Iraqi Journal of Science, 2026, Vol. 67, No. 6, pp: 3527-3549
DOI: 10.24996/ijs.2026.67.6.37

ISSN: 0067-2904

Enhancing the Performance of Cyberattack Detection Model on AWS Using Stacked Feature Selection and Hybrid Classification

Ibtihal A. Duayim[1]*, Nadia F. AL-Bakri[1], Mohammed Salih Mahdi[2]
[1]*Computer Science, Al-Nahrain University, Baghdad, Iraq*
[2]*BIT, Business Information College, University of Information Technology and Communications, Baghdad, Iraq*

Received: 14/ 11/2024 Accepted: 17/6/2025 Published: 30/6/2026

Abstract

One of the essential technologies worldwide is cloud computing. Increasing demand for its various Amazon Web Services (AWS) services make it more vulnerable to different types of attacks, such as DoS, DDoS, Infiltration, and others). The full list of attack types used in this study is presented in the classification section. Protecting cloud systems and their data mainly depends on various security technologies, including intrusion detection systems that monitor network traffic to detect suspicious activity or potential threats. This research proposes a model to contribute to solving the problem of identifying malicious activities in cloud networks. This model has three stages: the initial stage is pre-processing, encompassing data cleaning, normalization, and labelling. Stage two involved intersecting the outputs of the correlation and Information Gain methods to obtain a set of common features. Then these features were further filtered using the Chi-square method to identify the most relevant features. Lastly, a proposed classification technique was used utilizing (CNN -XGBoost) to classify traffic in the network into normal and abnormal. Experiment results on CSE-CIC-IDS2018 dataset showed that the proposed model achieved an accuracy of 99.3%, outperforming traditional filter-based feature selection.

Keywords: AWS, Cloud Computing, CSECICIDS2018, Data security, Filter Feature Selection, Intrusion Detection System (IDS).

تحسين اداء نموذج كشف الهجمات السيبرانية على AWS باستخدام تقنية اختيار الميزات المكدسة والتصنيف الهجين

ابتهال عباس دعيم[1]*, نادية فاضل البكري[1], محمد صالح مهدي[3]
[1]علوم الحاسوب, كلية العلوم, جامعة النهرين, بغداد, العراق
[2]تكنولوجيا معلومات الاعمال, كلية معلوماتية الاعمال, جامعة تكنولوجيا المعلومات والاتصالات, بغداد, العراق

الخلاصة

تعد الحوسبة السحابية من التقنيات الأساسية في جميع أنحاء العالم، حيث إن الطلب المتزايد على خدماتها المتنوعة من AWS يجعلها أكثر عرضة لأنواع مختلفة من الهجمات مثل (DoS، DDoS،

*Email: ibtihalabbas0@gmail.com

Infiltration، وغيرها). إن حماية أنظمة الحوسبة السحابية وبياناتها تعتمد بشكل أساسي على مجموعة متنوعة من تقنيات الأمان، بما في ذلك أنظمة كشف التسلل التي تراقب حركة الشبكة للكشف عن الأنشطة المشبوهة أو التهديدات المحتملة. يقترح هذا البحث نموذجًا للمساهمة في حل مشكلة تحديد الأنشطة الضارة في شبكات الحوسبة السحابية، ويتكون هذا النموذج من ثلاث مراحل: المرحلة الأولية هي المعالجة المسبقة، والتي تشمل تنظيف البيانات وتطبيعها ووضع العلامات عليها. المرحلة الثانية تتضمن تقاطع مخرجات طرق Corr و IG للحصول على مجموعة من الميزات المشتركة. ثم تمت تصفية هذه الميزات بشكل أكبر باستخدام طريقة Chi-s لتحديد الميزات الأكثر صلة. أخيرًا، تم استخدام تقنية تصنيف مقترحة باستخدام (CNN – XGBoost) لتصنيف حركة المرور في الشبكة إلى طبيعية وغير طبيعية. أظهرت نتائج التجربة على بيانات CSE-CIC-IDS2018 أن النموذج المقترح حققً دقة 99.3% متفوقا على الطرق التقليدية والقائمة على الفلترة في اختيار الميزات.

1. Introduction

The rapid development of technology makes life more comfortable and overcomes many security problems. Cloud Computing is one of these technologies; it is familiar and is obtaining attention from individual users and organizations. It refers to an Internet-based computing technology in which shared resources, such as storage, software, information, and platforms, are offered to users on demand. It is gaining popularity due to various advantages of on-demand service provision, flexible resource allocation, higher fault tolerance, and higher scalability.[1], [2], [3]

It provides services to its users in different ways. Complete virtual machine control is provided via Infrastructure as a Service (IaaS), such as Amazon Elastic Compute Cloud(EC2), and Amazon Simple Storage Service(S3). In addition, Platform as a Service (PaaS), where the user can deploy user-created applications in the cloud if the provider supports the languages, APIs, and tools used for creating applications, like Amazon Cognito, and Amazon DynamoDB. Lastly, Software as a Service (SaaS) enables users to execute providers' applications, such as Netflix and Dropbox. These services are provided via the Internet.[4]

Due to the networked nature of the cloud and its open and distributed nature, the cloud is vulnerable to intrusions. Thus, there is a risk to the cloud environment's security. Therefore, tackling security threats properly requires appropriate mechanisms.

Traditional network security, such as a firewall, is good for detecting and stopping many external attacks, but its effectiveness is weak for attacks from within the network. In return, intrusion detection systems (IDS) play a vital role in cloud security because they are software or devices that act as an additional security protective layer, which continuously monitors the network to detect well-known attacks, along with several unknown attacks.[5]

The IDS model faces many challenges. A large number of features that must be analyzed is one of the most significant, as they don't all really contribute to identifying the attack, since the model's performance may be affected throughout the classification process. For this reason, many previous researchers have used Feature Selection methods in IDS models to identify the most relevant and effective features.

In FS, the correlation and impact of each feature on the detection performance are determined by utilizing many strategies of FS. One of these strategies is the Filter FS, whose methods are independent of any classification algorithm and employ a feature ranking function to choose the best features.[6], [7], [8]

In this study, the aim is to use a proposed stacked filter feature selection method with a hybrid classifier to enhance the accuracy of intrusion attack detection within cloud environments. A new strategy is developed for feature selection called Correlation_Information Gain_Chi-squared (C_IG_C), which includes using three filter methods (Pearson Correlation(Corr), Information Gain(IG) and Chi-squared(Chi-s)). By utilizing the advantages of each method, the stacked approach offers a more effective feature selection procedure to choose the most relevant features, particularly focusing on those prevalent in the AWS cloud network environment (CSECICIDS2018 dataset). Also, a new strategy is proposed for classification using (CNN-XGBoost) classification algorithms to enhance the performance of the intrusion detection system.
The main objectives of this study are as follows:
1. Implement a suitable pre-processing Phase to include the clean traffic flows into the learning model.
2. Applying three filter feature selection methods to obtain the common attributes that are considered the best subset to be used in the classification phase. Also, a new feature selection method is proposed.
3. Four distinct classifiers are used in the classification phase to evaluate the NIDS model on the CSECICIDS2018 dataset.

The remaining section is organized as follows: **Section 2** provides details of related works. **Section 3** includes a description of key research concepts. The methodology is explained in **Section 4,** and in **Section 5,** the obtained Results are discussed. Finally, the conclusion and future works are shown in **Section 6**.

2. Related work

This section discusses the previous research related to this study, focusing on those that utilized the CSECICIDS2018 dataset. A list of the methods of these related works and their best performance scores is presented in Table 1.

The authors in [9] proposed a deep learning model that utilizes a filter-based FS method. They are leveraging an advanced CNNs-based model to detect cyberattacks in the cloud environment. SMOTE data strategy was used to fix the class imbalance issue on the utilized CSECICIDS2018 dataset, and a Pearson correlation coefficient matrix heatmap was used to identify highly correlated features. Model performance was evaluated through seven experimental scenarios, and the average results of the overall evaluation performance were 98.69%, 98.80%, 99.68%, and 98.79% for accuracy, precision, recall, and F1-score, respectively.

The approach in [10] for predicting cyber-attacks in cloud systems, proposed four machine learning models that were applied on a narrowed CSECICIDS2018 dataset to obtain a prediction. Without applying any FS methods, the MLP-BP model outperformed other models (LSTM, AdaBoost, and DAE-DNN). It demonstrates accuracy at 98.97. Although this has high accuracy, there are still some lingering problems with data processing and labelling. In addition, this study was limited to only attack predictions without classifying their types. The presented study in [11] proposed a deep learning model(CNN) to identify and classify only five types of network intrusions. The experiment was carried out on a part of CSECICIDS2018 dataset. A Random Forest model was implemented, and seventeen features were selected as input to the CNN model. 97.07% accuracy was achieved with only a 2.93% error rate. The limitation of this study is that the model is limited to classifying data into only five predefined attack types.

In [12], two models were used on the CSECICIDS2018 dataset to categorize into binary and multi-class classification. Multi-layer perceptron with backpropagation (MLP-BP) and MLP with particle swarm optimization (MLP_PSO). These models were the base for four experiments carried out by this study. Following the preprocessing stage, 24 features were chosen using the Random Forest classifier (RF). A dataset with chosen features is duplicated twice, once for multi-class classification and once for binary classification. Results indicate that 98.97% and 98.41% were the highest accuracy rates for binary and multi-class classification.

