National College of Ireland

Student ID: 22248056

School of Computing

National College of Ireland

# ADVANCING NETWORK INTRUSION DETECTION SYSTEMS THROUGH MACHINE LEARNING ALGORITHMS

MSc Research Project

MSc Data Analytics

Arya Eldho

Supervisor: Jorge Basilio

**National College of Ireland**

**MSc Project Submission Sheet**

**School of Computing**

**Student Name:** ……. ……ARYA ELDHO……

**Student ID:** ……22248056……..……

**Programme:** ……MSC DATA ANALYTICS…… **Year:** ……2024……..

**Module:** ……MSC RESEARCH PROJECT……

**Supervisor:** ……JORGE BASILIO……..……

**Submission Due Date:** ……..……
…

**Project Title:** ……ADVANCING NETWORK INTRUSION DETECTION SYSTEMS THROUGH MACHINE LEARNING ALGORITHMS……

**Word Count:** ……8230……
**PageCount**……20……..……..

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

**Signature:** ……

**Date:** ……

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST**

| | |
|---|---|
| Attach a completed copy of this sheet to each project (including multiple copies) | □ |
| **Attach a Moodle submission receipt of the online project submission,** to each project (including multiple copies). | □ |
| **You must ensure that you retain a HARD COPY of the project**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. | □ |

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

| **Office Use Only** | |
|---|---|
| Signature: | |
| Date: | |
| Penalty Applied (if applicable): | |

# ADVANCING NETWORK INTRUSION DETECTION SYSTEMS THROUGH MACHINE LEARNING ALGORITHMS

**ARYA ELDHO**
**22248056**

**Abstract**

Network intrusions pose a significant threat to cybersecurity. To secure the data being transferred online an Intrusion Detection System(IDS) is needed. IDSs detect intrusions in networks. In the study proposed here a machine learning based IDS is built. The machine learning models, SVM, Random Forest(RF), KNN, and the Convolutional Neural Network-Gated Recurrent Unit-Bidirectional Long Short Term Memory(CNN-GRU-BiLSTM) models are used in the study. The dataset used in the study is the NSL-KDD dataset. The dimensionality of the features in the dataset is reduced using both Recursive Feature Elimination (RFE) and Principal Component Analysis(PCA). The data is balanced using Synthetic Minority Oversampling(SMOTE). These models were trained and evaluated to get the best model to detect the intrusions in NIDS. The results of the study show that the best performance is shown by the CNN-GRU-BiLSTM when it is used along with the RFE as it achieved a validation accuracy of 97%. The study also shows that the accuracy of a machine learning model increases if the dimensionality of its features is reduced. The CNN-GRU-BiLSTM is used to build a desktop application and the desktop application is able to successfully detect and classify intrusions. The IDS model is developed successfully. The machine learning models and the desktop application are built using Python.

# 1 Introduction

Data transfer through networks has become an unavoidable part of a large number of companies in the world. However, a major issue that is present in transferring data over networks is cyber attacks on networks(Kala, 2023). Cyber attacks are occurring frequently on networks and this has led to the loss of important data. It was found that a cyber attack took place every 39 seconds in the world(Jain,2022). Cyber attackers can cause different kinds of harm on the networks like disrupting services and causing delays in services. The data being transferred through a network can be encrypted by an attacker and the original owners of the data will not be able to use the data(Kala, 2023). An attacker then may demand a ransom from the people who own the data. As cyber attacks have become a major issue in data transfer over networks, these attacks must be prevented or the damages caused by the cyber attacks on networks need to be mitigated. Network intrusion detection systems(NIDSs) are a way in which cyber attacks in networks can be prevented. There are different kinds of cyber attacks denial of service, and viruses that affect computer systems in networks. These cyber attacks enter into the network as a piece of code or malware and this malware causes disruption in services and leakage of data. Malware causes problems as long as it is present in a network and important services and data are also affected. If a malware is effectively detected in a network then it can be removed from a network and the problems caused by malware can be mitigated.

The NIDS built for detecting cyber attacks can be divided mainly into signature-based and anomaly-based (Einy, Oz and Navaei, 2021). Signature-based NIDSs work based on historical data like indicators of compromise. Patterns for normal behaviour and abnormal networks are identified in this type of NIDSs and these are used for attacks that are previously known. Anomaly-based NIDSs are able to detect both previously known and new attacks that cause abnormal behaviour in networks(Wang et al., 2022). However, the anomaly-based NIDSs generate a huge number of false positives during the detection of cyber attacks in networks and this is not desirable as the cyber attacks in networks will not be properly detected(Wang et al., 2022).

In this study NIDS based techniques are used for detecting and classifying cyber attacks. The major issue caused while working on the datasets related to this is the class imbalance problem. This is found when large number of samples belongs to one class. Class imbalance causes the ML model to be biased and it badly affects the performance of the NIDS model(Mduma, 2023). Another issue that negatively

affects the performance of the machine learning model is the presence of features that have dimensionality. Dimensionality is the number of dimensions that are needed to represent the data associated with a feature(Jia et al., 2022). However, features having a large number of dimensions negatively affect the performance of the machine learning models(Velliangiri, Alagumuthukrishnan and Thankumar joseph, 2019). Both the problems of class imbalance and dimensionality can be solved using data balancing and dimensionality reduction techniques. In the study performed here a NIDS based on machine learning will be built for detecting and classifying intrusions in networks. Data balancing and feature dimensionality reduction techniques will be used in the study to make sure that the NIDS based on machine learning shows the best performance.

In the study, section 2 will be discussing about the related work, section 3 will be discussing about the research procedure as well as research methodology, section 4 discusses the techniques that underline the implementation, section 5 will discuss the final stage of the implementation of the proposed solution, section 6 gives a comprehensive analysis of the results and main findings of the study, section 7 deals with conclusion and future work.

### 1.1 Aim

The aim of the study is to build a NIDS based on machine learning, whose prediction performance is improved using data balancing and feature dimensionality reduction techniques, for the detection and classification.
To Build a desktop application with the machine learning model having the best performance in network attack detection and classification.

### 1.2 Research Questions

- How can machine learning algorithms be harnessed to enhance the precision and speed of network intrusion detection systems by employing the Network Intrusion Detection dataset for the training and assessment of the performance of numerous ML models?
- Which particular machine learning methods showed optimal performance in the context of NIDSs using the dataset features?
- What is the contribution of different feature selection methods in NIDSs based on machine learning, and how can they be fine-tuned to have the best performance?

# 2 Related Work

The literature associated with the NIDSs based on deep learning and machine learning is discussed in this study.

