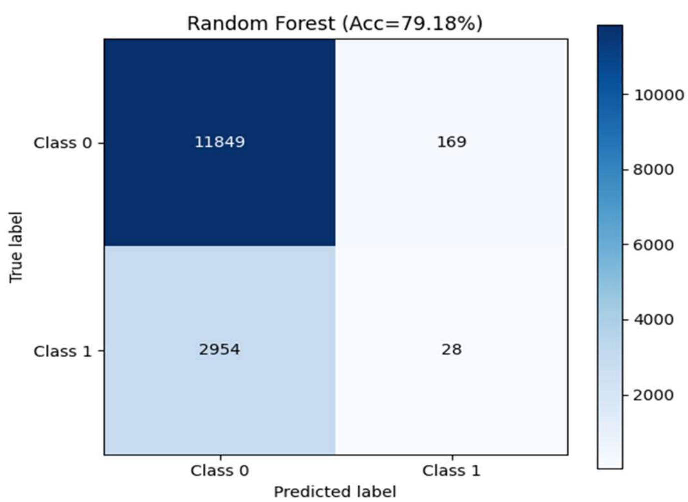


Figure 4.4: CFM for SVM using the Engineered Dataset

iii. Case 3: Engineered Features, Balanced Data

This configuration yielded the best results. With both feature engineering and class balancing applied, SVM significantly improved, correctly identifying 7887 instances of the NORMAL class, though it still misclassified 4131 instances (FN). False positives remained notable at 2577. In contrast, the RF classifier demonstrated superior performance, correctly classifying 9049 positives with only 2969 false negatives and just 194 false positives—substantially fewer errors compared to previous cases and the SVM model. It also achieved 11,824 true negatives, indicating balanced and accurate classification as shown in Figures 4.5 and 4.6.

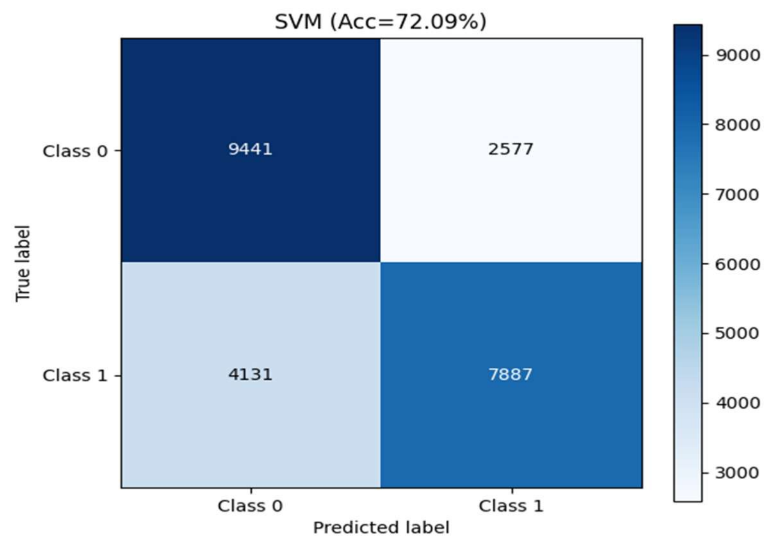


Figure 4.5: CFM for SVM using the Engineered and balanced Dataset

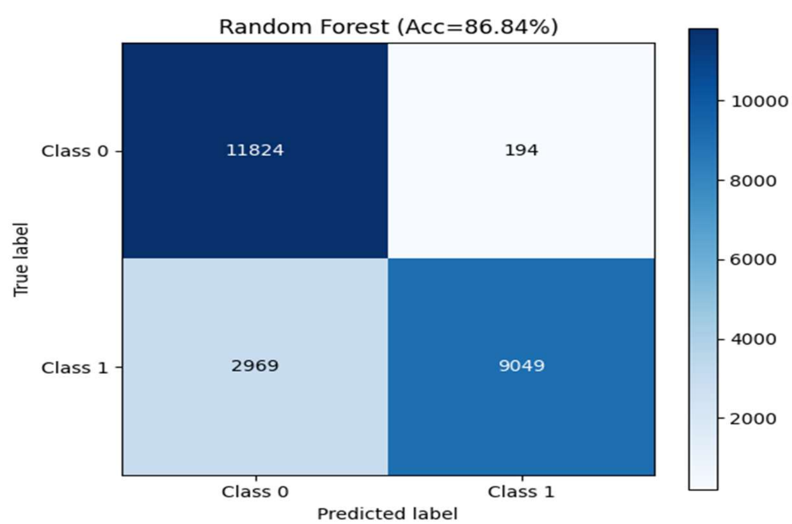


Figure 4.6: CFM for RF using the Engineered and balanced Dataset

The confusion matrix results clearly show that the combination of engineered features and data balancing (Case 3) leads to substantial gains in classification performance, particularly for the Random Forest model. RF outperformed SVM in every scenario, especially when both preprocessing steps were applied. The significant reduction in false negatives and false positives in Case 3 highlights the model's ability to generalize well across classes. These findings underscore the importance of proper feature selection and data preprocessing in improving model accuracy and reliability, particularly in scenarios involving class imbalance.

