smart cities

Article

Can Machine Learning Enhance Intrusion Detection to Safeguard Smart City Networks from Multi-Step Cyberattacks?

Jowaria Khan, Rana Elfakharany, Hiba Saleem, Mahira Pathan, Emaan Shahzad, Salam Dhou * and Fadi Aloul

Department of Computer Science and Engineering, American University of Sharjah, Sharjah P.O. Box 26666, United Arab Emirates; jowaria@umich.edu (J.K.); g00087725@aus.edu (R.E.); g00087239@aus.edu (H.S.); g00085003@aus.edu (M.P.); g00084456@aus.edu (E.S.); faloul@aus.edu (F.A.)

* Correspondence: sdhou@aus.edu

Highlights:

What are the main findings?

- Using machine learning was found to be effective in identifying and classifying multi-step network intrusion cyberattacks.
- Extreme Gradient Boosting (XGB) was identified as the top-performing machine learning model due to its high accuracy and computational efficiency, making it an ideal choice for an edge-based intrusion detection system (IDS).

What is the implication of the main finding?

- Identifying and classifying multi-step cyberattacks through advanced IDS help protect smart cities' systems from unauthorized access, ensuring the stability, security, and reliability of essential services.
- Given the dynamic nature of smart cities, where multiple systems work in tandem and data flow in real-time, this research highlights the importance of developing real-time intrusion detection systems that are able to detect intrusions as soon as they arise, enabling quick actions to isolate and mitigate risks before they can cause widespread damage.

Abstract: Intrusion detection systems are essential for detecting network cyberattacks. As the sophistication of cyberattacks increases, it is critical that defense technologies adapt to counter them. Multi-step attacks, which need several correlated intrusion operations to reach the desired target, are a rising trend in the cybersecurity field. System administrators are responsible for recreating whole attack scenarios and developing improved intrusion detection systems since the systems at present are still designed to generate alerts for only single attacks with little to no correlation. This paper proposes a machine learning approach to identify and classify multi-step network intrusion attacks, with particular relevance to smart cities, where interconnected systems are highly vulnerable to cyber threats. Smart cities rely on these systems seamlessly functioning with one another, and any successful cyberattack could have devastating effects, including large-scale data theft. In such a context, the proposed machine learning model offers a robust solution for detecting and mitigating multi-step cyberattacks in these critical environments. Several machine learning algorithms are considered, namely Decision Tree (DT), K-Nearest Neighbors (KNN), Naïve Bayes (NB), Support Vector Machine (SVM), Light Gradient-Boosting Machine (LGBM), Extreme Gradient Boosting (XGB) and Random Forest (RF). These models are trained on the Multi-Step Cyber-Attack Dataset (MSCAD), a recent dataset that is highly representative of real-world multi-step cyberattack scenarios, which increases the accuracy and efficiency of such systems. The experimental results show that the best performing model was XGB, which achieved a testing accuracy of 100% and an F1 Score of 88%. The proposed model

check for updates

Academic Editor: Pierluigi Siano

Received: 17 November 2024
Revised: 5 January 2025
Accepted: 13 January 2025
Published: 20 January 2025

Citation: Khan, J.; Elfakharany, R.; Saleem, H.; Pathan, M.; Shahzad, E.; Dhou, S.; Aloul, F. Can Machine Learning Enhance Intrusion Detection to Safeguard Smart City Networks from Multi-Step Cyberattacks? *Smart Cities* **2025**, *8*, 13. https://doi.org/10.3390/smartcities8010013

is computationally efficient and easy to deploy, which ensures a fast, sustainable and low power-consuming intrusion detection system at the cutting edge.

Keywords: cybersecurity; cyberattack; intrusion detection; machine learning; deep learning

1. Introduction

In today's digital world, the detection and prevention of cyberattacks is essential for securing computer networks. However, network invasions have evolved into multi-stage, sophisticated attacks. These threats, known as multi-step or multi-stage attacks, include a number of long-term associated processes and actions [1]. Compared to a single attack, this type of attack is more disruptive, since it is composed of several steps, making it more difficult to be identified by the victim. In addition, it is a challenge for Intrusion Detection Systems (IDSs) to handle multi-step attacks due to their multiple steps, multiple attack methods, and subtle nature. Usually, traditional IDSs struggle to detect the entire attack chain due to the stealthy nature and varied attack paths, requiring the correlation of multiple actions to understand the attack strategy and identify the threat. In other words, they cannot detect the original attack node and block the attack chain at its source [2].

In the case of smart cities, which depend on complex, interconnected networks that manage critical systems such as infrastructure, healthcare, transportation, energy and safety, these multi-step attacks present even greater risks. Due to the growing reliance on digital technologies in these cities, they have become vulnerable to cyberattacks. Any successful attack could greatly impact public safety and lead to significant consequences, including data breaches, system manipulations or service disruption. According to [3], anomaly-based detection systems that are implemented using machine learning models are best suited to providing the needed level of security in a smart environment. Therefore, it is crucial to develop advanced IDSs that are capable of detecting multi-step attacks, ensuring the security and stability of smart city networks.

Several IDS techniques have been implemented in the literature. Despite being effective with known threats, traditional IDSs rely on signature-based approaches that are often unable to detect new and emerging threats. Such a limitation has led to leveraging emerging machine learning technologies to improve IDSs [1]. In this paper, a machine learning-based framework is proposed for multi-step intrusion cyberattack detection on network devices. The MSCAD dataset is used to train the model [4]. Pre-processing and feature selection techniques are applied to clean the dataset and extract the relevant features. Then, several machine learning algorithms are applied, namely Decision Tree (DT), K-Nearest Neighbors (KNN), Naïve Bayes (NB), Support Vector Machine (SVM), Extreme Gradient Boosting (XGB), Light Gradient-Boosting Machine (LGBM) and Random Forest (RF).

The rest of the paper is structured as follows. Section 2 discusses previous work conducted on the topic of multi-stage attacks and IDSs. Section 3 explains the details of the dataset and the methods implemented in the paper. Section 4 elaborates on the results for each model. Section 5 discusses the findings and compares the results to previous studies in the field. Section 6 concludes the paper and discusses possible future work opportunities.

2. Literature Review

Several IDSs have been proposed in the literature. In this section, systems using machine learning, deep learning and other models are discussed.

2.1. Integrated Machine Learning and Deep Learning-Based Models

Several machine learning and deep learning-based IDSs have been proposed in the literature. For example, a detection system that leverages both machine learning and deep learning models to detect network attacks was proposed by Dhanya et al. [5]. The system has been applied on UNSW-NB15 datasets, and although this technique has achieved high detection rates for various types of attacks (an accuracy of 99.05%, as can be seen in Table 1), the current increase in the complexity of network attacks and the availability of more recent datasets makes the results outdated.

Table 1. Literature review summary for integrated ML and DL-Based Models.

Paper	Datasets	Technique	Best Results
Dhanya et al. [5]	UNSW-NB15	DT, RF, Adaboost, XGB and KNN	Accuracy of 99.05%
Dalal et al. [6]	Multi-Step Cyber-Attack Dataset (MSCAD)	Highly Boosted Neural Network	Accuracy of 99.72%
Taher et al. [7]	NSL-KDD	SVM and ANN	ANN accuracy of 94.02%
Pelletier and Abualkibash [8]	CIC IDS 2017	ANN, RF	RF was best model with accuracy of 96.4%
Gan et al. [9]	NSL-KDD	Data imbalance-based CNN-IDMDI	Average accuracy of 98.73% for binary intrusion detection, average accuracy of 94.55% for multi-class intrusion detection.
Maseer et al. [10]	IoTID 20	CNN-MLP	Highest accuracy of 98.1%
Qaddoura et al. [11]	IoTID 20	Clustering with reduction stage, oversampling and classification by Single Hidden Layer Feed-Forward Neural Network (SLFN)	G-mean of 0.9453 for SLFN-SVM-SMOTE. Highest accuracy of 94.81% for SMOTE

Dalal et al. proposed a deep learning-based method—a highly boosted neural network—to detect the multi-stage attacks [6]. Their work demonstrated the results of executing various machine learning algorithms in addition to their proposed enormously boosted neural network. The accuracy achieved in the prediction of multi-stage cyberattacks is 94.09% (Quest Model), 97.29% (Bayesian Network) and 99.09% (Neural Network). The evaluation results on the MSCAD dataset show that the proposed Extremely Boosted Neural Network could predict a multi-stage cyberattack with 99.72% accuracy. However, it is important to take into consideration the power consumption and complexity of neural network models when evaluating their performance in comparison to simpler and more efficient machine learning algorithms.

Similarly, Taher et al. worked on improving the accuracy of IDSs using machine learning methods such as SVMs and ANNs [7]. The best results for both models were observed using different numbers of features. The highest accuracy obtained for an SVM was 82.34% when using thirty-five features. On the other hand, ANNs achieved an accuracy of 94.02% using seventeen features. However, the study did not analyze more performance metrics such as F1 measures, to accurately determine the effectiveness of the model.

Pelletier and Abualkibash have also proposed training an ANN and RF to create a model that classifies labeled network traffic data [8]. Among the two, RF achieved a better accuracy of 96.4%. In the feature selection process, the model used only the 10 most important features out of over 80. While reducing the feature set can mitigate overfitting and improve computational efficiency, it may also discard critical features that capture subtle indicators of rare or complex attacks. In contrast, this proposed model retained 28 out of 66 features (see Section 3.2), ensuring a balance between reducing model complexity and preserving sufficient information for intrusion detection.

In addition, Gan et al. worked on improving the performance of IDSs with neural networks [9]. In their work, a data imbalance-based Convolutional Neural Network Intrusion Detection Method (CNN-IDMDI) was implemented. For binary intrusion detection, the average accuracy achieved was 98.73%, which resulted in improvements of 15.45%, 12.76% and 2.91% over CNN, CNN-LSTM and CNN-NSVM, respectively. As for multi-class intrusion detection, the average accuracy achieved was 94.55%, which is an enhancement by 16.09%, 12.71% and 3.66%, respectively, over the previously mentioned models. Despite the high accuracy of the results, the model has been tested using one-dimensional data, which may limit detection for sophisticated attacks.