### 2.1 NIDSs based on machine learning

The effectiveness of machine learning algorithms in the detection of cyber attacks is analysed in the study by (Saini and Dr. Arvind Kalia, 2023). Different feature-based, neural networks and state-based network intrusion detection models are considered in the study. Different datasets that can be used for training machine learning models for network intrusion detection like KDDCup and CCC are analysed in this study. The results of the study show that the machine learning models are effective in the detection of network intrusions and it was seen that the machine learning based models are able to achieve an accuracy of 98%. However, the main limitation of the study is that it is only a survey of the NIDSs based on machine learning.
Deep learning and Machine learning models are used for the detection of intrusions in networks in the study by (Dhanya et al., 2023). The machine learning models considered in the study included the SVM, Decision Tree(DT), RF, AdaBoost, XgBoost and Multi Layer Perceptron(MLP). A deep MLP is also considered in the study. The UNSW-NB15 dataset is used in the study. From the result of the study, it is seen that the best performance in network intrusion detection is achieved by the DT as it achieved an accuracy of 99.05%. The deep MLP model that used the ADAM optimiser achieved an accuracy of

98.44%. The study shows that machine learning and deep learning techniques are effective in the detection of network intrusions. However, the issues that may be associated with the data in the dataset like class imbalance are not addressed in this study.
Machine learning is utilised for the detection of cyber attacks in the study by (Sonule, 2021). The Naïve Bayes(NB) classifier is used in the study. The UNSW-NB15 dataset is used in the study. A binary classification is performed by the model proposed in the study. The results of the study show that the NB model proposed here achieves an accuracy of 66.7%. However, the main limitation of this study is that the accuracy achieved by the NB classifier is low.
Supervised learning algorithms are used for the detection of intrusion in IoT networks in the study by (Krishnan, Neyaz and Liu, 2021). The supervised learning algorithms considered in the study include the RF, Support Vector Classifier(SVC) and XgBoost. An IoT dataset utilised in a previous study is used here. Then the best features in the dataset are selected using Sequential Backward Processing. The results of the study show that the best performance is shown by the RF model as it achieves an accuracy of 99.78%. However, the operations like data balancing are not performed in this study.

Three machine learning algorithms are considered for the detection of intrusions in the study by(Alkasassbeh and Almseidin, 2018). The machine learning algorithms considered in the study include Bayesian network, MLP and J48. The KDD dataset is utilised in the study. The model proposed in the study classifies cyber attacks into four different categories. From the results of the study, it is seen that the best performance in detecting and classifying network intrusions is shown by the J48 model as it achieved an accuracy of 93.1%, a ROC value of 0.969 and a precision of 98.9%. However, techniques to improve the performances of the machine learning algorithms in detecting and classifying intrusions are not considered in the study.
Machine learning algorithms are utilised for identifying the cyber attacks in the study conducted by (Jaradat et..,2022). DT, RProp and SVM are the ML algorithms and CICIDS2017 dataset are utilised in the study. Konstanz Information Miner (KNIME) analytics platform process the data in the dataset which gives the best accuracy of 94.72%.
(Rincy N and Gupta, 2021) proposed NID-shield, a hybrid intrusion detection system which combines Wrapper approaches and Correlation-based Feature Selection (CFS) which selects the best subsets from the data for detecting the cyber-attacks. The NSL-KDD and UNSW-NB15 are utilised in the study. The string values in the data are converted to numerical values and then scaled. However the system proposed here is expensive.
In the study conducted by(Jamadar, 2018), CIDCIDS 2017 datasets are used, DT based systems are used to identify the anomaly based detections in the system. The research results shows that the proposed study receives an accuracy of 99%, a False Positive Rate(FPR) of 0.1% and a True Positive Rate(TPR) of 99.9%. The best features are retrieved from the model using RFE. But the DT tends can overfit during the training of the model and this can affect the performance of the model as well.

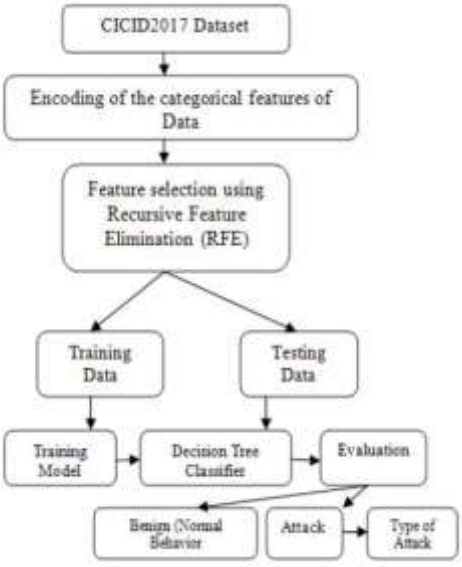



*Fig 1: the DT based anomaly detection system(Jamadar, 2018)*

Machine learning is used to build a network intrusion detection system in the study by (Awadallah Awad, 2021). The machine learning models used in the study include Artificial Neural Networks, KNearest Neighbour, Decision Tables, SVM, J48 DT and NB. The KDD99 dataset is used in the study. The model proposed in the study detects and classifies the attacks into 4 different types of attacks. The best performance is shown by the J48 Dt as it achieved an accuracy of 99.2% and an FPR of 0.9%. However, this method does not use techniques like feature dimensionality reduction which may improve the performance of the machine learning models.

### 2.2 NIDS based on deep learning

Many deep learning methods used for identifying cyber attacks are mentioned in the study by (Wu, Wei and Feng, 2020). They include CNN, recurrent neural networks (RNNs) and generative adversarial networks (GANs). Different datasets such as KDDcup 99, NSL-KDD, BoT-IoT and CSE-CIC-IDS2018 are identified in the study. The results concludes CNN to be the best deep learning model. The major drawback of the study is that, this is survey analysis and there is no actual NIDS proposed in the paper. The study by (Taher, Mohammed Yasin Jisan and Rahman, 2019), uses machine learning and deep learning techniques to identify the intrusions in the network. The Artificial Neural Network(ANN) and SVM are used in this study. The NSL-KDD dataset is used in the study to train the deep learning and machine learning models. The best features in the dataset are selected using Chi Square and correlation based feature selection. The irrelevant features in the dataset are analysed and removed in the study. The results of the study show that the ANN model proposed in the study shows the best performance as it achieves an accuracy of 96.88% while the SVM achieves an accuracy of 84.73%. This study shows that the NSL-KDD dataset is effective in training deep learning and machine learning models to detect intrusions in networks. However, the study does not perform data balancing and this may have affected the performances of the NIDS models.

Neural network classifiers and ensemble machine learning models are used in the detection of network attacks in the study by(Labonne, 2020). The NSL-KDD and KDD Cup 99 datasets are used in this study. Transfer learning is used in this study to make the models handle large amounts of unlabelled data. The results of the study show that the best performance is shown by a cascade-structured meta-specialists architecture and it achieves an accuracy of 88.39%, FPR of 1.94% for the NSL-KDD dataset and 94.44%, FPR of 0.33% for the KDD Cup 99 dataset. However, the architecture proposed in the study is computationally expensive.

Deep learning is used for the detection of intrusions in software defined networks(SDN) in the study by (Kurochkin and Volkov, 2020). The GRU is utilised in the study and the model based on the GRU detects and classifies intrusion into seven categories. The CSE-CIC-IDS2018 dataset is used in the study. The GRU contains a self-attention layer. The results of the study show that the GRU model achieves a precision of 0.94 and a recall of 0.99 in the detection of benign networks. The GRU model achieves both precision and recall as 1 in classifying the DDOS attack class. However, only a single deep learning model is utilised in the study.

A GRU-based lightweight neural network is proposed for the detection of vehicular intrusions in Controller Area Network (CAN) bus in the study by (Ma et al., 2022). The dataset containing data associated with networks containing cyber attacks and normal networks is used in the study. A feature extraction algorithm is utilised in the study. The results of the study show that the GRU model shows a good performance in detecting intrusions. However, the utilized of the GPU in the study for the detection of intrusions in the study is a limitation of the study.

