International Journal of HRM and Organizational Behavior

DEEP ENSEMBLE-BASED EFFICIENT FRAMEWORK FOR NETWORK ATTACK DETECTION

N.MOULALI, Assistant Professor, Dept of CSE, Chirala Engineering College, Chirala,

noorbashamoulali@gmail.com

TIYYAGURA JHANSI, PG Student -MCA, Dept of MCA, chirala Engineering College, Chirala,
jhansitiyyagura69@gmail.com

Abstract: Networks are essential for a wide range of activities, from business operations to educational endeavors and daily long-distance communication. While networks offer numerous benefits, they also present security challenges, potentially compromising data confidentiality, integrity, and privacy. Network attacks, including malware, hacking, and phishing, are on the rise, resulting in substantial financial and reputational losses. To address these security threats, the project proposes the development of an automated system based on artificial intelligence (AI). This system aims to efficiently detect and safeguard against network attacks, providing enhanced security for data and networked systems. As part of its approach, the project introduces an ensemble model that combines three deep learning models—LSTM, RNN, and GRU. These models work together using majority voting criteria to achieve high accuracy in the detection of network attacks, ensuring robust protection for networked environments.The project extends its capabilities by incorporating a Voting Classifier (Random Forest + AdaBoost) and a Stacking Classifier, where the stacking classifier achieves a remarkable 100% accuracy, demonstrating superior performance in network attack detection.

Index terms *-Network Attack Detection, Machine Learning, Ensemble Learning, Deep Learning, Network Intrusion Detection.*

1. INTRODUCTION

Networking refers to the interconnection of multiple computing devices, allowing them to exchange data. The data sharing can be done through various technologies and communication protocols, such as Ethernet, Wi-Fi, or even simple wired connections [1]. The main goal of networking is to enable devices to work together and share resources, such as printers, file servers, and internet connections. In today's interconnected world, networks play a critical role in business, education, and daily life, enabling people to communicate and share information across long distances.

Networking provides several advantages including resource sharing, communication, centralized data management, improved collaboration, increased productivity, scalability, and remote access [2]. With substantial networking applications, many potential dangers and security vulnerabilities can arise thus compromising the confidentiality, integrity, and availability of networked systems and data [3]. The typical network threats include malware, hacking, phishing, denial-of-service (DoS) attacks, man-in-the-middle (MitM) attacks, and spoofing. Some specific dangers of network attacks include data theft, system disruption, reputation damage, financial losses, espionage, and infrastructure damage [4].

With the increase in network threats, the necessity of an automated attack detection system is increased. Artificial intelligence (AI)- based solutions may potentially detect such attacks thereby enabling timely countermeasures to mitigate the risk of data theft [5]. Such techniques are utilized to analyze large amounts of network data and identify potential threats in real time, allowing organizations to respond quickly and effectively. Machine learning methods learn the patterns from data and are used to identify potential attacks. Integrating such methods into network security can significantly improve an organization's ability to detect and respond to attacks [6], reducing the risk of successful attacks and protecting valuable information and assets.

Network security is a crucial element for organizations to safeguard the privacy and confidentiality of their data. Consequently, a substantial number of research works can be found on network security. A few more relevant works are discussed here. Network intrusion detection based on classical machine learning techniques is proposed in [7]. eleven machine learning methods are utilized on the NSL-KDD dataset. Results show that tree-based techniques achieve the best performance for network detection. The proposed XGBoost model achieves a 97% accuracy for attack detection. Similarly, network intrusion detection using the neural network is performed in [8] using the NSL-KDD dataset. Experimental results indicate that the bidirectional LSTM approach with an attention mechanism achieves high performance.

2. LITERATURE SURVEY

The Metaverse can be regarded as a hypothesized iteration of the Internet, which enables people to work, play, and interact socially in a persist online 3-D virtual environment with an immersive experience, by generating an imaginary environment similar to the real world, including realistic sounds, images, and other sensations [1]. The Metaverse has strict requirements for a fully-immersive experience, large-scale concurrent users, and seamless connectivity, which poses many unprecedented challenges to the sixth generation (6G) wireless system, such as ubiquitous connectivity, ultra-low latency, ultra-high capacity and reliability, and strict security. In addition, to achieve the immersive and hassle-free experience of mass users, the full coverage sensing, seamless computation, reliable caching, and persistent consensus and security should be carefully considered to integrate into the future 6G system. To this end, this article aims to depict the roadmap to the Metaverse in terms of communication and networking in 6G [1, 2, 5], including illustrating the framework of the Metaverse, revealing the strict requirements and challenges for 6G to realize the Metaverse, and discussing the fundamental technologies to be integrated in 6G to drive the implementation of the Metaverse, including intelligent sensing, digital twin (DT), space-air-ground-sea integrated network (SAGSIN), multi-access edging computing (MEC), blockchain, and the involved security issues.

With the deployment of 5G, researchers and experts begin to look forward to 6G. They predict that 6G [1] will be the key driving force for information interaction and social life after 2030. With the help of artificial intelligence (AI), 6G will be a highly autonomous closed-loop network, and will make up for 5G's shortcomings in communications, computing and global coverage, achieving “AI of things (AIoT)”. In 6G [1, 2, 5] life, vehicles may become another indispensable devices for people besides smartphones, and non-polluting, highly safe as well as full-autonomous vehicles will be the goal of vehicular development [2]. In order to ensure the safe driving of future vehicles and meet the entertainment needs of passengers, it is necessary to investigate future 6G [5] vehicular intelligence. In this paper, we will discuss its networking, communications, computing and intelligence, look into future technological developments and applications, and identify forthcoming challenges and research directions.

