IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 26 June 2022, accepted 13 September 2022, date of publication 3 October 2022, date of current version 23 November 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3211313

APPLIED RESEARCH

Data Curation and Quality Evaluation for Machine Learning-Based Cyber Intrusion Detection

NGAN TRAN[1], HAIHUA CHEN[1], (Member, IEEE), JAY BHUYAN[2], AND JUNHUA DING[1]
[1]Department of Information Science, Denton, TX 76203, USA
[2]Department of Computer Science, Tuskegee University, Tuskegee, AL 36088, USA

Corresponding author: Junhua Ding (junhua.ding@unt.edu)

This work was supported in part by the National Science Foundation under Grant 1852249, and in part by the National Security Agency under Grant H98230-20-1-0417.

ABSTRACT Intrusion detection is an essential task for protecting the cyber environment from attacks. Many studies have proposed sophisticated models to detect intrusions from a large amount of data, yet they ignored the fact that poor data quality has a direct impact on the performance of intrusion detection systems. Examples of poor data quality include mislabeled, inaccurate, incomprehensive, irrelevant, inconsistent, duplicated, and overlapped data. In order to investigate how data quality may affect machine learning performance, we conducted a series of experiments on 11 host-based intrusion datasets using eight machine learning (ML) models and two pre-trained language models BERT and GPT-2. The experimental results showed: 1. BERT and GPT-2 outperformed the other models on every dataset. 2. Data duplications and overlaps in a dataset had different performance impacts on the pre-trained models and the classic ML models. The pre-trained models were less susceptible to duplicate and overlapped data than the classic ML models. 3. Removing overlaps and duplicates from training data with a normal range of sequence similarities could improve the pre-trained models' performances on most datasets. However, it may have adverse effects on model performance in datasets with highly similar sequences. 4. The reliability of model evaluation could be affected when testing data contains duplicates. 5. The overlapped rate between the normal class and the intrusion class seemed to have an inverse relationship to the performance of the pre-trained models in intrusion detection. Given the results, we proposed a framework for model selection and data quality assurance for building a high-quality machine learning-based intrusion detection system.

INDEX TERMS Data curation, data quality, intrusion detection, machine learning, deep learning, language model.

I. INTRODUCTION

The increasing usage of digital devices in a cyber-physical system has enhanced the efficiency of operating systems but has also led to security vulnerabilities. Cyber attacks on process control could lead to a significant control failure [1] and huge economic losses. Therefore, building an intrusion detection system (IDS) to predict and respond to attacks on a cyber-physical system has become an essential task among the cybersecurity community. However, it is a grand challenging task because of the range of novelty involved in cyber-attacks [1]. Recently, machine learning (ML) and deep learning (DL) approaches have been applied for intrusion detection at both the operation system level such as host-based intrusion detection system - HIDS and the network level such as network-based intrusion detection system - NIDS. As indicated by a Google Research [2], both models and data quality have a great impact on the performance of ML-based systems. The computing rule of ''garbage in, garbage out'' is still applicable to ML [3]. The lack of high-quality training data becomes a barrier to building high-performance ML systems [4]. Several data

The associate editor coordinating the review of this manuscript and approving it for publication was Shunfeng Cheng.

issues such as missing information, duplicates, lack of attack diversity, and data difficulty in intrusion datasets have been identified as challenges in ML-based IDS [5].

Recent studies in ML-based intrusion detection show immense efforts have been dedicated to model construction and optimization. For example, Sahu et al. compared the performance of various linear and non-linear classifiers for NIDS using the KDD dataset [6]. Al-Maksousy compared the performances of deep neural networks (DNN) and various traditional ML models in detecting network intrusions and found that DNN outperformed ML models in terms of accuracy, running time, and false positive rates [7]. Hu et al. proposed an incremental hidden Markov model (HMM) training framework that incorporates a simple data pre-processing method for identifying and removing similar sub-sequences of system calls for HIDS [8]. This training strategy has been widely applied since it can save on the training cost (especially on large data) without noticeable degradation in intrusion detection performance [8]. Convolutional neural network (CNN) and recurrent neural network (RNN) have also been used in HIDS [9], [10]. Recently, Liu and Lang conducted a comprehensive survey on ML and DL methods for IDS [11]. Focusing on model construction and optimization can only enhance the detection performance to a certain level as the core reason that hinders effective performance lies between data input quality and the model learning capabilities. However, very few studies have paid attention to data requirements, data quality issues, and data quality assurance in this specific research topic. It gives us the motivation to conduct an IDS study from a data-centric aspect to understand the impact of data quality on ML-based and DL-based IDS. The results would inspire future work to optimize both data quality and model capacity when developing an intrusion detection system.

With this goal in mind, we discussed nine data quality dimensions to systematically evaluate the quality of an intrusion dataset, and they are reputation, relevance, comprehensiveness, timeliness, variety, accuracy, consistency, duplication, and overlap. We proposed a data pre-processing and cleaning framework for HIDS tasks. Based on the framework, we conducted a series of experiments to investigate the impacts of duplications and overlaps on the conventional ML models and the pre-trained language models. The experiments were conducted on 11 host-based intrusion datasets collected by the University of New Mexico (UNM), the Massachusetts Institute of Technology (MIT), and the Australian Defence Force Academy (ADFA). The chosen conventional ML models, which are K-Means (KM), Logistic Regression (LR), Support Vector Machine (SVM), Neural Network (NN), Decision Tree (DT), Random Forest (RF), K Nearest Neighbors (KNN), Naïve Bayes (NB), are commonly used in IDS studies. The pre-trained language models such as BERT and GPT-2 are selected to represent the highly sophisticated and complex models. By choosing models at different complexity levels, we can thoroughly verify and evaluate the effects of data quality on the performances of ML-based and DL-based intrusion detection systems. From the experiment results, we discussed the data quality impacts on the detection performance and provided a guideline to ensure high quality in intrusion datasets based on nine data quality dimensions. The study is designed to answer four research questions below.

- RQ1: What is the performance difference between the pre-trained language models and the traditional ML models in the host-based intrusion detection tasks?
- RQ2: What are the impacts of the training data quality on the traditional ML performances versus the pre-trained language model performances?
- RQ3: What are the impacts of the testing data quality on the traditional ML performances versus the pre-trained language model performances?
- RQ4: How to evaluate the data quality of an intrusion detection dataset?

To the best of our knowledge, this is the first study that explores how to build a highly effective intrusion detection system from a data-centric rather than a model-centric perspective. The research results offer IDS researchers and practitioners with new insights and guidelines on improving intrusion detection performance. The rest of the paper is structured as follows. Section 2 presents the literature review related to machine learning-based IDS and data quality. Section 3 discusses the data preparation and quality requirements for intrusion detection. Section 4 introduces the proposed data cleaning and pre-processing method for host-based intrusion detection along with the experimental design. Section 5 presents the experimental results. Section 6 discusses the experiment results and implications. Finally, section 7 concludes the paper with a summary of our findings and potential directions for future work. The code and datasets used in this research are available on GitHub.[1]

II. RELATED WORK

A. MACHINE LEARNING-BASED INTRUSION DETECTION SYSTEMS

An intrusion detection system aims to detect anomalous activities or intrusions (active or passive attacks) in a cyber-physical system using traces of system events (HIDS) or network packets (NIDS). There are three approaches to intrusion detection: misused-based, anomaly-based and hybrid-based. In the misused-based approach, a ML model is trained with existing attack signatures, so any activities that match this pattern are flagged as intrusions [12], [13]. However, as the model is only trained with known attack signatures, the IDS is exposed to zero-day attacks, which leads to high false negative rate. In the anomaly-based approach, a large amount of normal activities are collected to train the model, and any instances that deviate from the normal profile are considered anomalous, or in this case, an intrusion [14], [15]. The advantage of this approach is the model's ability in detecting zero-day attacks, yet it requires a large amount of

[1] https://github.com/NganTran-0017/HIDS

normal activities or patterns to sufficiently construct a normal profile. Moreover, the anomaly-based approach is susceptible to high false alarm rate due to a wide variations of normal activities. Lastly, the hybrid approach is a combination of the misused-based and the anomaly-based approaches, where a ML model is trained with both normal patterns and intrusion patterns [7], [9], [10], [16], [17]. This approach inherits the advantages of both approaches above by leveraging intrusion signatures to increase known attack detection rate, and the collected normal patterns to lower false negative rate and at the same time, increase zero-day attack detection rate. By training the models with intrusion patterns and normal patterns, this approach does not require a relatively large amount of data, compared to the other two approaches. Therefore, we will apply the hybrid-based approach to our experiment in Section IV.

The input of ML-based IDS is a collection of instances consisting of one feature or more features. The features in an IDS can be system events in HIDS or network packets in NIDS. The instances can be treated as individual incidents or sequential incidents, and labeled as normal or anomalous. Additionally, the availability and sufficiency of labeled data are the key to determining the learning technique to used for building the IDS. In supervised learning, a model is trained on labeled data, and the quality and amount of labeled data are critical to the learning process [16]. In semi-supervised learning, a training dataset only contains a few labeled data and the rest of the training data are unlabeled instances. Since this approach uses only a fraction of labeled data, it reduces the cost of data labeling [16]. On the other hand, the unsupervised learning approach does not require labeled data, yet this method requires model tuning, and it often suffers from high false positives [18] and inferior results to those produced by supervised method [11]. Noisy labels, insufficient labeled data, and class imbalance are the most common data quality issues in supervised and semi-supervised learning-based IDS. Other data quality issues, such as inconsistency, duplication, incompleteness, incomprehension, imprecision, or lack of variety may also exist in ML input data.

We summarized the existing IDS studies in Table 1, including the selected learning techniques, models, and datasets, along with the type of IDS as NIDS or HIDS, publication year, and the references. We can see that the majority of existing studies focus on NIDS task. More datasets have been created for NIDS and as a result, more machine learning algorithms have been explored on the task. However, compared to NIDS, HIDS is more of a challenge due to: (1) more labeled data is required to reduce the false positive alarm rate; (2) it is difficult to design an efficient HIDS which can prevent outgoing denial-of-service attacks; and (3), in a shared system environment, the HIDS needs to work as an independent module since the shared parameters may cause the attack [51]. Nowadays, HIDS are becoming more important and play a major role in most of the intrusion detection systems [51]. Even though some studies have been conducted on HIDS [9], [10], [26], [33], [43], [49], more attention should be paid to HIDS. The lack of sophisticated model applications in HIDS along with the rise of powerful pre-trained language models such as BERT [52] and GPT [53] have inspired us to use both conventional ML models and pre-trained language models in our study.

B. DATASETS FOR INTRUSION DETECTION

NIDS datasets mainly include information from the packet itself, which is used to detect the malicious traffic traversing through the network, while datasets for HIDS usually include information about executed system events/logs or system calls to detect vulnerabilities of a target application or anomalous activities in an operating system [10]. We conducted an investigation in the popular datasets used for NIDS and HIDS tasks, as shown in Table 2.

The following properties are generally required when creating an IDS dataset. (1) Normal user behaviors. The quality of an IDS is primarily determined by its attack detection rate and false alarm rate. Therefore, the presence of normal user behavior is indispensable for evaluating an IDS [67]. (2) Intrusion behaviors. Any set of actions that are taken to exploit the system vulnerabilities with an intend of disrupting, destroying, spying, disabling or stealing a cyber-physical system is considered an intrusion behavior. There are various attack signatures resulted from different scenarios; hence, it is essential to clarify the attack types or names in an IDS dataset to enable a construction of multi-class intrusion detection systems. (3) Format. An IDS dataset can be in different formats, such as packet-based, flow-based, host-based log files, etc. (4) Anonymity. Some of the information may need to be anonymized due to privacy and security concerns, and this property indicates the affected attributes. (5) Duration. The monitored time (e.g., daytime vs. night or weekday vs. weekend) of the dataset should be indicated since time may play a critical role in the nature of the behavior. For example, a behavior might be regarded as an attack only when it occurs in a specific duration and/or at a specific time/day of the week. (6) Labeled. Labeled datasets are necessary for training and validating supervised learning and semi-supervised learning models. (7) Other information, such as attack scenarios, network structure, system information, IP addresses, recording environment, and download URL are also useful. Data quality issues can appear in the above information, which could have serious impacts on the intrusion detection performance. Despite the fact that data quality issues, such as duplication and imbalance, have been reported in the KDD dataset [57], only a few studies have included the qualities of IDS datasets in their discussions [58], [60].

C. DATA QUALITY EVALUATION AND ASSURANCE FOR MACHINE LEARNING

Poor data quality has a direct impact on the performance of the ML system that is built on flawed data. For example, a gender classification system from facial images was implemented with ML algorithms and produced a 0.8% error in lighter-skinned males; yet, it went as high as 34.7% error

TABLE 1. Machine learning techniques for IDS. Full names and abbreviations of the models are introduced in Appendix.

Technique used	Model	Datasets	HIDS/NIDS	Year	Reference
Supervised	KNN, SR	KDD-Cup99	NIDS	2017	[19]
Supervised	LR, RF	UNSW	NIDS	2017	[20]
Supervised	DNN, imbalanced network traffic, RF, VAE	CIDDS-001	NIDS	2018	[21]
Supervised	DNN	NSL-KDD	NIDS	2018	[17]
Supervised	DNN	NSL-KDD	NIDS	2018	[22]
Supervised	LR, NB, KNN, DT, AdaBoost, RF, CNN, CNN-LSTM, LSTM, GRU, SimpleRNN, DNN	CICIDS-2017, UNSW-NB15, ICS cyberattack	NIDS	2018	[23]
Unsupervised	Metric learning + clustering + SVM	Kyoto 2006, NSL-KDD	NIDS	2019	[24]
Supervised	NB, AODE, RBFN, MLP, J48 DT	UNSW-NB15	NIDS	2019	[25]
Supervised	Pruned exact linear time, quantile regression forests	NetFlow data	NIDS	2020	[26]
Unsupervised	Autoencoder, IF, KNN, K-Means, SCH, SVM	NSL-KDD, ISCX	NIDS	2020	[27]
Unsupervised	IF, HBOS, CBLOF, K-Means	BRO DNS, BRO CONN	NIDS	2020	[28]
Supervised	DNN	KDD-Cup99, NSL-KDD, UNSW-NB15	NIDS	2020	[29]
Supervised	KNN, RF, SVM-rbf, DNN, ResNet-50, one-vs-all classifier, multiclass classifier	NSL-KDD	NIDS	2020	[30]
Supervised	NB, DT, RF, ANN	KDD-Cup99	NIDS	2020	[31]
Supervised	NN, DT, linear discriminate analysis with the bagging algorithm	NSL-KDD	NIDS	2019	[32]
Supervised	SVM, MLP, NB, DT	ADFA-LD	HIDS	2017	[33]
Supervised	NN	NSL-KDD	NIDS	2021	[34]
Supervised	XGB-DNN	NSL-KDD	NIDS	2020	[35]
Supervised	BAT	NSL-KDD	NIDS	2020	[36]
Supervised	SGD, RF, LR, SVM, SM	NSL-KDD	NIDS	2019	[37]
Supervised	RNN-IDS	NSL-KDD	NIDS	2017	[38]
Supervised	CNN	NSL-KDD	NIDS	2017	[39]
Supervised	LGBM	NSL-KDD, UNSW-NB15, CICIDS-2017	NIDS	2021	[40]
Supervised	GAN-FS, NB, DT, RF, GBDT, SVM, KNN, NN	NSL-KDD, UNSW-NB15, CICIDS-2017	NIDS	2021	[41]
Unsupervised	S-NDAE, DBN,	NSL-KDD	NIDS	2020	[42]
Supervised	CNN	NGIDS-DS, ADFA-LD	HIDS	2017	[9]
Semi-Supervised	SC4ID	ADFA-LD, UNM	HIDS	2018	[43]
Supervised	GRU, LSTM, CNN+GRU	ADFA-LD	HIDS	2019	[10]
Supervised	ELM	KDDCup99, UNM, ADFA-LD	HIDS	2016	[44]
Supervised	KNN, NN	ADFA-LD	HIDS	2021	[45]
Supervised	Cycle-GAN	ADFA-LD	HIDS	2018	[46]
Supervised	RBF-NN	UNM	HIDS	2009	[47]
Supervised	DT, KNN, MLP, MNB, RF, SVM	ADFA-LD	HIDS	2020	[48]
Supervised	LR, SVM, DT, RF, ANN	DS2OS traffic traces	HIDS	2019	[26]
Supervised	NB, LR, KNN, SVM, IntruDTree	Kaggle cybersecurity datasets	HIDS	2020	[49]
Supervised	SVM, LR, NB, KNN, NN	NSL-KDD, ADFA	HIDS and NIDS	2020	[50]

in darker-skinned females [68]. The problem was due to a significant imbalance of different skin colors in the training data [68]. Recently, a study showed that ten of the most commonly-used computer vision, natural language, and audio datasets had serious data quality issues [69]. The computing rule of ''garbage in, garbage out'' is also applicable to ML-based anomaly detection. However, there is not much research on low-quality data and its impact on ML-based anomaly detection.