The hybrid deep learning-based strategy that is being suggested in [13] starts with preprocessing the raw data and subsequently normalizing it. Principal Component Analysis (PCA) was used to extract features from seventy-six fields to seven bottlenecks. The seven extracted features from each packet were then classified as two-way soft-clustered (attack and non-attack) using the Smart Monkey Optimized Fuzzy C-Means algorithm (SMO-FCM). The attack cluster data were additionally provided as inputs for the Auto-Encoder method, which is based on deep learning and produces attack classifications as outputs. Ultimately, across the CSECICIDS2018 dataset, the accuracy of the findings in intrusion detection utilizing the suggested technique (PCA + SMO-FCM + AE) is attained as 95%.

In a study [14], after the data had been preprocessed, the SMOTE technique was implemented to solve the class imbalance issue. Then, an autoencoder was applied in conjunction with deep learning. 97.79% accuracy was achieved on average across all files in the CSECICIDS2018 dataset. The lack of procedure specifics in this research paper is problematic. The PCA-DNN model was used in [15] to increase awareness about network security and improve threat detection rates by integrating Principal Component Analysis (PCA) with Deep Neural Networks(DNN). Preprocessing and normalization were utilized in the system's construction, and the one hot encoder technique was applied to cope with extremely significant categorical data. 20% of the dataset was used for testing, while the remaining 80% was for training. Prior to classification, PCA is used as a feature reduction technique to improve the prediction ability of neural network models. Using the CSECICIDS 2018 dataset, a DNN approach with and without dimensional reduction (PCA), the overall accuracy of binary classification detection with (PCA) was 97.97%.

The author in [16] used (CRNN) a convolutional recurrent neural network to create a DL-based hybrid IDS to predict and classify cyberattacks in the network. In the Hybrid CRNNIDS, local features were captured by a performed CNN, and the (RNN) captures temporal features to improve the IDS performance and prediction. The CSECICIDS2018 dataset was used, but one of the detected limitations of this research is that it appears not all available data was utilized. The analysis was based on a sample of approximately 8 million records. The highest detection rate accuracy was up to (97.75%) with 10-fold cross-validation.

In another study [17], the authors focused on three feature extraction methods applied in the pretrained phase: autoencoder (AE), deep autoencoder (DAE), and stack autoencoder (SAE). (DAE) yielded the best results in terms of performance measures for multiclass classification tasks. The proposed model employs a pretraining strategy utilizing a deep autoencoder (PTDAE) and a deep neural network (DNN). The model performs better in terms of detection on the NSL-KDD and CSE-CIC-ID2018 datasets. The overall intrusion detection accuracy result for the CSECICID2018 dataset is 95.79%.

The study [18] presented a fully connected dense deep neural network (DNN). Three fully connected layers were implemented in the classification stage after preprocessing the data, achieving a detection accuracy of 90%. No feature selection strategy was used.

The authors of [19] proposed an anomaly-based ensemble classifier using gradient boosting, decision trees, and logistic regression. After their tests with seven single classifiers to determine the most suited classifier for ensemble learning, the (CSE-CIC-IDS2018) dataset was used to evaluate the proposed model. Finding the data features that might not be used was made easier by using chi-square and Spearman's rank correlation. According to the experiment results, 23 out of the 80 features were chosen, and the model obtained scores of 98.8% for final accuracy, and 98.8% for precision.

In [20], the researcher used IDS on ANN and MLP to identify botnet attacks on the CSECICIDS2018 dataset. The accuracy obtained was 99.97%. This research's weakness is that, in addition to only addressing one type of attack, it misses information regarding data preparation before the classification process. Finally, the researchers in [21] proposed using the Deep Learning method. The LSTM is used to establish the model, and the Attention Mechanism (AM) is incorporated to deal with the time-correlated network traffic's classification issue and made some progress on the class imbalance issue by using SMOTE to get more samples and optimize the loss function. The overall accuracy of the system reached 96.2%.

Table 1: An overview of previous research

Author	Year	Method	Performance scores on CSECICIDS2018			
			acc	pre	recall	F1-score
[9]	2024	CNNs	98.67	98.80	99.68	98.79
[10]	2024	MLP-BP	98.97	99.00	98.00	99.38
[11]	2024	CNN + RF	97.07	98.11	96.93	97.52
[12]	2023	MLP-BP + MLP-PSO	98.97	99.98	98.80	98.00
[13]	2023	PCA+SMO-FCM+AE	95.30	94.70	-	-
[14]	2022	SMOTE+AE+DNN	97.79	97.04	96.85	-
[15]	2022	PCA-DNN	96.79	-	-	-
[16]	2021	HCRNNIDS	97.75	96.33	97.12	97.60
[17]	2021	PTDAE + DNN	95.79	95.38	95.79	95.11
[18]	2020	DNN	90.25	65.00	59.00	60.62
[19]	2020	Ensemble learning (LR+DT+GB)	98.8	98.8	97.1	97.9
[20]	2019	ANN + MLP	99.97	-	-	-
[21]	2019	LSTM + AM	96.19	96.00	96.00	93.00

3. Background

3.1. AWS Cloud Computing for IDS Applications

AWS is currently one of the oldest cloud computing platforms (it was launched in 2006) and is considered one of the most popular and widely used public platforms, offering hundreds of services. Among its broad range of services, it includes key services related to the Intrusion Detection System (IDS) model. These services include scalable virtual machines to host IDS components (EC2 service). It also provides services that facilitate the organized storage of metadata and alert information, including Amazon RDS (Relational Database Service) and Amazon S3 (Simple Storage Service). Network services such as Amazon VPC (Virtual Private Cloud) provide isolated network environments for deploying system

components, as well as availability-enhancing services such as Elastic Load Balancing (ELB). Furthermore, it provides services that ensure the isolation of system components and protect them from unauthorized access.

All of these services and more are provided by AWS, making this platform ideal for hosting and scaling Intrusion Detection Systems (IDSs) in a secure and cost-effective manner. In addition, since an intrusion detection system (IDS) is an important critical component of the AWS security environment, and due to AWS's services nature that are characterized by their flexibility and manageability, it cannot be only deployed within the cloud infrastructure, it also can be integrated with AWS security services to effectively detect and analyze threats. [22], [23], [24]

3.2. Cloud Computing Security

Security is one of the most important obstacles to the mainstream adoption of cloud computing technology. Data security is a major concern, and a vulnerability associated with cloud computing. Safety must be ensured when storing data because unreliable cloud service providers could lead to consumer data becoming infected. A cloud computing environment necessitates a large number of devices and applications, which, in turn, increases the number of access points, consequently raising the risk of a data breach.[25]

3.3. Intrusion Detection System (IDS)

Fundamentals of data security are classically classified by the CIA triad.

1. Confidentiality: This guarantees that user data is protected against access by unauthorized parties.
2. Integrity: This guarantees that information is correct and hasn't been altered from its original state.
3. Availability: Authorized people can easily obtain this guarantee of data dependability upon request. Cybersecurity must be taken into consideration as well as applied to protect user information to securely deploy cloud computing technologies.[26]

An attack that can threaten the availability, confidentiality, or integrity of information is considered an intrusion. An (IDS) is a software or hardware system whose main goal is to identify fraudulent activity in order to secure user data and cloud services.[27]

IDSs are divided into two categories: anomaly-based and misuse-based. An anomaly-based intrusion detection system is used to find novel intrusions by comparing recorded typical behavior in real-time traffic to previously recorded data. New threats may be detected by using this strategy, but it may also result in false-positive alarms, misidentifying benign packets as malicious. Conversely, in a misuse-based(signature-based) IDS, known attacks can be detected, and thus have high detection rates for these attacks. However, it performs poorly when it comes to identifying unknown attacks, this results in a high False Positive Rate (FPR) due to the slight similarity between arriving events and the modeled normal behavior.[26], [27]

3.4. IDS using ML and DL

For attack detection, Machine Learning (ML) algorithms such as decision trees (DT), support vector machines (SVM), and random forests are commonly applied in intrusion detection systems. Similarly, Deep Learning (DL) approaches such as a convolutional neural network (CNN), recurrent neural network (RNN), and deep autoencoder (AE) are also used in intrusion detection systems. However, DL methods have performed better than ML in certain cases, especially on large datasets. This has prompted researchers to investigate how deep learning theory can be applied to the problem of intrusion detection classification. While the improved performance of DL with increasing data size is the main feature that distinguishes it from traditional ML techniques, the continuous growth in the size of datasets used in intrusion detection research, as well as the growth in the dimensions of the classification

space and attack inputs, has led to a large number of misclassifications, thus reducing the efficiency of the system and degrading the overall performance of the system. Therefore, it is necessary to implement solutions that can select only the features required for the best possible classification procedure [28], [29]. It is worth noting that the hybrid of ML and DL algorithms can be used to build an IDS to obtain good results, as in the proposed system in this study.