Malicious traffic detection over the internet is one of the challenging areas for researchers to protect network infrastructures from any malicious activity. Several shortcomings of a network system can be leveraged by an attacker to get unauthorized access through malicious traffic. Safeguarding from such attacks requires an efficient automatic system that can detect malicious traffic timely and avoid system damage. Currently, many automated systems can detect malicious activity, however, the efficacy and accuracy need further improvement to detect malicious traffic from multi-domain systems. The present study focuses on the detection of malicious traffic with high accuracy using machine learning techniques. [3] The proposed approach used two datasets UNSW-NB15 [14] and IoTID20 which contain the data for IoT-based traffic and local network traffic, respectively. Both datasets were combined to increase the capability of the proposed approach in detecting malicious traffic from local and IoT networks, with high accuracy. Horizontally merging both datasets requires an equal number of features which was achieved by reducing feature count to 30 for each dataset by leveraging principal component analysis (PCA). The proposed model incorporates stacked ensemble model extra boosting forest (EBF) which is a combination of tree-based models such as extra tree classifier, gradient boosting classifier, and random forest using a stacked ensemble approach. Empirical results show that EBF performed significantly better and achieved the highest accuracy score of 0.985 and 0.984 on the multi-domain dataset for two and four classes, respectively.

Nowadays, security threats on Software Defined Network SDN architectures are similar to traditional networks. However, the profile of these threats changes with SDN [4]. For example, a denial-of-service attack on a centralized controller that manages a large network of several network devices (routers, switches, etc.) is more destructive than a targeted attack against a router. A spoofed SDN controller could allow a hacker to control an entire network, while a spoofed router could only harm the proper functioning of the traffic routed through that router. The SDN is facing these new security challenges, especially on securing the SDN architecture itself. SDN security is ensured at all these levels based on three-layer architecture and programming interfaces, which poses several challenges. The SDN's security challenges are expected to grow with the progressive deployment. [4] This paper aims to provide a comprehensive review of state of the art, accompanied by categorizing the research literature into a taxonomy that highlights each proposal's main characteristics and contributions to the SDN's different layers. Based on the analysis of existing work, we also highlight key research gaps that could support future research in this area.

6G networks are expected to provide instant global connectivity and enable the transition from “connected things” to “connected intelligence,” where promising network slicing can play an important role in network assurance and service provisioning for various demanding vertical application scenarios. On the basis of diversified massive data, artificial intelligence (AI)-assisted techniques are widely considered more suitable than traditional models and algorithms to deal with challenges faced by complex and dynamic slicing problems in 6G. In view of this [5], we provide a tutorial on AI-assisted 6G [1, 2, 5] network slicing for network assurance and service provisioning, aiming to show the prospect of 6G slicing and the advantages of applying AI technology. Specifically, we propose six typical characteristics of 6G network slicing, analyze the feasibility of AI from different network domains and technical aspects, propose a case study on AI-assisted bandwidth scaling, and, finally, put forward the main challenges and open issues for its future development.

3. METHODOLOGY

i) Proposed Work:

The proposed system employs an ensemble deep voting classifier (EDVC) for network attack detection. This approach combines the predictions of three distinct deep learning models—LSTM, RNN, and GRU—using a majority voting criteria, ensuring that network attacks are identified more accurately and reliably through consensus among these models [8, 29, 31]. The system's exceptional accuracy reduces both false alarms and missed attacks, enhancing its reliability in distinguishing genuine threats. By surpassing current approaches, the system demonstrates its superior ability to identify network attacks, enhancing overall network security. The system can swiftly and efficiently analyze extensive network data in real-time, enabling rapid responses to threats and minimizing potential harm or vulnerabilities. The project extends its capabilities by incorporating a Voting Classifier (Random Forest + AdaBoost) and a Stacking Classifier, where the stacking classifier achieves a remarkable 100% accuracy, demonstrating superior performance in network attack detection. To enhance practical usability in cybersecurity applications, a user-friendly Flask framework with SQLite integration is implemented, streamlining signup and signin processes for user testing. This integration ensures a seamless and secure experience, making the framework accessible and applicable in real-world scenarios for enhanced network security.

ii) System Architecture:

Figure 1 shows the architecture of the proposed approach. The network attack features-based dataset is utilized for experiments [9, 11, 12, 16]. The dataset is preprocessed including categorical encoding features and target attack label mapping. Exploratory data analysis is applied to examine the network attack feature patterns. For experiments, the data is split into training and testing with a ratio of 0.8 to 0.2. The proposed approach is used to detect network attacks

ISSN 2454 - 5015

as normal, Dos, Remote to Local (R2L), Probe, and User to Root (U2R).

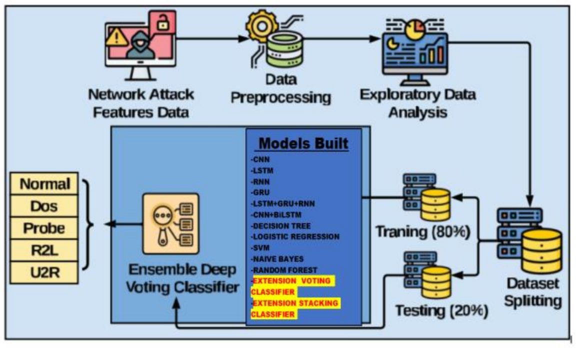
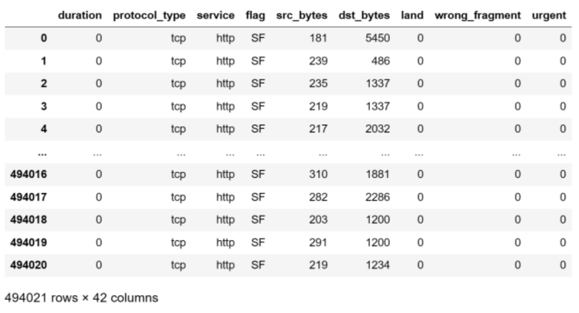