Maseer et al. identified critical gaps in IDSs, highlighting the over-reliance on outdated datasets like KDDCUP'99 and NSL-KDD, which fail to capture modern network complexities and the importance of adopting a hybrid DL and ML approach [10]. They advocate for newer datasets such as CICIDS2017, CICIDS2018 and ToN_IoT, which better reflect realistic traffic patterns and diverse attack types, including multi-step attacks and botnets. Hybrid approaches address these issues effectively by combining DL's feature extraction capabilities with the simplicity of traditional ML. For example, the Autoencoder-SVM Hybrid achieved an F1 score of 95.6% on CICIDS2017, the DBN-SVM Hybrid attained 97.3% accuracy with a 2.1% false positive rate on UNSW-NB15, and the CNN-MLP Hybrid demonstrated an F1 score of 98.1% on ToN_IoT, showcasing the efficacy of hybrid models in advancing IDS performance.

Similarly, Qaddoura et al. examined a multi-stage approach for classifying intrusion and normal activities [11]. The study focused on clustering, with the use of reduction, oversampling and classification by a Single Hidden-Layer Feed-Forward Neural Network (SLFN). Since the dataset was imbalanced, oversampling was conducted using different oversampling techniques, including the Synthetic Minority Oversampling Technique (SMOTE) and the SVM and Synthetic Minority Oversampling Technique (SVM-SMOTE). The highest accuracy of 94.81% was achieved by SMOTE, while the highest G-mean value of 94.53% was obtained by SVM-SMOTE. Nonetheless, considering that the study used a multi-label dataset, the analysis of the model excluded the other available labels. As a result, more analysis with the other two labels needs to be conducted. Table 1 summarizes the integrated ML and DL models for such a problem.

2.2. Machine Learning-Based Models

A wide variety of classical machine learning models have been shown in the literature to improve IDSs. Ingale et al. implemented two machine learning models, namely the Hidden Markov Model (HMM) and NB [12]. They applied these models to the KDDCUP'99 network intrusion dataset [12]. Their paper also proposed a multi-stage NB architecture that predicts each stage of a multi-stage attack scenario. The paper also provided a comparative study of these two models based on the accuracy of prediction. Experiments carried out in this research proved that HMMs (~98%) provided higher accuracy than NB (~85%).

Furthermore, Saheed et al. enhanced the performance of IDSs using machine learning algorithms including XGB, CatBoost, KNN, SVM, Quadratic Discriminant Analysis (QDA) and NB on the UNSWNB-15 dataset [13]. In addition, the study included Min–Max normalization as well as Principal Component Analysis (PCA) for dimensionality reduction. CatBoost had the best results, with an accuracy of 99.99%, F1 measure of 99.99% and a Mathew Correlation Coefficient (MCC) of 99.97%.

Chen et al. proposed a DT-based model for Distributed Denial of Service (DDoS) attack detection, which was then tested using their own private dataset [14]. The dataset is divided into two categories— sensor data and network data. For the sensor data flood, the model obtained an accuracy of 97.39%. For the network data flood, the model achieved high accuracy of 99.98% as well as an F1 measure of 99.98%. While they discussed other machine learning techniques like CNNs, MLPs, LSTMs and SVMs, they focused on DTs and RFs due to their high accuracy and efficiency in the experimental setup. However, relying solely on tree-based methods limits the model's ability to capture complex, non-linear relationships and scale with larger datasets. Although effective in their context, these limitations may restrict the model's generalization capabilities and its application in more complex, dynamic attack scenarios such as in the dataset used in this study (see Section 3.1).

Similarly, Hamza et al. conducted a study that used two datasets to detect malicious URLs and intrusion [15]. They used a publicly available 'Network Intrusion' dataset with six classes of cyberattacks for intrusion detection and tested them against various ML models. Additionally, the study made use of dimensionality reduction. Amongst all classification models used, such as DT, NB, Support Vector Machine (SVM) and KNN, the study applied dimensionality reduction to optimize the dataset and found that the DT algorithm performed best, achieving an F1 score of 98% and an accuracy of 95.09% when configured with 49 nodes. This number of nodes represents the optimal balance between performance and model complexity, rather than a limitation.

In addition, Malhotra and Sharma evaluated popular ML models to detect intrusions in real time [16]. The tested ML algorithms included Bayes Net, NB, Random Tree, RF, J48, Bagging, PART, OneR, ZeroR and Logistic (a multinomial logistic regression model). From the results, it was seen that RF was the best model, yielding an accuracy of 99.91% and a low false alarm rate of 0.001. However, the RF model had the longest build time. Thus, the high accuracy does not guarantee the time efficiency of the model.

Similarly, Yulianto et al. proposed an improved solution to enhance the performance of AdaBoost-based IDSs to handle the imbalanced training data [17]. The proposed solution uses SMOTE to solve training imbalances, along with PCA and Ensemble Feature Selection (EFS) for feature reduction. The results showed that, with SMOTE and EFS, the model provided an accuracy of 81.83%, precision of 81.83%, recall of 100% and an F1 score of 90.01%.

Chaturvedi's study [18] analyzed the performance of KNN and NB for host-based intrusion detection using the MSCAD dataset. KNN demonstrated superior accuracy of 99.6%, significantly outperforming NB which achieved 80.4%. Precision and recall were also higher for KNN across most attacks, except for ICMP_Flood, where NB had better recall (92.86% vs. 71.43%). Despite these results, the study did not discuss preprocessing, feature selection or parameter tuning. These omissions represent limitations, particularly considering the high dimensionality and potential class imbalance in the dataset.

Hammad et al. tested various classification and clustering techniques to analyze intrusion detection and classify cyberattacks [19]. The classification algorithms used were NB, RF, J48 and ZeroR. For clustering, K-means and Expectation Maximization (EM) were used. It was seen that for clustering, the accuracy was higher for K-means than EM. On the other hand, among all the classification models, RF provided the best results, with an accuracy of 97.6% and an FPR of 0.03. However, the results require more in-depth analysis, since high accuracy does not guarantee efficient execution time.

Gu and Lu proposed an IDS model based on SVM and NB feature embedding [20]. In this IDS model, the datasets were tested using SVM models after undergoing feature embedding based on NB. Combining both SVM and NB models yielded better results, with an accuracy of 93.75% for the UNSW-NB15 dataset and 98.92% for the CICIDS2017 dataset. Nevertheless, the study only considered the binary case for intrusion detection, which is not fully representative of real-life scenarios. Table 2 summarizes the results of the work conducted on this problem using ML algorithms.

Table 2. Literature review summary for machine learning-based models.

Paper	Datasets	Technique	Best Results
Ingale et al. [12]	KDDCUP'99	Hidden Markov Model (HMM) and NB	Accuracy of 97.87% for HMMs
Saheed et al. [13]	UNSWNB-15	XGB, Cat Boost, KNN, SVM, Quadratic Discriminant Analysis (QDA) and NB	Accuracy of 99.99%, F1 measure of 99.99%, and Mathew Correlation Coefficient (MCC) of 99.97%
Chen et al. [14]	Private Dataset	DT	Highest accuracy was 99.98% and highest F1 measure was also 99.98%
Hamza et al. [15]	Network Intrusion Dataset	DT, NB, SVM, KNN	DT had accuracy of 95.09% and F1 score of 98%
Malhotra and Sharma [16]	NSL-KDD	BayesNet, NB, Logistics, Random tree, RF, J48, Bagging, PART, OneR, ZeroR, Logistic	RF has accuracy of 99.91% and low false alarm rate of 0.001
Yulianto et al. [17]	CIC IDS 2017	AdaBoost with SMOTE and PCA	Accuracy of 81.83%, F1 score of 90.01%
Chaturvedi [18]	MSCAD	KNN, NB	Highest accuracy for KNN was 99.6%
Hammad et al. [19]	UNSW-NB15	NB, RF, J48, ZeroR, K-means and Expectation Maximization (EM)	RF was best model with accuracy of 97.6% and FPR of 0.03
Gu and Lu [20]	UNSW-NB15 and CICIDS2017	SVM, NB feature embedding	Highest accuracy for SVM was 98.92%

2.3. *Deep Learning-Based Models*

Emerging deep learning models have also been proposed in this field and examined in multiple studies. Zhou et al. focused on a new detection solution using a sequence-to-sequence (seq2seq) model, which makes use of an LSTM algorithm [21]. The highest F1 measure obtained by the proposed model was 99.9%. Despite not being compared to recent studies other than those using HMMs, the model consistently had better precision than HMMs in all cases.

Fredj proposed a natural language processing (NLP)-based model using LSTM to predict multi-step cyberattacks [22]. They used the text-based SMIA2012 dataset to interpret multi-step attacks and create a universal attack prediction model. Though the dataset used was outdated, the best accuracy achieved was 98%.

Another technique proposed by Sohail et al. introduced intrusion detection using a multi-tiered ANN model [23]. In this study, the classification was conducted based on label, category and subcategory. The resulting accuracies were 99.9%, 99.7% and 97.7% for the previously mentioned classifications, respectively. Despite the model being able to produce very highly accurate results, it is computationally expensive to train.

Abdullah et al.'s study addressed the gap in using older datasets and utilized the MSCAD dataset to develop IDS models on realistic multi-step cyberattacks [24]. They evaluated four classifiers: Adaptive Boosting Deep Learning (ABDL), RF, Multi-Layer Perceptron (MLP) and a Bayesian Networks Model (BNM). The researchers applied a Correlation Ranking Filter to reduce the features from 66 to 31, thereby enhancing computational efficiency. Among these, ABDL achieved the highest accuracy of 99.71%, with 31 features, outperforming RF (99.68%) and BNM (97.25%). However, while the reduction in dataset features improved accuracy and computational efficiency, the reliance on a correlation-based feature selection method may limit the model's ability to capture non-linear relationships between features.

Thamilarasu et al. developed an anomaly-based intrusion-detection model using a sequential deep learning model for IoT networks [25]. The model resulted in an average precision rate of 95% and a recall rate of 97% for different attack scenarios. In addition, the model obtained higher F1 scores for all scenarios when compared with other prototypes. However, the dataset used for such analysis was not described, making it difficult to determine the reliability of the results.

Furthermore, Vinayakumar et al. explored a Deep Neural Network (DNN) model to predict unforeseen and unpredictable cyberattacks [26]. The model was tested on public datasets, namely KDDCUP'99 and NSL-KDD. The results showed that DNN topologies performed better than classical ML algorithms and the accuracy reported was between 95% and 99%. However, the datasets are outdated and do not show real-world network traffic.