### 4.2.2 Model Performance Evaluation

The results in Table 4.6 and Table 4.7 summarizes the performance of the models in terms of the performance metrics.

#### 4.2.2.1 Case 1: Normal Dataset (Imbalanced)

In the first case, both SVM and Random Forest models were evaluated on a highly imbalanced dataset where the majority class dominates. The SVM model achieved an accuracy of 80%, which superficially appears strong; however, this is misleading because the model failed to identify any positive Sybil instances, resulting in zero precision, recall, and F1 score. This indicates that the SVM defaulted to predicting all instances as benign, effectively ignoring the minority class due to the overwhelming class imbalance. The Random Forest model showed a slightly lower accuracy of 78%, reflecting its attempts to identify Sybil instances. It achieved a precision of about 20%, meaning that only one in five positive predictions was correct, and a very low recall of approximately 4%, indicating it detected only a small fraction of the actual Sybil attacks. The F1 score remained low at 0.066, demonstrating poor overall performance in balancing false positives and false negatives. Overall, in this imbalanced setting, both models struggled to detect Sybil attacks effectively. While Random Forest showed some ability to identify positives, the high false positive rate and extremely low recall limited its practical utility. The SVM's complete failure to detect any positive cases underscores the critical challenge posed by class imbalance in cybersecurity detection tasks.

Table 4.6: SVM Performance

| Metric | Case 1 | Case 2 | Case 3 |
|---|---|---|---|
| Accuracy | 0.8 | 0.8 | 0.72 |
| Precision | 0 | 0.12 | 0.753727 |
| Recall | 0 | 0.001006 | 0.656266 |
| F1 Score | 0 | 0.001995 | 0.701628 |

Table 4.7: RF Performance

| Metric | Case 1 | Case 2 | Case 3 |
|---|---|---|---|
| Accuracy | 0.78 | 0.79 | 0.865 |
| Precision | 0.2038 | 0.142132 | 0.979011 |
| Recall | 0.039571 | 0.00939 | 0.752954 |
| F1 Score | 0.066274 | 0.017616 | 0.85123 |

4.2.2.2 Case 2: Engineered Dataset (Imbalanced with Feature Engineering)

In the second case, feature engineering was applied to the imbalanced dataset to improve model performance. The SVM model maintained an accuracy of 80%, similar to Case 1, but showed a slight improvement in precision (12%) and recall (0.1%). Despite this, the F1 score remained negligible at 0.002, indicating that the model's ability to detect Sybil attacks was still practically ineffective. The Random Forest model experienced a modest increase in accuracy to 79%, with precision rising to 14.2% and recall to just under 1%. The F1 score improved slightly to 0.018, reflecting marginal gains from feature engineering. However, these improvements were insufficient to overcome the inherent difficulties posed by the imbalanced data, as the models still missed the vast majority of Sybil instances.

This case highlights that while feature engineering can enhance detection capabilities, it is not enough on its own to address severe class imbalance. Both SVM and Random Forest models continued to struggle with low recall, emphasizing the need for additional strategies such as data balancing to improve Sybil attack detection.

4.2.2.3 Case 3: Engineered & Balanced Dataset

In the third case, the dataset was both engineered and balanced to address the class imbalance problem directly. The SVM model's accuracy decreased to 72%, reflecting a more realistic evaluation without majority class bias. However, precision significantly improved to 75.4%, and recall rose to 65.6%, resulting in a much stronger F1 score of 0.70. This demonstrates that balancing the dataset allowed the SVM to meaningfully detect Sybil attacks while maintaining a reasonable false positive rate. The Random Forest model excelled in this balanced scenario, achieving the highest accuracy of 86.5%. It delivered outstanding precision at 97.9%, indicating that nearly all positive predictions were correct, and a robust recall of 75.3%, capturing a substantial portion of Sybil attacks. The F1 score of 0.85 reflects an excellent balance between precision and recall, making Random Forest the superior model in this context.