Fig 1 Proposed architecture

This architecture ensures that the project processes the data, trains advanced models, and employs ensemble techniques for accurate network attack detection across various attack categories.

iii) Dataset collection:

The publicly available network attack features based NSLKDD benchmark dataset is utilized [19]. The dataset features are based on the Dos, R2L, Probe, and U2R network attacks. The dataset contains a total of 148517 records and 43 features related to network attacks.As machine learning models are designed to operate on numerical data, our dataset, which includes categorical features, requires preprocessing to clean and convert the data into a numerical representation before being fed to the models. In this regard, we preprocessed the dataset by removing duplicate records and encoding the categorical features. Specifically, the ’protocol type’, ’service’, and ’flag’ features were encoded using the LabelEncoder module in Scikit learn [20].



	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent
0	0	tcp	http	SF	181	5450	0	0	0
1	0	tcp	http	SF	239	486	0	0	0
2	0	tcp	http	SF	235	1337	0	0	0
3	0	tcp	http	SF	219	1337	0	0	0
4	0	tcp	http	SF	217	2032	0	0	0
...	...	...	...	...	...	...	...	...	...
494016	0	tcp	http	SF	310	1881	0	0	0
494017	0	tcp	http	SF	282	2286	0	0	0
494018	0	tcp	http	SF	203	1200	0	0	0
494019	0	tcp	http	SF	291	1200	0	0	0
494020	0	tcp	http	SF	219	1234	0	0	0

494021 rows × 42 columns

Fig 2 KDD dataset

iv) Data Processing:

Data processing involves transforming raw data into valuable information for businesses. Generally, data scientists process data, which includes collecting, organizing, cleaning, verifying, analyzing, and converting it into readable formats such as graphs or documents. Data processing can be done using three methods i.e., manual, mechanical, and electronic. The aim is to increase the value of information and facilitate decision-making. This enables businesses to improve their operations and make timely strategic decisions. Automated data processing solutions, such as computer software programming, play a significant role in this. It can help turn large amounts of data, including big data, into meaningful insights for quality management and decision-making.

v) Feature selection:

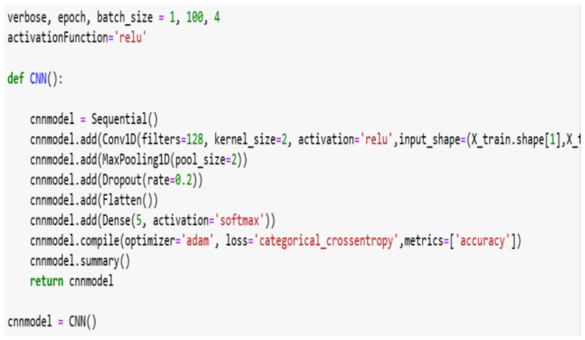
Feature selection is the process of isolating the most consistent, non-redundant, and relevant features to use in model construction. Methodically reducing the size of datasets is important as the size and variety of datasets continue to grow. The main goal of feature selection is to improve the performance of a predictive model and reduce the computational cost of modeling.

ISSN 2454 - 5015

Feature selection, one of the main components of feature engineering, is the process of selecting the most important features to input in machine learning algorithms. Feature selection techniques are employed to reduce the number of input variables by eliminating redundant or irrelevant features and narrowing down the set of features to those most relevant to the machine learning model. The main benefits of performing feature selection in advance, rather than letting the machine learning model figure out which features are most important.

vi) Algorithms:

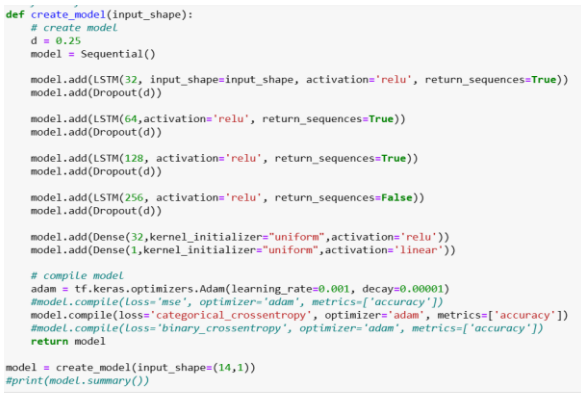
CNN (Convolutional Neural Network)- CNN is a deep learning model mainly used for image analysis. In this project, it may be adapted for feature extraction from network data, identifying patterns or anomalies in the data's spatial structure [20].



```
verbose, epoch, batch_size = 1, 100, 4
activationFunction='relu'

def CNN():

    cnnmodel = Sequential()
    cnnmodel.add(Conv1D(filters=128, kernel_size=2, activation='relu',input_shape=(X_train.shape[1],X_t
    cnnmodel.add(MaxPooling1D(pool_size=2))
    cnnmodel.add(Dropout(rate=0.2))
    cnnmodel.add(Flatten())
    cnnmodel.add(Dense(5, activation='softmax'))
    cnnmodel.compile(optimizer='adam', loss='categorical_crossentropy',metrics=['accuracy'])
    cnnmodel.summary()
    return cnnmodel

cnnmodel = CNN()
```