Analyzing the study by (Umair et al., 2022), hybrid multilayer deep learning model is utilized to enhance the intrusion detections. KDDCUP'99 and NSL-KDD are the datasets used in the study, and this model uses a multilayer CNN that uses a softmax classifier to classify the intrusions. The results shows that the best performance is achieved when CNN and LSTM are combined with an accuracy of 99%. These studies concluded that hybrid deep learning models are highly effective for intrusion detection.

Hybrid deep learning models are utilized by (Qazi, Muhammad Hamza Faheem and Zia, 2023), in his study for intrusion detection. The study proposed here combines CNN and RNN. CICIDS-2018 dataset

is utilized in the study. Using oversampling the class imbalance analysed in the data is removed. The results shows that the model achieves an accuracy of 98.9%.

# 3 Research Methodology

The research proposes to develop and evaluate anomaly detection models using traditional as well as deep learning architectures. The steps involved in the process are data collection, preprocessing, model development, evaluation and results interpretation.

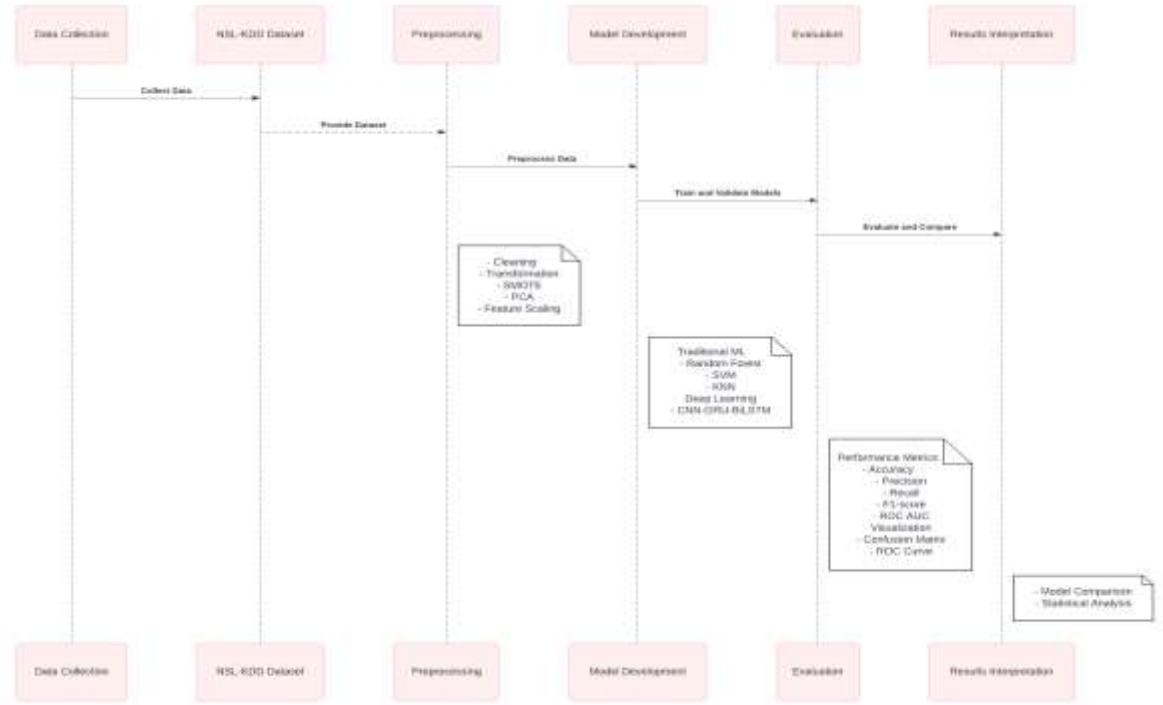



*Fig 2: The methodology overview*

### 3.1 Data Collection

The dataset used for the study is NSL-KDD dataset, taken from Kaggle, which is a very helpful source in evaluating and enhancing network intrusion detection systems. Each record in NSL-KDD dataset consists of 42 features, 41 characters related to network traffic, and two labels indicating normal or attack traffic.

| Feature No. | Feature Name | Description |
|---|---|---|
| 1 | duration | Length (number of seconds) of the connection |
| 2 | protocol_type | Type of protocol (e.g., tcp, udp, icmp) |
| 3 | service | Network service on the destination (e.g., http, telnet, ftp) |
| 4 | flag | Normal or error status of the connection |
| 5 | src_bytes | Number of data bytes from source to destination |
| 6 | dst_bytes | Number of data bytes from destination to source |
| 7 | land | 1 if connection is from/to the same host/port; 0 otherwise |
| 8 | wrong_fragment | Number of wrong fragments |
| 9 | urgent | Number of urgent packets |
| 10 | hot | Number of "hot" indicators |
| 11 | num_failed_logins | Number of failed login attempts |
| 12 | logged_in | 1 if successfully logged in; 0 otherwise |
| 13 | num_compromised | Number of compromised conditions |
| 14 | root_shell | 1 if root shell is obtained; 0 otherwise |

| | | |
|---|---|---|
| 15 | su_attempted | 1 if "su root" command attempted; 0 otherwise |
| 16 | num_root | Number of "root" accesses |
| 17 | num_file_creations | Number of file creation operations |
| 18 | num_shells | Number of shell prompts |
| 19 | num_access_files | Number of operations on access control files |
| 20 | num_outbound_cmds | Number of outbound commands in an ftp session |
| 21 | is_host_login | 1 if the login belongs to the "host" list; 0 otherwise |
| 22 | is_guest_login | 1 if the login is a "guest" login; 0 otherwise |
| 23 | count | Number of connections to the same host as the current connection in the past 2 seconds |
| 24 | srv_count | Number of connections to the same service as the current connection in the past 2 seconds |
| 25 | serror_rate | % of connections that have "SYN" errors |
| 26 | srv_serror_rate | % of connections that have "SYN" errors for the same service |
| 27 | rerror_rate | % of connections that have "REJ" errors |
| 28 | srv_rerror_rate | % of connections that have "REJ" errors for the same service |
| 29 | same_srv_rate | % of connections to the same service |
| 30 | diff_srv_rate | % of connections to different services |
| 31 | srv_diff_host_rate | % of connections to different hosts |
| 32 | dst_host_count | Number of connections to the same destination host as the current connection in the past 2 seconds |
| 33 | dst_host_srv_count | Number of connections to the same service as the current connection in the past 2 seconds |
| 34 | dst_host_same_srv_rate | % of connections to the same service in the past 2 seconds |
| 35 | dst_host_diff_srv_rate | % of connections to different services in the past 2 seconds |
| 36 | dst_host_same_src_port_rate | % of connections to the same source port in the past 2 seconds |
| 37 | dst_host_srv_diff_host_rate | % of connections to different hosts for the same service in the past 2 seconds |
| 38 | dst_host_serror_rate | % of connections that have "SYN" errors |
| 39 | dst_host_srv_serror_rate | % of connections that have "SYN" errors for the same service |
| 40 | dst_host_rerror_rate | % of connections that have "REJ" errors |
| 41 | dst_host_srv_rerror_rate | % of connections that have "REJ" errors for the same service |
| 42 | class | Label (either normal or specific attack type) |

Datasets Included:

- KDDTrain+: Training dataset with full 41 features plus class label.
- KDDTrain+_20Percent: 20% of the training dataset.
- KDDTest+: Testing dataset with full 41 features plus class label.
- KDDTest-21: A challenging test dataset.