3.5. Feature Selection

Feature Selection (FS) is A key element in improving IDSs. However, the increasing dimensionality of data presents a significant challenge to the feature extraction and selection techniques used today. While the performance of the IDS may be impacted by irrelevant data, the FS selects the strongly related features from the dataset, and improved IDS models are constructed using this most pertinent subset of the original features.[30]

FS methods can be categorized as filter, wrapper, and hybrid-based approaches. Each of those approaches has its own characteristics. The filter-based method chooses features based on their score, which is determined using various statistical techniques. The Wrapper-based FS chooses its features by producing and computing the features subset's prediction performance iteratively. The merits of filtering and wrapping techniques are combined in hybrid or embedded feature selection.[31]

3.6. Filter-based FS

Filter FS is a statistical measure used to evaluate the relevance of each feature and ranking based on their relevance score to select the best subset of top-ranking features according to a certain number or threshold for relevance.

3.6.1. Chi-squared (chi-s)

Chi-squared is a statistical hypothesis test used in feature selection to determine each feature's degree of dependence on the target variables. The higher value of Chi-s represents the more relevant features concerning the target. The formula of the Chi-squared test is shown by equation (1).

$$(Chi_s)^2 = \sum_{xz}(I_{xz} - J_{xz})^2 \ /J_{xz} \quad (1)$$

Where (x) and (z) are variables, (I) denotes the observed value, (J) the expected value, and (chi_s) represents the value of Chi-squared.[32]

3.6.2. Correlation (Corr)

This method involves using a correlation matrix, which contains the coefficient of correlation between two features in a dataset. There are three possible values for the correlation coefficient between -1 and 1. The relationship between the two features is represented by each individual cell in the matrix. Whether or not the feature is used going forward depends on the correlation coefficient's value. A feature is positively connected with another if the correlation coefficient between the two features is positive. They are negatively related if the correlation coefficient becomes below. We can decrease the dataset's dimensionality through these dependencies. To compute Pearson correlation, the mathematical formula is given by (2):

$$p_{xy} = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i-\bar{y})^2}} \quad (2)$$

where:

p_{xy}: the value of Pearson correlation coefficient, x_i: The individual sample points of each conditional feature, y_i: The individual sample points of the decision attribute, $\bar{x}$The average of all sample points of each conditional attribute, $\bar{y}$: Average of all sample points of the decision attribute.[31]

3.6.3. Information Gain (IG)

IG is one of the FS methods that is widely used. It first reduces noise from unimportant features using a basic attribute rank, and then it finds a feature with the majority of the information base in a specific class.
by calculating the feature's entropy. The best feature will be determined.
The entropy can be calculated using the formula (3):

$$Ent(s) = \sum_i^v -p_i \, log_2 \, p_i \quad (3)$$

where
Ent(s): The entropy of the sample, v: number of values in the classification class, p_i: number of samples for class i.
After an entropy value is obtained, the formula (4) is used to calculate the IG value.

$$G(s,a) = Ent(s) - \sum_{values(a)} \frac{|s_y|}{|s|} Ent(s_y) \quad (4)$$

where
s: refers to sample, a: represents a feature, y: is a possible value for attribute a, a set of possible values for a **are** represented by Values(a), $|s_y|$: is the number of samples for value y, |s|: number of samples for all data samples and Ent (s_y) is entropy for samples that have a value of y.
This method chooses IG as a feature selection to provide more stable sets of features that are selected due to its robust nature against overfitting.[33]

3.7. Multi-Layer-Perceptron (MLP)

MLP is an artificial neural network that is feed-forward and composed of a minimum of three layers consisting of an input layer, one or more hidden layer(s), and an output layer [Recent Developments in Multilayer Perceptron Neural Network]. In classification problems, the number of neurons in the MLP's input layer is correlated with the number of attributes, whereas the number of neurons in the output layer equals the number of classes that need to be classified.
Each layer's output during forward propagation is computed using the activation function from the preceding layer, along with matching weight and bias values in the formula (5). The activation function can transform the output of each layer to a certain range as shown in formula (6).[34]

$$y^i = w^i x^{i-1} + b^i \quad (5)$$

In this equation, y^i represents the resulting data frame, w^i represents weights, and b^i is the bias vector. Activated output data frames are represented by an $act^{i.}$

$$act^i = g(y^i) \quad (6)$$

3.8. XGBoost

A multiple decision tree is used in Extreme Gradient Boosting (XGBoost), as one tree might not be enough to produce good results. Initially, it was done by Tianqi Chen[35]. And further adopted by many developers. By selecting the best features automatically, the system eliminates unnecessary information. With XGBoost, the processing time can be decreased, and the accuracy can be increased. It gives the benefit of algorithm enhancement and tuning the model. It can also be deployed in computing environments.[35]

3.9. Convolution Neural Network (CNN)

One popular DL model for attack classification in intrusion detection systems is CNN. It consists of a convolutional layer that extracts features and a fully connected layer that determines which class the input data belongs to. By adding a pooling layer to the

convolutional layer, the size of the feature data can be reduced while still extracting unique features from the data and maintaining input/output information[36].

A series of filters is applied by the convolution layer using a mathematical process. In order to create the feature map, the filter must be applied to the input matrix. The feature map will be processed by a threshold-based activation function, which will decide whether or not the neuron will fire.[37]

4. Methods

This section explains the proposed model in this study. The proposed model combines the advantage of filter feature selection with the strengths of CNN for feature extraction and XGBoost for classification to improve network intrusion detection on the CSECICIDS 2018 dataset after preprocessing it.

The overall process of the proposed system is illustrated in Figure (1). It begins with data input from CSECICIDS2018 dataset. Then, three main phases were applied: preprocessing, feature selection, and classification.

4.1. Data Pre-processing Phase

In this phase, three steps must be completed before the data is ready for the feature-selection phase.

- **Step 1: Data Cleaning**: The presence of noise in data leads to a high computational cost, which in turn affects the accuracy and performance of the model. Therefore, this data must be cleaned and dealt with while maintaining the quality of the data. In this step, cleaning is split into row-cleaning and column-cleaning.

In row-cleaning, rows with missing values, invalid values (e.g. inf & -inf), and rows with duplicate values are removed.

While column-cleaning is represented by deleting incomplete outliers and irrelevant features, which are ('Forward Header Len', ' Destination Port', and 'Timestamp').

- **Step 2: Data normalization:** It is the technique of transforming features within a dataset to a common scale. This typically involves scaling the values to fall within a specific range, such as 0 to 1 or -1 to 1. It is a crucial step in improving model performance and making computations more efficient.

Min–Max Normalization is the approach used for reducing the values of a feature to a range between 0 and 1 in our research. It takes each data point and subtracts the feature's minimum value. Then, it divides the result by the feature's range.

The mathematical equation that corresponds to this method is as follows:

$$X' = (X - X_{min})/(X_{max} - X_{min}) \quad (7)$$

Where X is the original value, and X′ is the normalized value.[38]

- **Step 3: Data Labeling:** It describes the process of transforming category data into numerical inputs, which must be transformed to fit the model for training.[39]

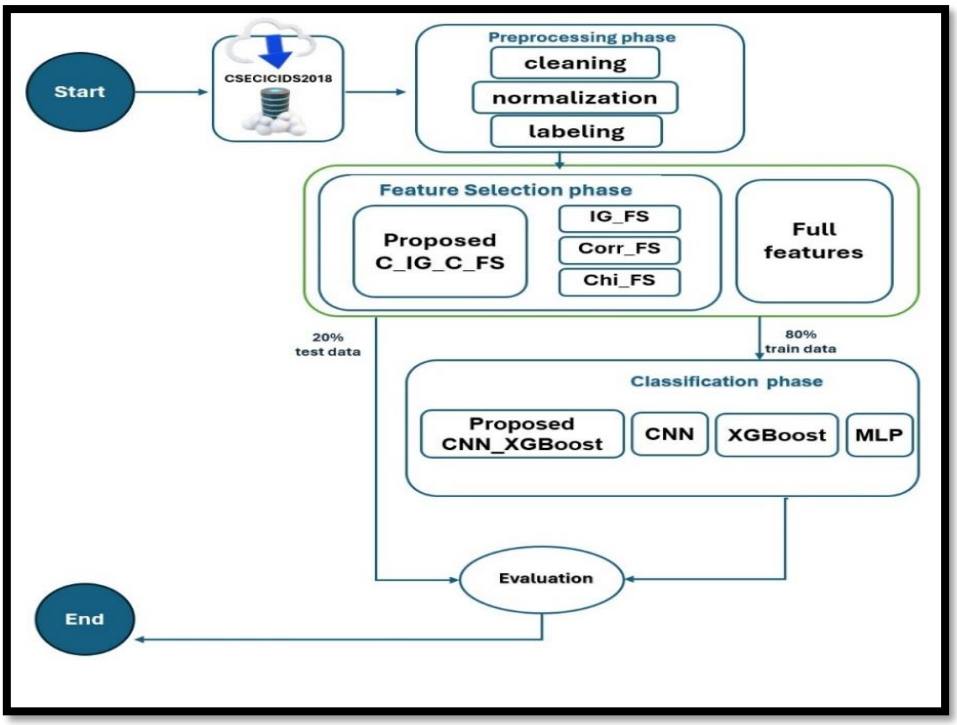

Figure 1: The proposed model

Label Encoder (LE) is one of the most frequently employed for feature mapping. In this study, LE was used and as a result, a dataset with 77 features and binary target attributes was obtained.