In another paper, Faker and Dogdu combined big data and deep learning techniques to improve the performance of IDSs [27]. For this, they tested several algorithms, such as K-means clustering, Deep Feed-Forward Neural Network (DNN), RF and Gradient Boosting Trees (GBTs) on public datasets like UNSW NB15 and CICIDS2017. DNN models provided the highest accuracy for multiclass classification, with 97% for the UNSW NB15 dataset and 99.56% for the CICIDS2017 dataset. Despite the high accuracies, it should be considered that the datasets have several missing or infinity values. Such values can introduce uncertainties, disrupt the training process, and lead to suboptimal model performance by distorting feature representation.

Later, Xiao et al. proposed a CNN model as an intrusion detection system [28]. Here, the model applied dimensionality reduction to remove redundant data and hence provide better results. The reported accuracy was 94.0%, with a TPR of 93.0% and FPR of 0.5%. However, within the dataset, certain attacks, like U2R and R2L attacks, have considerably fewer data in the training dataset, which may result in a biased model.

Similarly, Ahmad et al. also proposed anomaly detection methods but using Mutual Information (MI) through a DNN for IoT security [29]. The models tested were a DNN, CNN and Recurrent Neural Network (RNN). Among these, the DNN produced the highest detection accuracy of 99% and a low false alarm rate of 3.9%. Despite great results, this anomaly detection is limited to binary class problems, which restricts its ability to differentiate between various types of anomalies.

Lastly, Khan et al. proposed a novel Two-Stage Deep Learning (TSDL) model for efficient network intrusion detection [30]. Using TSDL increased the quality of the results, with an accuracy of 99.931% for the KDD99 dataset and 89.711% accuracy for the UNSW-NB15 dataset. Despite the high accuracies, it should be noted that the datasets are outdated, do not showcase more evolving attacks and the UNSW-NB15 dataset skewed towards attack instances, which does not indicate a real-world network traffic profile. Table 3 portrays a summary of the DL models implemented for this problem.

Table 3. Literature review summary for deep learning-based models.

Paper	Datasets	Technique	Best Results
Zhou et al. [21]	DARPA2000. ISCXIDS2012, CIC-IDS2017 and CSE-CIC-IDS2018	sequence-to-sequence (seq2seq) model (uses LSTM)	Highest F1 score was 0.999
Fredj [22]	SMIA2012	Natural Language Processing (NLP)	Accuracy of 98%

Table 3. *Cont.*

Paper	Datasets	Technique	Best Results
Sohail et al. [23]	IoTID 20	ANN	99.9% accuracy for label classification
Abdullah et al. [24]	MSCAD	ABDL, MLP, BNM, RF	Highest accuracy of 99.71% by ABDL
Thamilarasu et al. [25]	-	Sequential Deep learning model (linear stack of DNN layers)	F1 scores for all scenarios were greater than or equal to 97%
Vinayakumar et al. [26]	KDDCUP'99, NSL-KDD	DNN	DNN topologies showed training accuracy between 95% to 99%
Faker and Dogdu [27]	UNSW NB15, CICIDS2017	K-means clustering, DNN, RF, Gradient Boosting Tree (GBT)	DNN showed the highest accuracy of 99.56%
Xiao et al. [28]	KDDCUP'99	Convolutional Neural Network (CNN)	AC, DR (TPR), and FAR canreach 94.0%, 93.0%, and 0.5%
Ahmad et al. [29]	IoT-Botnet 2020	DNN, CNN, RNN	DNN showed the highest detection accuracy of 99.010% and an FAR of 3.9%
Khan et al. [30]	KDD99 and UNSW-NB15	Two-stage Deep Learning model (TSDL)	Highest accuracy of 99.931%

2.4. Additional Methods

In addition to machine learning and deep learning methods, various other models have been proposed in the literature. For example, Mao et al. used a multi-information fusion (MIF) model to reconstruct attack scenarios and extract attack chains [2]. The MIF model is a computation-based model that uses machine learning techniques. The model achieved high detection rates, low false alarm rates, and an accuracy of 98%, making it a promising approach for IDSs. However, the model relies on multiple information sources, which can be seen as a limitation since there is often missing information when dealing with networks.

Another method proposed by Li et al. utilizes a hierarchical and dynamic feature extraction framework (HDFEF) to extract features from network flows for IDSs [31]. Through experiments, the model has been shown to outperform other feature extraction methods, achieving a recall value of 99.96% and an F1 score of 99.84%. Even though the model's hierarchical structure is a major strength, its reliance on stationary network traffic data is a potential weakness.

Sen et al. proposed a technique for the detection of multi-stage cyberattacks with the aid of a graph-based cyber intelligence database and alert correlation approach [32]. Their approach detects multi-stage attacks by leveraging heterogeneous data to form a knowledge base and employs a model-based correlation approach to the generated alerts to identify multi-stage cyberattack sequences taking place in the network. They investigated the detection quality of the proposed approach by using a case study of a multi-stage cyberattack campaign in a future-orientated power grid pilot. Their alert correlation process results in a risk probability (P_r) for various attack steps.

Angelini et al. proposed a visual analytics solution that aims to help security operators enhance their network security. The solution uses interactive visualizations for quick insights, action prioritization and on-line correlation to identify and detect multi-step cyberattacks [33]. The on-line correlation method detects ongoing attacks based on the topological structure of attack paths. Though they discuss correlation metrics like Jaccard similarity and Cosine similarity, the paper does not report the results of these metrics, nor the accuracy. Instead, the focus of the paper is on the usefulness and testing of the model.

Shawly et al. proposed a multi-HMM-based detection architecture, also known as MulHMMs [34]. Using the DARPA2000 dataset, the model was tested based on detection accuracy. The results of the paper show that the detection accuracy for MulHMMs is better than that of the generic architecture for all the presented scenarios. However, the paper does not report exact accuracy values or other performance metrics, making it hard to analyze the results.

Furthermore, a technique called a Multi-Step Attack Alert Correlation (MAAC) system was proposed by Wang et al. to find attack paths and reduce the number of false alerts [35]. The metrics used to study the performance of the model included path detection rate, which is the ratio of detected attack paths to all attack paths, and false path rate, which is the rate of reporting wrong paths as attack paths. The results showed that the model was able to reduce duplicated alerts by 90%, had a path detection rate of 100%, and a false path rate of 0%. However, this model did not use generic metrics and used outdated datasets such as the DARPA 2000 LLDOS1.0 dataset, UNB ISCX IDS 2012 dataset and NDSec dataset. Furthermore, MAAC relied on pre-defined signatures to categorize alerts, and this dependency might let emerging threats, or zero-day attacks go undetected. In addition, the paper does not report precision or recall, even though it is dealing with false alerts.

Moreover, Zhang et al. proposed a method to cluster alerts with similar semantics into the same attack phase based on the semantic information of alerts [36]. Such a method included MSA detection based on HMM. Unfortunately, the paper does not mention the exact values of the performance metrics. However, the model was reported to be computationally inexpensive as compared to current models. Table 4 summarizes the previous studies conducted using additional methods aside from ML and DL.

Table 4. Literature review summary for additional methods.

Paper	Datasets	Technique	Best Results
Mao et al. [2]	DARPA1999/DARPA 2000 and CICIDS2017	Multi-information fusion (MIF) and CTnet	Accuracy of 99%
Li et al. [31]	CSE-CIC-IDS2018, CIC-IDS2017 and UNSW-NB15	Hierarchical and Dynamic Feature Extraction Framework (HDFEF)	Recall value of 99.96% and F1 score of 99.84%
Sen et al. [32]	-	Graph-Based Correlation	Predicted risk and actual occurrence of one class of 96%
Angelini et al. [33]	Private data from the ACEA organization	On-line correlation	Developed a visual analytics environment
Shawly et al. [34]	DARPA 2000	Multi-HMM based detection architecture (MulHMMs)	Detection accuracy for MulHMMs is better than that of generic architecture
Wang et al. [35]	DARPA 2000 LLDOS 1.0, UNB ISCX IDS 2012 and NDSec	Multi-Step Attack Alert Correlation (MAAC)	Reduced duplicated alerts by 90%, had a path detection rate of 100%, and a false path of 0
Zhang et al. [36]	DARPA 2000, DEFCON21 CTF and ISCXIDS 2012	MSA detection based on HMM	Proposed model had higher accuracy, precision, recall and F1 score than Baum–Welch, K-means, and TL

3. Datasets and Methods

3.1. Dataset

The dataset used in this work is MSCAD [4]. It consists of two multi-step cyberattack scenarios, which are a password cracking attack and a volume-based DDoS attack [1]. The dataset was used to train the IDS. It is a numerical dataset in which the type of attack is labeled as either, 'Brute Force', 'Port Scan', 'Normal', 'HTTP DDoS, 'ICMP Flood' or 'Web Crawling'. The dataset consists of 128,800 labeled records representing different multi-step cyberattacks, each having 66 features. These features for the forward and backward traffic direction include the following:

- Total, size, minimum, maximum, mean, standard deviation and average size of the packet.
- Minimum, maximum, mean, standard deviation and total time between two packets sent.
- Number of flow bytes and packets per second.
- Number of times PSH and URG flags were set.
- Total bytes used for headers.
- Minimum, maximum, mean, standard deviation and variance length of a packet.
- Number of packets with FIN, SYN, RST, PUSH, ACK, URG, CWE and ECE.
- Average number of packets and bytes in a sub flow.
- Download and upload ratio.
- Total number of bytes sent in an initial window.
- Count of packets with at least 1 byte of TCP data payload in the forward direction.
- Minimum, maximum, mean and standard deviation time of a flow-changing state between active and idle.

As mentioned earlier, the MSCAD includes two multi-step cyberattack scenarios. The two multi-step attack scenarios were performed as follows:

- Multi-Step Attack Scenario A

In this scenario, an attacker aims to perform a password cracking attack (Brute Force) on any host within the victim network. The attacker executes this attack in three main sequential steps. Firstly, the Port Scan was executed simultaneously. Secondly, the HTTrack Website Copier was used as a website crawler tool to make an offline copy of the web application pages. Using a password list of 47 entries and a user list of 10 entries resulted in 470 attempts to crack the password. Finally, the Brute Force script was executed [6].