### 3.2 Data Preprocessing

In the preprocessing stage several steps were taken to prepare the NSL-KDD dataset, for effective anomaly detection modelling. For a reliable dataset for analysis, incomplete rows were removed, categorical features were transformed into numerical values, which helps the machine learning algorithms to interpret these variables efficiently during training and evaluation. The dataset also exhibits class imbalance, so the Synthetic Minority Over-sampling Technique (SMOTE) was employed to mitigate the imbalance and ensured all models learned effectively from all the attacks.
Feature scaling was applied following this using StandardScaler from scikit-learn. By defining the range of numerical values, the normalization has prevented larger scale variables from controlling the training process. Inorder to resolve all the scaling differences, StandardScalar was enabled, guaranteed each feature contributed in relation with the learning process without favouring any variables.
The preprocessing steps includes feature scaling, management of class imbalance using SMOTE, data cleaning and transformation. These make sure that NSL-KDD dataset is ready to predict the anomaly detection and extract insights from the traffic data.

### 3.3 Model Development

For the anomaly detection in the network traffic, both traditional and deep learning methods were used. Traditional models include Random Forest Classifier, Support Vector Machine (SVM), and K-Nearest Neighbours (KNN). These models were first trained on the preprocessed dataset NSL-KDD.
Simultaneously, deep learning model, CNN-GRU-BiLSTM was constructed. During the training phase Adam optimizer was used to update the model weights, and early stopping and model check pointing mechanisms were used to prevent overfitting and preserve the best-performing model configuration.

### 3.4 Evaluation Methodology

In the evaluation methodology, several metrices were used to analyse the model performance to classify the network traffic as normal or anomalous. These include accuracy, precision, recall, F1 score and area under the receiver operating characteristic curve (ROC AUC). The output from these matrices efficiently identifies models capacity to differentiate normal traffic and anomalous traffic in the network. Confusion matrices were also generated to graphically represent the distributions. Additionally ROC curves also analyses the true positive and false positive rates. Statistical methods were also employed in the study to support the performance evaluation of the models.

# 4 Design Specification

CNN-GRU-BiLSTM is one of the most advanced deep learning technique to identify anomalies in the network. To enhance the network security, its architecture provides feature extraction, sequence learning and bidirectional context understanding. The architecture described here is used to detect the anomalies in the NSL-KDD dataset for the research study.

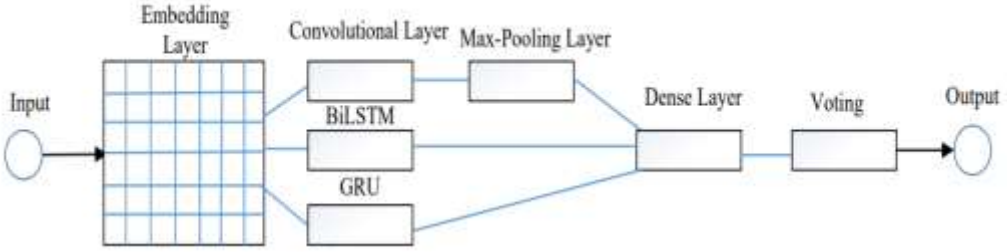



*Fig 3: CNN-GRU-BiLSTM architecture*

The initial layer receives pre-processed data, which then flows into 1D Convolutional (Conv1D) layer, with 64 filters and ReLU activation for extracting local patterns. Then the data get flowed into Max Pooling Layer simplifying the future processing. For enhancing feature extraction capabilities, the data is then passed through a fully connected Dense layer with 128 units. The output obtained is reshaped into a 3D tensor suitable for processing. The GRU layer with 64 units captures all the temporal dependencies in the data maintaining the sequences. The generated output is further handled by Bidirectional LSTM layer which analyse the data in both forward and backward directions improving the model performance. Finally the output layer and dense layer produces class probability for each network. Throughout the training, the model achieves the optimal performance using Adam optimizer with early stopping and model checkpoint mechanisms. With the aim of obtaining reliable and accurate anomaly detection, the complex architecture provides the use of recurrent and bidirectional layers to efficiently detect abnormalities in the network traffic data.

# 5 Implementation

### 5.1 Loading the dataset and Preprocessing

First step is to import all the libraries for data preprocessing, model building, evaluation and visualization. The pandas libraries are used for data manipulation while numpy is used for numerical operations. **SelectKBest** and f_classif from sklearn.feature_selection are used to implement feature

selection techniques. Output matrices such as F1 score, recall, precision, and accuracy are computed using functions from sklearn.metrics. The joblib library is used for model and scaler saving. Deep learning models were build using tensorflow library. Dimensionality reduction technique such as PCA and feature selection method like RFE are also included.

**Data loading**
Using the pandas library, the dataset stored in csv format is loaded. The functions within the library reads the data, assign column names and creates a structured dataframe.

**Data Exploration**
After loading the dataset, the structure of the data is analyzed such as number of rows and columns, and details about the dataset is derived for further processing.

**Dropping Irrelevant Features**
The process identify and eliminates unnecessary rows and columns that is no longer required for the model building process.

**Categorical Feature Encoding**
Categorical values in the dataset should be transformed before being used by the models. A separate function (*replace*) is used to convert the categorical values into numerical representations.

**Saving the Preprocessed Data**
The preprocessed data is stored in a new csv file for further machine learning modelling tasks.

### 5.2 Training of Random Forest Classifier model without any feature dimensionality reduction method

The Random forest model was trained using the NSL-KDD dataset. Initially data was loaded, then cleaned and then divided into training and testing sets (80% training and 20% testing). SMOTE is used in the dataset to resolve the class imbalance issue and hence obtained more balanced class distribution. Both the training and testing data were then subjected to standard scaling to normalize the features and guarantee equal contribution to the model training. The accuracy and precision matrix generated is used to evaluate the model performance. A confusion matrix was also generated

### 5.3 Training of Support Vector Machine Model (SVM) without any feature dimensionality reduction method

Here SMOTE is used to produce more balanced dataset. This helps the model learn effectively from minority classes. Before feeding the data into the SVM model, both the testing and training sets might undergo feature scaling using StandardScaler from sklearn.preprocessing. This ensures that all features equally contribute during model training by normalizing their scales. The core of the training involves creating an SVM model. Here, crucial hyperparameters like the kernel function (e.g., linear, radial basis function) and the cost parameter (C) would be chosen or tuned to achieve optimal performance. The chosen kernel function defines how the SVM model maps the data into higher dimensions for classification, while the cost parameter controls the trade-off between fitting the training data and allowing for some margin of error. With these hyperparameters set, the model is then trained on the pre-processed training data. Once trained, the model's performance is evaluated on the unseen testing data using metrics like accuracy, precision, and recall. Additionally, a confusion matrix can be generated using libraries like mlxtend. Plotting to visualize how well the model classified different attack types. This matrix helps identify potential weaknesses in the model, such as misclassifying certain attack types more frequently. Finally, the trained model is saved using joblib for future use or deployment. This allows for applying the trained model to new data for real-world intrusion detection tasks.