4.2. Feature Selection Phase

As a well-chosen feature set has the potential to enhance the accuracy of classifier techniques. The methodology presented in this study focuses on a filter-based approach as the two other FS strategies (Wrapper- and Embedded-based) are computationally expensive

After the data pre-processing phase, seventy-seven features remained, and before using any filter feature selection, features with constant values were eliminated to get a set of 69 features, shown in Table (2, stored in the data frame called (p_data) ready to use in filter methods.

Table 2: The 69 features that enter the filter feature selection methods

Features name			
(1)-'Protocol'	(2)-'Flow IAT Std'	(3)-'Bwd Pkts/s'	(4)-'Bwd Seg Size Avg'
(5)-'Flow Duration'	(6)- 'Flow IAT Max'	(7)-'Pkt Len Min'	(8)-'Subflow Fwd Pkts'
(9)-'Tot Fwd Pkts'	(10)-'Flow IAT Min'	(11)-'Pkt Len Max'	(12)-'Subflow Fwd Byts'
(13)-'Tot Bwd Pkts'	(14)-'Flow IAT Tot'	(15)-'Pkt Len Mean'	(16)-'Subflow Bwd Pkts'
(17)-'TotLen Fwd Pkts'	(18)-'Fwd IAT Mean'	(19)-'Pkt Len Std'	(20)-'Subflow Bwd Byts'
(21)-'TotLen Bwd Pkts'	(22)-'Fwd IAT Std'	(23)-'Pkt Len Var'	(24)-'Init Fwd Win Byts'
(25)-'Flow Byts/s'	(26)-'Fwd IAT Max'	(27)-'FIN Flag Cnt'	(28)-'Init Bwd Win Byts'
(29)-'Flow Pkts/s'	(30)-'Fwd IAT Min'	(31)-'SYN Flag Cnt'	(32)-'Fwd Act Data Pkts'
(33)-'Fwd Pkt Len Mean'	(34)-'Bwd IAT Tot'	(35)-'RST Flag Cnt'	(36)-'Fwd Seg Size Min'
(37)-'Fwd Pkt Len Std'	(38)-'Bwd IAT Mean'	(39)-'PSH Flag Cnt'	(40)-'Active Mean'
(41)-'Bwd Pkt Len Max'	(42)-'Bwd IAT Std'	(43)-'ACK Flag Cnt'	(44)-'Active Std'
(45)-'Bwd Pkt Len Min'	(46)-'Bwd IAT Max'	(47)-'URG Flag Cnt'	(48)-'Active Max'
(49)-'Bwd Pkt Len Mean'	(50)-'Bwd IAT Min'	(51)-'CWE Flag Count'	(52)-'Active Min'
(53)-'Bwd Pkt Len Std'	(54)-'Fwd PSH Flags'	(55)-'ECE Flag Cnt'	(56)-'Idle Mean'
(57)-'Fwd Pkt Len Max'	(58)-'Fwd URG Flags'	(59)-'Down/Up Ratio'	(60)-'Idle Std'
(61)-'Fwd Pkt Len Min'	(62)-'Bwd Header Len'	(63)-'Pkt Size Avg'	(64)-'Idle Max'
(65)-'Flow IAT Mean'	(66)-'Fwd Pkts/s'	(67)-'Fwd Seg Size Avg'	(68)-'Idle Min'
(69)-'Label'			

Then three filter feature selection techniques are applied separately:

4.2.1. **Pearson Correlation**: This method calculates the correlation between each feature and label in (p_data) according to equation (2) and the instruction code data_after_Corr = p_data.corr()['Label'], which returns data with 51 features stored in the (data_after_Corr) data frame. The eliminated features illustrated in Table (3) are stored in the (eliminated_Corr) data frame.

Table 3: Eliminated features numbers after using the Pearson Correlation method

(1)	(2)	(5)	(6)	(7)	(14)	(22)	(26)	(28)	(33)	(34)	(36)	(37)
(45)	(46)	(57)	(61)									

4.2.2. **Information Gain**: mutual information between each feature and the target label in (p_data) is calculated by a call method -using Python from Google Colab- as follows

mutual_information = mutual_info_classif(X, y)

where X represents 68 columns, which are features, and y represents the last column, which is the target Label. The features are sorted by their mutual information scores, and then, by trial and error, the number of features that give the best performance score is determined. The result is data with 50 features stored in the (data_after_IG) data frame. The eliminated features shown in Table(4) are stored in (eliminated_IG).

Table 4: Eliminated features numbers after using the Information Gain method

(7)	(27)	(31)	(35)	(39)	(40)	(43)	(44)	(45)	(47)	(48)
(51)	(52)	(54)	(55)	(58)	(60)	(61)				

4.2.3. **Chi-square**: The number of features that give the best performance score are selected by the following instruction code,

transformer= enericUnivariateSelect(score_func=chi2, mode='k_best', param=29)

The result is data with 29 features stored in the data frame (data_after_Chi), eliminated features are illustrated in Table (5).

Table 5: Eliminated features numbers after using the Chi-squared method

(2)	(3)	(4)	(7)	(8)	(9)	(11)	(12)	(13)	(15)	(16)	(17)	(19)
(23)	(25)	(27)	(29)	(32)	(33)	(38)	(39)	(40)	(42)	(44)	(47)	(48)
(49)	(50)	(51)	(52)	(56)	(58)	(60)	(62)	(63)	(64)	(66)	(67)	(68)

Then the proposed FS method is applied as follows

4.2.4. **C_IG_C_FS**: the proposed feature selection method, which represents applying Corr and IG methods over data with 69 features, then the intersection of the result of applying these two FS methods provides 36 common features shown in Table (6) and eliminates the difference. The 36 common features are passed on to the Chi-s method, and finally, the 29 most relevant features, shown in Table (9), are the result after applying the Chi-s method. The steps to obtain features from the proposed FS method are illustrated in Algorithm 1.

Algorithm 1: Proposed feature selection method (C_IG_C_FS)

Input: (p_data) with 69 features

Output: an optimal subset of features

Begin

Step 1: // Apply Corr FS method

data_after_Corr ← **correlation** over (p_data)

data_after_Corr = p_data.corr()['Label']

Step 2: // Apply IG FS method

data_after_IG ← **mutual information** over (p_data)

data_after_IG = mutual_info_classif(p_data.drop ('Label',axis=1),p_data['Label'])
Step 3: // Feature intersection process
ins_data ← data_after_Corr ∩ data_after_IG
Step 4: // Apply Chi-squared FS method
selected_ features ← **chi-squared** over ins_data
Selected_ features = GenericUnivariateSelect(score_func=chi2, mode='k_best', param=29)
Step 5: // Return results
return selected features
End

Table 6: common features numbers from the intersection operation

(3)	(4)	(8)	(9)	(10)	(11)	(12)	(13)	(15)	(16)	(17)	(18)
(19)	(20)	(21)	(23)	(24)	(25)	(29)	(30)	(32)	(38)	(41)	(42)
(49)	(50)	(53)	(56)	(59)	(62)	(63)	(64)	(65)	(66)	(67)	(68)

4.3. Classification Phase

In this sub-section, the resulting data from the previous phase; data with 29 features and the target Label, which has binary dependent variables represented by 0 for the benign class and 1 for the attack class, were utilized to classify network traffic using different types of classifiers (MLP, XGBoost, CNN, as well as the proposed method CNN_XGBoost). The main objective of classification in IDS is to identify attacker behavior and distinguish it from normal network traffic. Attacks are classified as a binary in the context of an IDS to distinguish between malicious and benign network traffic. The fourteen distinct attack types (Benign, Bot, Brute Force -Web, Brute Force -XSS, DDOS attack-HOIC, DDOS attack-LOIC-UDP, DDoS attacks-LOIC-HTTP, DoS attacks-GoldenEye, DoS attacks-Hulk, DoS attacks-SlowHTTPTest
DoS attacks-Slowloris, FTP-BruteForce, Infiltration, SQL Injection, SSH-Bruteforce), which are included in the utilized CICIDS2018 dataset, were labelled with the value 1 to represent malicious traffic and enabling binary classification.
In this research, four classifiers are applied to the training dataset to classify network traffic into normal and abnormal traffic. Here is a simple description for each of these:

4.3.1. **MLP classifier**: As mentioned in (3.6), each MLP has at least three layers. In this research, the MLP classifier is used with four layers. It consists of an input layer with several neurons equal to the number of x_train instances, two hidden layers with a 'relu' activation function, a dropout function, and an output layer with a softmax function that produces the attack class as an output of the network.
Also, the Adam optimizer is used, and the model is trained with 20 epochs and a batch size equal to 1000.

4.3.2. **XGBoost classifier**: As explained in the previous section (3.8), XGBoost provides efficient memory utilization and significantly decreases computation time.
In this research, the XGBoost classifier was built in Python by importing the XGBClassifier class from the XGBoost library and making an instance of the class. We trained the model on the training data for (features and target labels) to learn the relationships between features and labels and make accurate predictions on unseen data.