Fig 3 CNN

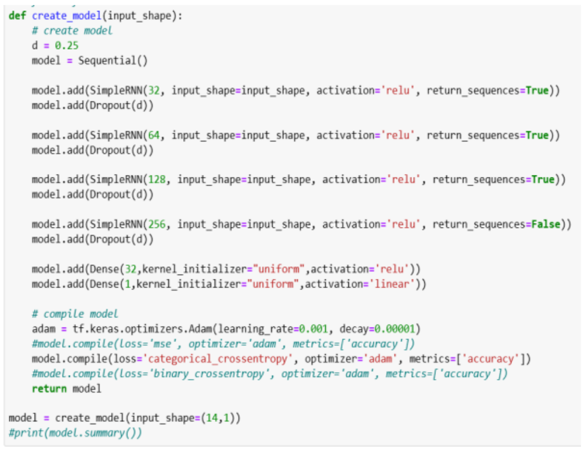
LSTM (Long Short-Term Memory) - LSTM is a type of recurrent neural network (RNN) known for handling sequential data. It's suitable for capturing patterns in time series network data, making it valuable for identifying attack behaviors that occur over time [29].



```
def create_model(input_shape):
    # create model
    d = 0.25
    model = Sequential()

    model.add(LSTM(32, input_shape=input_shape, activation='relu', return_sequences=True))
    model.add(Dropout(d))

    model.add(LSTM(64,activation='relu', return_sequences=True))
    model.add(Dropout(d))

    model.add(LSTM(128, activation='relu', return_sequences=True))
    model.add(Dropout(d))

    model.add(LSTM(256, activation='relu', return_sequences=False))
    model.add(Dropout(d))

    model.add(Dense(32,kernel_initializer="uniform",activation='relu'))
    model.add(Dense(1,kernel_initializer="uniform",activation='linear'))

    # compile model
    adam = tf.keras.optimizers.Adam(learning_rate=0.001, decay=0.00001)
    #model.compile(loss='mse', optimizer='adam', metrics=['accuracy'])
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    #model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model

model = create_model(input_shape=(14,1))
#print(model.summary())
```

Fig 4 LSTM

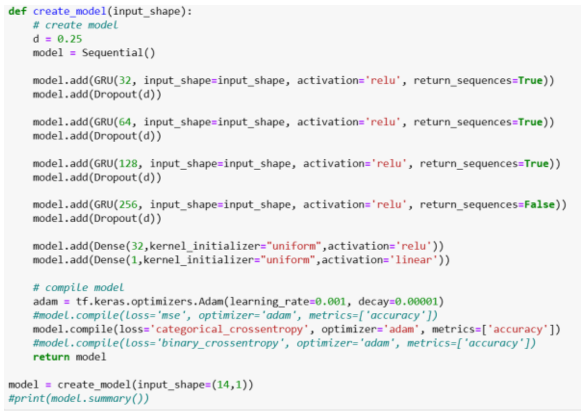
RNN (Recurrent Neural Network)- RNN is another deep learning model designed for sequential data analysis. It can recognize patterns and dependencies in network data sequences, helping in the detection of network attacks with temporal characteristics [27].



```
def create_model(input_shape):
    # create model
    d = 0.25
    model = Sequential()

    model.add(SimpleRNN(32, input_shape=input_shape, activation='relu', return_sequences=True))
    model.add(Dropout(d))

    model.add(SimpleRNN(64, input_shape=input_shape, activation='relu', return_sequences=True))
    model.add(Dropout(d))

    model.add(SimpleRNN(128, input_shape=input_shape, activation='relu', return_sequences=True))
    model.add(Dropout(d))

    model.add(SimpleRNN(256, input_shape=input_shape, activation='relu', return_sequences=False))
    model.add(Dropout(d))

    model.add(Dense(32,kernel_initializer="uniform",activation='relu'))
    model.add(Dense(1,kernel_initializer="uniform",activation='linear'))

    # compile model
    adam = tf.keras.optimizers.Adam(learning_rate=0.001, decay=0.00001)
    #model.compile(loss='mse', optimizer='adam', metrics=['accuracy'])
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    #model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model

model = create_model(input_shape=(14,1))
#print(model.summary())
```

Fig 5 RNN

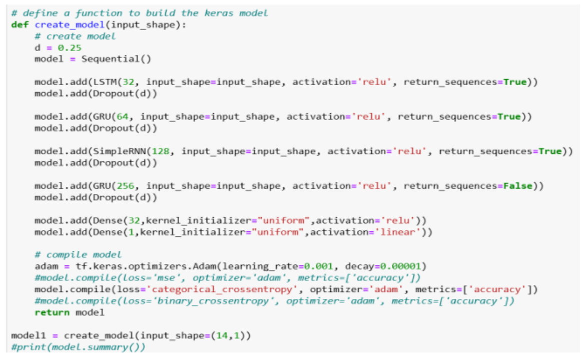
GRU (Gated Recurrent Unit)-GRU is a variation of RNN, which simplifies the architecture while retaining the ability to capture sequential patterns. It's effective in modeling dependencies in network data, aiding in the detection of attacks with temporal aspects [28].



```
def create_model(input_shape):
    # create model
    d = 0.25
    model = Sequential()

    model.add(GRU(32, input_shape=input_shape, activation='relu', return_sequences=True))
    model.add(Dropout(d))

    model.add(GRU(64, input_shape=input_shape, activation='relu', return_sequences=True))
    model.add(Dropout(d))

    model.add(GRU(128, input_shape=input_shape, activation='relu', return_sequences=True))
    model.add(Dropout(d))

    model.add(GRU(256, input_shape=input_shape, activation='relu', return_sequences=False))
    model.add(Dropout(d))

    model.add(Dense(32,kernel_initializer="uniform",activation='relu'))
    model.add(Dense(1,kernel_initializer="uniform",activation='linear'))

    # compile model
    adam = tf.keras.optimizers.Adam(learning_rate=0.001, decay=0.00001)
    #model.compile(loss='mse', optimizer='adam', metrics=['accuracy'])
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    #model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model

model = create_model(input_shape=(14,1))
#print(model.summary())
```