- Multi-Step Attack Scenario B

In scenario B, the attacker aims to execute the volume-based DDoS on any host within the victim network. The volume-based DDoS was performed based on three sequential steps. The first step of the volume-based DDoS attack is to execute the Port Scan attack (Full, SYN, FIN and UDP Scan) simultaneously. Then, the next step is to launch the APP-based DDoS attack using HTTP Slowloris DDoS attack. Finally, the volume-based DDoS attack was executed using the Radware tool. This scenario took an hour, and three hosts were infected by the volume-based DDoS attack [6].

Table 5 shows the distribution of the MSCAD records in each of the classes. As can be seen from the table, these classes are heavily imbalanced. The classes—Web Crawling, ICMP Flood and HTTP DDoS—comprise a very small number of records. Thus, a class imbalance handling approach was considered, as can be seen in Section 4.

Table 5. Distribution of MSCAD dataset.

Type of Attack	Number of Records
Web Crawling	28
Port Scan	11,081
Brute Force Attack	88,502
HTTP DDoS	641
ICMP Flood	45
Normal Traffic	28,502

3.2. Data Preprocessing Methods

To handle the class imbalance problem that is evident in the dataset, an Adaptive Synthetic (ADASYN) technique was employed to generate new synthetic data [37]. The core concept of an ADASYN is to employ a weighted distribution for various minority class examples depending on how challenging they are to learn. More synthetic data are created for difficult-to-learn minority class examples than for easier-to-learn minority examples. In light of this, the ADASYN technique enhances learning with regard to the data distribution in two ways: (1) minimizing the bias caused by the class imbalance, and (2) adaptively pushing the classification decision boundary toward the challenging examples, which is an added feature compared to the usual class balancing techniques, such as SMOTE, which may otherwise lead to overfitting. Furthermore, techniques such as oversampling also increase the likelihood of overfitting, since they make exact copies of the minority classes, which is undesirable.

Figure 1 shows the data pre-processing pipeline considered in this paper. As can be seen in the figure, the correlated features were dropped from the dataset to reduce the model complexity. This process resulted in retaining 28 out of the originally existing 66 features. Following the feature reduction, the labels were encoded as follows: 0 for Brute Force, 1 for HTTP DDoS, 2 for ICMP Flood, 3 for Normal, 4 for Port Scan and 5 for Web Crawling. After that, the remaining features were scaled to ensure that each feature had a zero mean and unit standard deviation. Subsequently, the ADASYN technique was used for class balancing the training set. Lastly, PCA was performed on the training data for dimensionality reduction as well. It is worth noting that we applied the ADASYN technique only on the training set, which comprises 67% of our full dataset. The testing data, which comprise 33% of our full dataset, were not oversampled and were used for the purpose of testing our trained model to correctly determine the goodness of our models without considering the generated synthetic data. Therefore, the models used in this paper were solely tested on the original data.

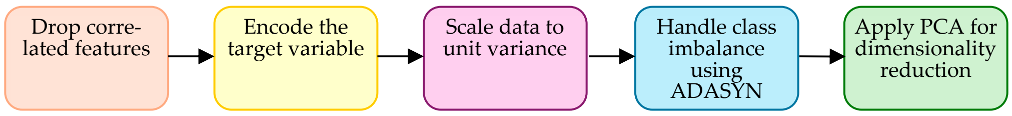

Drop corre-lated features
Encode the target variable
Scale data to unit variance
Handle class imbalance using ADASYN
Apply PCA for dimensionality reduction

Figure 1. Data pre-processing pipeline considered on the dataset.

3.3. Machine Learning Methods

Several machine learning methods were employed in this paper, namely DT, KNN, NB, SVM, XGB, LGBM and RF. Each of these methods is described as follows:

Decision Tree (DT): This is a well-known machine learning algorithm that can be used for classification and regression. DT is based on the concept of recursive partitioning of the samples with respect to a certain attribute [38]. The algorithm allows users to build a tree-like model by splitting the given data into smaller subsets based on the most significant features. The objective of this method is to create a model that predicts the target variable by minimizing the impurity of the subsets created by splitting. The impurity can be calculated using the *Gini index*, $p_i(t)$ is the frequency of class i at node t, and c is the total number of classes, as in

$$Gini\ index = 1 - \sum_{i=0}^{c-1} p_i(t)^2 \tag{1}$$

In addition, the impurity can also be calculated using entropy, as in

$$Entropy = -\sum_{i=0}^{c-1} p_i(t) \log_2 p_i(t) \tag{2}$$

DT can handle categorical and numerical data and is also useful for real-life applications, since it is robust to missing data.

K-Nearest Neighbor (KNN): This is used for classification and regression problems and classifies new samples by calculating the distance between the new sample and all the records using distance measures [39]. In this work, the Euclidean distance measure was considered where i corresponds to the feature number, as in

$$D(p,q) = \sqrt{\sum_i (p_i - q_i)^2} \tag{3}$$

Next, the algorithm selects the K nearest records and assigns the new sample to the most common class among those K neighbors. Despite being sensitive to the choice of K, this algorithm can handle complex decision boundaries and is easy to implement. However, it can suffer from issues with dimensionality when handling data with high dimensions.

Naïve Bayes (NB): This is a classification algorithm that depends on probability. The algorithm is based on Bayes' theorem, which is implemented by obtaining the probability of having a record belonging to a specific class given its features [40]. Equation (4) estimates the probability of class Y given the feature vector X comprising the different features X_i, where i corresponds to the feature number:

$$P(Y|X_1X_2\ldots X_d) = \frac{P(X_1X_2\ldots X_d|Y)P(Y)}{P(X_1X_2\ldots X_d)} \tag{4}$$

NB assumes that the features being used are mutually independent. Therefore, it struggles with correlated features. Regardless, the algorithm can work with large datasets and is computationally efficient.

Support Vector Machine (SVM): works by finding the best hyperplane that separates all data points of one class from those of another class. The best hyperplane can be defined as the one with the largest margin between two classes [41]. If the given data are linear, a decision boundary can be found where $\vec{w}$ includes the coefficients of the features and b is the y-intercept, as in

$$\vec{w}\cdot\vec{x} + b = 0 \tag{5}$$

However, if the given data are non-linearly separable, the algorithm then maps the data into a higher-dimensional space in order to separate the data using certain parameters. The decision boundary can be found, as in

$$\vec{w}\cdot\Phi(\vec{x}) + b = 0 \tag{6}$$

This equation represents the hyperplane in the transformed feature space that separates the data. $\vec{x}$ is the input feature vector which is transformed into a higher-dimensional space using a function $\Phi(\vec{x})$. $\vec{w}$ is the weight vector that defines the orientation of the decision boundary. b is the bias term that allows us to shift the decision boundary as needed. The goal of this is to find the optimal values of the parameters such that the decision boundary maximally separates the classes while minimizing classification error.

As a result, the algorithm works well with high-dimensional spaces and can handle non-linear decision boundaries. Nevertheless, the method depends on the choice of the kernel function used, such as the radial basis function kernel (RBF) where the squared Euclidean distance between two features is computed, as in

$$K(x,y) = e^{-||x-y||^2/(2\sigma^2)} = e^{-\gamma||x-y||^2} \tag{7}$$

where $||x-y||$ represents the Euclidean distance between vectors x and y and σ^2 is a free parameter describing the width of the Gaussian bell curve and influencing the smoothness of the decision boundary. Having a smaller value for this parameter leads to more influence of individual data points, which in return makes the decision boundary more complex.

The decision boundary can be controlled making sure there is a good compromise between its smoothness and the accuracy of the classification.

Light Gradient-Boosting Machine (LGBM): This is another well-known machine learning algorithm used for classification and regression [42]. It is mainly known for its speed, efficiency and performance. The algorithm works by combining weak models together and gradually improving their accuracy. Through the implementation of smart optimization techniques based on gradients, the algorithm trains the models in a boosting manner. In other words, each model works on reducing the errors made by previous models. An LGBM can handle large datasets and provide good results with limited resources. Using hyperparameters related to the tree structure, learning rate and early stopping, it is possible to adapt the model to specific requirements.

Extreme Gradient Boosting (XGB): This is a machine learning algorithm used for classification and regression purposes [43]. Similar to LGBM, it is also known for its efficiency and speed. While LGB follows a leaf-wise tree growth approach, XGB follows a level-wise tree growth strategy. By doing so, the algorithm processes all the leaves at a certain level before moving on to the next level. The algorithm is suitable for large scale datasets and offers more flexibility when it comes to fine-tuning the model.

Random Forest (RF): This is also used for classification and regression. This algorithm combines several decision trees together to provide predictions [44]. The main concept of this method is to create a forest of decision trees that are trained on different subsets of the data. In this case, each decision tree is constructed based on a random subset of data and features. Such a technique ensures that there is diversity among the models. The final prediction is obtained through an averaging mechanism based on the predictions of each tree in the forest. RF aids in overcoming the likelihood of overfitting since it makes use of a combination of trees. Moreover, the model is easy to use and only requires tuning of a small number of parameters.

3.4. Feature Selection and Dimensionality Reduction

Feature selection and dimensionality reduction are used to improve computational efficiency and aid in attaining a much clearer approach to the more useful features in a dataset, as machine learning algorithms would now have more specific features to learn. Through these methods, it is possible to reduce the number of input features while maintaining the most relevant information. Feature selection is achieved by using a correlation matrix and eliminating the correlated features with Pearson correlation coefficients greater than 95%. The choice of a 95% threshold was made to ensure that only highly redundant features were removed, preserving the diversity of features while avoiding overfitting.

Next, PCA is used to reduce the dimensions in the dataset. PCA is a data transformation technique that places the data into a new feature space where the first coordinate of the new space (known as the first principal component) represents the majority of the variance in the data, the second principal component represents the second greatest source of variance in the data and so on. PCA has been found to be an efficient dimensionality reduction method in the domain of cybersecurity [45,46]. In this project, the features with the highest explained variance ratio were chosen to be the classifiers' input features. The number of PCA components chosen is discussed in Section 4. However, the results without the use of PCA appeared to be higher than those with PCA. For this reason, the evaluation of the models was then performed without PCA, as shown in the experimental results.

3.5. Model Selection

In this work, hyperparameter tuning was conducted using GridSearchCV with five-fold stratified cross-validation. During this process, a weighted F1 score was used as the scoring metric to identify the optimal hyperparameter configuration for each model.