4.3.3. **CNN Classifier**: The advantage of CNN's ability to extract spatial features is taken. In this research, the CNN classifier structure shown in Figure (2) is used. A Convolutional Neural Network with 1Dconvolutional, Max pooling, Flatten, and Dense layers. These layers work together to extract features from input data. The convolutional layers learn to identify patterns, and the max pooling layers reduce the dimensionality while preserving important information. Then the flattened output is ready to be fed into fully connected (Dense) layers for further processing. Relu activation function is used, which enables CNN to learn more complex patterns that may indicate intrusive behavior by introducing nonlinearity.
The first column in Figure (2) represents the layer type utilized in this CNN model, each with a specific role. The output format in the second column refers to the number of positions(which decreases after convolution and pooling) and the number of features or channels generated by the filters. The last column represents the trainable parameter (weights and biases).

Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 25, 128)	768
max_pooling1d (MaxPooling1D)	(None, 12, 128)	0
conv1d_1 (Conv1D)	(None, 8, 64)	41024
max_pooling1d_1 (MaxPooling1D)	(None, 4, 64)	0
flatten (Flatten)	(None, 256)	0
dense_4 (Dense)	(None, 128)	32896
dropout_3 (Dropout)	(None, 128)	0
dense_5 (Dense)	(None, 2)	258

Total params: 74946 (292.76 KB)

Figure 1: Structure of the CNN classifier used for classification

4.3.4. ***CNN_XGBoost classifier****:* In this subsection, the classification procedure, which utilizes (29) features produced by the C_IG_C_FS method, will be discussed and explained. Then, a Convolutional Neural Network was used for its feature extraction capabilities. The proposed CNN consists of the following:
1) The first convolutional layer has 128 filters, each with a kernel size of (5). The activation function used is ReLU, which introduces non-linearity, and the selected (29 attributes) specify the shape of the input data.
2) A max pooling layer with a pool size of 2 is used.
3) The second convolutional layer has 64 filters, each with a kernel size of (5). The activation function is again ReLU.
4) Another max pooling layer, similar to the first one, is used.
5) After that, the layer that flattens the output of the convolutional layers into a 1D vector.
In summary, these layers, which are illustrated in Algorithm 2, work together to extract features from input data. The convolutional layers learn to identify patterns, and the max pooling layers reduce the dimensionality while preserving important information. The flattened output is then ready to be fed into the XGBoost classifier for training.

Algorithm 2: CNN used in the proposed classifier for Feature Extraction:
Input: 29 featured results after applying C_IG_C_FS method

Output: 256 extracted features

Begin:
Step 1: Add Convolutional Layer 1:
Filters = 128, kernel size = (5), activation function = ReLU
Output shape: (None, 25, 128)
Step 2: Add Max Pooling Layer1:
Pool size =2
Output shape: (None, 12, 128)
Step 3: Add Convolutional Layer 2:
Filters = 64, kernel size = (5), activation function = ReLU
Output shape: (None, 8, 64)
Step 4: Max Pooling Layer2:
Pool size =2
Output shape: (None, 4, 64)
Step 5: Flatten:
Flatten the output of the previous layer into a 1D vector.
Output shape: (None, 256)
End

The next step is to train an XGBoost classifier model using the extracted (256) features from the CNN model. Then the performance metrics are used to evaluate the proposed classification model.

5. Results and Discussion

This section encompasses several key aspects in all three phases of building the proposed model.

5.1. Results during the pre-processing phase

CSECICIDS2018 raw data has 80 features and about 15 million instances, as shown in Table (7). After pre-processing, the number of instances became less due to deletion (instances containing null, infinite, and duplicate values) using the pandas library in Python. On the other hand, three irrelevant features are deleted. As a result, the amount of data that moved to the next phase is 77 features, with many instances represented in Table(8).

Table 7: Number of Benign and Attack Instances in the CSECICIDS2018 dataset before preprocessing

class	Value count
Benign	12960361
Attack	2555191

Table 8: Number of benign and attack instances in the CSECICIDS2018 dataset after preprocessing

class	Value count
Benign	9809012
Attack	1330511

5.2. Results during Feature Selection Phase

The purpose of this part is to deal with features and remove some that negatively impact or might not improve the accuracy of the system.

Columns with constant values are eliminated at the beginning of this phase in order to simplify the training process and improve the model's performance by focusing on features that carry valuable information that help build a more accurate model. Table (2) shows the remaining 69 attributes.

After that, tables (3,4,5) show eliminated attributes from applying (Corr., IG, and chi_s) filter feature selection methods.

The intersection operation applied to the subsets results from applying Corr and IG methods and provides 36 common features, as shown in Table (6). The intersection of feature selection helps increase confidence in selecting highly important features, as each feature selection method has its own strengths and weaknesses. By taking the intersection of features selected by two different methods, this increases confidence that these features are indeed of importance to the model.

Ultimately, Table (9) displays the 29 most relevant attributes obtained from applying the Chi-squared method over 36 common features, results from the previous operation with the target feature.

Table 7: Selected Features From C_IG_C_FS the Proposed Method with the Target (Label)

(2)	(8)	(9)	(10)	(11)	(12)	(13)	(16)	(17)	(18)	(19)	(20)
(21)	(20)	(23)	(24)	(25)	(29)	(30)	(32)	(36)	(38)	(41)	(42)
(46)	(53)	(59)	(62)	(63)	(65)		(69)-'Label'				

5.3. Evaluation metrics

The precision, accuracy, recall, and F1 score are computed with the help of the confusion matrix to evaluate the proposed model's performance. These metrics can be expressed by equations 8, 9, 10, and 11.

$$\boldsymbol{Accuracy} = \frac{\boldsymbol{TP+TN}}{\boldsymbol{TP+FP+TN+FN}} \quad (8)$$

$$\boldsymbol{Precision} = \frac{\boldsymbol{TP}}{\boldsymbol{TP+FP}} \quad (9)$$

$$\boldsymbol{Recall} = \frac{\boldsymbol{TP}}{\boldsymbol{TP+FN}} \quad (10)$$

$$\boldsymbol{F1_{Score}} = \frac{\boldsymbol{2*Precision*Recall}}{\boldsymbol{Precision+Recall}} \quad (11)$$

Table 8: Confusion Matrix

predict / actual	**normal**	**abnormal**
normal	TP	FN
abnormal	FP	TN

True Positive (TP) in the confusion matrix refers to normal activity that is correctly detected as benign; True Negative (TN) represents an abnormal activity that is correctly detected as an attack; False Positive (FP) is abnormal activity incorrectly classified as benign; and False Negative (FN) refers to normal activity that is wrongly predicted as an attack. To fulfill the aim of IDS, the (80%) training set is used to learn the general pattern of network traffic and then predict normal and abnormal traffic in the (20%) test set.

5.4. Results during the classification phase

The main objective of the evaluation is to assess the proposed method's performance thoroughly.

The equations (8-11) based on the confusion matrix shown in Table (10) are used to evaluate the performance metrics of CNN_XGBoost, the proposed model, and other options

Table (11) provides performance metrics scores for each FS method in different classifiers.

Table 9: Evaluation metrics for four models with four FS techniques and clean data without FS

FS method	Model	acc	pre	recall	F1_score	FPR	FNR
Full (69-features)	**MLP**	0.989117	0.996081	0.912539	0.952482	0.08746	0.00048
	XGBoost	**0.989669**	**0.998406**	**0.915029**	**0.954901**	**0.08497**	**0.00019**
	CNN	0.989289	0.997051	0.913088	0.953224	0.08691	0.00036
	Proposed classifier CNN_XGBoost	0.989289	0.997010	0.913125	0.953226	0.08687	0.00037
Corr (51-features)	**MLP**	0.988405	0.994457	0.908434	0.949501	0.09156	0.00069
	XGBoost	**0.989735**	**0.998330**	0.915982	**0.955385**	0.08401	**0.00020**
	CNN	0.989347	0.997296	0.913693	0.953666	0.08630	0.00033
	Proposed classifier CNN_XGBoost	0.989723	0.997941	**0.916242**	0.955348	**0.08375**	0.00025
Chi-s (2-features)	**MLP**	0.988554	0.995241	0.908940	0.950135	0.09105	0.00059
	XGBoost	0.989504	**0.997993**	0.914350	0.954342	0.08564	**0.00025**
	CNN	0.988081	0.995814	0.904451	0.947936	0.09554	0.00051
	Proposed classifier CNN_XGBoost	**0.991052**	0.994444	**0.930591**	**0.961459**	**0.06940**	0.00070
IG (50-features)	**MLP**	0.988968	0.994831	0.912299	0.951779	0.08770	0.00064
	XGBoost	0.989662	**0.998401**	0.914969	0.954866	0.08503	**0.00019**
	CNN	0.989187	0.997171	0.912119	0.952750	0.08788	0.00035
	Proposed classifier CNN_XGBoost	**0.989769**	0.998151	**0.915962**	**0.955292**	**0.08403**	0.00022
Proposed method C_IC_C_FS (29-features)	**MLP**	0.988581	0.995741	0.908678	0.950219	0.09132	0.00052
	XGBoost	0.989752	**0.998548**	0.915886	0.955432	0.08411	**0.00018**
	CNN	0.989044	0.997216	0.911191	0.952265	0.08880	0.00034
	Proposed classifier CNN_XGBoost	**0.993819**	0.994571	**0.998427**	**0.996495**	**0.039993**	0.00157