### 5.4 Training of K-Nearest Neighbour Model without any feature dimensionality reduction method

For the KNN model also data is split into testing and training sets (typically 80% training and 20% testing) using train_test_split from sklearn.model_selection ensuring the model is assessed on unseen data for generalization. For the KNN model development, the K NeighborsClassifier from sklearn.neighbors is initialized with a key hyperparameter (*random default value n_neighbors=5*) to classify new data points based on the majority vote of their five closest neighbours. The model is then trained on the pre-processed training data (x_train_sm and y_train_sm) to learn feature-attack type relationships. Evaluation is performed on unseen testing data to calculate metrics like accuracy and precision, and to generate a confusion matrix for visualizing prediction performance across attack categories.

### 5.5 Train test split the dataset and Training of CNN-GRU- BI_LSTM model without any feature dimensionality reduction methods

The dataset underwent a preprocessing phase similar to supervised models, involving splitting into testing and training sets, applying SMOTE for handling class imbalance. The core of the model architecture consists of a Convolutional Neural Network (CNN) and Gated Recurrent Unit (GRU) layers. The input data is reshaped to a suitable 3D format for CNN processing, followed by a 1D convolutional layer (Conv1D) with 64 filters and max pooling for feature extraction and down sampling. The data is flattened and passed through a dense layer with ReLU activation, reshaped for GRU processing, and then fed into a unidirectional GRU layer and bidirectional LSTM layer for sequence learning. During training, ModelCheckpoint and EarlyStopping callbacks are utilized to save the best model and prevent overfitting. The model is trained for 20 epochs with a batch size of 64 and a validation split of 20% to monitor performance. After training, the model's performance is evaluated on the testing data, and visualizations including accuracy/loss curves and a confusion matrix are generated to assess its classification performance across attack categories. This comprehensive approach ensures effective preprocessing, model development, training, and evaluation for anomaly detection within network traffic data using a CNN-GRU architecture.

### 5.6 Training of RandomForestClassifier model with Principal Component Analysis (PCA)

A PCA model is trained on the transformed training data (x_train_sm), and both training (x_train_sm_pca) and testing (x_test_pca) data are transformed accordingly to obtain lower-dimensional representations. The trained PCA model is saved using joblib for future transformations. Predictions are made on the transformed testing data (x_test_pca), and evaluation metrics including a confusion matrix to visualize performance across attack categories, accuracy score (acc1), and weighted precision score (pre1) are calculated.

### 5.7 Training of Support Vector Machine Model with PCA

A Support Vector Machine (SVM) classifier is employed for intrusion detection. The SVM model is initialized with a linear kernel and a default regularization parameter. The kernel function defines how the model transforms the data for classification, and the regularization parameter controls the model's complexity. These parameters can be further tuned to optimize performance. During training, the SVM learns a separating hyperplane that maximizes the margin between different attack classes in the transformed feature space. The trained model is then used to make predictions on the transformed testing data. The model predicts the attack class labels for each data point in the testing set. To evaluate the model's performance, a confusion matrix is generated.

### 5.8 Training of K-Nearest Neighbour Model with PCA

Similar to the above two models the KNN model is initialized with a hyperparameter n_neighbors set to 5. This parameter determines the number of nearest neighbours to consider when classifying a new data point. During training, the KNN model learns the characteristics of different attack classes based on their proximity in the transformed feature space. The trained model is then used to make predictions on the transformed testing data. The model predicts the attack class label for each data point in the testing set by considering its nearest neighbours. To evaluate the model's performance, a confusion matrix is generated. The confusion matrix visualizes how well the model classified different attack

types. Additionally, accuracy and precision scores are calculated using accuracy_score and precision_score from sklearn.metrics to provide quantitative measures of performance.

### 5.9 Train test split the dataset and Training of CNN-GRU-BI_LSTM model with Principal Component Analysis (PCA)

The core of this approach is the CNN-GRU model built using Sequential from tensorflow.keras.models. This model combines the strengths of LSTMs and CNNs to exploit both spatial and temporal features within the network traffic data. The CNN layer, with its ability to extract local patterns, operates on the transformed training data after it's reshaped into a format compatible with CNNs (typically 3D tensors). To further improve the model's ability to learn from both past and future information in the sequences, a Bidirectional LSTM layer is incorporated. Finally, a dense layer with softmax activation predicts the class probabilities for each network traffic sample, essentially classifying it as normal traffic or one of the different attack types. Accuracy is chosen as the primary metric to monitor model performance during training. To train the model effectively, callbacks are utilized. Predictions are made on the testing data, and the predicted class labels are obtained. A confusion matrix, was generated using confusion_matrix from sklearn.metrics, helps visualize the model's performance on different attack categories.

### 5.10 Training of Random Forest Classifier model with RFE

After all the initial procedure, an RFE object is created, specifying the Random Forest Classifier and the desired number of features to retain. The RFE object is then trained on the scaled training data, essentially ranking features based on their significance for classification. Subsequently, the RFE object's transform method is used to select the most important features from both the training and testing sets. Finally, the names of the chosen features are extracted based on the RFE model's selections.

### 5.11 Training of Support Vector Machine model with RFE

After the initial data loading, preprocessing, and train-test split, RFE is introduced for feature selection. As described previously, RFE iteratively fits a model (in this case, not a Random Forest but a chosen model) and eliminates features that have the lowest ranking based on a scoring function. An RFE object is created, specifying the SVM classifier and the desired number of features to retain. The RFE object is then trained on the scaled training data, essentially ranking features based on their significance for classification. Subsequently, the RFE object's transform method is used to select the most important features from both the testing and training sets. Following feature selection, an SVM classifier model is trained using the reduced training data containing only the features that are selected. The SVM model used in this case employs a specific hyperparameter configuration. The kernel function influences how the SVM transforms the data for classification, and the C parameter controls the trade-off between fitting the training data and avoiding overfitting. These are just some of the hyperparameters that can be tuned to optimize SVM performance. The model's performance is then evaluated on the reduced testing data containing only the selected features. This involves making predictions, generating a confusion matrix to visualize performance on different attack categories, and calculating accuracy and precision to assess the model's effectiveness in classifying network traffic.

### 5.12 Training of K-Nearest Neighbour model with RFE

After the initial steps, an RFE object is created, specifying the KNN classifier and the desired number of features to retain. The RFE object is then trained on the scaled training data, essentially ranking features based on their significance for classification. Subsequently, the RFE object's transform method is used to select the features that are most important from both the testing and training sets. Finally, the names of the chosen features are extracted based on the RFE model's selections. Following feature selection, a KNN classifier model is trained using the reduced training data containing only the features that are selected. The KNN model used in this case employs a specific hyperparameter configuration. Here, a crucial hyperparameter is the number of neighbours, which is set to 5 in this instance. This parameter determines how many of the closest data points in the training data are used to predict the class label for a new data point. The model's performance is then evaluated on the reduced testing data containing only the features that are selected. This involves making predictions, generating a confusion

matrix to visualize performance on different attack categories, and calculating accuracy and precision to assess the model's effectiveness in classifying network traffic.