Noisy labels are one of the most serious data quality issues. To investigate the impact of the data labeling on the performance of ML-based NIDS, Lauría and Tayi compared the performances of two classifiers DT and NB, which were trained on poor quality data [70]. The experiments showed that data with high-quality labels may not be required to further train a classifier that has already performed at an acceptable level [70]. Class imbalance is another common issue in IDS. As pointed out by [6], both of the CICIDS and KDD datasets are imbalanced and not uniform. Particularly, the two most dominant classes in KDD have more than 40,000 instances while the 16 minor classes have less than 1,000 instances [6]. Their experiments demonstrated that balancing data could not help improve the model performance in nonuniform training datasets, but the data balancing could

TABLE 2. Overview of public datasets for IDS.

Dataset	Volume	Information	Format	Labeled	Balanced	Year
		NIDS				
KDD-Cup99 [54]	4.9 millions for training, 2 million for testing	41 features, 20 types of attacks	packet, logs	yes	no	1999
Kyoto 2006 [55]	3,054,682 for training, 1,563,923 for testing	14 features derived from the KDD-Cup99 and 10 additional features	packet, logs	yes	yes	2006
DARPA-2009 [56]	673,931 records for training and 74,880 records for testing	16 network features and 26 packet features; 7,000 pcap files	packets	yes	no	2009
NSL-KDD [57]	125,973 for training, 22,544 for testing	41 features, 22 types of attacks	packet, logs	yes	no	2009
ISCX [58]	30,814 normal and 15,375 attack traces for training, 13,154 normal and 6,580 attack traces for testing	1.5 million network traffic packets, with 20 features and covers seven days of network activity	packets	yes	no	2012
UNSW-NB15 [59]	175,341 records for training, 82,332 records for testing	49 features in pcap file format and 9 categories of attacks	packets	yes	no	2015
NGIDS-DS [60]	63,185,992 records for training and 34,987,493 records for testing	88,791,734 records for benign and 1,262,426 records for malicious activities. 7 features for ground-truth cs; 9 features for the 99 csv files of host logs; 18 features for NGIDS.pcap	packet, logs	yes	no	2016
CICIDS2017 [61]	75,561 records for training and 25,187 records for testing	3,119,345 instances and 83 features containing 15 class labels	packets	yes	no	2017
CSE-CIC-IDS2018 [61]	16,000,000 data instances in total	The number of features range from 79 to 83, containing 7 class labels	packets	yes	no	2018
		HIDS				
DARPA 98/99 [62]	4,898,431 records for training and 2,984,154 records for testing	97,277 normal and 311,744 intrusion traces with 41 features and classes labeled as either normal or any of the 22 types of attacks	packet, logs	yes	no	1998
UNM dataset [63]	627 system-call sequences for training and 3,136 system-call sequences for testing	4,298 normal traces and 1,001 intrusion traces and 467 features	logs	yes	no	1998
KDD99 [54]	494,021 for training and 311,029 for testing	1,033,372 normal and 4,176,086 attack traces; 41 features	packets	yes	no	1999
NSL-KDD [57]	1,152,281 distinct records from KDD99: 860,725 normal and 291,556 attack traces	41 features per record	packets	yes	no	2000
ADFA-LD [64]	833 traces and 308,077 system calls for training, 4,373 traces and 2,122,085 system calls for testing	746 trace attack sequences and 317,388 system call attack sequences		yes	no	2014
ADFA-WD [64]	355 traces and 13,504,419 system calls for training, 1,827 traces and 117,918,735 system calls for testing	5,542 trace attack sequences and 74,202,804 system call attack sequences		yes	no	2014
DARPA-2009 [56]	673,931 records for training and 74,880 records for testing	16 network features and 26 packet features; 7,000 pcap files	packets	yes	no	2009
ADFA-IDS [65]	308,077 system calls and 833 traces for training, 212,2085 system calls and 4,372 traces for testing	15 attack types and 36,636 malicious system call traces	logs	yes	no	2013
NGIDS-DS [60]	313,926 records with 7 attributes in ground-truth csv file; 1,262,426 attack and 88,791,734 normal with 9 attributes; 1,094,231 capture packets with 18 unique IPs in NGIDS.pcap file	cyber normal and abnormal traffic scenarios for different enterprises	packet, host logs	yes	no	2017
DS2OS traffic traces [66]	61.52 MB	traces captured in the IoT environment DS2OS for different services	–	yes	yes	2018
Kaggle cybersecurity datasets [49]	25,000 instances	3 qualitative features and 38 quantitative features	–	yes	yes	2020

improve a neural network performance in minor classes [6]. Oversampling and undersampling have been used to handle class imbalance in IDS [71], [72], [73]. Missing information, duplicate data, lack of attack diversity, dataset difficulty, and feature sparsity are other factors that can impact the performance of ML-based IDS [5], [6], [74].

As a result, different quality dimensions have been proposed to measure the data quality for ML systems. Fan [75]

argued that data consistency, data de-duplication, information completeness, data currency, and data accuracy are central to data quality. These dimensions are related to the data itself. The dimensions that are related to users include accessibility, integrability, interpretability, rectifiability, relevance, and timeliness [76]. Recently, Chen et al. defined ''data quality'' as data metrics for fitting the purpose of building an ML system [4]. Dimensions such as comprehensiveness, correctness, and variety, are critical to the evaluation of the data quality for ML systems [4].

We found that most studies focused on optimizing detection performance from a model perspective, and the common choices of models were conventional ML models and neural network-based models, not pre-trained models. Furthermore, the literature on data quality revealed that data quality has a great impact on ML performance. Yet, only a few studies focus on improving ML performance from a data perspective, much less on intrusion detection tasks. In light of these findings, we designed this study to investigate the impacts of data quality on both ML models and pre-trained models in HIDS tasks and then, provide a guideline to ensure high quality in intrusion datasets using our proposed framework along with nine data quality dimensions that we gathered from the literature.

III. DATA PREPARATION AND QUALITY REQUIREMENTS FOR INTRUSION DETECTION

As discussed above, data quality is crucial for ML-based IDS. However, each stage of the data preparation can be plagued with problems of data quality, such as scattered presence of the data source, insufficient data during data collection, label noise during the data annotation, and overlapping issues during the training-testing data partition. Therefore, it is necessary to identify the dataset attributes, and then develop a guideline for quality assurance during data preparation.

A. DATA PREPARATION WORKFLOW

Data preparation for ML-based IDS mainly includes four steps: (1) Selecting one or multiple data sources. (2) Collecting the data from the selected data source. Data can be collected from one source or integrated from multiple sources. (3) Labeling the data for training and testing. (4) Pre-processing the data as the model input. The workflow is shown in Figure 1. The data sources for HIDS and NIDS are different: data for HIDS can be collected from audit records, log files, the application program interface (API), rule patterns, and system calls, while data for NIDS is usually collected from the Simple Network Management Protocol (SNMP), network packets (TCP/UDP/ICMP), Management Information Base (MIB), and router NetFlow records. Once the data source is confirmed, additional information should be collected, such as metadata, format, duration, etc., for further analysis. The data collection process should be well-designed. For example, in HIDS data, sequential system events should be correctly organized by the execution order. Data labeling is an essential step for supervised ML-based IDS. Most ML algorithms make the assumption that training data are accurate. However, mislabeled data in training samples do exist and can impact the predictive accuracy of ML-based classifiers [70]. Data pre-processing involves removing outliers and redundant data, extracting useful features, and data partition for training and testing. If it is handled improperly, it can cause data quality issues such as sparsity, bias, and overlaps between training and testing data.

B. DATASET PROPERTIES AND ATTRIBUTES

Certain properties of a dataset should be collected and evaluated beforehand based on a specific scenario [67]. We believe that to construct a high-quality IDS dataset, the following information should be collected during the data curation.

1) GENERAL INFORMATION

General information of a dataset should include the year of creation, public availability, metadata, format, and data volume. Both network traffic and system logs face concept drift issue over time, and new attacks might appear in any scenario. A ML model built in 1990 might not be useful to predict the data in 2022. Therefore, the year of creation (age) of an intrusion detection dataset is critical for determining the scope of an IDS. Intrusion detection datasets should be publicly available to serve as a basis for comparing different intrusion detection methods and for quality evaluation by third parties [67]. Metadata, such as network structure, hosts, IP addresses, configuration of the clients, and attack scenarios, can provide users content-related explanations of the results. IDS datasets can be roughly divided into three formats: (1) packet, (2) flow, and (3) logs. The processing for different data formats varies. The format is directly associated with the volume of data. Data volume can be described by the number of packets, flows, points, and instances, which is essential to model selection in ML.

2) DURATION

Duration refers to the timestamps when data was collected. For example, the MIT lpr dataset includes real-world normal usage of the lpr command (a Linux command for submitting files for printing) for two weeks using 77 hosts, while the ISCX dataset includes seven days of network activities (normal and malicious) from the Information Security Center of Excellence (ISCX) at the University of New Brunswick. As mentioned in [77], a long duration trace is needed when detecting the cyclostationary traffic signal (i.e., differences in traffic between daytime/nighttime or weekdays/weekends).

3) CONTEXT

Context is information of the environment where the data were captured, and it gives us a deeper understanding of the operating systems and the processes. In NIDS, context information delineates the network environment and conditions, in which the data were captured. For example, traffic type and network type indicate the context information in an NIDS dataset. In HIDS, the context information is the information

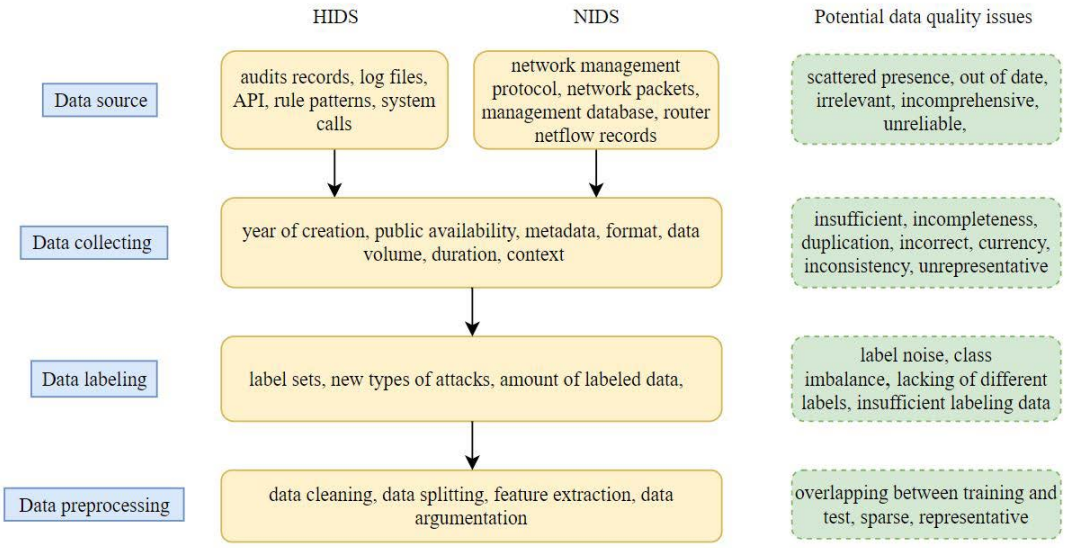

FIGURE 1. Data preparation workflow for machine learning-based IDS.

of the monitored system. For example, the context of the ADFA-LD dataset lies in the system information, which is an Ubuntu Linux system version 11.04.

4) NORMAL TRACES AND ATTACK TYPES IN INTRUSION TRACES

This information is related to data labeling, which is the foundation of ML-based IDS. The training/validation/testing data must be correctly labeled as normal or not. In an attack record, the type of attacks should be recorded also [77]. Maciá-Fernández et al. applied the following strategy to label an instance: a) an attack label for the flows that they positively know correspond to an attack, b) a normal label for those that are generated synthetically based on normal patterns, and c) a background label for the instances no one knows exactly if they are attacks or not [77]. Ring et al. summarized the specific types of attacks used in different datasets [67].

5) FEATURES

The input of ML-based IDS is a collection of instances consisting of features. As shown in Table 2, most of the datasets include the feature information, which can be qualitative or quantitative features, network features or packet features, n-gram features, and others.

C. DATASET QUALITY MEASUREMENT

Different principles, metrics, and dimensions have been proposed to measure data quality [5], [75], [78], [79], [80]. However, only a few of them were discussed in the context of building ML systems [4] and especially intrusion detection systems [5]. By analyzing the literature, we identified nine quality dimensions or attributes that are central to data quality to be investigated in this research. They are reputation, relevance, comprehensiveness, timeliness, variety, accuracy, consistency, duplication and overlap. The duplication and overlap of instances in datasets are particularly related to the training and validation of machine learning models.

1) REPUTATION

"Reputation" is related to reliability, credibility, and trustworthiness. Reputation is evaluated using an information-theoretic concept such as the Kullback-Leibler distance. For datasets from a single source, one can use a collective measure of trustworthiness (in the sense of reliability) based on the referrals or ratings from members in a community of practice to evaluate the reputation [81]. PageRank is another method for reputation measurement [82]. For datasets integrated from multiple sources, an "opinion" (a metric indicating the degree of belief) can be gathered to represent the uncertainty in the aggregated result.

2) RELEVANCE

"Relevance" indicates why the data is collected. Data should be collected and evaluated for "fit for purpose" [4]. For example, HIDS data should be collected from the host system and audit sources, such as operating systems, window server logs, firewall logs, application system audits, or database logs. While NIDS data should be extracted from a network through packet capture or sniffer, NetFlow, etc. Moreover, if the IDS targets a specific type of attacks, then the collected data must be relevant to this type of attack.