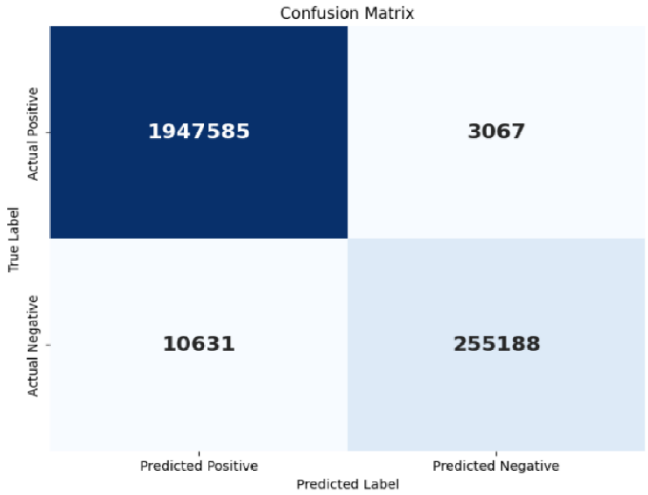

Figure 2: Confusion matrix of the proposed model

Table (11) presents a comparison between classifiers that were tested on features chosen in several ways. The heavy type indicates that this is the best result obtained using the corresponding classifier and based on the number of features chosen in a particular way, shown in the table. The results show that the proposed method achieved higher accuracy compared to others in the experiment, and the false positive rate was statistically significantly lower than the other results.

Figure (3) shows that performance metrics on test data achieved with the proposed method were 1947585 normal activities that were correctly detected as benign, and only 10631 abnormal activities incorrectly classified as benign. It also shows that 255188 abnormal activities were correctly detected as an attack and 3067 normal activities were wrongly predicted as an attack.

Table (12) highlights some of the key findings. It illustrates the highest accuracy obtained after applying four types of classifiers over selected features from applying chosen filter feature selection methods, in addition to the proposed method. It also indicates the classifier that achieved this accuracy.

Table 10: The number of features and the highest accuracy obtained after applying four strategies

Strategy	No. of features	Instances before FS with duplicate	Instances after FS without duplicate	accuracy	classifier
Corr	51	11139523	11069452	**98.9735%**	XGBoost
IG	50	11139523	11139354	**98.9769%**	Proposed(CNN_XGBoost)
Chi-s	29	11139523	11085280	**99.1052%**	Proposed(CNN_XGBoost)
C_IG_C_FS	29	11139523	11082351	**99.3819%**	Proposed(CNN_XGBoost)

Figures (4,5,6) illustrate the confusion matrix with results from applying four classifiers on selected features from each filter FS method, and Figure (7) shows the confusion matrix with results from applying four classifiers on the proposed FS method