### 5.13 Train test split the dataset and Training of CNN-GRU- BI_LSTM model with Recursive Feature Elimination (RFE)

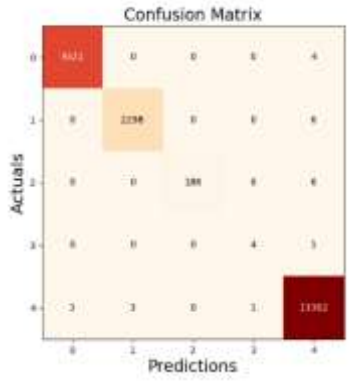
Following feature selection, a CNN-GRU-BiLSTM model is trained using the reduced training data containing only the selected features. The model architecture utilizes a sequential approach. The model architecture employs a sequential approach, comprising Convolutional Layers, Flattening, Dense Layer, Reshaping, GRU Layer, Bidirectional LSTM Layer, and Output Layer. The Conv1D layer initiates the process by extracting local features from the data using filters and a specified kernel size, followed by ReLU activation for non-linearity and max pooling for dimensionality reduction. Subsequently, the data is flattened to prepare it for fully-connected layers, where further feature extraction occurs. The data is then reshaped into a suitable format for the GRU layer, which captures sequential dependencies within the features.

# 6 Evaluation

The system built here achieved a remarkable 97% accuracy in network attack detection using a CNN-GRU-BiLSTM model with Recursive Feature Elimination (RFE). This performance significantly outshines the 66.7% accuracy achieved by a Naive Bayes classifier (Sonule, 2021), highlighting the potential of deep learning for this task. While comparable to Decision Trees (94.72% - Jaradat et al., 2022) and Artificial Neural Networks (96.88% - Taher et al., 2019), our model offers potential advantages in feature extraction due to its CNN architecture and might benefit from addressing class imbalance through techniques like SMOTE. Although a cascade-structured architecture (Labonne, 2020) reached similar accuracy on different datasets (88.39% - 94.44%), a future investigation into the computational efficiency of our CNN-GRU-BiLSTM with RFE would provide a more well-rounded comparison. In conclusion, this study establishes the CNN-GRU-BiLSTM with RFE as a highly effective solution for real-world network intrusion detection, demonstrating its superiority over simpler models and offering potential improvements over existing deep learning approaches.

### 6.1 Confusion matrix of models without any feature dimensionality reduction methods

Each cell in the confusion matrix represents the number of predictions made by the model for a combination of true and predicted classes. Fig 4 represents the confusion matrix for Random Forest Model, where most of the instances are on the diagonal of the matrix which indicates it predicts majority of instances correctly in the class. There is noticeable imbalance among the classes where class 4 has a high count (13362) compared to class 3. Fig 5 shows that KNN model generally performs well but few notable issues are with class 4. Fig 6 indicates that majority of the predictions are correct but there exists a notable challenge in distinguishing class 4. The confusion matrix for CNN-GRU-BI_LSTM in Fig 7 also shows the majority of predictions as correct which shows the accuracy of the model.



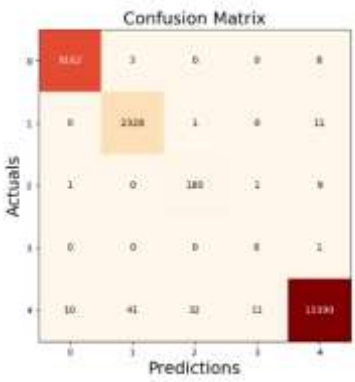
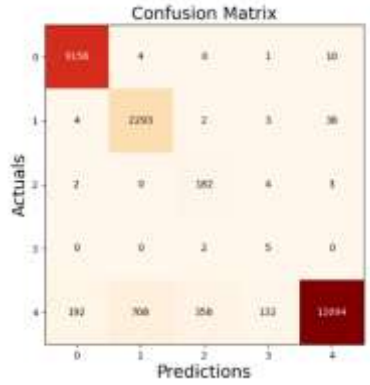



*Fig 4 : confusion matrix of Random Forest model without Feature dimensionality reduction*

*Fig 5 : confusion matrix of KNN model without any Feature dimensionality reduction*



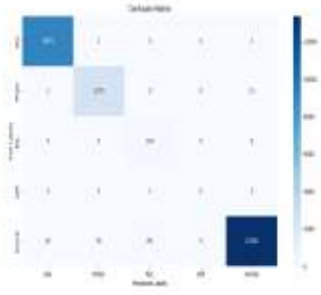



*Fig 6:confusion matrix of SVM model without Feature dimensionality reduction*

*Fig 7: confusion matrix of CNN-GRU-BI_LSTM without feature dimensionality reduction*

### 6.2 Confusion matrix of models with PCA

From the evaluation results, CNN-GRU-BI_LSTM indicates to be the most balanced model (Fig 8), generates high true positive rates especially for normal and DOS classes. The KNN model in Fig 9 shows fewer false positives and fewer true positive rates in some class as well. Fig 10 and Fig 11 indicates the matrix for random forest and SVM model which shows to be more conservative, with lowest false positive rates.

*Fig 8: Shows the confusion matrix of CNN-GRU-BI_LSTM model*

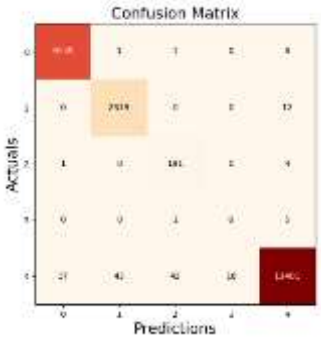



*Fig 9: Confusion matrix of KNN model*

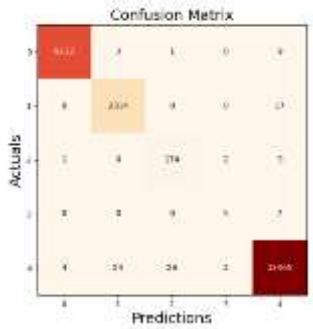



*Fig 10: confusion matrix of random forest model*

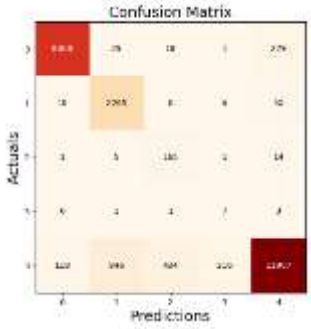



*Fig 11: confusion matrix of SVM model*

### 6.3 Confusion matrix of models with RFE

From the analysis, CNN-GRU-BI_LSTM indicates the efficient model with RFE. Fig 12 represents the confusion matrix for KNN model, where most of the instances are on the diagonal of the matrix which indicates it predicts majority of instances correctly in the class. There is noticeable imbalance among the classes where class 4 has a high count (13180). Fig 13 shows that Random forest model with fewer false positive rates. Fig 14 indicates that majority of the predictions in SVM model are correct but there

exists a notable challenge in distinguishing class 4. The confusion matrix for CNN-GRU-BI_LSTM in Fig 15 also shows the majority of predictions as correct which shows the efficiency of the model.