This case underscores the critical importance of dataset balancing combined with feature engineering for effective Sybil attack detection. While SVM showed marked improvement, Random Forest's ensemble approach leveraged the balanced data most effectively, minimizing false positives and maximizing detection rates, which is essential for practical deployment in security systems.

Figure 4.7 compares the accuracy of the models for each case while figures 4.8 and 4.9 compares the f1-score, precision and recall of the models and cases.

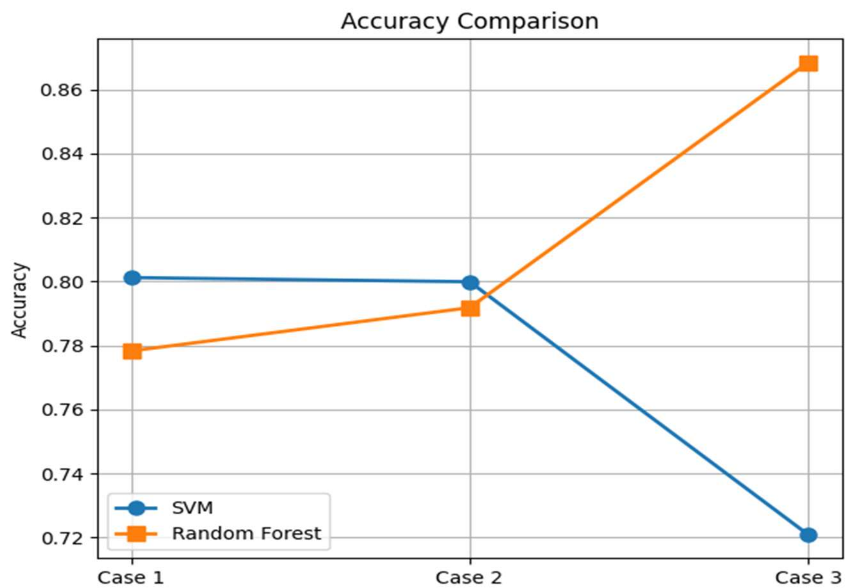


Figure 4.7: Accuracy Comparison

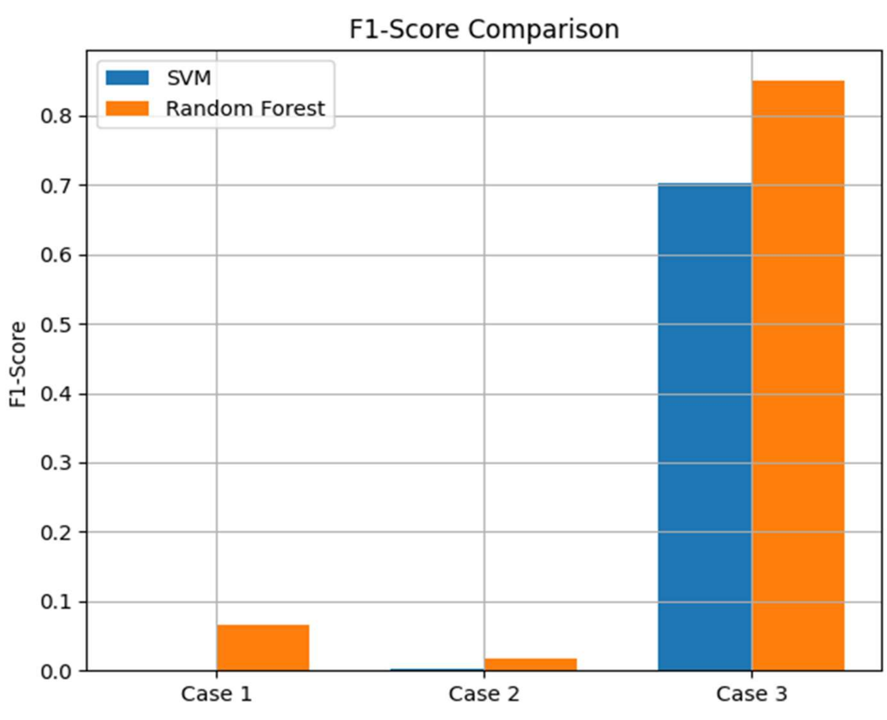


Figure 4.8: F1-score comparison

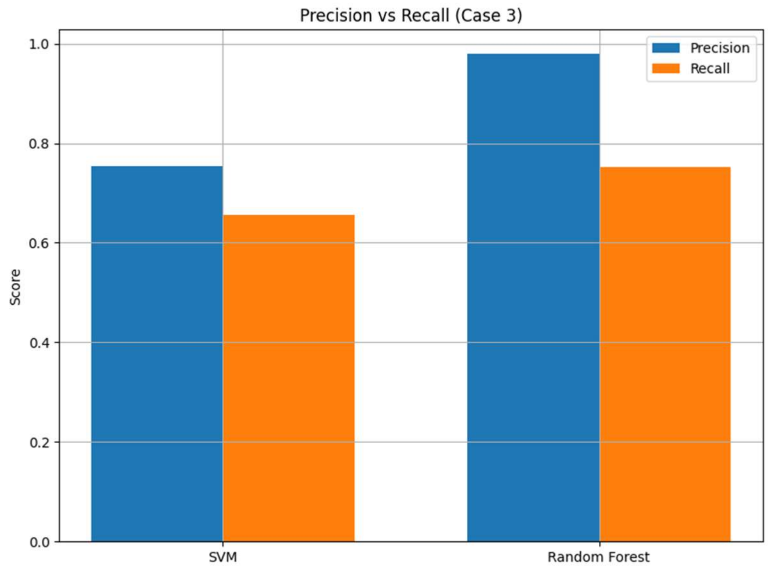


Figure 4.9: Precision and Recall Comparison,

## 5. CONCLUSION

The development and implementation of automated feature engineering algorithm generated extra and relevant features for the sybil attack simulation dataset. By integrating the SMOTE technique, the dataset was balanced to achieve better performance. The accuracy of sybil attack detection increased using Random Forest model excelled in this balanced scenario a high accuracy of 86.5%. It delivered outstanding precision at 97.9%, indicating that nearly all positive predictions were correct, and a robust recall of 75.3%, capturing a substantial portion of Sybil attacks. The F1 score of 0.85 reflects an excellent balance between precision and recall, making Random Forest the superior model in this context. Future research should focus on Improving the Federated Learning Technique with genetic algorithms for improved performance.

## REFERENCES

Albdour, L., Manaseer, S., & Sharieh, A. (2020). IoT crawler with behavior analyzer at fog layer for detecting malicious nodes. *International Journal of Communication Networks and Information Security*, *12*(1), 83–94. https://doi.org/10.17762/ijcnis.v12i1.4459