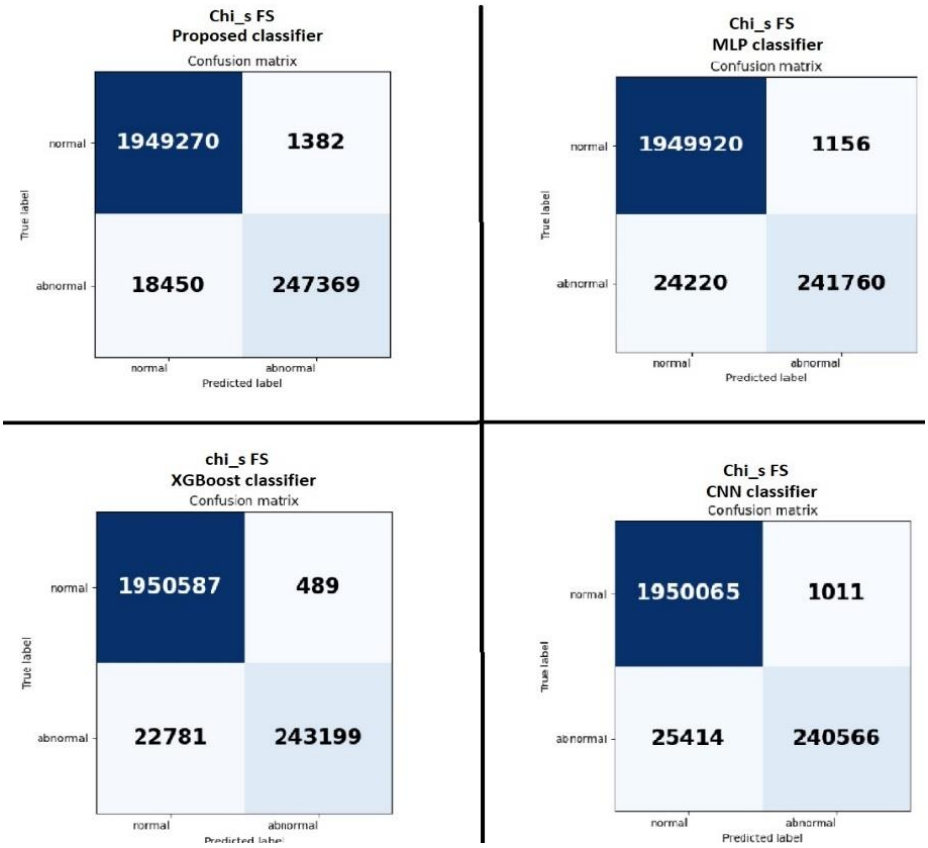

Figure 4 : CM results from inserting 29 features into 4 types of classifiers

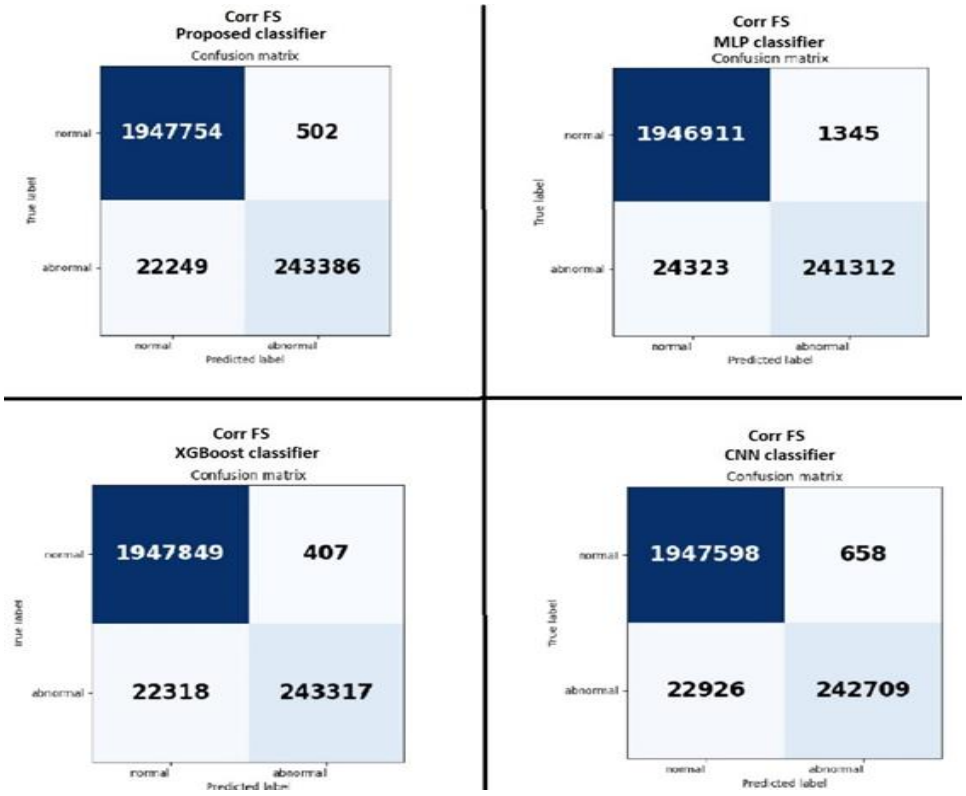

Figure 5: CM results from inserting 51 features into 4 types of classifiers

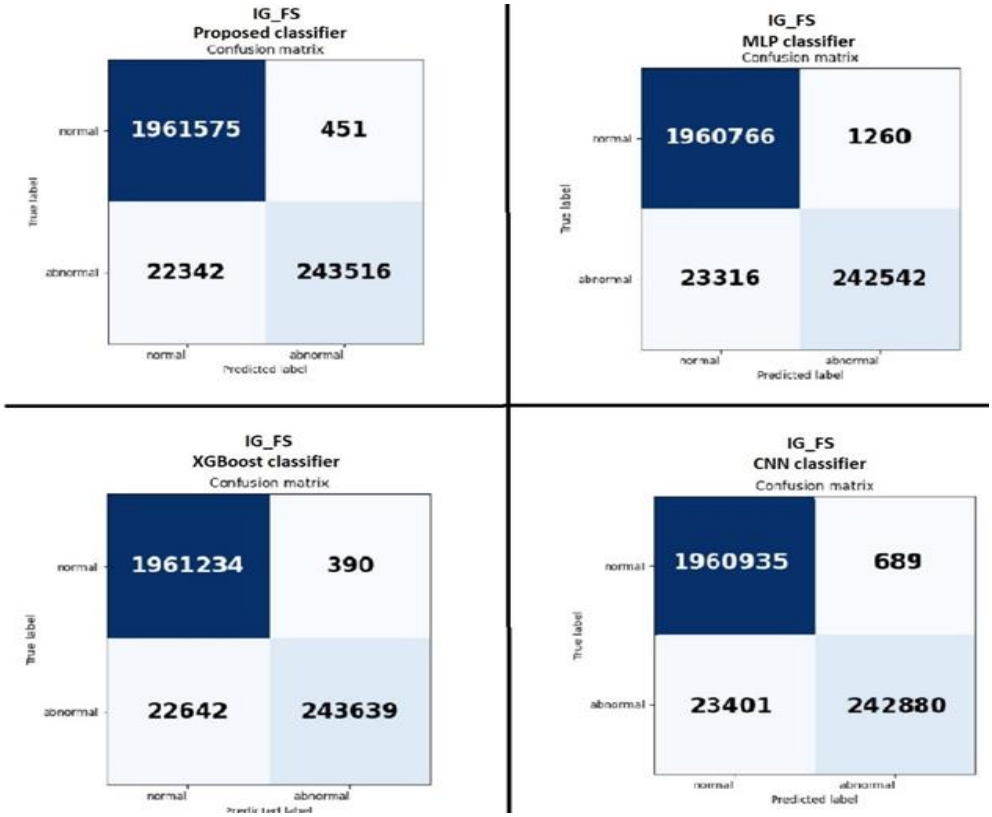

Figure 6: CM results from inserting 50 features into 4 types ofclassifiers

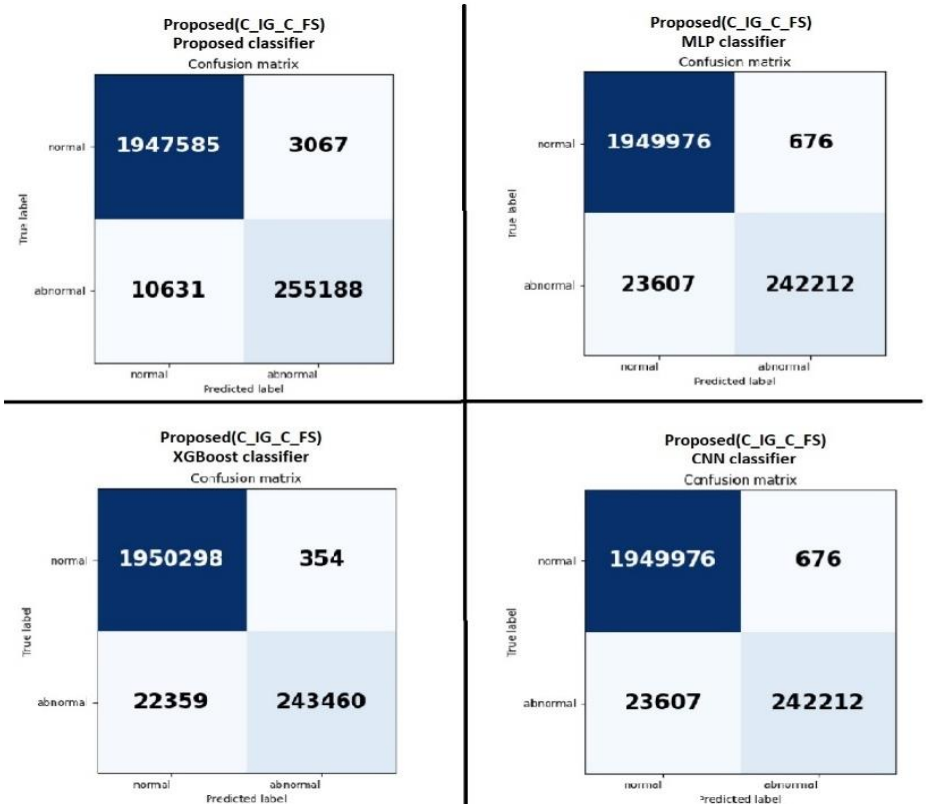

Figure 3 : CM results from inserting 29 features into 4 types of classifiers

5.5. Discussion

The previous results confirmed three significant findings: First, using straightforward methods (Filter FS) in an appropriate context (C_IG_C_FS) to select important attributes can achieve positive results and maintain the accuracy of the system. Results verified that using the classifiers in conjunction with the proposed (C_IG_C_FS) method produced better performance metrics scores compared to using it in conjunction with other straightforward FS methods (Corr, IG, Chi_s). Additionally, it demonstrated how well deep learning performs with vast amounts of data and how its ability to extract several data features makes it useful for detecting and classifying threats. Therefore, it was included in the proposed model, and it extracted (256) further features that helped to enhance the system's performance. Results can differ according to the used dataset or strategy. During the experiment, it was found that TensorFlow and Keras have sufficient tools to achieve outstanding results without needing an external optimization algorithm. However, the optimizer algorithms may be able to improve the system's performance even more. Ultimately, hybrid CNN with XGBoost presents results are best than using it with a SoftMax classifier, The algorithm was able to reduce false positive alarms, which is the most important part of an intruder detection system, because false negative (FN) - which is identifying benign traffic as an attack - does not pose a risk to the network performance, although it reduces the accuracy of the system, but the opposite is more dangerous, false positive (FP) which identifying malicious traffic as benign. Tabel (13) shows the amount of FP detection in each method.

Table 11: The amount of detecting false positive traffic flow by each classifier for the proposed FS method C_IG_C_FS

classifier	Total FP detections
MLP	24275
XGBoost	22359
CNN	24221
CNN_XGBoost	**10631**

The best accuracy of the proposed model compared with those using FS methods in related works mentioned in Table (1) is shown in Table (14). It illustrates that the proposed model provides higher accuracy compared to those related works that used other feature selection methods and classification algorithms; the related work that was not compared did not show that it used FS strategies.

Table 12: Best accuracy scores of the proposed model compared with related works

Author	features	FS method	model	acc
[12]	24	RF	MLP-BP(binary classification)	98.9%
[13]	-	PCA	PCM+FCM-SMO+AE	95.3%
[14]	-	AE	MLP	97.7%
[15]	12	PCA	PCA-DNN(binary classification)	96.7%
[19]	23	Chi-squared + Pearson correlation	Ensemble learning (LR+DT+GB)	98.8%
Proposed	29	C_IG_C	CNN-XGBoost	99.3%

6. Conclusion

This study proposed a stacked feature selection strategy (C_IG_C_FS). This strategy uses the intersection of Pearson correlation and Information Gain scores, followed by chi-square, which is applied on the common selected feature results from the two previous methods. As a result, the number of features was reduced from sixty-nine to only twenty-nine features.

After filtering, four classifiers were evaluated using these selected features. The results in Table (11) illustrate that the proposed module CNN_XGBoost outperforms other classifiers across four of the most important performance criteria, accuracy, recall, F1-score, and False Positive Rate. These results reflect the effectiveness of combining feature selection with hybrid classification. Thus, the result of the proposed model achieved its objectives, improved preprocessing, reduced dimensions while maintaining its performance stability, and increased Classification efficiency. However,, further improvement is still needed. In the future, the model could be tested on other datasets to verify its generalizability. The current feature selection strategy could also be compared with Wrapper or embedded methods to gain a broader view of efficiency. It is also proposed to explore the integration of nature-inspired optimization algorithms, such as Particle Swarm Optimization or Grey Wolf Optimizer, into the feature selection process.

References

[1] [1] R. A. Azeez, M. K. Abdul-Hussein, M. S. Mahdi, and H. T. S. ALRikabi, "Design a system for an approved video copyright over cloud based on biometric iris and random walk generator using watermark technique," Periodicals of Engineering and Natural Sciences, vol. 10, no. 1, Art. no. 1, Dec. 2021, doi: 10.21533/pen.v10i1.2577.

[2] "Intrusion Detection Systems in Cloud Computing Paradigm: Analysis and Overview - Rana - 2022 - Complexity - Wiley Online Library." Accessed: Jul. 27, 2024. [Online]. Available: https://onlinelibrary.wiley.com/doi/full/10.1155/2022/3999039

[3] "Applied Sciences | Free Full-Text | Enhanced Chimp Optimization-Based Feature Selection with Fuzzy Logic-Based Intrusion Detection System in Cloud Environment." Accessed: Jul. 27, 2024. [Online]. Available: https://www.mdpi.com/2076-3417/13/4/2580

[4] "Search," Amazon Web Services, Inc. Accessed: Jul. 27, 2024. [Online]. Available: https://aws.amazon.com/search/

[5] Y. Mehmood, M. A. Shibli, U. Habiba, and R. Masood, "Intrusion Detection System in Cloud Computing: Challenges and opportunities," in 2013 2nd National Conference on Information Assurance (NCIA), Rawalpindi, Pakistan: IEEE, Dec. 2013, pp. 59–66. doi: 10.1109/NCIA.2013.6725325.