3) COMPREHENSIVENESS

"Comprehensiveness" requires a dataset to be a representative sample of a population [4], [5]. Existing ML-based IDSs frequently suffer from the issues of bias and lacking of robustness, which are mainly caused by data incomprehensiveness. For example, the NSL-KDD dataset includes 39 attack types, which are further categorized into the following four

categories: DoS, R2L, U2R, and probe. Suppose that all attack types should have an instance in the training set. However, 17 of these attacks, which are not in the training set, are in the testing set. This dataset cannot then be considered as comprehensive as it does not represent the entire population. The importance of comprehensive data to ML, and especially to DL, is well-understood as a DL model contains millions of parameters and it requires a large amount of data for training. If a comprehensive dataset with various types of attacks can be developed for intrusion detection like the ImageNet dataset for computer vision, it would greatly enhance the detection performance.

4) TIMELINESS

''Timeliness'' (also called ''currency'') refers to the extent to which the age of the data [78] is appropriate for the task. Timeliness is an important factor that can affect the performance of ML models since new attacks are emerging constantly and the existing datasets such as DARPA and KDD99 are too old to reflect them. For that reason, evaluating ML-based IDS using out-of-date datasets like DARPA and KDD99 does not offer a true evaluation of current attack trends and can result in inaccurate claims for their effectiveness [83]. Ideally, an IDS dataset should include most of the common attacks in current environments [11].

5) VARIETY

''Variety'' concerns the degree of data coverage in the selected features, and it is considered as a subset of comprehensiveness in constructing a ML-based IDS. For example, the selected features in the KDD-Cup99 dataset are supposed to have a normal distribution, otherwise it would induce the data sparsity issue. Moreover, to improve the robustness in ML models, validation data and testing data should be heterogeneous for a thorough model evaluation.

6) ACCURACY

According to Wang and Strong [78], ''accuracy'' means ''the extent to which data are correct, reliable and certified.'' Accuracy can be defined in syntactic and semantic aspects [11]. Syntactic accuracy aims to check how close a data instance is to the elements of the corresponding defined feature, while semantic accuracy requires an instance to be semantically represented appropriately. Labeling accuracy means that an instance should be correctly labeled as normal or a specific attack type. For ML systems, labeling accuracy plays a critical role in detection performance as inaccurate labeling can cause data poisoning, which exposes detection systems to poisoning attacks [72].

7) CONSISTENCY

Data ''consistency'' refers to the validity and integrity of data, which represents real-world samples [75]. It aims to detect errors, such as inconsistencies and conflicts in the data, and it is typically identified as violations of data dependencies [75]. For example, the system calls in HIDS data should be grouped by its process ID to ensure the integrity of each trace and sorted by the execution order to ensure the sequential order within a trace. The value of an attribute's domain (feature) should also be constrained to the range of admissible values. For instance, the range of a system call number is between 1 and 137 in the Inetd dataset that was collected by the UNM, so there should be no system call number that is greater than 137 in the dataset.

8) DUPLICATION

Data duplication happens when the same data instance occurs multiple times in a dataset. It is a longstanding issue that has been studied for decades [4], [5]. Duplication is usually caused by ''data from a multiple data sources that was not fused appropriately.'' Particularly, the CICIDS2017 dataset is known to have duplicate data [5], [84]. Having duplications in a dataset can exacerbate the class imbalance issue, especially when duplicate records are from major classes. Additionally, failing to remove duplications from a testing dataset can result in a misleading high performance. Hence, we propose a data cleaning method to remove duplications within each class in Section IV-B.

9) OVERLAP

Overlapped data instances between different classes pose an issue to ML performance [73], [85], [86]. Overlaps can occur among different classes or among the training and testing data. When the same data instance appears in different classes (in this case, the normal class and the intrusion class), it causes overlaps in the decision boundaries and hence, confuses the models. If a model is trained to perfectly identify the overlapped decision boundaries, then it is overfitted. It is a common issue in NIDS studies, where the original feature dimensions are reduced through feature selection and, inadvertently, the reduced feature dimensions create overlaps between different classes and duplications within the same class. Overlapped decision boundaries can lead to high false positive rate (FPR) and high false negative rate (FNR). As a result, high performances from many NIDS studies are often misleading [73]. Moreover, having a large overlap between the training and the testing data can potentially introduce bias to the model, which leads to inaccurate model evaluation, as pointed out by Chen et al. [4]. One solution to the issue is to remove the overlapped instances from either class. A similar approach was also adopted in [85], where the overlapped data in the training set were removed to generate a smoother decision boundary and avoid model overfitting.

IV. METHODOLOGY

In this section, we will first describe the workflow and the datasets used in this study. We will then detail our pre-processing method and propose a data cleaning method to remove the overlaps and duplicates from the dataset. After that, we will describe ten learning models including eight conventional ML models and two pre-trained

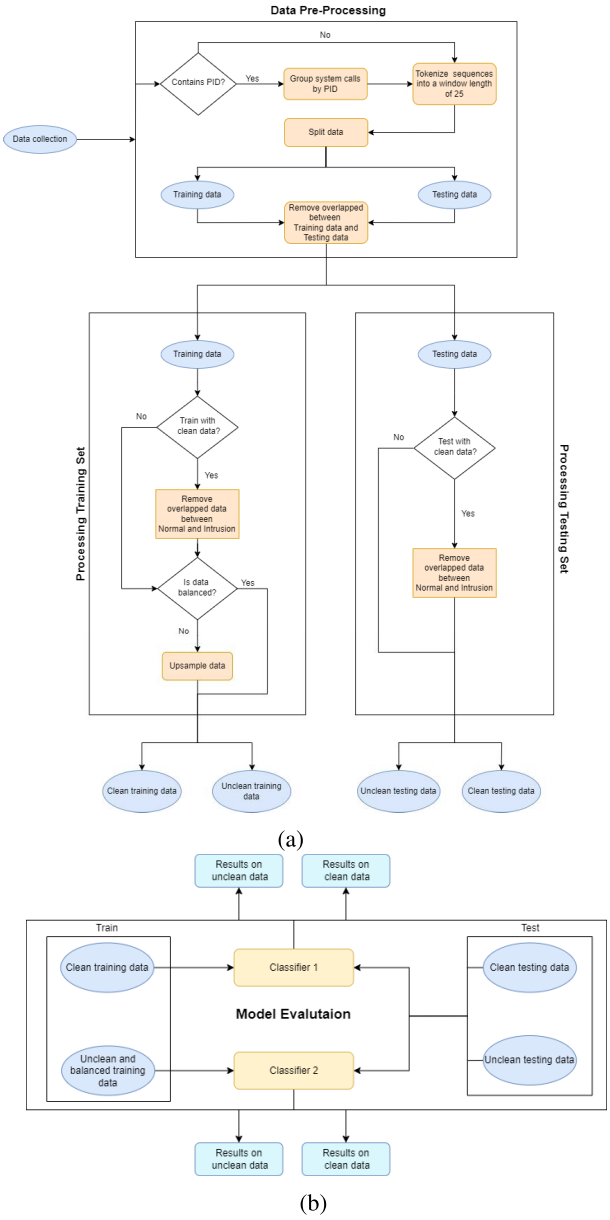

FIGURE 2. Experiment workflow (a) Data processing (b) Model evaluation.

language models. Lastly, we will describe four evaluation metrics that are used for model evaluation and comparison.

The experimental workflow is shown in Figure 2. The workflow begins with data pre-processing, which ensures that each trace of system calls are consistent and in sequential order. In this step, we grouped the system calls by PID, then tokenized and parsed each trace of system calls into smaller sequences (or fragments), and partitioned the data into the training and testing sets in a 70-30 ratio.

The next step is to design several training sets and testing sets with different quality levels for the purpose of validating the effect of data quality on model performance. The quality level of each dataset (training set and testing set) is either clean or unclean. In a clean set of data, we get rid of overlaps and duplicates using our data cleaning method, and in an unclean set of data, we leave the data as is, so there may be duplicates and overlaps. The difference in the data processing procedure for a training set versus a testing set is an extra step of data imbalance handling in the training data. The reason is we do not want data imbalance to affect the model learning capabilities, yet we want a reliable model evaluation in the testing data.

To reduce the effect of class imbalance, we apply either oversampling or downsampling technique depending on the size of the training data. Oversampling is applied to randomly bootstrap data from a minor class when the training size is below 500,000 sequences. Otherwise, downsampling is applied to randomly lower the size of a major class. In the end, the training dataset will have the same amount of data in each class. At the end of the data pre-processing step (Fig. 2a), we will have four sets of data: a clean training dataset, an unclean training dataset, a clean testing dataset, and an unclean testing dataset.

This workflow is set up to conduct two experiments, in which each experiment contains two scenarios. Experiment one is designed to answer RQ2 by exploring the performance variations of the pre-trained models and the classic ML models when they are trained with clean data versus with unclean data. In the first scenario, all models are trained with clean data and, in the second scenario, all models are trained with unclean data. The models from both scenarios are then evaluated with clean testing data and compared against each other. Experiment two is designed to answer RQ3 by exploring the performance variations of the pre-trained models and the ML models when they are evaluated with clean data versus with unclean data. In this experiment, all models are trained with clean data, then in the first scenario, the trained models are evaluated with clean testing data, and in the second scenario, the trained models are evaluated with unclean testing data. The performances from both scenarios are compared against each other. Following the same framework, we ran experiments with ten candidate models, listed in section IV-C, on 11 datasets collected by the ADFA, MIT, and UNM.

A. DATASETS

The 11 datasets include: ADFA-LD, MIT lpr, Inetd (an Internet service daemon in Linux), Live lpr, Live Named, Login and Ps, Stide, Synthetic ftp, Synthetic lpr, Synthetic Sendmail, and Xlock (a linux command that locks the server until the user enters the password). The first dataset was collected by the ADFA, and the last nine datasets were collected by the UNM.

The ADFA-LD[2] dataset was collected on a Ubuntu operating system version 11.04 in 2013 by the University of New South Wales at the Australian Defence Force Academy. The dataset contains three folders: training, validation, and testing. The training and validation folders contain normal traces, whereas the testing folder contains only intrusion

[2]https://research.unsw.edu.au/projects/adfa-ids-datasets

traces. In this dataset, each trace contains the system calls number that were executed from a single process.

The MIT Live lpr[3] dataset was collected on a SunOS version 4.1.4 by the Massachusetts Institute of Technology (MIT) in 1997. The rest of the datasets were collected from specific Linux programs, such as sendmail, lpr, login and ps, xlock, etc. by UNM.[4] Instead of having each trace stored separately like the ADFA-LD dataset, the MIT lpr and all of the UNM datasets contain various traces in the format of a process ID (PID) and the system call number in each text file. The difference in the data format between the ADFA-LD dataset and the other datasets explains the essential role of grouping system calls by PID in our data pre-processing step (section IV-B).

B. DATA PRE-PROCESSING AND CLEANING

Data pre-processing includes grouping system calls by PID, tokenizing and parsing traces of system calls into smaller sequences and finally, data partition. Since processes are executed concurrently, data needs to be pre-processed to ensure that each trace is in a sequential order and belongs to only one process. This step applies specifically to the data from the UNM and MIT, where multiple traces are stored together in the form of the PIDs and system call numbers. Hence, we added a conditional statement to check whether the data contains a PID column. If it does, then we re-organize the traces by grouping the system calls by their PID. As a result, the data from the UNM and MIT will have the same format as the ADFA-LD data.

As traces of system calls have various lengths from a few hundreds to hundreds of thousands, we tokenized and parsed each trace into 25-grams sequences to create a consistent data pool. The parsed sequences have the same label as the entire trace. Hence, instead of using an entire trace, we use the parsed sequences (or fragments) of each trace to train the models. Additionally, training with extremely large traces can be expensive in terms of time and resources. Therefore, parsing traces into smaller sequences allows us to train and test various models in a timely manner.

The last step of data pre-processing is data partition, where data is split into a training set and a testing set in a 70-30 ratio. This is where duplication and overlap issues start occurring.

As a trace is parsed into smaller sequences, the chances of duplicating and overlapping data between the normal class and the intrusion class are also increased. Duplicate sequences are created when a sequence occurs multiple times in a class. An overlapped sequence is created when a sequence appears in both classes. It happens when an intrusion trace starts out as a normal trace and performs anomalous tasks later on, or when an intrusion trace imitates the normal patterns and performs anomalous activities in between the normal patterns.

When the data pool contains duplicate sequences, the partitioned training and testing sets will be overlapping each other. As a result, model evaluation becomes unreliable because the model already knows the labels of some sequences during training. When the data pool contains overlapped sequences, the same sequence with different labels could exist in both the training data and the testing data. It may confuse the model and compromise the model's performance.

We propose a data cleaning method to remove duplicate sequences and overlapped sequences from the dataset. First, training data and testing data are condensed into two sets. This step removes any duplication within each set. Then, we remove the overlapped sequences, which exist in the training set, from the testing set. It is completed by taking the difference between the testing set and the training set, as shown in (1), where A represents the testing data, and B represents the training data.

After data partition, a training dataset most likely contains duplicate sequences within each class and overlapped sequences between both classes. Therefore, we use the same technique described above to remove these sequences from a training dataset and a testing dataset. The only difference is that in (1), A represents the intrusion class, and B represents the normal class. The clean training data and clean testing data are stored separately from the original unclean training data and unclean testing data. When a fragment of a system call trace exists in both classes, and its behavior has not deviated from the normal pattern, then the fragment is still considered normal; though, it may be part of a longer anomaly sequence.

$$set(CleanA) = set(A) - set(B) \tag{1}$$

C. MACHINE LEARNING ALGORITHMS

This study implements eight commonly used ML models for HIDS: K-means (KM), Logistic Regression (LR), Support Vector Machine (SVM), Neural Network (NN), Decision Tree (DT), Random Forest (RF), K Nearest Neighbor (KNN), and Naïve Bayes (NB). Their configurations are described as follows:

- KM: We choose the number of clusters as two because there are two categories in the datasets: normal labeled as 0 and intrusion labeled as 1. In the end, a cluster that resembles a normal profile should contain only normal sequences, and a cluster that resembles an intrusion should contain only intrusion sequences.
- LR: All parameters are set as the default.
- SVM: To accelerate the training process, we randomly sample a smaller amount of data for training if the dataset is too large. Specifically, we sample 5% of the training data if the size is over 500,000, 20% if the size is between 100,000 and 500,000, and 100% otherwise.
- NN: We set the input size as 25, corresponding to the parsed sequence length, and it is comprised of two layers: a fully connected layer with the number of nodes the same as the sequence length, a ReLU activation

[3]https://www.cs.unm.edu/ immsec/data/live-lpr.html
[4]https://www.cs.unm.edu/ immsec/systemcalls.htm

function, and a sigmoid output layer with two output nodes, where each one represents the probability of a sequence belonging to each class.
- DT: The minimum samples per split is ten and samples per leaf is five.
- RF: Other settings are the same as DT. The only difference is that RF bootstraps samples to build 100 trees.
- KNN: We choose three as the number of neighbors. The weights parameter is set to be uniform, where all observations in each neighborhood are weighted equally regardless of their distances.
- NB: All parameters are set as the default.

In addition, we implement two deep learning models, Bidirectional Encoder Representations from Transformers (BERT) and Generative Pre-trained Transformers (GPT), which are proven state-of-the-art models for sequential data. As for BERT, we use the BERT Base version, which contains 110 million parameters. The sequences are passed to the BERT tokenizer (bert-base-uncased), where each system call number is tokenized and converted into token IDs. Each sequence is padded with [CLS] and [SEP] at the beginning and the end of the sequence. The output of BERT is passed to a single feed-forward network with 768 neurons and then a sigmoid activation function to generate the probability of an intrusion. For GPT, we use the smaller GPT-2 model for the experiment. For both models, when the data size is less than 2,000 training sequences, we train the model with a batch size of 16 and three to five epochs. However, when the data size is between 500,000 and 2 million, between 2 million to 6 million, and over 10 million, we train the model with 10% of the data, 5% of the data, and 1% of the data, respectively. In these situations, we set the batch size as 32 and the number of epochs as two.