For each algorithm, the mean weighted F1 score across the five folds during cross-validation is reported to provide insights into model performance during tuning. After selecting the best hyperparameters, the models were retrained on the entire training dataset and evaluated on a separate test dataset.

3.6. Evaluation Metrics

There are several evaluation metrics used in this work to evaluate the performance of different machine learning models.

A Confusion Matrix is a tabular matrix that displays the predictions of a classification problem. The matrix plots the actual values and predicted values for each class, thereby providing a visual representation of the results.

Accuracy describes the number of correctly classified samples divided by the total number of samples. From the confusion matrix, accuracy is mathematically defined as the ratio of the sum of true positive and true negatives samples over the total number of samples, as in

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \tag{8}$$

Precision provides a measure of how exactly the samples were classified. Thus, precision is a metric that checks the quality of the classification process, as in

$$Precision = \frac{TP}{TP + FP} \tag{9}$$

Recall indicates a measure of completeness, or how many of the samples were able to be classified. Thus, recall relies on the quantity of samples that were classified, as in

$$Recall = \frac{TP}{TP + FN} \tag{10}$$

F1 Score is a metric that measures accuracy by taking into consideration both precision and recall. This is obtained from the weighted harmonic mean of the precision and recall, as in

$$F - measure = \frac{2TP}{2TP + FN + FP} \tag{11}$$

The Receiver Operating Characteristic (ROC) Curve is a probability curve that displays the trade-off between the actual positive instances correctly identified, and the actual negative instances that are incorrectly identified as positive. The plot is obtained between the True Positive Rate (TPR) vs. False Positive Rate (FPR), and the performance of the model is depicted as a point on the ROC curve. TPR is equal to the recall found in (10), while FPR is calculated as

$$FPR = \frac{FP}{TN + FP} \tag{12}$$

The Area Under Curve (AUC) is a measure of the classifier's ability to distinguish between positive and negative classes. The higher the area under the ROC curve, the better the performance of that model is in distinguishing positive and negative classes.

4. Implementation and Results

4.1. Implementation and Experimental Setup

Before presenting the results, it is essential to outline the implementation process used for applying the machine learning models. The models were implemented in Python using well-known libraries such as scikit-learn for machine learning algorithms, imbalanced-learn for the ADASYN technique, and NumPy and Pandas for data preprocessing and manipulation. Visualization of results was performed using Matplotlib and Seaborn.

The experiments were conducted on Google Colab, utilizing its free-tier environment with 51 GB of RAM. While no dedicated GPU was employed for this work, the high RAM capacity facilitated efficient handling of the dataset and preprocessing steps.

Each algorithm underwent hyperparameter optimization using GridSearchCV combined with a five-fold stratified cross-validation strategy. This approach ensured robust performance evaluation by maintaining consistent class distributions across folds and minimizing bias in performance metrics.

The dataset was preprocessed to enhance model performance and reliability. This preprocessing included balancing class distributions using the ADASYN technique, which addresses imbalances by generating synthetic samples for minority classes. Features were scaled to unit variance to standardize their range, and target labels were encoded for compatibility with the learning algorithms.

After preprocessing, hyperparameter tuning was performed to identify the optimal configurations for each model. The models were then trained on the processed training dataset and evaluated on a separate test dataset using key metrics such as F1 score, accuracy, precision, recall and the area under the curve (AUC). This systematic approach ensured that the results were both reliable and comparable across the evaluated models.

4.2. Data Preprocessing and Feature Selection

In this section, the results of data preprocessing and feature selection are presented. Table 6 summarizes the instances of each class in the dataset after carrying out the ADASYN technique.

Table 6. Distribution of training data after applying the ADASYN technique.

Type of Attack	Number of Samples
Web Crawling	59,188
Port Scan	59,192
Brute Force Attack	59,188
HTTP DDoS	59,207
ICMP Flood	59,182
Normal Traffic	59,162

Figure 2 shows a correlation matrix heatmap that visually shows the features with highest correlation. Pearson correlation coefficients were employed to compute feature correlations. It was noted that 38 features were found to be highly correlated (with correlation coefficients greater than 95%) and were subsequently dropped. The correlation matrix after dropping the correlated features is illustrated in Figure 2. Note that only the features with correlations higher than 95% were dropped while correlated features with a correlation value below the threshold would still be considered.

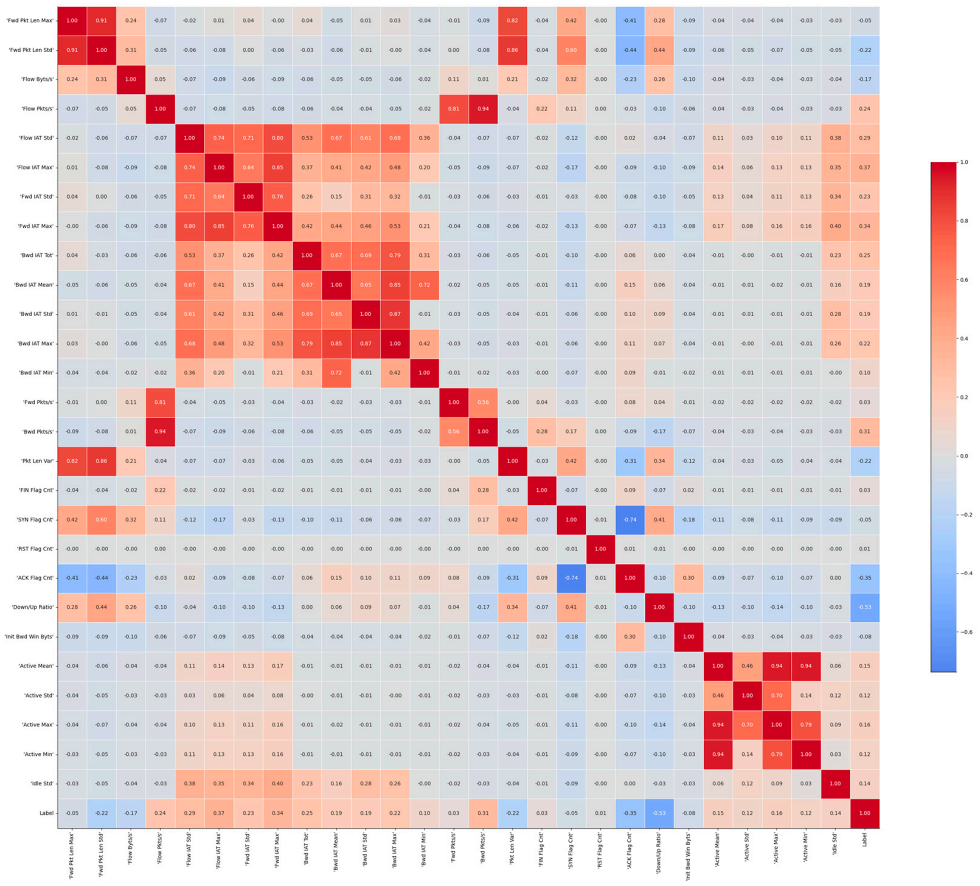


Figure 2. Correlation matrix between the dataset features.

4.3. Principal Component Analysis (PCA)

In order to effectively carry out PCA, we first created a machine learning pipeline, which is defined as the end-to-end system that coordinates the input of data into and output from a machine learning model (or group of several models) [47]. A machine learning pipeline was created for every model to search for the best combination of the number of PCA components and classifier parameters through a five-fold stratified cross validation. Table 7 shows the different models used in this paper with the best number of principal components considered, best values for the hyperparameters and the resulting training and testing F1 scores. As can be seen, KNN attained the highest training and testing F1 scores with its respective optimal number of components and parameters.

Table 7. Results of applying PCA to different models.

Model	Number of PCA Components Chosen	Best Values of Hyperparameters	Training F1 Score	Testing F1 Score
DT	20	Min samples split: 10 Max depth: 20 Criterion: gini	97%	80%
KNN	25	N_neighbors: 2	98%	82%
Naive Bayes	20	Var_smoothing: 1×10^{-8}	54%	45%
SVM (Linear)	25	tol: 1×10^{-5}	80%	50%
SVM (RBF)	20	C: 10, gamma: 0.1	90%	54%
LGB	20	Learning rate: 0.1, boosting: gbdt	97%	81%

Table 7. *Cont.*

Model	Number of PCA Components Chosen	Best Values of Hyperparameters	Training F1 Score	Testing F1 Score
XGB	20	Learning rate: 0.1, Booster: gbtree	97%	80%
RF	25	Criterion: entropy, Min_samples_split: 15	96%	80%

4.4. Machine Learning

In this section, the results of the different machine learning methods utilized in this work are presented. In this work, several methods were considered, namely DT, KNN, NB, SVM, XGB, LGBM and RF. As the evaluations of the models were conducted, it was shown that the use of PCA did not achieve the highest results. In fact, results were higher when the models were implemented without PCA. Therefore, a GriFid search was performed with five-fold cross validation on each of the models to identify the best set of hyperparameters that maximize the models' performance. The following subsections describe the specifications of the machine learning models used without the use of PCA.

4.4.1. Decision Tree

It was found that the best hyper parameters for the DT are the entropy as a criterion with a max number of nodes of 20 and minimum samples split of 30. We were able to obtain a testing accuracy score of 100% and an F1 score of 83% on the testing set. The Decision Tree model was able to accurately identify the classes for many samples, subsequently increasing the accuracy. Furthermore, we were able to obtain a 91% AUC score as well on the testing data, thereby showing that the model can efficiently distinguish between the positive and negative classes.

4.4.2. KNN

Similarly, a grid search yielded the best hyperparameter for the number of neighbors 'n_neighbors' to be three. This classifier's performance was also quite satisfactory as it attained results quite close to that of DT with a testing accuracy of 99%, an F1 Sore of 82% and an AUC score of 92% on the testing data. The confusion matrix for KNN can detect the samples accurately for most classes; however, it is less accurate when compared to the previously seen DT model. Despite this, the KNN model performed decently at classifying class 5, as suggested by its ROC performance. While the ROC curve for KNN is not explicitly shown, the figure presenting the ROC of the best-performing model illustrates the trends for this challenging and underrepresented class in the dataset.