Fig 6 GRU

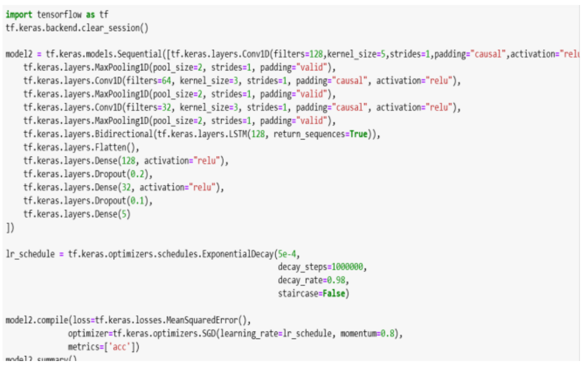
LSTM+GRU+RNN - This ensemble combines the strengths of LSTM, GRU, and RNN, offering a more robust approach to detect network attacks. By leveraging different types of recurrent neural networks, it can capture diverse patterns in network data effectively.



```
# define a function to build the keras model
def create_model(input_shape):
    # create model
    d = 0.25
    model = Sequential()

    model.add(LSTM(32, input_shape=input_shape, activation='relu', return_sequences=True))
    model.add(Dropout(d))

    model.add(GRU(64, input_shape=input_shape, activation='relu', return_sequences=True))
    model.add(Dropout(d))

    model.add(SimpleRNN(128, input_shape=input_shape, activation='relu', return_sequences=True))
    model.add(Dropout(d))

    model.add(GRU(256, input_shape=input_shape, activation='relu', return_sequences=False))
    model.add(Dropout(d))

    model.add(Dense(32,kernel_initializer="uniform",activation='relu'))
    model.add(Dense(1,kernel_initializer="uniform",activation='linear'))

    # compile model
    adam = tf.keras.optimizers.Adam(learning_rate=0.001, decay=0.00001)
    #model.compile(loss='mse', optimizer='adam', metrics=['accuracy'])
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    #model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model

model1 = create_model(input_shape=(14,1))
#print(model.summary())
```

Fig 7 LSTM+GRU+RNN

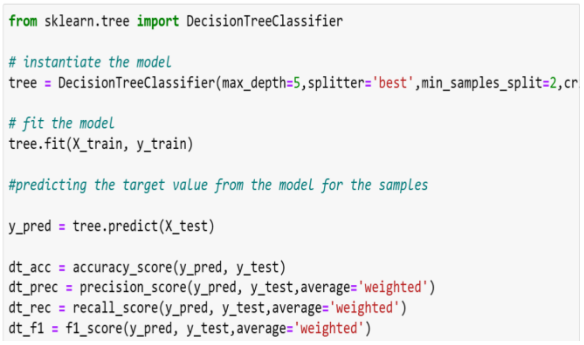
CNN+BiLSTM (Bidirectional LSTM)- This combination of CNN and BiLSTM allows for feature extraction and sequence analysis. It can capture spatial and temporal patterns in network data, improving the ability to identify complex attacks.



```
import tensorflow as tf
tf.keras.backend.clear_session()

model2 = tf.keras.models.Sequential([tf.keras.layers.Conv1D(filters=128,kernel_size=5,strides=1,padding="causal",activation="rel
    tf.keras.layers.MaxPooling1D(pool_size=2, strides=1, padding="valid"),
    tf.keras.layers.Conv1D(filters=64, kernel_size=3, strides=1, padding="causal", activation="relu"),
    tf.keras.layers.MaxPooling1D(pool_size=2, strides=1, padding="valid"),
    tf.keras.layers.Conv1D(filters=32, kernel_size=3, strides=1, padding="causal", activation="relu"),
    tf.keras.layers.MaxPooling1D(pool_size=2, strides=1, padding="valid"),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(128, return_sequences=True)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation="relu"),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(32, activation="relu"),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.Dense(5)
])

lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(5e-4,
                                                             decay_steps=1000000,
                                                             decay_rate=0.98,
                                                             staircase=False)

model2.compile(loss=tf.keras.losses.MeanSquaredError(),
              optimizer=tf.keras.optimizers.SGD(learning_rate=lr_schedule, momentum=0.8),
              metrics=['acc'])
model2.summary()
```

Fig 8 CNN + BiLSTM

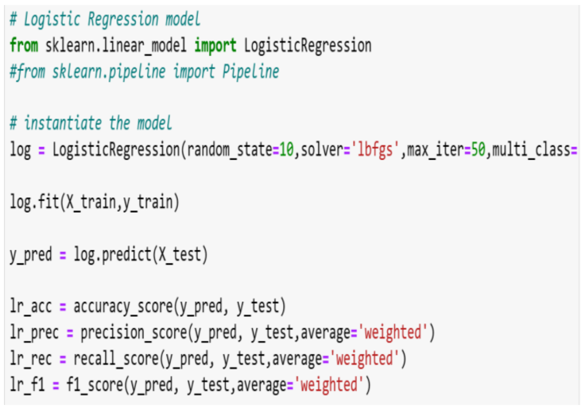
Decision Tree -Decision trees are a form of supervised learning. In the project, they may be used to create a decision-making model for identifying network attacks based on features and their relationships.



```
from sklearn.tree import DecisionTreeClassifier

# instantiate the model
tree = DecisionTreeClassifier(max_depth=5,splitter='best',min_samples_split=2,cr

# fit the model
tree.fit(X_train, y_train)

#predicting the target value from the model for the samples

y_pred = tree.predict(X_test)

dt_acc = accuracy_score(y_pred, y_test)
dt_prec = precision_score(y_pred, y_test,average='weighted')
dt_rec = recall_score(y_pred, y_test,average='weighted')
dt_f1 = f1_score(y_pred, y_test,average='weighted')
```