D. EVALUATION METRICS

We use four evaluation metrics to measure the model performances:

- Macro F1 score: The F1 score is a harmonic mean between recall and precision. A model can achieve a high F1 score only when it has high recall and high precision. Data imbalance is a common issue in HIDS datasets; unlike weighted average, macro average is not affected by class size when measuring model performance; instead, it measures all classes equally. Therefore, the macro F1 score is a great indicator to measure a model performance regardless of the class size.
- AUC score: measures the area under the ROC curve, which indicates the model's ability in distinguishing between the normal class and the intrusion class in comparison to random classification. Hence, the higher the AUC score is, the better the model.
- False positive rate (FPR): measures how well a model can recognize a normal (negative) class through the number of normal instances that are incorrectly classified as an intrusion. An IDS with a high FPR indicates that it has not yet learned the characteristics of the normal class, and it can set back the process of finding actual threats.
- False negative rate (FNR:) measures how well a model can recognize an intrusion (positive) class through the number of intrusion instances that are misclassified as normal. An IDS with a high FNR can be detrimental since an unidentified threat can cause more damage than a false alarm.

The performance of a model is the best when it achieves the highest macro F1 and AUC scores with the lowest FPR and FNR.

V. EXPERIMENTAL DESIGN

In this section, we develop the training data and the testing data for the experiments.

A. TRAINING DATASET DESIGN

After processing training data, two subsets are produced: one is the clean training data and the other is the unclean training data. The differences between them are the presence of duplicate sequences and overlapped sequences between two classes.

Table 3 presents the training data from 11 datasets. The table records four features before and after cleaning, such as size, overlapped rate, normal duplication rate, and intrusion duplication rate. After cleaning, there is no overlap between the intrusion class and the normal class, and there is no duplication in each class.

The train size column details the number of training sequences before and after cleaning. There is a notable difference in the training size before and after removing duplicate and overlapped sequences. Before cleaning, Stide has the largest training size (over 11 million), whereas Inetd has the smallest training size (around 6,200). After cleaning, ADFA-LD has the largest training size (over 336,000), whereas Inetd has the smallest (346). On average, a training size is reduced by almost 90% after cleaning. A high overlapped rate or high duplication rate within each class can contribute to the size reduction in training data after cleaning. The dataset that loses most of its data after cleaning is Stide, with a reduced rate of 99.99%. In contrast, the dataset that loses the least amount of training data is ADFA-LD, with a reduced rate of 23.27%.

The overlapped rate column details a percentage of overlaps between the normal class and the intrusion class in a training dataset. It is calculated as the percentage of Jaccard index in equation 2. The Jaccard index or Jaccard similarity coefficient is used to measure the similarity between two data samples by calculating the intersection proportion out of the union of both sets. Data samples with high similarity yield a higher Jaccard index. Based on Table 3, the overlapped rate of all the datasets are between 0% and 38.9%, with the MIT Live lpr, ADFA-LD, Xlock, and Live Named datasets having almost no overlap (less than 1.5%) while Inetd and Login and

Ps datasets have the highest overlapped rate (above 34%). An overlapped rate of 34% indicates 34% of the fragments are the same in both classes in the training dataset. On the other hand, the fragments from each class are very distinctive in the MIT Live lpr, ADFA-LD, Xlock, and Live Named datasets.

$$Overlapped\ Rate = \frac{set(intrusion) \cap set(normal)}{set(intrusion) \cup set(normal)} * 100\% \quad (2)$$

$$Duplication\ Rate = 1 - \frac{set(data)}{data} * 100\% \quad (3)$$

We define the ''normal duplication'' rate as the percentage of duplicate sequences within the normal class, and the ''intrusion duplication'' rate as the percentage of duplication sequences within the intrusion class. The duplication rate within each class can be calculated following equation 3. With this definition, Stide has the highest normal duplication rate (99.97%), whereas ADFA-LD has the lowest rate (41.41%). Synthetic lpr, MIT lpr, and Live lpr have the highest duplication rate in the intrusion class (around 99.70%), whereas ADFA-LD has the lowest rate (5.31%). Based on these statistics, the ADFA-LD dataset has a very high quality with a low overlapped rate between both classes and a low duplication rate within each class.

B. TESTING DATASET DESIGN

After processing the testing data, two types of datasets will be produced: one is the clean testing data and the other one is the unclean testing data. A difference between the processing step of the testing data and the training data is that there is no countermeasure to the class imbalance in the testing data. Therefore, the class distribution in the unclean testing data remains the same as in the original data. On the other hand, the class distribution in the clean testing data is altered as we remove the overlaps and duplications in both classes.

Table 4 includes four attributes: testing size, overlapped rate, normal duplication rate, and intrusion duplication rate before and after cleaning the testing datasets. After cleaning, the overlapped rate, normal duplication rate, and intrusion duplication rate are 0% on all of the datasets.

The test size column details the number of sequences in each testing dataset before and after cleaning. ADFA-LD has the biggest testing size, whereas Synthetic lpr has the smallest testing size before and after cleaning. Overall, the difference in the testing sizes before and after cleaning is not as remarkable as the difference in the training sizes. On average, a testing size is reduced by 11% after removing duplicates and overlaps. After cleaning, ADFA-LD has the lowest reduction rate in its testing size (0.72%), whereas Synthetic Sendmail has the highest rate reduction rate (21.43%).

The overlapped rate column details the percentage of overlaps between the normal class and the intrusion class, before and after cleaning. Overall, the overlapped rates between two classes in the testing datasets is lower than that of the training datasets. There are seven datasets with no overlap to 0.07% of overlap: Live lpr, MIT Live lpr, Synthetic lpr, Synthetic Sendmail, Xlock, Live Named, and ADFA-LD. On the other hand, Inetd has the highest overlapped rate, which is 11.26%.

The duplication rate column details the percentage of duplicate sequences within each class in each testing dataset. The duplication rate within each class is lower in the testing data compared to the training data. Live lpr has the highest duplication rate in the normal class (24.71%), whereas Inetd and ADFA-LD have less than 1.70% duplication. Xlock and Synthetic Sendmail have the highest duplication rate in the intrusion class (above 29%), whereas MIT Live lpr and ADFA-LD have almost no duplication (below 0.2%). Based on these statistics, the clean testing data and unclean testing data from the ADFA-LD dataset are very similar to each other as it has low overlapped and duplication rates.

For the second experiment, we created 19 sets of unclean testing data, whose duplication percentage ranges from 5% to 95% with an increment of 5% each, to evaluate the models with different duplication percentages. To create a testing set with a particular duplication percentage, we removed the same data proportion from the clean testing dataset accordingly to each class size, and then bootstrapped the data up to the original clean testing size. The amount of the bootstrapped data is the total percentage of duplication in both classes.

C. EXPERIMENT FRAMEWORK

The use of the training and testing data at different data quality levels for answering RQ2 and RQ3 is shown in Table 5. We use the clean training data and the unclean training data to train the models separately for the second research question. To ensure a fair evaluation, we evaluate these models using clean testing data and compare the model learning capabilities through their performance evaluations. For the third research question, we train the models with clean data and then evaluate them with testing data, which contain an increasing percentage of duplication from 0 (clean testing data) to 95 (extremely unclean testing data). The goal is to see how reliable the model performances can be as they are evaluated with an increasing percentage of duplication versus with clean data. As each model behaves differently on the same dataset, we analyze the performance variation in a group of traditional ML models such as DT and RF and a group of pre-trained language models like BERT and GPT.

A higher macro F1 score and AUC score with lower FPR and FNR indicate a better performance. Comparing to the benchmark results from Table 6, if the difference in each metric is lower than 0.05, then we assume that there is no difference in that performance metric. The benchmark results are from the models that were trained and tested with clean data. A model performance is considered as improved when at least two of the performance metrics are improved, which are higher macro F1 score and AUC score and lower FPR and FNR. For example, comparing to when trained with unclean data, if a model trained with clean data has an increase of 0.06 in the macro F1 score, 0.05 in the AUC score, and

TABLE 3. Training data characteristics.

Index	Dataset	Train Size		Overlapped Rate		Normal Duplication Rate		Intrusion Duplication Rate	
		Unclean	Clean	Unclean	Clean	Unclean	Clean	Unclean	Clean
1	ADFA-LD	437,825	336,020	0.21%	0.00%	41.41%	0.00%	5.31%	0.00%
2	Inetd	6,238	346	34.39%	0.00%	55.67%	0.00%	94.93%	0.00%
3	Live lpr	502,297	2,152	10.87%	0.00%	99.47%	0.00%	99.71%	0.00%
4	Live Named	6,462,660	26,563	1.32%	0.00%	99.59%	0.00%	47.30%	0.00%
5	Login and Ps	18,801	2,136	38.90%	0.00%	87.98%	0.00%	79.43%	0.00%
6	MIT lpr	2,164,086	3,806	0.00%	0.00%	99.83%	0.00%	99.72%	0.00%
7	Stide	11,076,920	3,268	8.72%	0.00%	99.97%	0.00%	99.48%	0.00%
8	Synthetic Ftp	127,174	2,347	8.30%	0.00%	98.46%	0.00%	37.11%	0.00%
9	Synthetic lpr	116,641	618	28.54%	0.00%	81.95%	0.00%	99.73%	0.00%
10	Synthetic Sendmail	1,264,563	3,301	23.66%	0.00%	99.78%	0.00%	71.95%	0.00%
11	Xlock	238,088	17,582	0.44%	0.00%	92.75%	0.00%	31.18%	0.00%

TABLE 4. Testing data characteristics.

Index	Dataset	Test Size		Overlapped Rate		Normal Duplication Rate		Intrusion Duplication Rate	
		Unclean	Clean	Unclean	Clean	Unclean	Clean	Unclean	Clean
1	ADFA-LD	137,855	136,869	0.01%	0.00%	1.68%	0.00%	0.18%	0.00%
2	Inetd	170	151	11.26%	0.00%	0.00%	0.00%	1.82%	0.00%
3	Live lpr	209	168	0.00%	0.00%	24.71%	0.00%	6.98%	0.00%
4	Live Named	5,722	5,122	0.02%	0.00%	10.06%	0.00%	23.43%	0.00%
5	Login and Ps	505	425	1.18%	0.00%	10.76%	0.00%	16.72%	0.00%
6	MIT lpr	373	342	0.00%	0.00%	9.45%	0.00%	0.00%	0.00%
7	Stide	619	571	0.53%	0.00%	7.12%	0.00%	7.97%	0.00%
8	Synthetic Ftp	538	493	1.22%	0.00%	9.14%	0.00%	2.58%	0.00%
9	Synthetic lpr	65	61	0.00%	0.00%	12.50%	0.00%	6.67%	0.00%
10	Synthetic Sendmail	602	473	0.00%	0.00%	17.35%	0.00%	29.05%	0.00%
11	Xlock	4,659	4,363	0.00%	0.00%	5.60%	0.00%	34.75%	0.00%

TABLE 5. Experiment design.

Train	Test	Models	Research Question
Clean	Clean	All	2
Unclean	Clean	All	2
Clean	Clean	All	3
Clean	Unclean	All	3

0.02 in the FNR, yet a decrease of 0.08 in the FPR, that model performance is considered as improved.

VI. EXPERIMENTAL RESULTS

In this section, we present the overall result, and the results for answering research questions 2 and 3.

A. OVERALL RESULTS

We use the evaluation results from models that are trained and tested with clean data as a benchmark to compare with the results from the experiments below. We only include the benchmark macro F1 scores in Table 6. The benchmark AUC score, FPR, and FNR are also consistent with macro F1 score.

1) MODEL PERFORMANCE

Overall, BERT and GPT generate the highest macro F1 scores compared to the other models on all of the datasets. Based on Table 6, both performances are almost perfect (above 0.93) on the MIT lpr, ADFA-LD, Xlock, and Synthetic lpr datasets, whose testing sets barely have any overlap between both classes. However, their performances are not always remarkable. The lowest macro F1 score that BERT achieves is 0.58 on the Login and Ps dataset. The lowest macro F1 score that GPT achieves is 0.53 on the Live lpr and Login and Ps datasets. Among the ML models, DT and RF achieve the highest macro F1 scores. DT and RF achieve the highest macro F1 of 0.85 and 0.92 respectively on both ADFA-LD and MIT lpr. On the other hand, KM generates the lowest macro F1, which is 0.1 on Xlock.

We look further into the performances of different models on the other evaluation metrics: AUC score, FPR, and FNR. The findings are described as follows:

- Regarding the AUC score, BERT and GPT outperform the traditional models on all datasets except Live

Named. GPT performs worse than the traditional models on Live lpr. The pre-trained models yield the highest AUC scores on the MIT lpr, ADFA-LD, Xlock, and Synthetic lpr datasets. However, their AUC scores are not so good (between 0.55 and 0.66) on Live lpr, and Login and Ps. Compared to each other, BERT achieves higher AUC on five datasets (ADFA-LD, Inetd, Live lpr, Live Named, and Synthetic lpr), whereas GPT achieves higher AUC on four datasets (Login and Ps, Stide, Synthetic Ftp, and Synthetic Sendmail). Among the conventional ML models, DT and RF yield the highest AUC scores. Their highest scores are 0.87 (DT on Xlock) and 0.92 (RF on ADFA-LD), and their lowest scores are 0.4 on Login and Ps.

- Regarding the FPR, the pre-trained models outperform the traditional models on all datasets, except BERT on Login and Ps. The pre-trained models yield the lowest FPR (below 0.01) on Live lpr, Live Named, MIT lpr, and Xlock. The highest FPR that BERT and GPT yield are 0.45 on Login and Ps and 0.22 on Inetd respectively. Based on their performances on the 11 datasets, GPT yields similar to lower FPR than BERT. Compared to each other, BERT achieves a lower FPR on two datasets (ADFA-LD and Inetd), whereas GPT achieves a lower FPR on four datasets (Live lpr, Login and Ps, Stide, and Synthetic Ftp). Among the conventional ML models, DT and RF achieve the lowest FPR. Their lowest FPR are 0 (RF on MIT lpr and Xlock) and 0.01 (DT on Xlock). Their highest FPR are 0.52 and 0.49 respectively on Login and Ps.
- Regarding FNR, the pre-trained models outperform the traditional models on seven out of the 11 datasets (except on Inetd, Live lpr, Live Named, and Synthetic Ftp). They achieve the lowest FNR on MIT lpr (both achieve 0.00), ADFA-LD (both achieve 0.01), Synthetic lpr (both achieve FNR lower than 0.09), and Xlock (both achieve 0.16 FNR). Compared to each other, BERT achieves a lower FNR on four datasets, whereas GPT achieves a lower FNR on three datasets. Among the conventional ML models, NN achieves lower and more stable FNR on most datasets. However, the lowest FNR are from DT and RF on ADFA-LD (0.12 and 0.08 respectively) compared to the other ML models.

The above findings demonstrate that the pre-trained language models outperform the traditional ML models on most datasets. Compared to each other, BERT achieves higher macro F1 scores, higher AUC scores, and lower FNRs. However, BERT tends to yield higher FPRs. On the other hand, GPT does not always generate high macro F1 scores, AUC scores, and low FNRs, but it yields lower FPRs than BERT on most datasets.