Hollmann, N., Müller, S., & Hutter, F. (2023). *Large Language Models for Automated Data Science: Introducing CAAFE for Context-Aware Automated Feature Engineering*. http://arxiv.org/abs/2305.03403

Li, S., Cheng, Y., Wang, W., Liu, Y., & Chen, T. (2020). *Learning to Detect Malicious Clients for Robust Federated Learning*. http://arxiv.org/abs/2002.00211

Muhammad Zunnurain Hussain, M. Z. H. (2024). *Sybil Attack Simulation Dataset*. IEEE Dataport.

Nargesian, F., Samulowitz, H., Khurana, U., Khalil, E. B., & Turaga, D. (2017). Learning Feature

Engineering for Classification. *International Joint Conferences on Artificial Intelligence (IJCAI)*, 2529–2535. https://www.ijcai.org/Proceedings/2017/0355.pdf

Prabhu, C. S. R. (2019). *Fog Computing, Deep Learning and Big Data Analytics- Research Directions*. Springer Nature Singapore.

Priyadarshini, R., & Barik, R. K. (2019). A Deep Learning Based Intelligent Framework to Mitigate DDoS Attack in Fog Environment. *Journal of King Saud University - Computer and Information Sciences*. https://doi.org/10.1016/j.jksuci.2019.04.010

Sihombing, D. J. C. (2024). Application of Feature Engineering Techniques and Machine Learning Algorithms for Property Price Prediction. *JITSI : Jurnal Ilmiah Teknologi Sistem Informasi*, *5*(2), 72–76. https://doi.org/10.62527/jitsi.5.2.241

Sultana, R., Grover, J., Tripathi, M., Sachdev, M. S., & Taneja, S. (2024). Detecting Sybil Attacks in VANET: Exploring Feature Diversity and Deep Learning Algorithms with Insights into Sybil Node Associations. *Journal of Network and Systems Management*, *32*(3), 51. https://doi.org/10.1007/s10922-024-09827-7

Yakubu, J., Abdulhamid, M., Atabo, H., & Haruna, C. (2019). Security challenges in fog - computing environment : a systematic appraisal of current developments. *Journal of Reliable Intelligent Environments*. https://doi.org/10.1007/s40860-019-00081-2