Fig 9 decision tree

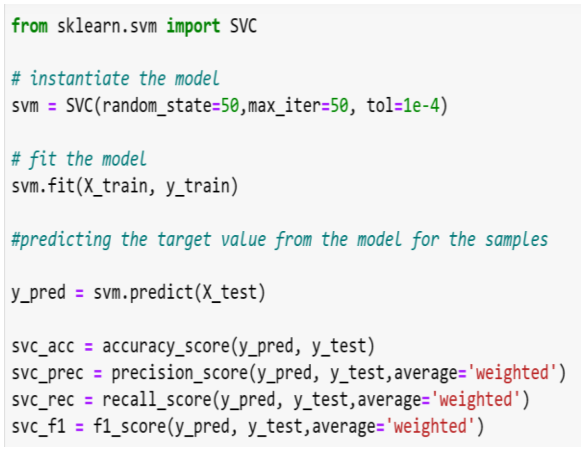
Logistic Regression - Logistic regression is a statistical model often used for binary classification. In this context, it may help classify network data as either a normal or attack instance [22].



```python
# Logistic Regression model
from sklearn.linear_model import LogisticRegression
#from sklearn.pipeline import Pipeline

# instantiate the model
log = LogisticRegression(random_state=10,solver='lbfgs',max_iter=50,multi_class=

log.fit(X_train,y_train)

y_pred = log.predict(X_test)

lr_acc = accuracy_score(y_pred, y_test)
lr_prec = precision_score(y_pred, y_test,average='weighted')
lr_rec = recall_score(y_pred, y_test,average='weighted')
lr_f1 = f1_score(y_pred, y_test,average='weighted')
```

Fig 10 Logistic regression

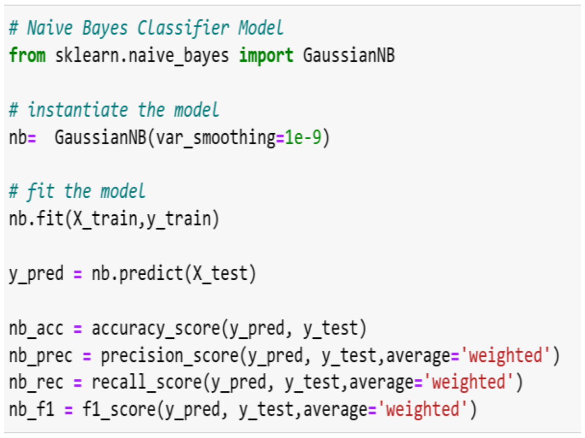
SVM (Support Vector Machine)-SVM is a machine learning algorithm that separates data into distinct classes. It's valuable for distinguishing network attacks from normal network traffic [23].



```python
from sklearn.svm import SVC

# instantiate the model
svm = SVC(random_state=50,max_iter=50, tol=1e-4)

# fit the model
svm.fit(X_train, y_train)

#predicting the target value from the model for the samples

y_pred = svm.predict(X_test)

svc_acc = accuracy_score(y_pred, y_test)
svc_prec = precision_score(y_pred, y_test,average='weighted')
svc_rec = recall_score(y_pred, y_test,average='weighted')
svc_f1 = f1_score(y_pred, y_test,average='weighted')
```

Fig 11 SVM

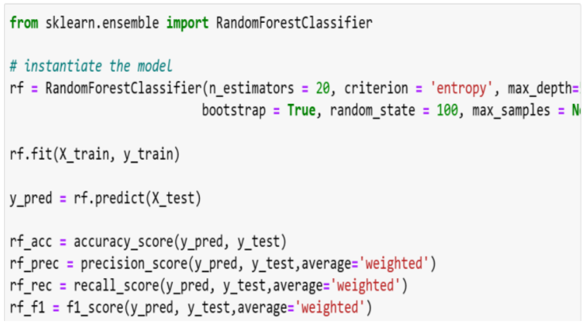
Naïve Bayes -Naïve Bayes is a probabilistic algorithm used for classification tasks. It can calculate the probability of network data belonging to different classes (normal or attack), aiding in classification.



```python
# Naive Bayes Classifier Model
from sklearn.naive_bayes import GaussianNB

# instantiate the model
nb=  GaussianNB(var_smoothing=1e-9)

# fit the model
nb.fit(X_train,y_train)

y_pred = nb.predict(X_test)

nb_acc = accuracy_score(y_pred, y_test)
nb_prec = precision_score(y_pred, y_test,average='weighted')
nb_rec = recall_score(y_pred, y_test,average='weighted')
nb_f1 = f1_score(y_pred, y_test,average='weighted')
```