4.4.3. NB

For the NB algorithm, a Gaussian model was implemented since we are dealing with continuous features. The best hyperparameter value found after a grid search was 1×10^{-20} for 'var_smoothing'. The confusion matrix identifies some of the samples correctly, but for class 5, there are zero samples predicted to be true. Additionally, many samples have been misclassified and predicted incorrectly to belong to different classes.

This model had the worst performance in comparison to the other models with a testing accuracy score of 76%, an F1 score of 29% and an AUC score of 74% on the testing data. Although this model is one of the least computationally expensive models, it assumes the independence of all features, which is a major drawback, especially for a dataset that describes network traffic where the likelihood of a feature being affected by a combination of other features, which in turn predict the likelihood of a type of attack, is usually high.

4.4.4. SVM with RBF Kernel

For SVM with RBF kernel, a grid search yielded the best hyperparameters of 10 for C and 0.1 for gamma. The results attained using this model are quite average, with an accuracy of 93% on the testing data, an F1 score of 58% and an AUC score of 92% on the testing data. It is worth noting that this classifier, however, did the best job at classifying class 5, as was noted from its confusion matrix. Furthermore, the AUC scores for most of the classes were relatively high, thereby showing the model's good performance in distinguishing different classes, including the hard to learn classes.

4.4.5. SVM with Linear Kernel

The best hyperparameter found through grid search is tol: 1×10^{-5}. This model also showed quite average results, with the testing accuracy being 88%, an F1 score of 52% and an AUC score of 86% on the testing data. Furthermore, it is worth noting that this model does not detect class 5 at all which is also the most underrepresented class in the original dataset.

4.4.6. LGBM

The results obtained using LGBM were similar to those obtained from DT and KNN. A grid search showed the best hyperparameter for the boosting technique to be gbdt and the learning rate to be 0.01. The classifier's performance was also quite satisfactory, with a testing accuracy of 100%, an F1 score of 81% and an AUC score of 90% on the testing data. The confusion matrix for the LGBM can detect the samples quite accurately for all the classes.

4.4.7. XGB

XGB was our best performing model, with the grid search showing the best hyperparameters for the booster to be gbtree and the learning rate to be 0.01. The model results for the testing data included an accuracy of 100%, precision of 93%, recall of 85%, an F1 score of 88% and an AUC score of 93%. Similar to the LGBM, the confusion matrix for XGB can also detect the samples accurately for all the classes.

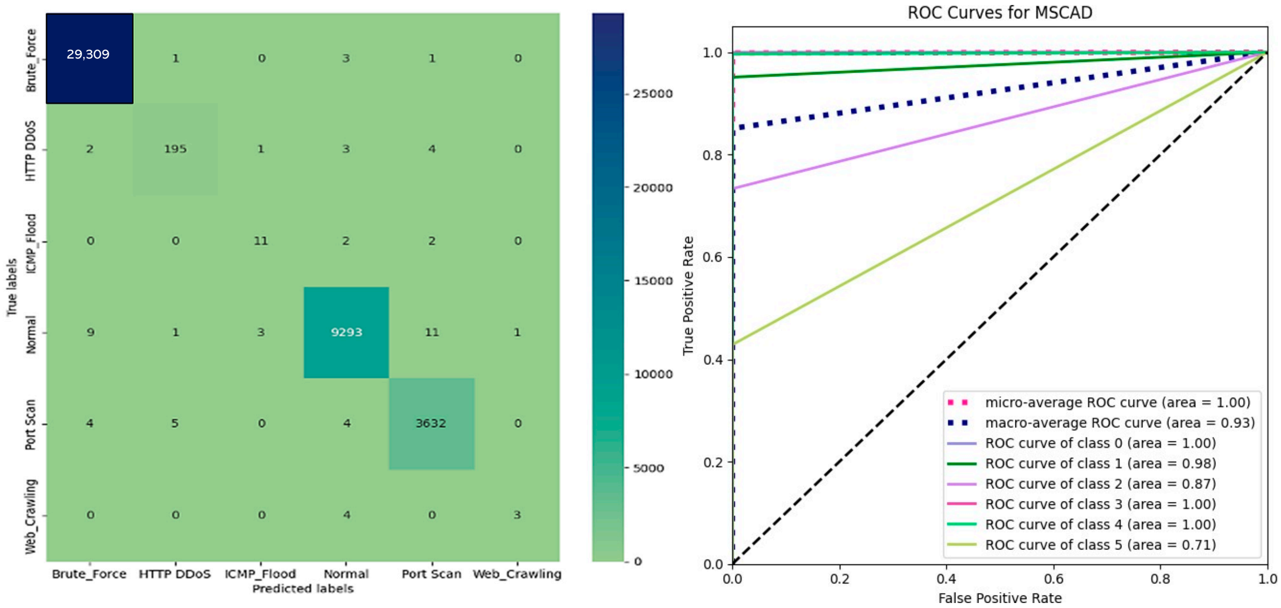
4.4.8. RF

The best hyperparameters found through grid search for RF are entropy for criterion and a minimum sample split of 25. The model results for the testing data were similar to those of XGB, with an accuracy of 100%, precision of 91%, recall of 83%, an F1 score of 86% and an AUC score of 91%. The confusion matrix of RF was able to detect most classes accurately; however, it was unable to appropriately classify class 5. Table 8 presents a comprehensive summary of the performance metrics for each of the machine learning models used.

Table 8. Best performing models' results for testing data without PCA.

Metric	DT	KNN	Naïve Bayes	SVM (Linear)	SVM (RBF)	LGB	XGB	RF
Mean F1 Score (CV)	74%	96%	50%	73%	86%	96%	99%	99%
Accuracy	100%	99%	76%	88%	93%	100%	100%	100%
Precision	83%	81%	38%	55%	54%	78%	93%	91%
Recall	83%	84%	54%	76%	86%	85%	85%	83%
F1 Score	83%	82%	29%	52%	58%	81%	88%	86%
AUC	91%	92%	74%	86%	92%	90%	93%	91%

Seeing that XGB performs the best, we further analyzed the confusion matrix and the ROC curve, as shown in Figure 3.



(**a**) (**b**)

Figure 3. XGB model performance resulting from applying the model to evaluation data: (**a**) confusion matrix and (**b**) ROC curve.

In Table 8, the results of the machine learning models implemented are presented and compared. Grid Search was used to obtain the best set of hyperparameters for each of the models. Subsequently, each model was implemented and the respective confusion matrix, ROC curves, AUC score and other evaluation metrics were obtained. From Table 8, we can see that XGB had the best results, with an accuracy of 100%, precision of 93%, an AUC score of 93% and an F1 score of 88%. However, nonlinear SVM had the best results for recall, with 86%. Additionally, NB had the worst results for our data, since the algorithm assumes independence of features. The results for the models presented in this paper are compared with previous studies in Table 9.

Table 9. Comparison with previous studies.

Work	Best Model	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)	AUC (%)
Mao et al. [2]	CTnet	99	97	95	96	-
Dhanya et al. [5]	DT	99.05	99	99	99	-
Dalal et al. [6]	Extremely Boosted Neural Network	99.72	-	-	-	-
Pelletier et al. [8]	RF	96.4	-	-	-	-
Gan et al. [9]	CNN-IDMDI	98.73	98.75	98.73	98.74	-
Qaddoura et al. [11]	SLFN	98.42	98.79	99.94	-	-
Ingale et al. [12]	Service-Based HMM	97.87	-	-	-	-
Saheed et al. [13]	XGB	99.99	100	-	99.99	-
Chen et al. [14]	DT	99.98	99.98	99.98	99.98	-
Hamza et al. [15]	DT	95.09	79–100	78–100	78–100	-
Yulianto et al. [17]	AdaBoost	81.83	81.83	100	90.01	-

Table 9. *Cont.*

Work	Best Model	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)	AUC (%)
Hammad et al. [19]	RF	97.6	97.6	97.6	97.6	-
Zhou et al. [21]	LSTM	-	-	-	99.9	-
Fredj [22]	LSTM	98.18	98	-	98	-
Sohail et al. [23]	ANN	99.9	99.0	99.8	99.4	-
Thamilarasu et al. [25]	DL	-	95	97	97	-
Khan et al. [30]	TSDL	99.93	-	-	-	-
Li et al. [31]	HDFEF	99.73	99.73	99.96	99.84	99.83
Wang et al. [35]	Logistic regression	90.56	-	-	90.47	98
Proposed Work	XGB	100	93	85	88	93

5. Discussion

In this section, the results of the proposed approach are discussed and compared to similar studies in the literature. In this study, several machine learning models were utilized, namely DT, KNN, NB, SVM, LGBM, XGB and RF. Among these models, the XGB model was able to outperform other models due to its speed, efficiency and ability to work well with large scale datasets such as the one used in this paper. In addition, XGB is flexible enough to incorporate several fine-tuning methods. In this paper, the ADASYN technique was used to handle class imbalances by generating synthetic data for the minority classes. Furthermore, the benefit of the ADASYN technique is that it uses weighted distribution to decide the amount of synthetic data generated in such a way that the complex minority class will have more synthetic data than the simpler minority classes. However, it is important to note that oversampling was performed only on the training set and the testing set was kept as it was. This ensured that the results of the model's performance on the test set were as reliable and accurate as they were on the original data samples. From the models implemented in this paper, XGB produced 100% accuracy and an 88% F1 score, which are more representative and reliable metrics for model performance as compared to using only accuracy, which can be misleading.

In comparison to other studies, Table 9 summarizes the results of machine learning models that are comparable to the one proposed in this paper. As seen in Table 9, the highest AUC results are in [31]. However, as mentioned in Section 2, the models in [31] rely mainly on stationary network traffic data, which is not representative of current multi-stage attacks. Similarly, in [2], the results indicate a very high accuracy of 99% for both the datasets. But it is important to note that the model used in [2] does not handle fake IP addresses.

In addition, [5] employed several models, such as SVM, DT, RF, Adaboost, XGB and KNN. Among these, the highest accuracy was recorded for DT as 99.05%. However, there was no discussion about feature selection or the implementation details in [5]. Next, [6] used the same dataset (MSCAD) as this study, but it only checked the accuracy of the neural network models. The neural network models provided an accuracy of 99.72%, but the neural network architecture was ambiguous, and the paper did not report any performance metrics other than accuracy, which makes it difficult to evaluate the reliability of the methods used. A similar case is seen in [12], which used a Hidden Markov Model and NB with preprocessing and dimensionality reduction, but only reported accuracy as a metric. It was found that the Hidden Markov Model provided the highest accuracy of 97.87%.