Among the traditional ML models, RF outperforms the others regarding the macro F1 score, AUC score, FPR, and FNR on six of the datasets. RF achieves the highest performance on the ADFA-LD, MIT lpr, and Xlock datasets and the lowest on the Login and Ps dataset. Additionally, DT performs similarly to RF, and it outperforms the other models on four of the datasets. Therefore, we will analyze their performances on behalf of the traditional ML models. Additionally, we will analyze the performances of BERT and GPT-2 on behalf of the pre-trained models.

2) DATASET PERFORMANCE

Based on Table 6, the model performances are inconsistent on the 11 datasets. There are some datasets where the pre-trained models and the traditional ML perform very well, yet there are some datasets where both groups of models perform very poorly. Hence, we compare the overall performances of all datasets through the median performance across all models to identify the datasets with the highest and the lowest overall performances. We use median instead of mean because it is not affected by the extreme values from the performances of the pre-trained models and the classic ML models.

Among the datasets, ADFA-LD and Xlock generate the highest median macro F1 score (0.71) and AUC score (0.75) across all of the models. On the other hand, Login and Ps generates the lowest median macro F1 score (0.45) and AUC score (0.46) across all of the models. In terms of FPR, Live Named and Xlock generate the lowest median rate (0.025), whereas Inetd and Login and Ps generate the highest median rate (0.56 and 0.51, respectively). Regarding FNR, ADFA-LD and Xlock generate the lowest median rate (0.24), whereas Live lpr generates the highest rate (0.73).

Overall, BERT, GPT, RF, and DT yield very high performances on the ADFA-LD, Xlock, and MIT lpr dataset. On the other hand, both groups of models perform poorly on the Login and Ps dataset. Regardless of its complexity, a model does not perform consistently on all datasets. The differences between ADFA-LD, Xlock, and Login and Ps are their training sizes, overlapped rates, and duplication rates, which are different dimensions in data quality.

3) OVERLAPPED RATE AND MODEL PERFORMANCE

We further explored the association between the model performances when trained with unclean data and the overlapped rate in Fig. 3. In this figure, we plotted the pre-trained models' macro F1 scores on each dataset along with the dataset's overlapped rate. The reason behind our model selection is because the BERT and GPT represent the best-performing models in this experiment, and their performances are more stable than the others. The association between the overlapped rate and the other performance metrics are consistent with Fig. 3, hence we only included one visualization.

Figure 3 demonstrates the macro F1 scores of the pre-trained models when trained with unclean data and the overlapped rates between two classes, which varies from 0% to 39% in increasing order. Based on this figure, the datasets with a high overlapped rate tend to yield a lower macro F1 score, except Synthetic lpr. On the datasets with almost no overlap between the classes, which are MIT lpr, ADFA-LD, and Xlock, the pre-trained models achieved one of the highest

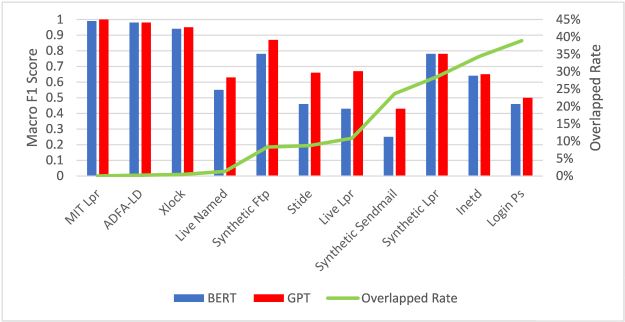

FIGURE 3. The pre-trained models' performances when trained on unclean data and the overlapped rates of each dataset.

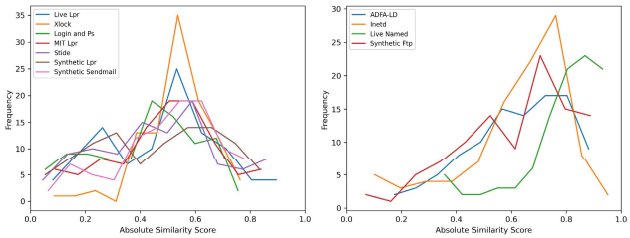

FIGURE 4. Cosine similarity distribution (a) Normal distribution group, (b) Negatively skewed distribution group.

macro F1 scores (above 0.94) and AUC scores (above 0.98) and one of the lowest FPR (below 0.03) and FNR (below 0.04). Additionally, on the datasets with an overlapping rate above 30% - Login and Ps and Inetd, BERT and GPT only achieved macro F1 scores between 0.50 and 0.65, AUC scores between 0.47 and 0.67, FPRs between 0.15 and 0.78, and FNRs between 0.29 and 0.48.

Nevertheless, the pre-trained models' performances on highly overlapped datasets are not necessarily the lowest among all of the datasets. In particular, Live lpr has an overlapped rate of 10.87%, and the pre-trained models generated one of the lowest performances compared to the other datasets. On the other hand, Synthetic lpr has an overlapped rate of 28.5%, and the pre-trained models generated a very high performance.

This indicates that the overlaps between two classes may not be the leading indicator of low performance on a dataset. However, based on Fig. 3, the overlapped rate seems to negatively correlate to the pre-trained models' performances on most datasets. This is understandable since overlapped data can affect model performance, which is consistent with the results from [86].

4) SEQUENCE SIMILARITY IN DATASETS

Suspecting that model performance may be affected by sequence similarity, we calculated the cosine similarity scores between 100 random pairs of normal sequences and intrusion sequences from each dataset. We plotted the similarity score distributions in Fig. 4. The figure demonstrates two groups of datasets. Fig. 4a represents the datasets with a normal distribution of similarity scores: Live lpr, Xlock, Login and Ps, MIT lpr, Stide, Synthetic lpr, and Synthetic Sendmail. On these datasets, the fragments of system call sequences from both classes are distinctive from each other. Fig. 4b represents the datasets with a negatively skewed distribution of similarity scores: ADFA-LD, Inetd, Live Named, and Synthetic Ftp. On these datasets, the fragments of the system call sequences from the normal class and the intrusion class are very similar to each other. The categorization of datasets helps us distinguish the effects of duplicates and overlaps in training data, between the ones with a normal distribution of sequence similarity versus the ones with a negative distribution of sequence similarity.

Regardless of its complexity, models learn and perform differently depending on the data. It is important to explore and investigate which data quality attributes can affect the model performance, such as whether clean training data achieves a better model performance, and how accurate a model's performance evaluation is when it is evaluated on unclean data.

B. PERFORMANCE VARIATION ON CLEAN TRAINING DATA VERSUS UNCLEAN TRAINING DATA

This section presents the results of experiment one, where the models are evaluated with clean testing data after trained with clean data versus with unclean data. In this experiment, we trained the models with clean and unclean data separately. We evaluated their performances on clean testing data to ensure a fair comparison. On some datasets, the traditional ML models show different performance trends compared to the pre-trained models, so we present the results based on the model types. RF and DT represent the traditional ML models, whereas BERT and GPT are the pre-trained models.

When investigating the performance difference between the models trained with clean data and those trained with unclean data, we observe four patterns in the performance changes: unaffected, increasing, decreasing, and contradicting. We demonstrate the differences between the model performances when they are trained with clean data (orange bars) versus with unclean data (blue bars) in Fig. 5 to 7. From left to right and top to bottom, the model performances are BERT, GPT, DT, and RF, respectively. Each pair of bars represent the model performance in terms of the macro F1 score, AUC score, FPR, and FNR, respectively. In each subsection below, we will describe a performance pattern along with the datasets where at least a group of models exhibits this pattern.

1) UNAFFECTED PERFORMANCE

In this performance pattern, the majority of the performance metrics in a model performance have minimal to no variation when the model is trained with clean or unclean data. ADFA-LD is the only dataset, in which the pre-trained models and the classic ML models exhibit no changes in both scenarios. Also, cleaning the training data from MIT lpr does not affect

TABLE 6. Overall macro F1 score results. Bold values indicate the highest macro F1 scores.

Index	Dataset	KM	LR	SVM	NN	DT	RF	KNN	NB	BERT	GPT-2
1	ADFA-LD	0.3	0.67	0.68	0.79	0.85	0.92	0.65	0.68	**0.99**	**0.99**
2	Inetd	0.31	0.52	0.47	0.57	0.44	0.52	0.60	0.57	**0.69**	0.65
3	Live lpr	0.37	0.47	0.52	0.57	0.56	0.52	0.56	0.48	**0.69**	0.53
4	Live Named	0.68	0.60	0.59	0.59	0.63	0.64	0.64	0.60	**0.77**	0.74
5	Login and Ps	0.52	0.46	0.40	0.40	0.38	0.38	0.45	0.45	**0.58**	0.53
6	MIT lpr	0.40	0.45	0.50	0.49	0.85	0.92	0.56	0.45	**1.00**	**1.00**
7	Stide	0.44	0.46	0.58	0.57	0.59	0.68	0.62	0.52	0.84	**0.90**
8	Synthetic Ftp	0.44	0.53	0.61	0.56	0.65	0.69	0.67	0.59	0.79	**0.84**
9	Synthetic lpr	0.50	0.53	0.60	0.59	0.64	0.70	0.68	0.57	**0.98**	0.93
10	Synthetic Sendmail	0.42	0.46	0.52	0.47	0.53	0.53	0.56	0.45	0.79	**0.81**
11	Xlock	0.1	0.49	0.66	0.59	0.82	0.89	0.76	0.57	**0.96**	**0.96**

BERT and GPT performances as they already achieve the best performance. The performances of DT and RF remain unchanged when they are trained with clean data from the Login and Ps, Synthetic Ftp, and Xlock datasets.

Based on Fig. 5, there is barely any difference between the orange bars and the blue bars across the performance metrics of BERT (a) and GPT (b). Training BERT with clean data yields 0.01 higher in the macro F1 score and AUC score, 0.01 lower in the FPR, and no change in the FNR. Training GPT with clean data yields a 0.01 higher macro F1 score and FPR and no change in the AUC and FNR. Additionally, training DT with clean data leads to a 0.01 higher macro F1 and AUC score, 0.02 higher in the FNR, and 0.04 lower in the FPR. Training RF with clean data leads to 0.01 higher in the AUC score, 0.02 higher in the FNR, 0.04 lower in the FPR, and no change in the macro F1 score. As these changes are lower than 0.05, it can be due to random chance. Hence, we conclude that training with clean data or unclean data from ADFA-LD yield no difference in the performances of BERT, GPT, DT and RF.

Similar variations in the performance metrics (less than 0.05) of the pre-trained models were observed on MIT lpr dataset and of the conventional ML models on Login and Ps (Fig. 6), Synthetic Ftp, and Xlock datasets.

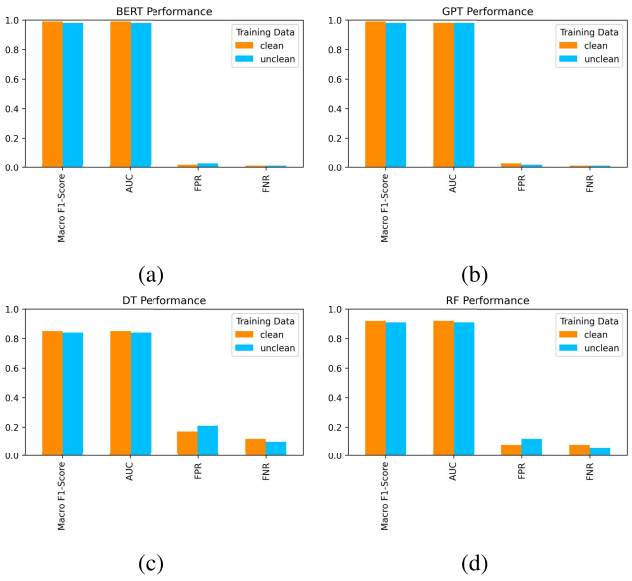

FIGURE 5. Performance comparison when trained with clean data versus with unclean data from ADFA-LD dataset (a) DT performance, (b) RF performance, (c) BERT performance, (d) GPT performance.

2) INCREASING PERFORMANCE

In this pattern, a model performs better when trained with clean data than with unclean data, indicating that the models can learn and perform better when we clean the training data from specific datasets. An increasing model performance exhibits improvements on the majority of its performance metrics. Synthetic lpr and Synthetic Sendmail are the only datasets where both groups of models (the pre-trained and the classic models) performed better when trained with clean data. Additionally, the pre-trained models also performed better when trained with clean data from Stide and Login and Ps. Lastly, the classic ML models performed better when trained with clean data from Live lpr and MIT lpr.

Both groups of models improved their performances when trained with clean data on Synthetic lpr. BERT and GPT had an increase in macro F1 scores (0.2 and 0.15 higher, respectively) and AUC scores (0.16 and 0.12 higher, respectively), and a decrease in the FNR (0.31 and 0.23 lower, respectively); yet their FPRs remained unchanged. Similarly, DT and RF performed better with higher macro F1 scores (0.12 and 0.13 higher, respectively) and AUC scores (0.09 and 0.08 lower, respectively), and lower FPRs (0.17 and 0.14 lower, respectively).

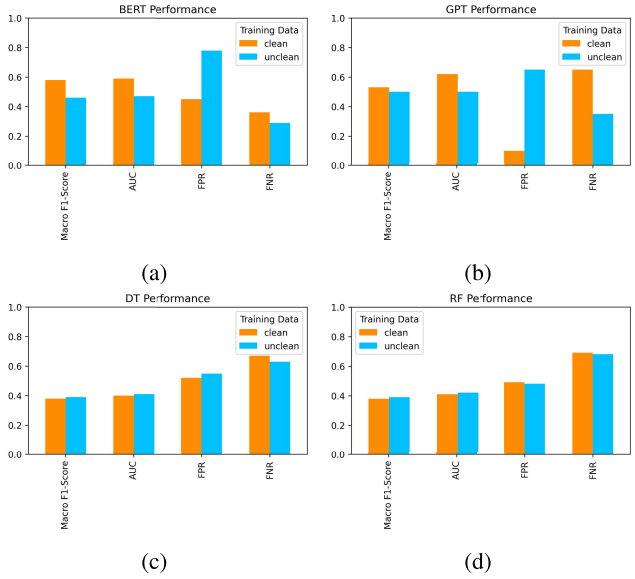

FIGURE 6. Performance comparison when trained with clean data versus with unclean data from the Login and Ps dataset (a) DT performance, (b) RF performance, (c) BERT performance, (d) GPT performance.

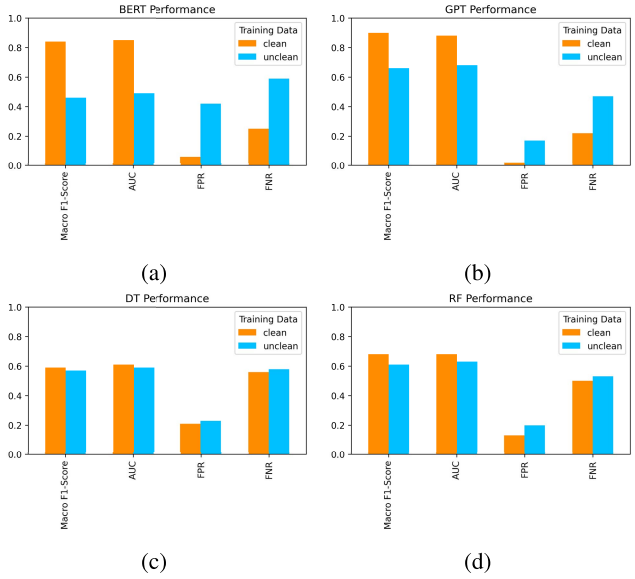

FIGURE 7. Performance comparison when trained with clean data versus with unclean data from the Stide dataset (a) DT performance, (b) RF performance, (c) BERT performance, (d) GPT performance.