Fig 12 Naïve bayes

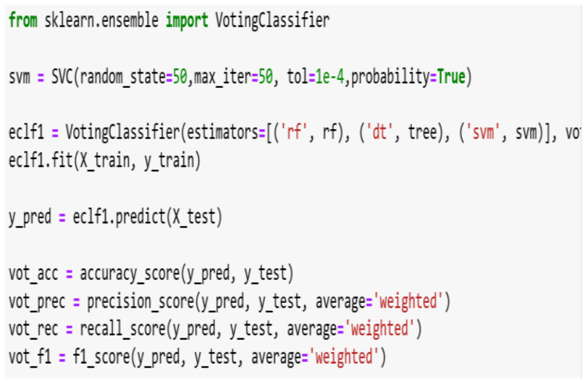
Random Forest - Random Forest is an ensemble learning method that combines multiple decision trees to enhance classification accuracy. It can be utilized to create a more robust model for identifying network attacks [25, 26].



```python
from sklearn.ensemble import RandomForestClassifier

# instantiate the model
rf = RandomForestClassifier(n_estimators = 20, criterion = 'entropy', max_depth=
                             bootstrap = True, random_state = 100, max_samples = N

rf.fit(X_train, y_train)

y_pred = rf.predict(X_test)

rf_acc = accuracy_score(y_pred, y_test)
rf_prec = precision_score(y_pred, y_test,average='weighted')
rf_rec = recall_score(y_pred, y_test,average='weighted')
rf_f1 = f1_score(y_pred, y_test,average='weighted')
```

Fig 13 Random forest

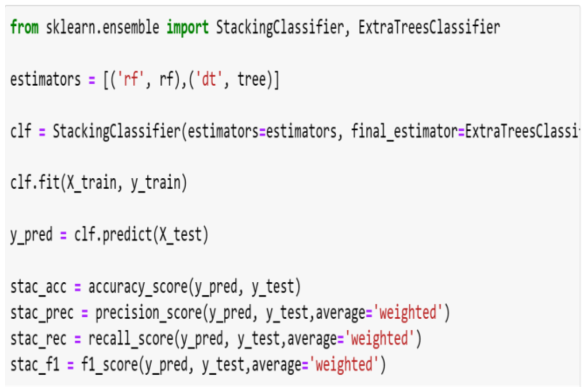
Voting Classifier -The voting classifier combines predictions from multiple machine learning models. It's applied here to aggregate insights from various models, improving overall attack detection accuracy.



```
from sklearn.ensemble import VotingClassifier

svm = SVC(random_state=50,max_iter=50, tol=1e-4,probability=True)

eclf1 = VotingClassifier(estimators=[('rf', rf), ('dt', tree), ('svm', svm)], vo
eclf1.fit(X_train, y_train)

y_pred = eclf1.predict(X_test)

vot_acc = accuracy_score(y_pred, y_test)
vot_prec = precision_score(y_pred, y_test, average='weighted')
vot_rec = recall_score(y_pred, y_test, average='weighted')
vot_f1 = f1_score(y_pred, y_test, average='weighted')
```

Fig 14 Voting classifier

Stacking Classifier- Stacking combines the predictions of multiple models using a meta-classifier. It enhances the network attack detection system by taking into account diverse model outputs and providing a more accurate final decision.



```
from sklearn.ensemble import StackingClassifier, ExtraTreesClassifier

estimators = [('rf', rf),('dt', tree)]

clf = StackingClassifier(estimators=estimators, final_estimator=ExtraTreesClassi

clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

stac_acc = accuracy_score(y_pred, y_test)
stac_prec = precision_score(y_pred, y_test,average='weighted')
stac_rec = recall_score(y_pred, y_test,average='weighted')
stac_f1 = f1_score(y_pred, y_test,average='weighted')
```

Fig 15 Stacking classifier

4. EXPERIMENTAL RESULTS

Precision: Precision evaluates the fraction of correctly classified instances or samples among the ones classified as positives. Thus, the formula to calculate the precision is given by:

Precision = True positives/ (True positives + False positives) = TP/(TP + FP)

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

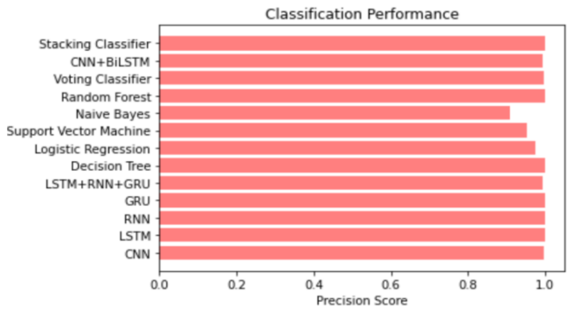


Fig 16 Precision comparison graph

Recall: Recall is a metric in machine learning that measures the ability of a model to identify all relevant instances of a particular class. It is the ratio of correctly predicted positive observations to the total actual positives, providing insights into a model's completeness in capturing instances of a given class.

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

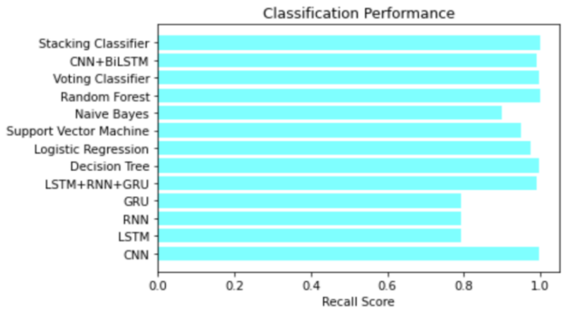


Fig 17 Recall comparison graph

ISSN 2454 - 5015

Accuracy: Accuracy is the proportion of correct predictions in a classification task, measuring the overall correctness of a model's predictions.