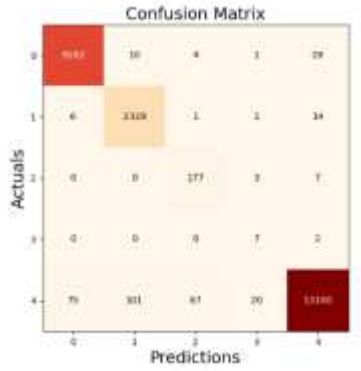



*Fig 12: confusion matrix of KNN model*

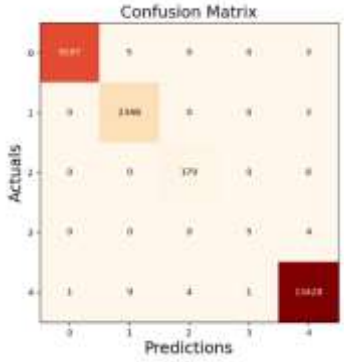



*Fig 13: confusion matrix of random forest model*

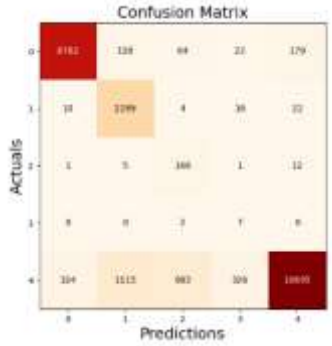



*Fig 14: confusion matrix of SVM model*

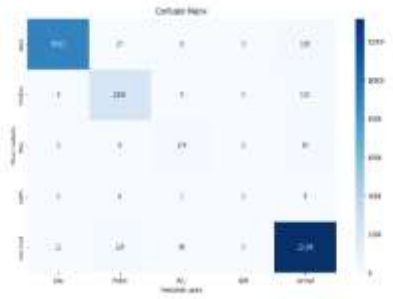


*Fig 15: confusion matrix of CNN-GRU-BI_LSTM model*

### 6.4 Analysis of Evaluation matrix of the models without any feature reduction methods, with PCA and RFE

The Isolation Forest model achieved an accuracy of 38.66% and a precision of 41.15% on anomaly detection in network traffic data without using any feature reduction methods.

The supervised learning models, including Random Forest, have an acquired accuracy of 99.9% and precision of 99.9%, K-Nearest Neighbours (KNN) acquired an accuracy of 99.5% and precision of 99.9% where Support Vector Machine (SVM) acquired an accuracy of 94.19% and precision of 99.9%. The CNN-GRU-BI_LSTM model has acquired a training accuracy of 0.97, validation loss of 0.07 and validation accuracy of 0.97 with RFE. The Isolation Forest model achieved an accuracy of 45.90% and a precision of 41.15% on anomaly detection in network traffic data with RFE. The supervised learning models, including Random Forest, have acquired an accuracy of 99.8% and precision of 99.8%, K-Nearest Neighbours (KNN) acquired an accuracy of 98.7% and precision of 99.8% where Support Vector Machine (SVM) acquired an accuracy of 87.1% and precision of 99.8%. The Isolation Forest model achieved an accuracy of 42.66% and a precision of 41.39% on anomaly detection in network traffic data with PCA. The supervised learning models, including Random Forest has acquired an accuracy of 99.6% and precision of 99.6%, K-Nearest Neighbours (KNN) acquired an accuracy of 99.4% and of precision: 99.6% where Support Vector Machine (SVM) acquired an accuracy of 92.10% and precision of 99.6%

The comparison table of evaluation matrix is given in Fig:16

| **Results without any feature reduction method** | **KNN** | **Random Forest** | **SVM** | **CNN-GRU-BI_LSTM** |
|---|---|---|---|---|
| | Precision=99.9% | Precision=99.9% | Precision=99.9% | Precision=99.9%<br>Val_loss=3.14 |
| | Accuracy=99.5% | Accuracy=99.9% | Accuracy=94.19% | Accuracy=0.99% |

| | | | | |
|---|---|---|---|---|
| **Results with PCA** | Precision=99.6% | Precision=99.6% | Precision=99.6% | Precision=99.8%<br>Val_loss=2.54 |
| | Accuracy= 99.4% | Accuracy= 99.6% | Accuracy= 92.10% | Accuracy= 0.99% |
| **Results with RFE** | Precision=99.8% | Precision=99.8% | Precision=99.8% | Precision=99.9%<br>Val-loss=0.07 |
| | Accuracy= 98.7% | Accuracy= 99.8% | Accuracy= 87.1% | Accuracy= 0.97% |

*Fig 16: Comparison table of evaluation matrix of models*

### 6.5 ROC curves showing the validation accuracy of CNN-GRU-BI_LSTM

From the proposed research work CNN-GRU-BI LSTM model shows the highest efficiency with RFE in analyzing the intrusions in the network traffic. Fig 17 shows the ROC curve of the model without any feature reduction methods which leads to a poor and unstable validation accuracy. In Fig 18, with PCA the model shows some slight improvement, still its not fully stable. Fig 19 shows the accuracy of the model with RFE where both training and validation accuracies are high suggesting successful generalization and effective feature selection.


*Fig 17: Shows the validation accuracy of CNN-GRU-BI_LSTM without Feature reduction methods*

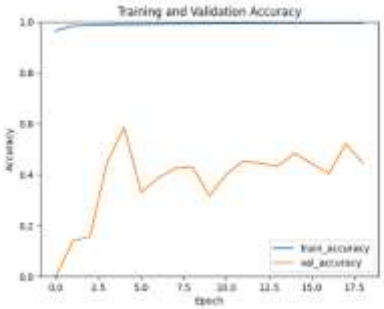



*Fig 18: Shows the validation accuracy of CNN-GRU-BI LSTM with PCA*

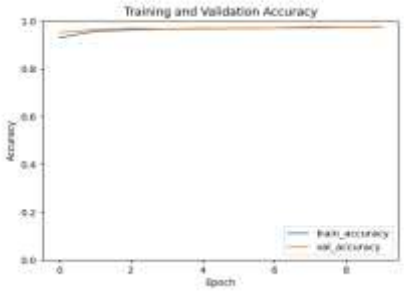



*Fig 19 : Shows the validation accuracy of CNN-GRU-BI_LSTM With RFE*

### 6.6 .The interface for prediction in the desktop application.

As part of the research study, the desktop application developed here utilizes the CNN-GRU-BiLSTM model, provides a user friendly solution for the network intrusion. The UI is developed using python libraries such as Tkinter. Here the users can input the network traffic data in various formats. The preprocessed data is fed into the model, generate the output window by predicting the intrusion type. Fig 20 shows the GUI interface.


*Fig 20: GUI interface for intrusion prediction.*

### 6.7 Discussion

This study aimed to develop a cutting-edge machine learning model for network attack detection and classification. To achieve this, a dataset rich in network features was collected to train the models. The data underwent meticulous pre-processing to eliminate irrelevant components and address class imbalance through Synthetic Minority Oversampling (SMOTE). The study has used Principal component analysis (PCA) and Recursive feature elimination (RFE), without reduction methodologies. Support Vector Machine (SVM), Random Forest, K-Nearest Neighbors, Isolation forest and Convolutional Neural Network-Gated Recurrent Unit-Long Short-Term Memory (CNN-GRU-LSTM) to train in three different scenarios : without any feature reduction method, with features reduced by RFE and PCA. Performance matrix were used to analyse the performance of the models. The combination of CNN-GRU-LSTM with RFE derived essential features from the network traffic data and proves to be the most accurate anomaly detection.

# 7 Conclusion And Future Work

Looking ahead there are more areas to be researched to enhance the study. Evaluation matrix can be expanded to include the outputs of precision, recall and F1 score to provide a more understanding on models performance. The computational efficiency of CNN-GRU-LSTM with RFE can be optimized to make them utilized in the real world deployments. By leveraging through this approaches, machine learning can provide a crucial role in safeguarding the network from cyber attacks. This study provides valuable insights for researchers and developers working towards the development of next-generation network intrusion detection systems.