Likewise, when trained with clean data from Synthetic Sendmail, both groups of models improved their performances, yet it came at a cost of increasing their FNRs. BERT and GPT also had most of their performance metrics improved on Login and Ps (Fig. 6), yet their FNRs were worsened. As DT performance remained unchanged, the pre-trained models and RF exhibited an increase in their performances when trained with clean data from the Stide dataset (Fig. 7a, b, and c). Regarding the classic ML models, they performed better when trained with clean data from the Live lpr and MIT lpr datasets.

3) DECREASING PERFORMANCE

In this pattern, a model performs worse when trained with clean data than with unclean data, indicating that cleaning data in some datasets reduces the models' learning capabilities. A decreasing model performance exhibits declines in the majority of its performance metrics. When trained with clean data, the ML models' performances were lower on the Inetd dataset, and the pre-trained models' performances were lower on Synthetic Ftp and Xlock.

The performances of DT and RF were lower when trained with clean data from the Inetd dataset as they both had a decrease in the AUC scores and an increase in the FPRs. Also, they both had a decrease in FNRs. Regarding the pre-trained models, they performed worse when trained with clean data from Synthetic Ftp and Xlock datasets. On Synthetic Ftp, BERT and GPT had a decrease in the AUC scores and an increase in the FNRs; however, they had a decrease in their FPRs. Although their macro F1 scores and FPRs remained unchanged, BERT and GPT had a decrease in their AUC scores and an increase in their FNRs when trained with clean data from Xlock.

4) INCONCLUSIVE PERFORMANCE

In this performance pattern, the variations in the performance metrics of a model contradict each other. For example, when trained with clean data, a model has an increase in macro F1 score and AUC score yet at the cost of increasing FPR and FNR. This model is considered to have an inconclusive performance. Live Named was the only dataset generating this pattern in the pre-trained models' and DT performances.

When trained with clean data from Live Named, the performances of BERT and GPT were inconclusive due to an increase in their macro F1 and FNRs along with a decrease in their AUC and FPRs. On the same dataset, DT had an increase in the macro F1 score and FNR along with a decrease in the AUC score.As there may be other factors causing the models' performances to be inconclusive, we will exclude the Live Named dataset from our discussion.

In this experiment, we find that:

- The pre-trained models and the classic ML models behaved differently on the same datasets, before and after duplicates and overlaps were removed. This once again proves that data quality has a direct impact on model performance regardless of the model complexity.
- Removing duplicates and overlaps increased the pre-trained models' performances on four datasets (Stide, Login Ps, Synthetic lpr, and Synthetic Sendmail) and had no effect on their performances on two datasets (ADFA-LD and MIT lpr), while reducing their performances on two datasets (Synthetic Ftp and Xlock).
- Removing duplicates and overlaps increased the classic ML models' performances on four datasets (Live lpr, MIT lpr, Synthetic lpr, and Synthetic Sendmail) and had no effects on their performances on four datasets (ADFA-LD, Login and Ps, Synthetic FTP,

and Xlock), while reducing their performances on one dataset (Inetd).

- The pre-trained models are more capable of learning from redundant data as their performances are higher than the classic ML models' performances on unclean data. Notably, the pre-trained models excelled their performances on MIT lpr regardless of the cleaning status of the training data; yet, the classic ML models only performed better after the duplicates and overlaps were removed from the same training data.

C. PERFORMANCE VARIATION ON CLEAN TESTING DATA VERSUS UNCLEAN TESTING DATA

We trained the models with clean data and evaluated them on testing data with various duplication rates ranging from 0% (clean testing data) to 95% (extremely repetitive testing data) with an increment of 5% each. To ensure a fair evaluation, the performance on the clean testing dataset (Table 6) serves as the baseline to compare against the performance on the unclean testing dataset. Therefore, we can observe how reliable or accurate the models' performances are as the duplication rate increases. With the same approach used in the previous experiment, we consider the model performance improved when the majority of its performance metrics are improved by at least 0.05.

Based on the models' performance trends as the duplication rate increases on the testing data, we categorized the datasets into four groups: stable (or unchanged), inflated, deflated, and unstable. We demonstrated different performance trends from Fig. 8 to Fig. 10. In each figure, from left to right and top to bottom, the performance trends are presented in terms of the macro F1 score, AUC score, FPR, and FNR, respectively. Each line chart contains the performance trends of DT (yellow), RF (green), BERT (blue), and GPT (red). In each subsection below, we will describe a performance trend along with the datasets where at least a group of models exhibits this trend.

1) DATASETS WITH A STABLE PERFORMANCE TREND

A performance is considered stable or unchanged when the variations of the performance metrics across different duplication rates on the x-axis are within 0.05. ADFA-LD and MIT lpr were the only datasets that showed unchanged performance trends.

Fig. 8 demonstrates BERT, GPT, DT, and RF performance trends as the duplication rate increases. Based on Fig. 8 a and b, the macro F1 scores and AUC scores of these models varied within 0.04, and their FPRs and FNRs varied within 0.02; except BERT, whose FNR variation was within 0.05. As the variations in the performance metrics were minimal, both groups of models generated very stable performances regardless of the increasing duplication rate in the ADFA-LD testing data.

On the MIT lpr dataset, we observed two different performance trends in the pre-trained and classic ML models. There was no variation in terms of all performance metrics of the pre-trained models. On the other hand, the ML models have a decreasing performance trend, which we will discuss in the next subsection.

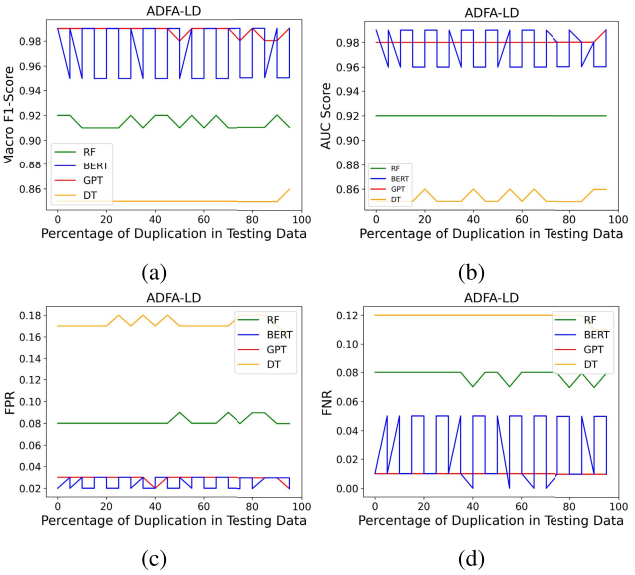

FIGURE 8. Stable performances of the pre-trained models and the classic ML models when evaluated with an increasing duplication rate on the ADFA-LD dataset (a) Macro F1 score, (b) AUC score, (c) FPR, and (d) FNR.

2) DATASETS WITH A DEFLATED PERFORMANCE TREND

A model performance deflates when its performance on unclean data is lower than its actual performance on clean data. A deflated model performance generates a decreasing trend in macro F1 and AUC scores and an increasing trend in FPR and FNR as the duplication rate increases. Five datasets generated this performance trend and they were Inetd (both pre-trained and ML models), Stide (both pre-trained and ML models), MIT lpr (only the ML models), Login and Ps (only the pre-trained models), and Live Named (only the pre-trained models).

Fig. 9 demonstrates the performance trends of all models on Inetd in terms of four metrics. All models had slight decreasing trends in the macro F1, AUC scores, and FPRs as the duplication rate increased on the x-axis. Their FNRs, on the other hand, became unstable and started increasing from a 50% duplication rate.

On the Stide dataset, all models had a stable performance trend until the duplication rate reached 60% (for DT and RF) and 85% (for BERT and GPT).Towards the end, their macro F1 and AUC scores dropped, while their FPRs and FNRs increased sharply.

On the MIT lpr dataset, the ML models' performances were worsened as the duplication rate increased. DT and RF had decreasing trends in macro F1 and AUC scores and an increasing trend in FNRs. In terms of FPR, DT has an increasing trend towards the highest duplication rate, whereas RF yields a constant FPR of 0.00.

On the Login and Ps dataset, the pre-trained models had slightly decreasing trends in their macro F1, AUC scores

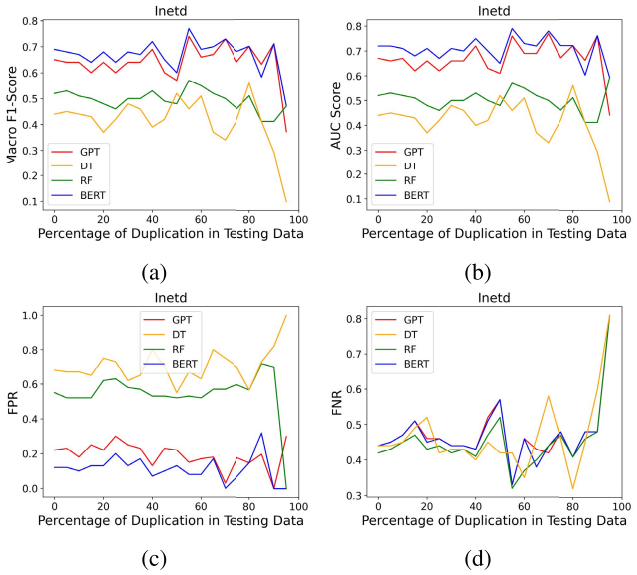

FIGURE 9. Deflated performances of the pre-trained models and the classic ML models when evaluated with an increasing duplication rate on the Inetd dataset (a) Macro F1 score, (b) AUC score, (c) FPR, and (d) FNR.

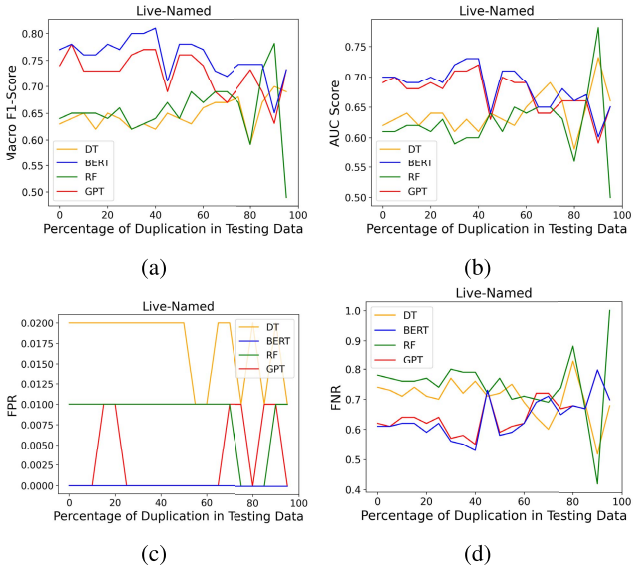

FIGURE 10. Deflated performances of the pre-trained models and inflated performances of the classic ML models when evaluated with an increasing duplication rate on the Live Named dataset (a) Macro F1 score, (b) AUC score, (c) FPR, and (d) FNR.

and FPRs, along with an increasing trend in FNRs. Overall, their performances deteriorate, especially their FNRs, as the duplication rate in the testing data increases.

Two observations were made on the Live Named dataset. Based on Fig. 10, the pre-trained models' performances became worse as the duplication rate increased. Notably, they had decreasing trends in macro F1 and AUC scores and an increasing trend in FNRs; yet their FPRs barely changed with less than 0.01 variation. On the other hand, the ML models exhibited an inflated performance, which we will discuss in the next subsection.

3) INFLATED PERFORMANCE

A performance is considered inflated when it has positive trends in macro F1 and AUC scores and negative trends in FPR and FNR across the x-axis. This indicates that the evaluation of unclean data is exaggerated or inflated. The datasets yield this performance trend in both groups of models are Synthetic Ftp, Synthetic lpr, Synthetic Sendmail, and Xlock, and in the ML models are Live Named and Login and Ps.

The overall performances of the pre-trained models and the ML models are inflated on the Synthetic Ftp, Synthetic lpr, and Synthetic Sendmail datasets. Towards the highest duplication rate, both groups of models had increasing trends in their macro F1 and AUC scores and a decreasing trend in their FNRs. Furthermore, as the duplication rate reached 90%, the pre-trained models had a decrease in their FPRs, whereas the ML models had a sharp increase in their FPRs.

On the Xlock dataset, both groups of models had inflated performances as the duplication rate increased. Their macro F1 and AUC scores were inflated, whereas their FNRs were deflated. Their FPRs stayed unchanged regardless of the duplication rate.

On the Live Named and Login and Ps datasets, DT and RF performances were inflated. As the duplication rate increased, they both had an increase in their macro F1 and AUC scores and a decrease in their FNRs; however, there were minimal variations in their FPRs.

4) UNSTABLE PERFORMANCE

Finally, a model performance is considered unstable when its performance metric variations contain unusual spikes. Live lpr was the only dataset where both groups of models exhibited unstable performance trends as shown in Fig. 11. The pre-trained models' performances were stable until the duplication rate reached 60%, and the ML models' performances were stable until the duplication rate reached 40%. Interestingly, the FPRs of all 4 models remained below 0.04 (BERT's and GPT's) and 0.15 (DT's and RF's) across the x-axis.

In this experiment, we find that:

- The pre-trained models and the classic ML models were susceptible to duplicate data in the testing set, leading to unreliable performance evaluation.
- When evaluated with duplicate data, the pre-trained models' performances were deflated on four datasets (Inetd, Stide, Login Ps, and Live Named) and inflated on four datasets (Synthetic Ftp, Synthetic lpr, Synthetic Sendmail, and Xlock), while remaining stable on two datasets (ADFA-LD and MIT lpr).
- When evaluated with duplicate data, the ML models' performances were deflated on three datasets (Inetd, Stide, and MIT lpr) and inflated on six datasets (Live Named, Login and Ps, Synthetic Ftp, Synthetic lpr, Synthetic Sendmail, and Xlock), while remaining stable on one dataset (ADFA-LD).

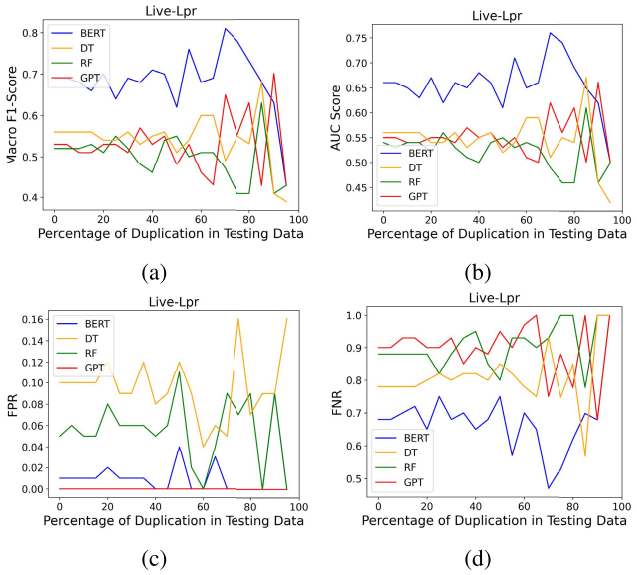

FIGURE 11. Unstable performances of both groups of models when evaluated with an increasing duplication rate on the Live Lpr dataset (a) Macro F1 score, (b) AUC score, (c) FPR, and (d) FNR.