Furthermore, in [32], graph-based correlations were used to detect multi-stage attacks and the technique used was cross-domain descriptive ontology. This was tested on real-time population; however, common metrics were not used for reporting the results. In this paper, correlation was used for pre-processing and not as a results metric; hence, the results of [32] cannot be compared with those achieved in this study.

A similar case can be seen in [33], where on-line correlations were used to identify and detect attacks but the accuracy of the correlation results is not reported. Thus, the results are unclear, and the paper shifts focus from detection to usefulness and testing of the environment. Another example is seen in [35], where an alert correlation method is used to find attack paths and reduce the numbers of false alerts. However, it does not utilize generic metrics; instead, it introduces path detection rates and false path rates as result metrics.

Also, [35] did not report precision and recall, although the study did work with false alerts. Moreover, in [22], an NLP prediction model was used which was based on LSTM to predict multi-stage cyberattacks and the accuracy reached 98%. This result was highly accurate; however, since the study used an outdated text-based dataset from 2012, it cannot be compared with the results of this paper. Datasets which were introduced earlier, like the previous one in [22], can be considered as outdated datasets. It is important to note that the MSCAD dataset used in this paper was introduced in 2022 and, hence, it is considered to be more relevant as compared to datasets introduced before that.

In addition, [34] used multi-HMM-based detection architecture to detect multi-stage attacks; however, it did not report accuracy, and other results were also unclear. Further, in [9], the authors aimed to improve the performance of IDS using neural networks and the Convolutional Neural Network Intrusion Detection Method and the accuracy obtained was 98.73%. Since a highly complex model was used, there is a chance that overfitting led to high accuracy.

Similarly, the highest precision and F1 scores are in [13]. However, the paper used an outdated dataset. Furthermore, DT was employed in [14] to detect DDoS attack and the highest accuracy recorded was 99.98%, with an F1 score of 99.98% as well. Nevertheless, the paper only focused on four types of attacks. Likewise, [15] also used a DT classifier to detect five types of cyber intrusion attacks, with an accuracy of 95.09% and an F1 score of 98%. However, they faced challenges in distinguishing between TCP_SYN and Port Scan attacks, which shared similar features. This resulted in precision values of 79% for both attack types, with recall values of 83% for TCP_SYN and 78% for Port Scans, reflecting an increase in false positives and false negatives.

Another model was seen in [23], where a multi-tiered ANN model was used for intrusion detection and the best result was an accuracy of 99.9%. However, the model is computationally expensive to train. Moreover, in [11], clustering and an SLFN (Single Hidden-Layer Feed-Forward Neural Network) were used for multi-stage detection, and a high accuracy of 98.42% was reported. However, the SLFN was performed with oversampling, so the rare classes could be overfitted.

Additionally, in [7], SVM and ANN were used to improve the accuracy of intrusion detection and results showed that ANN performed better, with an accuracy of 94.02%. But this paper does not report any metrics besides accuracy, hence, we cannot completely evaluate the model's performance. Another instance was in [25], where a sequential deep learning model was used to detect intrusions in IoT networks and an F1 score of 97% was recorded, but the dataset used was not mentioned.

In [12], HMM was used to detect multi-step attacks. Though the model performed with less delay, the paper did not mention the exact values of the performance metrics. It is also seen that in [21], a sequence-to-sequence model was used for detecting multi-stage attacks and this yielded a high F1 score of 99.9%. However, the paper did not report other metrics and only compared the performance with HMM.

Furthermore, a DNN model was used in [26], which yielded an accuracy ranging from 95% to 99%, but this used public datasets that do not represent real world network traffic. Similarly, in [27], big data and deep learning techniques were combined and DNN yielded the highest accuracy of 99.56%. However, the dataset had several missing and infinity values. Moreover, the highest recall was seen in [17], but the training data was overfitted due to oversampling of the minority class, which resulted in inaccurate measurements. When compared with previously implemented models for IDSs, our DT, LGBM, XGB and RF models obtained the highest accuracy result of 100%.

Next, in [8], an ANN and RF evaluated the CIC IDS 2017 dataset using R language, and the best result was provided by RF, with an accuracy of 96.4%. But feature selection only used the 10 most important features, hence there were several missing data points. It was seen in [19] that different classifiers were used on the same dataset and RF performed the best with an accuracy of 97.6%. As mentioned earlier, this high accuracy does not guarantee efficient execution time.

Moreover, in [29], different deep learning models like CNN and DNN were used to detect intrusion. Here, the DNN yielded the highest accuracy of 99.01% and the DNN also predicted anomalies with a 99.3% accuracy. However, these deep learning models are complex and were sometimes limited to binary classification. Also, in [20], an SVM and NB were used with feature embedding and both models reported an accuracy of 98.92%. However, the dataset was large and showed class imbalance; hence, there was a bias towards the majority class.

Additionally, in [30], a two-stage deep learning model was used and this showed an accuracy of 99.93%, but this was tested on public datasets that do not represent real world network traffic.

When comparing our results with [6], which is the only paper from the research that used our dataset, we can conclude that our proposed models were relatively efficient in terms of results and model complexity. The researchers in [6] used Extremely Boosted Neural Networks and relied only on accuracy, which was reported as 99.72%, as shown in Table 1. Our proposed models are less computationally expensive when compared to the previous studies mentioned in Section 2. The best performing model in [6] was a combination of a gradient boosted tree and a feedforward neural network, whereas our paper used the Extreme Gradient-Boosting machine learning algorithm. Moreover, models were compared in [6] solely based on accuracy, while in this study, several metrics, like accuracy, precision, recall, AUC and F1 score, were employed. Thus, the work presented in this paper achieved better results on the MSCAD dataset than other previous studies.

6. Conclusions

In this paper, we proposed a machine learning-based approach for network IDS. The MSCAD dataset was considered because it was found to be highly representative of real-world multi-step cyberattack scenarios and increases the accuracy and efficiency of such systems. Several pre-processing steps were applied to this dataset, such as dimensionality reduction and data balancing using the ADASYN technique. Moreover, correlated features were subsequently used to eliminate features supplying redundant information to the models. Several machine learning models were deployed, including DT, KNN, NB, SVM, LGBM, XGB and RF. Grid searches were used to find the most suitable hyperparameters to tune each model. The experimental results showed that the best performing model was XGB, which achieved a testing accuracy of 100% and an F1 score of 88%.

In conclusion, the proposed method presents a computationally efficient model that can ensure ease of deployment for a fast, sustainable and low power consuming IDS at the cutting edge. Furthermore, the proposed model can be utilized in smart cities where interconnected systems require advanced intrusion detection capabilities. With the growing importance of securing critical infrastructures in smart cities, it was found that this machine learning-based IDS can effectively detect sophisticated multi-step cyberattacks, ensuring the protection of essential services. In future studies, the proposed model can be trained and tested on more datasets that may include other classes of cyberattacks. This will ensure that the model can be generalized to detect more cyberattacks, which will contribute to enhancing its capabilities and ensuring the security of various smart city systems.

Author Contributions: Conceptualization, S.D. and F.A.; data curation, J.K., R.E., H.S., M.P. and E.S.; funding acquisition, F.A.; investigation, J.K., R.E., H.S., M.P. and E.S.; methodology, J.K., R.E., H.S., M.P., E.S., S.D. and F.A.; project administration, S.D. and F.A.; resources, S.D. and F.A.; software, J.K., R.E., H.S., M.P. and E.S.; supervision, S.D. and F.A.; validation, J.K., R.E., H.S., M.P. and E.S.; writing—original draft, J.K., R.E., H.S., M.P. and E.S.; writing—review and editing, S.D. and F.A. All authors have read and agreed to the published version of the manuscript.

Funding: The work in this paper was supported, in part, by the Open Access Program from the American University of Sharjah (Award # OPFY25-3152-OE2520). This paper represents the opinions of the authors and does not mean to represent the position or opinions of the American University of Sharjah.

Data Availability Statement: The original data presented in the study are openly available in IEEE Dataport at [doi: https://dx.doi.org/10.21227/phr0-e264], reference number [4].

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Almseidin, M.; Al-Sawwa, J.; Alkasassbeh, M. Generating a benchmark cyber multi-step attacks dataset for intrusion detection. *J. Intell. Fuzzy Syst.* **2022**, *43*, 3679–3694. [CrossRef]
2. Mao, B.; Liu, J.; Lai, Y.; Sun, M. MIF: A multi-step attack scenario reconstruction and attack chains extraction method based on multi-information fusion. *Comput. Netw.* **2021**, *198*, 108340. [CrossRef]
3. Anand, R.; Jain, M.; Jain, L.; Narwal, B.; Jaiswal, A. Application of an Intrusion Detection System in Smart Cities: A Review. In Proceedings of the 2022 4th International Conference on Artificial Intelligence and Speech Technology (AIST), Delhi, India, 9–10 December 2022; pp. 1–6. [CrossRef]
4. Almseidin, M.; Al-Sawwa, J.; Alkasassbeh, M. *Multi-Step Cyber-Attack Dataset (MSCAD for Intrusion Detection)*; IEEE Dataport; IEEE: Piscataway, NJ, USA, 2022. [CrossRef]
5. Dhanya, K.; Vajipayajula, S.; Srinivasan, K.; Tibrewal, A.; Kumar, T.G. Detection of Network Attacks using Machine Learning and Deep Learning Models. *Procedia Comput. Sci.* **2023**, *218*, 57–66. [CrossRef]
6. Dalal, S.; Manoharan, P.; Lilhore, U.K.; Seth, B.; Alsekait, D.M.; Simaiya, S.; Hamdi, M.; Raahemifar, K. Extremely boosted neural network for more accurate multi-stage Cyber attack prediction in cloud computing environment. *J. Cloud Comput.* **2023**, *12*, 14. [CrossRef]
7. Taher, K.A.; Jisan, B.M.Y.; Rahman, M. Network intrusion detection using supervised machine learning technique with feature selection. In Proceedings of the 2019 International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST), Dhaka, Bangladesh, 10–12 January 2019. [CrossRef]
8. Pelletier, Z.; Abualkibash, M. Evaluating the CIC IDS-2017 Dataset Using Machine Learning Methods and Creating Multiple Predictive Models in the Statistical Computing Language R. *Int. Res. J. Adv. Eng. Sci.* **2020**, *5*, 187–191.
9. Gan, B.; Chen, Y.; Dong, Q.; Guo, J.; Wang, R. A convolutional neural network intrusion detection method based on data imbalance. *J. Supercomput.* **2022**, *78*, 19401–19434. [CrossRef]
10. Maseer, Z.K.; Kadhim, Q.K.; Al-Bander, B.; Yusof, R.; Saif, A. Meta-analysis and systematic review for anomaly network intrusion detection systems: Detection methods, dataset, validation methodology, and challenges. *IET Netw.* **2024**, *13*, 339–376. [CrossRef]
11. Qaddoura, R.; Al-Zoubi, A.M.; Almomani, I.; Faris, H. A Multi-Stage Classification Approach for IoT Intrusion Detection Based on Clustering with Oversampling. *Appl. Sci.* **2021**, *11*, 3022. [CrossRef]
12. Ingale, S.; Paraye, M.; Ambawade, D. Enhancing Multi-Step Attack Prediction using Hidden Markov Model and Naive Bayes. In Proceedings of the 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India, 2–4 July 2020; pp. 36–44.