Deep learning is used for the detection of intrusions in software defined networks(SDN) in the study by (Kurochkin and Volkov, 2020). The GRU is utilised in the study and the model based on the GRU detects and classifies intrusion into seven categories. The CSE-CIC-IDS2018 dataset is used in the study. The GRU contains a self-attention layer. The results of the study show that the GRU modelachieves a precision of 0.94 and a recall of 0.99 in the detection of benign networks. The GRU model achieves both precision and recall as 1 in classifying the DDOS attack class. However, only a single deep learning model is utilised in the study.
A GRU-based lightweight neural network is proposed for the detection of vehicular intrusions in Controller Area Network (CAN) bus in the study by (Ma et al., 2022). The dataset containing data associated with networks containing cyber attacks and normal networks is used in the study. A feature extraction algorithm is utilised in the study. The results of the study show that the GRU model shows a good performance in detecting intrusions. However, the utilised of the GPU in the study for the detection of intrusions in the study is a limitation of the study.

# References


Ahmetoglu, H. and Das, R. (2022). A comprehensive review on detection of cyber-attacks: Data sets, methods, challenges, and future research directions. Internet of Things, p.100615. doi:https://doi.org/10.1016/j.iot.2022.100615.

Alkasassbeh,M.,Almseidin, M.(2018).Machine Learning Methods for Network Intrusion Detection.ICCCNT 2018. [online] available at: https://arxiv.org/ftp/arxiv/papers/1809/1809.02610.pdf

Alshammari, A. and Aldribi, A. (2021). Apply machine learning techniques to detect malicious network traffic in cloud computing. Journal of Big Data, 8(1). doi:https://doi.org/10.1186/s40537-021-00475-1.

Awadallah Awad, N. (2021). Enhancing Network Intrusion Detection Model Using Machine Learning Algorithms. Computers, Materials & Continua, 67(1), pp.979–990. doi:https://doi.org/10.32604/cmc.2021.014307.

D, K. (2023). Optimizing Performance: SelectKBest for Efficient Feature Selection in Machine Learning. [online] Medium. Available at: https://medium.com/@Kavya2099/optimizing-performance-selectkbest-for-efficient-feature-selection-in-machine-learning-3b635905ed48.

Data Science | Machine Learning | Python | C++ | Coding | Programming | JavaScript. (2020). StandardScaler in Machine Learning. [online] Available at: https://thecleverprogrammer.com/2020/09/22/standardscaler-in-machine-learning/.

Dey, R. (2023). Understanding Principal Component Analysis (PCA). [online] Medium. Available at: https://medium.com/@roshmitadey/understanding-principal-component-analysis-pca-d4bb40e12d33.

Dhanya, K.A., Vajipayajula, S., Srinivasan, K., Tibrewal, A., Kumar, T.S. and Kumar, T.G. (2023). Detection of Network Attacks using Machine Learning and Deep Learning Models. Procedia Computer Science, 218, pp.57–66. doi:https://doi.org/10.1016/j.procs.2022.12.401.

E.,G. (2024). Recursive Feature Elimination: A Powerful Technique for Feature Selection in Machine Learning. [online] The Modern Scientist. Available at: https://medium.com/the-modern-scientist/recursive-feature-elimination-a-powerful-technique-for-feature-selection-in-machine-learning-89b3c2f3c26a [Accessed 22 Apr. 2024].

Einy, S., Oz, C. and Navaei, Y.D. (2021). The Anomaly- and Signature-Based IDS for Network Security Using Hybrid Inference Systems. Mathematical Problems in Engineering, 2021, pp.1–10. doi:https://doi.org/10.1155/2021/6639714.

FasterCapital. (2024). Advantages Of Random Forests. [online] Available at: https://fastercapital.com/topics/advantages-of-random-forests.html.

Jain,S.(2022). 160 Cybersecurity Statistics: Updated Report 2023. [online] Astra. Available at: https://www.getastra.com/blog/security-audit/cyber-security-statistics/#:~:text=Cybersecurity%20statistics%20indicate%20that%20there.

Jamadar, R.A. (2018). Network Intrusion Detection System Using Machine Learning. Indian Journal of Science and Technology, 11(48), pp.1–6. doi:https://doi.org/10.17485/ijst/2018/v11i48/139802.

Jaradat,A,S.,Barhoush,M,M.,Easa,R,B.(2022). Indonesian Journal of Electrical Engineering and Computer Science, 25(2). doi:https://doi.org/10.11591/ijeecs.v25.i2.

Jia, W., Sun, M., Lian, J. and Hou, S. (2022). Feature dimensionality reduction: a review. Complex & Intelligent Systems. doi:https://doi.org/10.1007/s40747-021-00637-x.

Kaggle. (2024). NSL-KDD. [online] Available at: https://www.kaggle.com/datasets/hassan06/nslkdd.

Kala,E,M. (2023). The Impact of Cyber Security on Business: How to Protect Your Business. Open Journal of Safety Science and Technology, 13(02), pp.51–65. doi:https://doi.org/10.4236/ojsst.2023.132003.

Kanade, V. (2022). What Is a Support Vector Machine? Working, Types, and Examples. [online] Spiceworks. Available at: https://www.spiceworks.com/tech/big-data/articles/what-is-support-vector-machine/.

Krishnan, S., Neyaz, A. and Liu, Q. (2021). IoT Network Attack Detection using Supervised Machine Learning. Ashar Neyaz & Qingzhong Liu International Journal of Artificial Intelligence and Expert Systems (IJAE), [online] (10), pp.2021–2039. Available at: https://shsu-ir.tdl.org/server/api/core/bitstreams/19328a7a-eaa5-4831-8166-9adb00032794/content [Accessed 30 Mar. 2024].

Kurochkin, I.I. and Volkov, S.S. (2020). Using GRU based deep neural network for intrusion detection in software-defined networks. IOP Conference Series: Materials Science and Engineering, 927, p.012035. doi:https://doi.org/10.1088/1757-899x/927/1/012035.

Learnenough.com. (2024). Learn Enough to Be Dangerous. [online] Available at: https://www.learnenough.com/blog/how-to-import-Pandas-in-python.

Li, Y. and Liu, Q. (2021). A comprehensive review study of cyber-attacks and cyber security; emerging trends and recent developments. Energy Reports, [online] 7(7), pp.8176–8186. doi:https://doi.org/10.1016/j.egyr.2021.08.126.

Ma, H., Cao, J., Mi, B., Huang, D., Liu, Y. and Li, S. (2022). A GRU-Based Lightweight System for CAN Intrusion Detection in Real Time. Security and Communication Networks, 2022, pp.1–11. doi:https://doi.org/10.1155/2022/5827056.

Mduma, N. (2023). Data Balancing Techniques for Predicting Student Dropout Using Machine Learning. Data, 8(3), p.49. doi:https://doi.org/10.3390/data8030049.

MLNerds (2019). How does KNN algorithm work ? What are the advantages and disadvantages of KNN ? [online] Ace the Data Science Interview! Available at: https://machinelearninginterview.com/topics/machine-learning/how-does-knn-algorithm-work-what-are-the-advantages-and-disadvantages-of-knn/.