VII. DISCUSSION AND IMPLICATIONS

In this section, we discuss the insights behind the experiment results regarding model selection, quality assurance in training data, and how to design testing data for a reliable model evaluation.

A. MODEL SELECTION FOR HIDS

Cyberattacks are often conducted through a series of actions to avoid being detected by IDS and intrusion prevention systems. All actions executed within a system are recorded as the system call data. Therefore, features in a system call sequence are not independent but interconnected in their positions and the sequential order, which contains essential information that can distinguish intrusion activities from normal activities.

BERT and GPT are large language models that were pre-trained on text data to learn and extract conceptual information from natural language through different types of embeddings. In contrast, the traditional ML models are designed to learn from the selected features by developing a particular set of rules. Given the data nature, the pre-trained models can learn more from sequential data than the classic ML models. This is supported by the difference in their performances in Table 6. The performance gap between the pre-trained models and the conventional ML models is roughly 10% on all datasets.

Additionally, the pre-trained models are less susceptible to redundant data (overlaps and duplicates) in the training and testing data than the conventional ML models. Regardless of the cleaning status of the training data, the pre-trained models achieved higher performance than the conventional ML models. The pre-trained models' evaluations were more stable than the conventional ML models when tested with highly duplicated data. Hence, using the classic ML models in HIDS is not the best approach as the models cannot learn the context from sequential data. Based on our experiment results, we recommend using the pre-trained models, such as BERT or GPT, to achieve the best results in detecting intrusions through an input of system call sequences.

B. QUALITY ASSURANCE IN HIDS TRAINING DATA

Based on the experiment results, we propose guidance based on nine data quality dimensions from section III-C to assure data quality in a HIDS dataset. These quality dimensions are reputation, relevance, comprehensiveness, variety, timeliness, accuracy, consistency, duplication, and overlap.

First, a dataset must be collected by a reputable organization with specific goals in mind over a certain of time period to ensure that the data distribution reflects the population distribution. Additionally, when collecting HIDS data, one must ensure that the dataset contains various examples of the latest attacks. By satisfying these conditions, a dataset has achieved the reputation, relevance, comprehensiveness, variety, and accuracy quality dimensions.

Regarding the timeliness quality and consistency quality, we have the following suggestions. Latest datasets are preferred to old datasets due to the constant changes in intrusion signatures and normal patterns. In our study, the latest dataset like ADFA-LD is preferred over the other datasets as it contains more novel attacks and up-to-date normal traces. The consistency quality can be ensured by checking if the data is organized sequentially by each PID (like the ADFA-LD dataset) or by grouping the data by their PID as we proposed in the data pre-processing procedure.

Regarding the last two data qualities, our experiment results provided us the rationales to remove duplicates and overlaps from a dataset. Firstly, the inconsistency of each model performance on 11 datasets proved that data quality has a direct impact on the detection performance. Secondly, we found that the overlapped percentage has an inverse correlation with the pre-trained models' performances (section VI-A3). Thirdly, our second experiment showed that the performances of the pre-trained models and the classic ML models were either inflated or deflated on most testing datasets containing duplicates.

Combining the first experiment results and the insights from Fig. 4, we have the following suggestions. For the datasets containing distinctive sequences or with a normal distribution of cosine similarities, we recommend removing duplicate and overlapped sequences as it mostly yields neutral to positive effects on the pre-trained models' performances. For the datasets containing many similar sequences from both classes or with a negative skew distribution of the cosine similarities among the sequences, we do not recommend removing duplicate and overlapped sequences as it can yield negative or inconclusive performance results. As many HIDS datasets contain mostly overlapped and duplicate sequences (section VI-A), cleaning data can result in insufficient data for training and testing. Hence, if there are sufficient data, we recommend removing duplicate and overlapped data using our

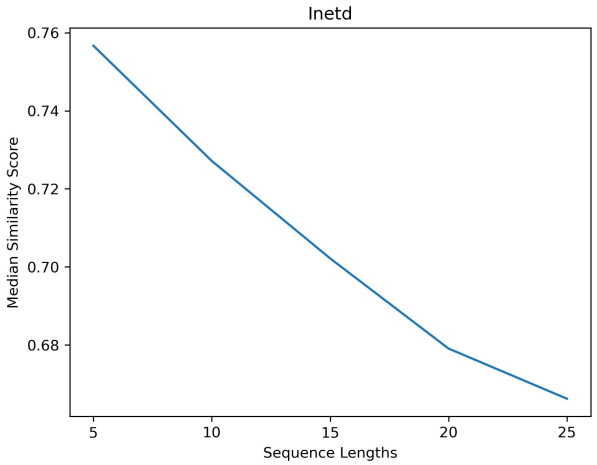

FIGURE 12. Cosine similarity is reduced in longer input sequence length from the Inetd dataset.

proposed cleaning method (section IV-B) to ensure reliability in model evaluation. Moreover, removing overlaps and duplicates can be beneficial to intrusion detection expectancy on high-quality datasets as models can distinguish each class by the most distinctive patterns. Besides, our previous experiment showed that a longer sequence length has a lower sequence similarity score (as shown in Fig. 12); this suggests increasing the input sequence length is another way to reduce duplicate and overlapped data between both classes.

C. DESIGNING TESTING DATA FOR RELIABLE MODEL EVALUATION

From our second experiment, we found that the performance evaluations of both pre-trained and classic ML models on most datasets were either deflated or inflated as the duplication rate increased. The only exception datasets were ADFA-LD and MIT lpr, which both had a great amount of unique data instances with low overlapped and duplication rates. As a result, having duplicates in the testing data can result misleading results to model evaluation unless the dataset is high quality. Therefore, we recommend removing duplicate instances from testing data to ensure reliability and accuracy in model evaluation. Additionally, testing data should represent diverse attacks and normal patterns to ensure an up-to-date model evaluation. It can be achieved by collecting data from diverse attacks and randomly stratifying the collected data to include various attacks in testing data.

D. IMPLICATIONS

The variation of a model performance on different datasets demonstrates that data quality has a direct impact on model performance. Compared to the classic ML models, the pre-trained models were more capable of learning from overlapped and duplicate data, which explained their superior performances even on unclean data. Therefore, we recommend choosing pre-trained models for HIDS tasks. Moreover, the inverse relationship between the overlapped rate and the pre-trained models' performances provides a reason to remove overlaps from the the training data. Applying our proposed cleaning method on the training datasets with a normal distribution of sequence similarities gives us more chance at improving the model performances. However, it may cause an inverse effect on ML performances, especially on low-quality data. Additionally, evaluating models with data containing duplicates mostly generates unreliable results. Therefore, we recommend removing overlaps and duplicates from a dataset before training and evaluating a model. Lastly, our guidance based on nine data quality dimensions and attributes can be used to assure data quality in an intrusion detection dataset.

There were two limitations in our study: (1). Manually splitting a trace of system calls into multiple fragments can reduce the semantic context in the data, and increase duplicates and overlaps. Hence, the entire sequence of system calls should be used to train ML-based IDS so that the models can capture and learn from the contextual differences between the normal class and the intrusion class. However, the sequence could be too long to be processed. (2). Pre-trained language models do not always generate high performance as they were pre-trained on natural language, not system events. Hence, using a significant amount of sequences of system events to re-train a language model can be a future direction. Furthermore, transfer learning [4] can be a promising direction in which one can leverage a model's experience in detecting previously known attacks to enhance its zero-day detection capability.

VIII. CONCLUSION

The research findings in this research show that regardless of the model complexity, ML and DL models learn and perform differently depending on the data quality. In the first experiment, we found that the pre-trained models are less susceptible to duplicates and overlaps compared to the traditional ML models, which explains the pre-trained models' superior performances. By removing duplicates and overlaps, the pre-trained models' performances either remained unchanged or improved on most datasets used in this study. However, applying the proposed data cleaning method may lead to an adverse effect such as decreasing detection performance or unstable performance on low-quality data. In the second experiment, we found that evaluating the pre-trained models and ML models on duplicate data leads to unreliable evaluations in most cases, where the actual performance was either inflated, deflated, or become unstable. The only exception to this effect occur in two high-quality datasets - ADFA-LD and MIT lpr, in which the performances of the pre-trained models and the ML models were unaffected regardless of the cleaning status of the testing data. Therefore, it is necessary to use our proposed data cleaning method to process the training data to increase detection performance. It is also important to use the proposed data cleaning method to process the testing data to ensure a reliable performance evaluation.

APPENDIX
ABBREVIATIONS OF THE MACHINE LEARNING MODELS

Logistic regression (LR), support vector machine (SVM), decision tree (DT), random forest (RF), neural network (NN), artificial neural network (ANN), isolation forest (IF), k nearest neighbor (KNN), scaled convex hull (SCH), histogram-based outlier score (HBOS), cluster-based local outlier factor (CBLOF), naïve bayes (NB), averaged one dependence estimator (AODE), radial basis function network (RBFN), multi-layer perceptron (MLP), deep neural networks (DNN), convolutional neural network (CNN), long short-term memory (LSTM), recurrent neural network (RNN), gated recurrent unit (GRU), softmax regression (SR), hidden Markov model (HMM), variational autoencoder (VAE), sequence covering for intrusion detection (SC4ID), intrusion detection tree (IntruDTree), gradient boosting decision tree (GBDT), generative adversarial network (GAN), extreme gradient boosting (XGB), deep belief network (DBN), light gradient boosting machine(LGBM), extreme learning machine (ELM), radial basis functions neural network (RBF-NN), multi-variable naïve bayesian (MNB), stochastic gradient decent (SGD), sequential model (SM).

ACKNOWLEDGMENT

The authors would like to thank Abdullah Gadi at the University of North Texas for conducting the investigation of existing ML models and public datasets for IDS and creating tables 1 and 2 and the paragraphs to describe the two tables and we also would like to thank Marie Bloechle at the University of North Texas for proofreading the paper.

REFERENCES

[1] A. Ayodeji, Y.-K. Liu, N. Chao, and L.-Q. Yang, "A new perspective towards the development of robust data-driven intrusion detection for industrial control systems," *Nucl. Eng. Technol.*, vol. 52, no. 12, pp. 2687–2698, Dec. 2020.

[2] N. Sambasivan, S. Kapania, H. Highfill, D. Akrong, P. Paritosh, and L. M. Aroyo, "'Everyone wants to do the model work, not the data work': Data cascades in high-stakes AI," in *Proc. CHI Conf. Hum. Factors Comput. Syst.*, New York, NY, USA, 2021, pp. 1–15, doi: 10.1145/3411764.3445518.

[3] J. Buolamwini and T. Gebru, "Gender shades: Intersectional accuracy disparities in commercial gender classification," in *Proc. Conf. Fairness, Accountability Transparency*, 2018, pp. 77–91.

[4] H. Chen, J. Chen, and J. Ding, "Data evaluation and enhancement for quality improvement of machine learning," *IEEE Trans. Rel.*, vol. 70, no. 2, pp. 831–847, Jun. 2021.

[5] R. M. Verma, V. Zeng, and H. Faridi, "Data quality for security challenges: Case studies of phishing, malware and intrusion detection datasets," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2019, pp. 2605–2607, doi: 10.1145/3319535.3363267.

[6] A. Sahu, Z. Mao, K. Davis, and A. E. Goulart, "Data processing and model selection for machine learning-based network intrusion detection," in *Proc. IEEE Int. Workshop Tech. Committee Commun. Quality Rel. (CQR)*, May 2020, pp. 1–6.

[7] H. H. Al-Maksousy, M. C. Weigle, and C. Wang, "NIDS: Neural network based intrusion detection system," in *Proc. IEEE Int. Symp. Technol. Homeland Secur. (HST)*, Oct. 2018, pp. 1–6.

[8] J. Hu, X. Yu, D. Qiu, and H. H. Chen, "A simple and efficient hidden Markov model scheme for host-based anomaly intrusion detection," *IEEE Netw.*, vol. 23, no. 1, pp. 42–47, Jan. 2009.

[9] N. N. Tran, R. Sarker, and J. Hu, "An approach for host-based intrusion detection system design using convolutional neural network," in *Proc. Int. Conf. Mobile Netw. Manage.* Cham, Switzerland: Springer, 2017, pp. 116–126.

[10] A. Chawla, B. Lee, S. Fallon, and P. Jacob, "Host based intrusion detection system with combined cnn/rnn model," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*. Cham, Switzerland: Springer, 2018, pp. 149–158.

[11] H. Liu and B. Lang, "Machine learning and deep learning methods for intrusion detection systems: A survey," *Appl. Sci.*, vol. 9, no. 20, p. 4396, Oct. 2019.

[12] L. Le Jeune, T. Goedeme, and N. Mentens, "Machine learning for misuse-based network intrusion detection: Overview, unified evaluation and feature choice comparison framework," *IEEE Access*, vol. 9, pp. 63995–64015, 2021.

[13] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O. M. Caicedo, "A comprehensive survey on machine learning for networking: Evolution, applications and research opportunities," *J. Internet Services Appl.*, vol. 9, no. 1, pp. 1–99, Jun. 2018.

[14] Z. Ahmad, A. S. Khan, C. W. Shiang, J. Abdullah, and F. Ahmad, "Network intrusion detection system: A systematic study of machine learning and deep learning approaches," *Trans. Emerg. Telecommun. Technol.*, vol. 32, no. 1, Jan. 2021, doi: 10.1002/ett.4150.

[15] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff, "A sense of self for unix processes," in *Proc. IEEE Symp. Secur. Privacy*, May 1996, p. 120.

[16] K. Hara and K. Shiomoto, "Intrusion detection system using semi-supervised learning with adversarial auto-encoder," in *Proc. NOMS IEEE/IFIP Netw. Oper. Manage. Symp.*, Apr. 2020, pp. 1–8.

[17] S. Behera, A. Pradhan, and R. Dash, "Deep neural network architecture for anomaly based intrusion detection system," in *Proc. 5th Int. Conf. Signal Process. Integr. Netw. (SPIN)*, Feb. 2018, pp. 270–274.

[18] G. Pu, L. Wang, J. Shen, and F. Dong, "A hybrid unsupervised clustering-based anomaly detection method," *Tsinghua Sci. Technol.*, vol. 26, no. 2, pp. 146–153, Apr. 2021.

[19] S. Zhao, W. Li, T. Zia, and A. Y. Zomaya, "A dimension reduction model and classifier for anomaly-based intrusion detection in Internet of Things," in *Proc. IEEE 15th Int. Conf. Dependable, Autonomic Secure Comput., 15th Int. Conf. Pervasive Intell. Comput., 3rd Int. Conf. Big Data Intell. Comput. Cyber Sci. Technol. Congr. (DASC/PiCom/DataCom/CyberSciTech)*, Nov. 2017, pp. 836–843.

[20] T. Salman, D. Bhamare, A. Erbad, R. Jain, and M. Samaka, "Machine learning for anomaly detection and categorization in multi-cloud environments," in *Proc. IEEE 4th Int. Conf. Cyber Secur. Cloud Comput. (CSCloud)*, Jun. 2017, pp. 97–103.

[21] R. Abdulhammed, M. Faezipour, A. Abuzneid, and A. AbuMallouh, "Deep and machine learning approaches for anomaly-based intrusion detection of imbalanced network traffic," *IEEE Sensors Lett.*, vol. 3, no. 1, pp. 1–4, Jan. 2019.

[22] S. Naseer, Y. Saleem, S. Khalid, M. K. Bashir, J. Han, M. M. Iqbal, and K. Han, "Enhanced network anomaly detection based on deep neural networks," *IEEE Access*, vol. 6, pp. 48231–48246, 2018.