[6] A. Muhsen, G. Jumaa, N. Al Bakri, and A. Sadiq, "Feature Selection Strategy for Network Intrusion Detection System (NIDS) Using Meerkat Clan Algorithm," International Journal of Interactive Mobile Technologies (iJIM), vol. 15, p. 158, Aug. 2021, doi: 10.3991/ijim.v15i16.24173.

[7] "Filter Feature Selection - an overview | ScienceDirect Topics." Accessed: Jul. 25, 2024. [Online]. Available: https://www.sciencedirect.com/topics/computer-science/filter-feature-selection

[8] "Optimizing Feature Selection Method in Intrusion Detection System Using Thresholding," IJIES, vol. 17, no. 3, pp. 214–226, Jun. 2024, doi: 10.22266/ijies2024.0630.18.

[9] W. H. Aljuaid and S. S. Alshamrani, "A Deep Learning Approach for Intrusion Detection Systems in Cloud Computing Environments," Applied Sciences, vol. 14, no. 13, Art. no. 13, Jan. 2024, doi: 10.3390/app14135381.

[10] M. Gopalsamy, "Predictive Cyber Attack Detection in Cloud Environments with Machine Learning from the CICIDS 2018 Dataset," vol. 10, no. 10, 2024.

[11] A. D. Vibhute and V. Nakum, "Deep learning-based network anomaly detection and classification in an imbalanced cloud environment," Procedia Computer Science, vol. 232, pp. 1636–1645, Jan. 2024, doi: 10.1016/j.procs.2024.01.161.

[12] S. Alzughaibi and S. El Khediri, "A Cloud Intrusion Detection Systems Based on DNN Using Backpropagation and PSO on the CSE-CIC-IDS2018 Dataset," Applied Sciences, vol. 13, no. 4, Art. no. 4, Jan. 2023, doi: 10.3390/app13042276.

[13] B. R M and J. K. M K, "Intrusion Detection on AWS Cloud through Hybrid Deep Learning Algorithm," Electronics, vol. 12, no. 6, Art. no. 6, Jan. 2023, doi: 10.3390/electronics12061423.

[14] M. P. P. J. C. Haripriya, "An Efficient Autoencoder Based Deep Learning Technique to Detect Network Intrusions," International Transaction Journal of Engineering, vol. Management, p. 13A7P: 19, 2022, doi: 10.14456/ITJEMAST.2022.142.

[15] M. Al-Fawa'reh, M. Al-Fayoumi, S. Nashwan, and S. Fraihat, "Cyber threat intelligence using PCA-DNN model to detect abnormal network behavior," Egyptian Informatics Journal, vol. 23, no. 2, pp. 173–185, Jul. 2022, doi: 10.1016/j.eij.2021.12.001.

[16] M. A. Khan, "HCRNNIDS: Hybrid Convolutional Recurrent Neural Network-Based Network Intrusion Detection System," Processes, vol. 9, no. 5, Art. no. 5, May 2021, doi: 10.3390/pr9050834.

[17] Y. N. Kunang, S. Nurmaini, D. Stiawan, and B. Y. Suprapto, "Attack classification of an intrusion detection system using deep learning and hyperparameter optimization," Journal of Information Security and Applications, vol. 58, p. 102804, May 2021, doi: 10.1016/j.jisa.2021.102804.

[18] R. I. Farhan, A. T. Maolood, and N. F. Hassan, "Performance analysis of flow-based attacks detection on CSE-CIC-IDS2018 dataset using deep learning," Indonesian Journal of Electrical Engineering and Computer Science, vol. 20, no. 3, Art. no. 3, Dec. 2020, doi: 10.11591/ijeecs.v20.i3.pp1413-1418.

[19] Q. R. S. Fitni and K. Ramli, "Implementation of Ensemble Learning and Feature Selection for Performance Improvements in Anomaly-Based Intrusion Detection Systems," in 2020 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT), Jul. 2020, pp. 118–124. doi: 10.1109/IAICT50021.2020.9172014.

[20] V. Kanimozhi and T. P. Jacob, "Artificial Intelligence based Network Intrusion Detection with hyper-parameter optimization tuning on the realistic cyber dataset CSE-CIC-IDS2018 using cloud computing," ICT Express, vol. 5, no. 3, pp. 211–214, Sep. 2019, doi: 10.1016/j.icte.2019.03.003.

[21] P. Lin, K. Ye, and C.-Z. Xu, "Dynamic Network Anomaly Detection System by Using Deep Learning Techniques," in Cloud Computing – CLOUD 2019, D. Da Silva, Q. Wang, and L.-J. Zhang, Eds., Cham: Springer International Publishing, 2019, pp. 161–176. doi: 10.1007/978-3-030-23502-4_12.

[22] V. Engström, P. Johnson, R. Lagerström, E. Ringdahl, and M. Wällstedt, "Automated Security Assessments of Amazon Web Services Environments," ACM Trans. Priv. Secur., vol. 26, no. 2, pp. 1–31, May 2023, doi: 10.1145/3570903.

[23] B. Gupta, P. Mittal, and T. Mufti, "A Review on Amazon Web Service (AWS), Microsoft Azure & Google Cloud Platform (GCP) Services," in Proceedings of the 2nd International Conference on ICT for Digital, Smart, and Sustainable Development, ICIDSSD 2020, 27-28 February 2020, Jamia Hamdard, New Delhi, India, New Delhi, India: EAI, 2021. doi: 10.4108/eai.27-2-2020.2303255.

[24] S. Mathew, "Overview of Amazon Web Services," 2014.

[25] S. El Kafhali, I. El Mir, and M. Hanini, "Security Threats, Defense Mechanisms, Challenges, and Future Directions in Cloud Computing," Arch Computat Methods Eng, vol. 29, no. 1, pp. 223–246, Jan. 2022, doi: 10.1007/s11831-021-09573-y.

[26] "Future Internet | Free Full-Text | A Survey on Intrusion Detection Systems for Fog and Cloud Computing." Accessed: Jul. 25, 2024. [Online]. Available: https://www.mdpi.com/1999-5903/14/3/89

[27] S. Lata and D. Singh, "Intrusion detection system in cloud environment: Literature survey & future research directions," International Journal of Information Management Data Insights, vol. 2, no. 2, p. 100134, Nov. 2022, doi: 10.1016/j.jjimei.2022.100134.

[28] "A Deep Learning Method with Filter Based Feature Engineering for Wireless Intrusion Detection System | IEEE Journals & Magazine | IEEE Xplore." Accessed: Jul. 25, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/8668403

[29] H. Attou et al., "Towards an Intelligent Intrusion Detection System to Detect Malicious Activities in Cloud Computing," Applied Sciences, vol. 13, no. 17, Art. no. 17, Jan. 2023, doi: 10.3390/app13179588.

[30] S. Lata and D. Singh, "Intrusion detection system in cloud environment: Literature survey & future research directions," International Journal of Information Management Data Insights, vol. 2, no. 2, p. 100134, Nov. 2022, doi: 10.1016/j.jjimei.2022.100134.

[31] M. A. Siddiqi and W. Pak, "Optimizing Filter-Based Feature Selection Method Flow for Intrusion Detection System," Electronics, vol. 9, no. 12, Art. no. 12, Dec. 2020, doi: 10.3390/electronics9122114.

[32] M. Bakro et al., "Efficient Intrusion Detection System in the Cloud Using Fusion Feature Selection Approaches and an Ensemble Classifier," Electronics, vol. 12, no. 11, Art. no. 11, Jan. 2023, doi: 10.3390/electronics12112427.

[33] Kurniabudi, D. Stiawan, Darmawijoyo, M. Y. B. Idris, A. M. Bamhdi, and R. Budiarto, "CICIDS-2017 Dataset Feature Analysis with Information Gain for Anomaly Detection," IEEE Access, vol. 8, p. 132911, 2020.

[34] Accessed: Jul. 25, 2024. [Online]. Available: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-023-00694-8

[35] J. Brownlee, "A Gentle Introduction to XGBoost for Applied Machine Learning," MachineLearningMastery.com. Accessed: Jul. 25, 2024. [Online]. Available: https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/

[36] J. Kim, J. Kim, H. Kim, M. Shim, and E. Choi, "CNN-Based Network Intrusion Detection against Denial-of-Service Attacks," Electronics, vol. 9, p. 916, Jun. 2020, doi: 10.3390/electronics9060916.

[37] "(PDF) CNN-LSTM: Hybrid Deep Neural Network for Network Intrusion Detection System." Accessed: Jul. 25, 2024. [Online]. Available: https://www.researchgate.net/publication/363570685_CNN-LSTM_Hybrid_Deep_Neural_Network_for_Network_Intrusion_Detection_System

[38] S. Songma, T. Sathuphan, and T. Pamutha, "Optimizing Intrusion Detection Systems in Three Phases on the CSE-CIC-IDS-2018 Dataset," Computers, vol. 12, no. 12, Art. no. 12, Dec. 2023, doi: 10.3390/computers12120245.

[39] D. Yadav, "Categorical encoding using Label-Encoding and One-Hot-Encoder," Medium. Accessed: Jul. 25, 2024. [Online]. Available: https://towardsdatascience.com/categorical-encoding-using-label-encoding-and-one-hot-encoder-911ef77fb5bd