13. Saheed, Y.K.; Abiodun, A.I.; Misra, S.; Holone, M.K.; Colomo-Palacios, R. A machine learning-based intrusion detection for detecting internet of things network attacks. *Alex. Eng. J.* **2022**, *61*, 9395–9409. [CrossRef]
14. Chen, Y.-W.; Sheu, J.-P.; Kuo, Y.-C.; Van Cuong, N. Design and Implementation of IoT DDoS Attacks Detection System based on Machine Learning. In Proceedings of the 2020 European Conference on Networks and Communications (EuCNC), Dubrovnik, Croatia, 15–18 June 2020; pp. 122–127.
15. Hamza, A.; Hammam, F.; Abouzeid, M.; Ahmed, M.A.; Dhou, S.; Aloul, F. Malicious URL and Intrusion Detection using Machine Learning. In Proceedings of the 2024 International Conference on Information Networking (ICOIN), Ho Chi Minh City, Vietnam, 17–19 January 2024; pp. 795–800.
16. Prachi, H.M.; Malhotra, H.; Sharma, P. Intrusion Detection using Machine Learning and Feature Selection. *Int. J. Comput. Netw. Inf. Secur.* **2019**, *11*, 43–52. [CrossRef]
17. Yulianto, A.; Sukarno, P.; Suwastika, N.A. Improving AdaBoost-based Intrusion Detection System (IDS) Performance on CIC IDS 2017 Dataset. *J. Phys. Conf. Ser.* **2019**, *1192*, 012018. [CrossRef]
18. Chaturvedi, P. A Comparative Approach for Host Based Intrusion Detection Using Naiyve Bayes and KNN Algorithm. *Int. J. Innov. Res. Comput. Sci. Technol.* **2024**, *12*, 87–90. [CrossRef]
19. Hammad, M.; El-Medany, W.; Ismail, Y. Intrusion Detection System using Feature Selection with Clustering and Classification Machine Learning Algorithms on the UNSW-NB15 dataset. In Proceedings of the 2020 International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT), Sakheer, Bahrain, 20–21 December 2020; pp. 1–6.
20. An Effective Intrusion Detection Approach Using SVM with Naïve Bayes Feature Embedding | Elsevier Enhanced Reader. Available online: https://www.sciencedirect.com/science/article/abs/pii/S0167404820304314 (accessed on 29 April 2023).
21. Zhou, P.; Zhou, G.; Wu, D.; Fei, M. Detecting multi-stage attacks using sequence-to-sequence model. *Comput. Secur.* **2021**, *105*, 102203. [CrossRef]
22. Ben Fredj, O. An NLP-inspired method to predict multi-step cyberattacks. In Proceedings of the 2022 15th International Conference on Security of Information and Networks (SIN), Sousse, Tunisia, 11–13 November 2022; pp. 1–6.
23. Sohail, S.; Fan, Z.; Gu, X.; Sabrina, F. Multi-tiered Artificial Neural Networks model for intrusion detection in smart homes. *Intell. Syst. Appl.* **2022**, *16*, 200152. [CrossRef]
24. Abdullah, M.Z.; Jassim, A.K.; Hummadi, F.N.; Al Khalidy, M.M.M. New strategies for improving network security against cyber attack based on intelligent algorithms. *J. Eng. Sustain. Dev.* **2024**, *28*, 342–354. [CrossRef]
25. Thamilarasu, G.; Chawla, S. Towards Deep-Learning-Driven Intrusion Detection for the Internet of Things. *Sensors* **2019**, *19*, 1977. [CrossRef] [PubMed]
26. Vinayakumar, R.; Alazab, M.; Soman, K.P.; Poornachandran, P.; Al-Nemrat, A.; Venkatraman, S. Deep Learning Approach for Intelligent Intrusion Detection System. *IEEE Access* **2019**, *7*, 41525–41550. [CrossRef]
27. Faker, O.; Dogdu, E. Intrusion Detection Using Big Data and Deep Learning Techniques. In Proceedings of the 2019 ACM Southeast Conference, Kennesaw, GA, USA, 18–20 April 2019. [CrossRef]
28. Xiao, Y.; Xing, C.; Zhang, T.; Zhao, Z. An Intrusion Detection Model Based on Feature Reduction and Convolutional Neural Networks. *IEEE Access* **2019**, *7*, 42210–42219. [CrossRef]
29. Ahmad, Z.; Khan, A.S.; Nisar, K.; Haider, I.; Hassan, R.; Haque, M.R.; Tarmizi, S.; Rodrigues, J.J.P.C. Anomaly Detection Using Deep Neural Network for IoT Architecture. *Appl. Sci.* **2021**, *11*, 7050. [CrossRef]
30. Khan, F.A.; Gumaei, A.; Derhab, A.; Hussain, A. TSDL: A Two-Stage Deep Learning Model for Efficient Network Intrusion Detection. *IEEE Access* **2019**, *7*, 30373–30385. [CrossRef]
31. Li, Y.; Qin, T.; Huang, Y.; Lan, J.; Liang, Z.; Geng, T. HDFEF: A hierarchical and dynamic feature extraction framework for intrusion detection systems. *Comput. Secur.* **2022**, *121*, 102842. [CrossRef]
32. Sen, O.; Eze, C.; Ulbig, A.; Monti, A. On Holistic Multi-Step Cyberattack Detection via a Graph-Based Correlation Approach. In Proceedings of the 2022 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm), Singapore, 25–28 October 2022; pp. 380–386.
33. Angelini, M.; Bonomi, S.; Lenti, S.; Santucci, G.; Taggi, S. MAD: A visual analytics solution for Multi-step cyber Attacks Detection. *J. Comput. Lang.* **2019**, *52*, 10–24. [CrossRef]
34. Shawly, T.; Khayat, M.; Elghariani, A.; Ghafoor, A. Evaluation of HMM-Based Network Intrusion Detection System for Multiple Multi-Stage Attacks. *IEEE Netw.* **2020**, *34*, 240–248. [CrossRef]
35. Wang, X.; Gong, X.; Yu, L.; Liu, J. MAAC: Novel Alert Correlation Method to Detect Multi-step Attack. In Proceedings of the 2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Shenyang, China, 20–22 October 2021; pp. 726–733.
36. Zhang, X.; Wu, T.; Zheng, Q.; Zhai, L.; Hu, H.; Yin, W.; Zeng, Y.; Cheng, C. Multi-Step Attack Detection Based on Pre-Trained Hidden Markov Models. *Sensors* **2022**, *22*, 2874. [CrossRef]

37. He, H.; Bai, Y.; Garcia, E.A.; Li, S.; Li, S. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In Proceedings of the 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), Hong Kong, China, 1–8 June 2008; pp. 1322–1328. [CrossRef]
38. Rokach, L.; Maimon, O. Decision Trees. In *Data Mining and Knowledge Discovery Handbook*; Maimon, O., Rokach, L., Eds.; Springer: Boston, MA, USA, 2005; pp. 165–192.
39. Mucherino, A.; Papajorgji, P.J.; Pardalos, P.M. k-Nearest Neighbor Classification. In *Data Mining in Agriculture*; Mucherino, A., Papajorgji, P.J., Pardalos, P.M., Eds.; Springer: New York, NY, USA, 2009; pp. 83–106. [CrossRef]
40. Webb, G.I. Naïve Bayes. In *Encyclopedia of Machine Learning [Internet]*; Sammut, C., Webb, G.I., Eds.; Springer: Boston, MA, USA, 2010; pp. 713–714. [CrossRef]
41. Cristianini, N.; Ricci, E. Support VEctor Machines. In *Encyclopedia of Algorithms*; Kao, M.-Y., Ed.; Springer: Boston, MA, USA, 2008; pp. 928–932. [CrossRef]
42. Ke, G.; Meng, Q.; Finley, T.; Wang, T.; Chen, W.; Ma, W.; Ye, Q.; Liu, T.-Y. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Red Hook, NY, USA, 2017. Available online: https://proceedings.neurips.cc/paper_files/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html (accessed on 1 January 2025).
43. XGBoost | Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Available online: https://dl.acm.org/doi/10.1145/2939672.2939785 (accessed on 1 January 2025).
44. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
45. Alshamkhany, M.; Alshamkhany, W.; Mansour, M.; Khan, M.; Dhou, S.; Aloul, F. Botnet Attack Detection using Machine Learning. In Proceedings of the 2020 14th International Conference on Innovations in Information Technology (IIT), Al Ain, United Arab Emirates, 17–18 November 2020; pp. 203–208. [CrossRef]
46. Al Ali, S.; Suleiman, A.; Hallal, G.; Alseiari, S.; Ma, Y.; Dhou, S.; Aloul, F. Android Malware Detection Using Machine Learning. In Proceedings of the 2024 IEEE International Conference on Internet of Things and Intelligence Systems (IoTaIS), Bali, Indonesia, 28–30 November 2024; pp. 79–84. [CrossRef]
47. Machine Learning Pipeline. Available online: https://c3.ai/glossary/machine-learning/machine-learning-pipeline/ (accessed on 6 May 2024).