[23] N. Elmrabit, F. Zhou, F. Li, and H. Zhou, "Evaluation of machine learning algorithms for anomaly detection," in *Proc. Int. Conf. Cyber Secur. Protection Digit. Services (Cyber Secur.)*, Jun. 2020, pp. 1–8.

[24] R. Aliakbarisani, A. Ghasemi, and S. F. Wu, "A data-driven metric learning-based scheme for unsupervised network anomaly detection," *Comput. Electr. Eng.*, vol. 73, pp. 71–83, Jan. 2019.

[25] M. Nawir, A. Amir, N. Yaakob, and O. B. Lynn, "Effective and efficient network anomaly detection system using machine learning algorithm," *Bull. Electr. Eng. Informat.*, vol. 8, no. 1, pp. 46–51, 2019.

[26] M. Evangelou and N. M. Adams, "An anomaly detection framework for cyber-security data," *Comput. Secur.*, vol. 97, Oct. 2020, Art. no. 101941.

[27] J. Meira, R. Andrade, I. Praça, J. Carneiro, V. Bolón-Canedo, A. Alonso-Betanzos, and G. Marreiros, "Performance evaluation of unsupervised techniques in cyber-attack anomaly detection," *J. Ambient Intell. Humanized Comput.*, vol. 11, no. 11, pp. 4477–4489, Nov. 2020.

[28] A. Hariharan, A. Gupta, and T. Pal, "CAMLPAD: Cybersecurity autonomous machine learning platform for anomaly detection," in *Proc. Future Inf. Commun. Conf.* Cham, Switzerland: Springer, 2020, pp. 705–720.

[29] S. Choudhary and N. Kesswani, "Analysis of KDD-cup'99, NSL-KDD and UNSW-NB15 datasets using deep learning in IoT," *Proc. Comput. Sci.*, vol. 167, pp. 1561–1573, Jan. 2020.

[30] M. Wang, K. Zheng, Y. Yang, and X. Wang, "An explainable machine learning framework for intrusion detection systems," *IEEE Access*, vol. 8, pp. 73127–73141, 2020.
[31] H. Alqahtani, I. H. Sarker, A. Kalim, S. M. M. Hossain, S. Ikhlaq, and S. Hossain, "Cyber intrusion detection using machine learning classification techniques," in *Proc. Int. Conf. Comput. Sci., Commun. Secur.* Cham, Switzerland: Springer, 2020, pp. 121–131.
[32] E. Besharati, M. Naderan, and E. Namjoo, "LR-HIDS: Logistic regression host-based intrusion detection system for cloud environments," *J. Ambient Intell. Humanized Comput.*, vol. 10, no. 9, pp. 3669–3692, Sep. 2019.
[33] B. Subba, S. Biswas, and S. Karmakar, "Host based intrusion detection system using frequency analysis of n-gram terms," in *Proc. TENCON IEEE Region Conf.*, Nov. 2017, pp. 2006–2011.
[34] M. Choraś and M. Pawlicki, "Intrusion detection approach based on optimised artificial neural network," *Neurocomputing*, vol. 452, pp. 705–715, Sep. 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231220319032
[35] P. Devan and N. Khare, "An efficient XGBoost-DNN-based classification model for network intrusion detection system," *Neural Comput. Appl.*, vol. 32, pp. 12499–12514, Jan. 2020, doi: 10.1007/s00521-020-04708-x.
[36] T. Su, H. Sun, J. Zhu, S. Wang, and Y. Li, "BAT: Deep learning methods on network intrusion detection using NSL-KDD dataset," *IEEE Access*, vol. 8, pp. 29575–29585, 2020.
[37] F. Yihunie, E. Abdelfattah, and A. Regmi, "Applying machine learning to anomaly-based intrusion detection systems," in *Proc. IEEE Long Island Syst., Appl. Technol. Conf. (LISAT)*, May 2019, pp. 1–5.
[38] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.
[39] Z. Li, Z. Qin, K. Huang, X. Yang, and S. Ye, "Intrusion detection using convolutional neural networks for representation learning," in *Neural Information Processing*, D. Liu, S. Xie, Y. Li, D. Zhao, and E.-S. M. El-Alfy, Eds. Cham, Switzerland: Springer, 2017, pp. 858–866.
[40] J. Liu, Y. Gao, and F. Hu, "A fast network intrusion detection system using adaptive synthetic oversampling and LightGBM," *Comput. Secur.*, vol. 106, Jul. 2021, Art. no. 102289.
[41] X. Liu, T. Li, R. Zhang, D. Wu, Y. Liu, and Z. Yang. (Jul. 2021). *A GAN and Feature Selection-Based Oversampling Technique for Intrusion Detection*. [Online]. Available: https://www.hindawi.com/journals/scn/2021/9947059/
[42] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 41–50, Feb. 2018.
[43] P.-F. Marteau, "Sequence covering for efficient host-based intrusion detection," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 4, pp. 994–1006, Apr. 2019.
[44] S. A. Maske and T. J. Parvat, "Advanced anomaly intrusion detection technique for host based system using system call patterns," in *Proc. Int. Conf. Inventive Comput. Technol. (ICICT)*, Aug. 2016, pp. 1–4.
[45] Y. Lu and S. Teng, "Application of sequence embedding in host-based intrusion detection system," in *Proc. IEEE 24th Int. Conf. Comput. Supported Cooperat. Work Design (CSCWD)*, May 2021, pp. 434–439.
[46] M. Salem, S. Taheri, and J. S. Yuan, "Anomaly generation using generative adversarial networks in host-based intrusion detection," in *Proc. 9th IEEE Annu. Ubiquitous Comput., Electron. Mobile Commun. Conf. (UEMCON)*, Nov. 2018, pp. 683–687.
[47] U. Ahmed and A. Masood, "Host based intrusion detection using RBF neural networks," in *Proc. Int. Conf. Emerg. Technol.*, Oct. 2009, pp. 48–51.
[48] X. Zhang, Q. Niyaz, F. Jahan, and W. Sun, "Early detection of host-based intrusions in Linux environment," in *Proc. IEEE Int. Conf. Electro Inf. Technol. (EIT)*, Jul. 2020, pp. 475–479.
[49] I. H. Sarker, Y. B. Abushark, F. Alsolami, and A. I. Khan, "IntruDTree: A machine learning based cyber security intrusion detection model," *Symmetry*, vol. 12, no. 5, p. 754, May 2020.
[50] J. Liu, K. Xiao, L. Luo, Y. Li, and L. Chen, "An intrusion detection system integrating network-level intrusion detection and host-level intrusion detection," in *Proc. IEEE 20th Int. Conf. Softw. Qual., Rel. Secur. (QRS)*, Dec. 2020, pp. 122–129.
[51] S. Jose, D. Malathi, B. Reddy, and D. Jayaseeli, "A survey on anomaly based host intrusion detection system," *J. Phys., Conf. Ser.*, vol. 1000, Apr. 2018, Art. no. 012049.
[52] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, vol. 1, Jun. 2019, pp. 4171–4186.
[53] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, p. 9, 2019.
[54] S. J. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. K. Chan, "Cost-based modeling for fraud and intrusion detection: Results from the JAM project," *Proc. DARPA Inf. Survivability Conf. Expo.*, vol. 2, Jan. 2000, pp. 130–144.
[55] J. Song, H. Takakura, and Y. Okabe. (2006). *Description of Kyoto University Benchmark Data*. Accessed: Apr. 12, 2021. [Online]. Available: http://www.takakura.com/Kyoto_data/BenchmarkData-Description-v5.pdf
[56] DARPA2009. (2009). *Darpa 2009 Intrusion Detection Dataset*. Accessed: Apr. 12, 2021. [Online]. Available: http://www.darpa2009.netsec.colostate.edu/
[57] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symp. Comput. Intell. Secur. Defense Appl.*, Jul. 2009, pp. 1–6.
[58] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Comput. Secur.*, vol. 31, no. 3, pp. 357–374, 2012.
[59] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. Mil. Commun. Inf. Syst. Conf. (MilCIS)*, Nov. 2015, pp. 1–6.
[60] W. Haider, J. Hu, J. Slay, B. P. Turnbull, and Y. Xie, "Generating realistic intrusion detection system dataset based on fuzzy qualitative modeling," *J. Netw. Comput. Appl.*, vol. 87, pp. 185–192, Jun. 2017.
[61] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy*, 2018, pp. 108–116.
[62] MIT Lincoln Laboratory. (1998). *1998 Darpa Intrusion Detection Evaluation Dataset*. Accessed: Apr. 12, 2021. [Online]. Available: https://www.ll.mit.edu/r-d/datasets/1998-darpa-intrusion-detection-evaluation-dataset
[63] S. A. Hofmeyr, S. Forrest, and A. Somayaji, "Intrusion detection using sequences of system calls," *J. Comput. Secur.*, vol. 6, no. 3, pp. 151–180, 1998.
[64] G. Creech and J. Hu, "A semantic approach to host-based intrusion detection systems using contiguousand discontiguous system call patterns," *IEEE Trans. Comput.*, vol. 63, no. 4, pp. 807–819, Apr. 2014.
[65] D. Čeponis and N. Goranin, "Towards a robust method of dataset generation of malicious activity for anomaly-based HIDS training and presentation of AWSCTD dataset," *Baltic J. Modern Comput.*, vol. 6, no. 3, pp. 217–234, 2018.
[66] M.-O. Pahl and F.-X. Aubet, "All eyes on you: Distributed multi-dimensional IoT microservice anomaly detection," in *Proc. 14th Int. Conf. Netw. Service Manage. (CNSM)*, Nov. 2018, pp. 72–80.
[67] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Comput. Secur.*, vol. 86, pp. 147–167, Oct. 2019.
[68] J. Buolamwini and T. Gebru, "Gender shades: Intersectional accuracy disparities in commercial gender classification," in *Proc. 1st Conf. Fairness, Accountability Transparency*, vol. 81, S. A. Friedler and C. Wilson, Eds. New York, NY, USA, Feb. 2018, pp. 77–91.
[69] C. G. Northcutt, A. Athalye, and J. Mueller, "Pervasive label errors in test sets destabilize machine learning benchmarks," 2021, *arXiv:2103.14749*.
[70] E. J. M. Lauría and G. Tayi, "Statistical machine learning for network intrusion detection: A data quality perspective," *Int. J. Services Sci.*, vol. 1, pp. 179–195, Jan. 2008.
[71] A. Divekar, M. Parekh, V. Savla, R. Mishra, and M. Shirole, "Benchmarking datasets for anomaly-based network intrusion detection: KDD CUP 99 alternatives," in *Proc. IEEE 3rd Int. Conf. Comput., Commun. Secur. (ICCCS)*, Oct. 2018, pp. 1–8.
[72] G. Apruzzese, P. Laskov, and A. Tastemirova, "SoK: The impact of unlabelled data in cyberthreat detection," in *Proc. IEEE 7th Eur. Symp. Secur. Privacy (EuroS&P)*, Jun. 2022, pp. 20–42.
[73] T. Ngan, C. Haihua, J. Janet, B. Jay, and D. Junhua, "Effect of class imbalance on the performance of machine learning-based network intrusion detection," *Int. J. Performability Eng.*, vol. 17, no. 9, p. 741, 2021. [Online]. Available: http://www.ijpe-online.com/EN/abstract/article_4617.shtml

[74] R. M. Verma, V. Zeng, and H. Faridi, "Data quality for security challenges: Case studies of phishing, malware and intrusion detection datasets," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2019, pp. 2605–2607.
[75] W. Fan, "Data quality: From theory to practice," *SIGMOD Rec.*, vol. 44, no. 3, pp. 7–18, 2015.
[76] A. F. Karr, A. P. Sanil, and D. L. Banks, "Data quality: A statistical perspective," *Stat. Methodol.*, vol. 3, no. 2, pp. 137–173, 2006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1572312705000638
[77] G. Maciá-Fernández, J. Camacho, R. Magán-Carrión, P. García-Teodoro, and R. Therón, "UGR'16: A new dataset for the evaluation of cyclostationarity-based network IDSs," *Comput. Secur.*, vol. 73, pp. 411–424, Mar. 2018.
[78] R. Y. Wang and D. M. Strong, "Beyond accuracy: What data quality means to data consumers," *J. Manage. Inf. Syst.*, vol. 12, no. 4, pp. 5–33, 1996.
[79] C. Batini, C. Cappiello, C. Francalanci, and A. Maurino, "Methodologies for data quality assessment and improvement," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 1–52, Jul. 2009.
[80] H. Chen, G. Cao, J. Chen, and J. Ding, "A practical framework for evaluating the quality of knowledge graph," in *Proc. China Conf. Knowl. Graph Semantic Comput.* Cham, Switzerland: Springer, 2019, pp. 111–122.
[81] A. Jøsang, R. Ismail, and C. Boyd, "A survey of trust and reputation systems for online service provision," *Decision Support Syst.*, vol. 43, no. 2, pp. 618–644, 2007.
[82] A. Bradai and H. Afifi, "Game theoretic framework for reputation-based distributed intrusion detection," in *Proc. Int. Conf. Social Comput.*, Sep. 2013, pp. 558–563.
[83] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, pp. 1–22, Dec. 2019.
[84] R. Panigrahi and S. Borah, "A detailed analysis of CICIDS2017 dataset for designing intrusion detection systems," *Int. J. Eng. Technol.*, vol. 7, no. 3.24, pp. 479–482, 2018.
[85] M. Rani, S. B. Dhok, R. B. Deshmukh, and P. Kumar, "Overlap aware compressed signal classification," *IEEE Access*, vol. 8, pp. 52950–52967, 2020.
[86] V. López, A. Fernández, J. G. Moreno-Torres, and F. Herrera, "Analysis of preprocessing vs. cost-sensitive learning for imbalanced classification. Open problems on intrinsic data characteristics," *Expert Syst. Appl.*, vol. 39, no. 7, pp. 6585–6608, Jun. 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417411017143

NGAN TRAN received the B.S. degree in computer science from the University of North Texas (UNT), in 2020, where she is currently pursuing the Ph.D. degree in information science with a concentration in data science with the Department of Information Science. Her research interests include intrusion detection systems, machine learning, and data analytics.

HAIHUA CHEN (Member, IEEE) received the B.S. degree from Central China Normal University, in 2014, the M.S. degree from Wuhan University, in 2017, and the Ph.D. degree in information science from the Department of Information Science, University of North Texas (UNT), in 2022. He is currently a Clinical Assistant Professor in data science with the Department of Information Science, UNT. He is the Co-Editor of *The Electronic Library* and the Guest Editor of *Information Discovery and Delivery* and *Frontiers in Big Data*. His research interests include data quality in machine learning, legal artificial intelligence, data security and privacy, health informatics, and academic data mining.

JAY BHUYAN received the Ph.D. degree in computer science from the University of Louisiana at Lafayette. He is currently a Professor with the Department of Computer Science, Tuskegee University. His research and teaching interests include telecom software architecture and development, network security, education research, and machine learning. He has over 30 years of full-time and part-time teaching experience and over 15 years of research and development experience at Telecom Industry. He is a member of ACM and IEEE Computer Societies.

JUNHUA DING is currently the Reinburg Endowed Professor in data science with the Department of Information Science, University of North Texas (UNT), where he is also the Director of the Data Science Program. He was a Faculty Member of the Department of Computer Science, East Carolina University (ECU), before he joined UNT, in 2018. His research interests include data analytics, machine learning, computational law, data security and privacy, and software engineering.