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

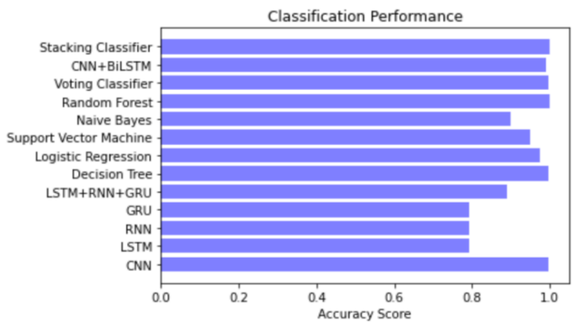


Fig 18 Accuracy graph

F1 Score: The F1 Score is the harmonic mean of precision and recall, offering a balanced measure that considers both false positives and false negatives, making it suitable for imbalanced datasets.

$$\text{F1 Score} = 2 * \frac{\text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} * 100$$

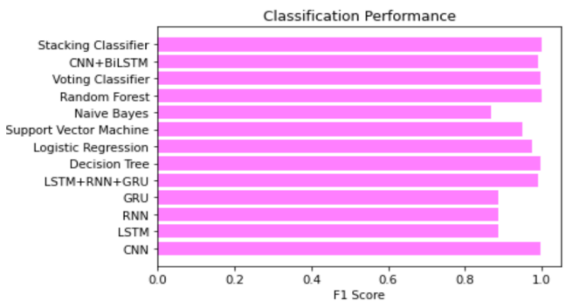
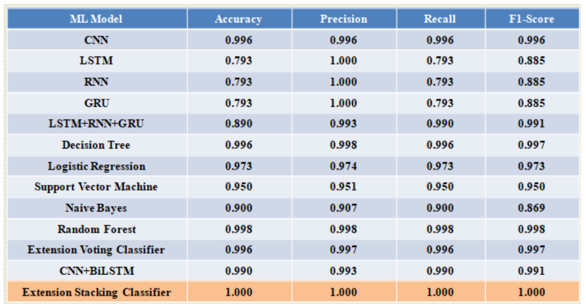
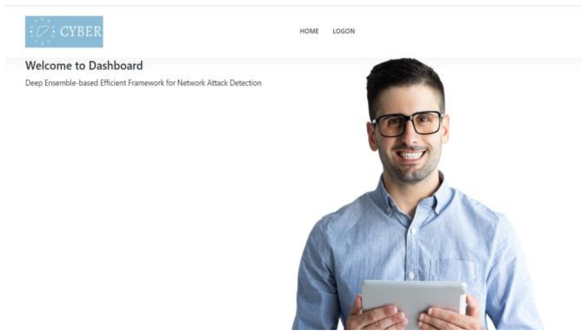


Fig 19 F1Score



ML Model	Accuracy	Precision	Recall	F1-Score
CNN	0.996	0.996	0.996	0.996
LSTM	0.793	1.000	0.793	0.885
RNN	0.793	1.000	0.793	0.885
GRU	0.793	1.000	0.793	0.885
LSTM+RNN+GRU	0.890	0.993	0.990	0.991
Decision Tree	0.996	0.998	0.996	0.997
Logistic Regression	0.973	0.974	0.973	0.973
Support Vector Machine	0.950	0.951	0.950	0.950
Naive Bayes	0.900	0.907	0.900	0.869
Random Forest	0.998	0.998	0.998	0.998
Extension Voting Classifier	0.996	0.997	0.996	0.997
CNN+BiLSTM	0.990	0.993	0.990	0.991
Extension Stacking Classifier	1.000	1.000	1.000	1.000

Fig 20 Performance Evaluation



Welcome to Dashboard

Deep Ensemble-based Efficient Framework for Network Attack Detection

Fig 21 Home page

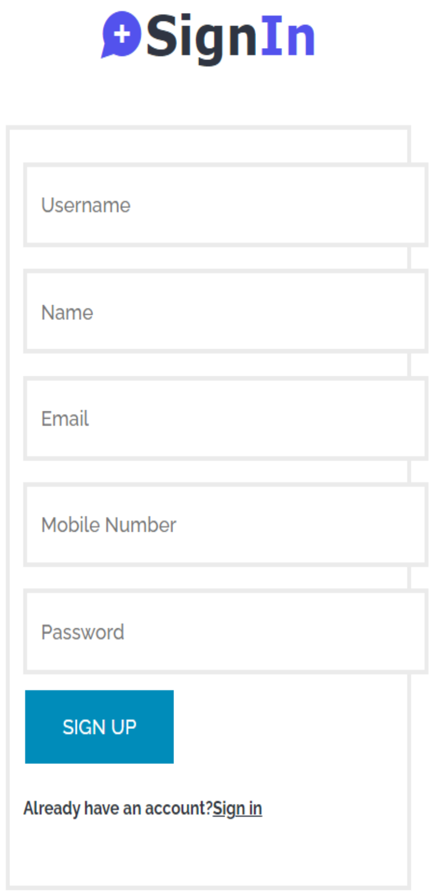
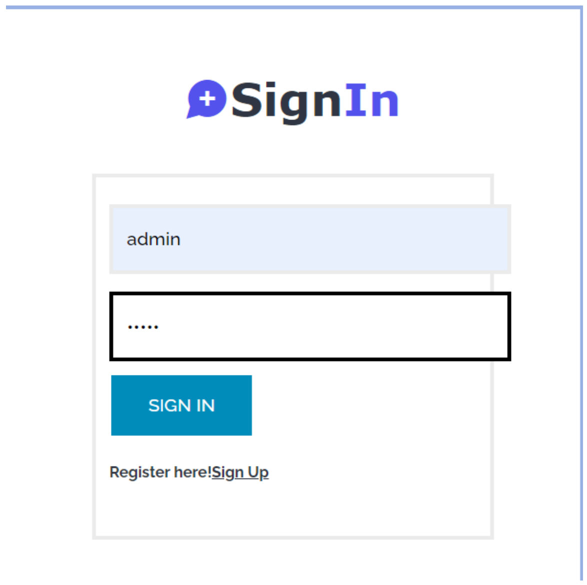


Fig 22 Signin page



admin

.....

Fig 23 Login page

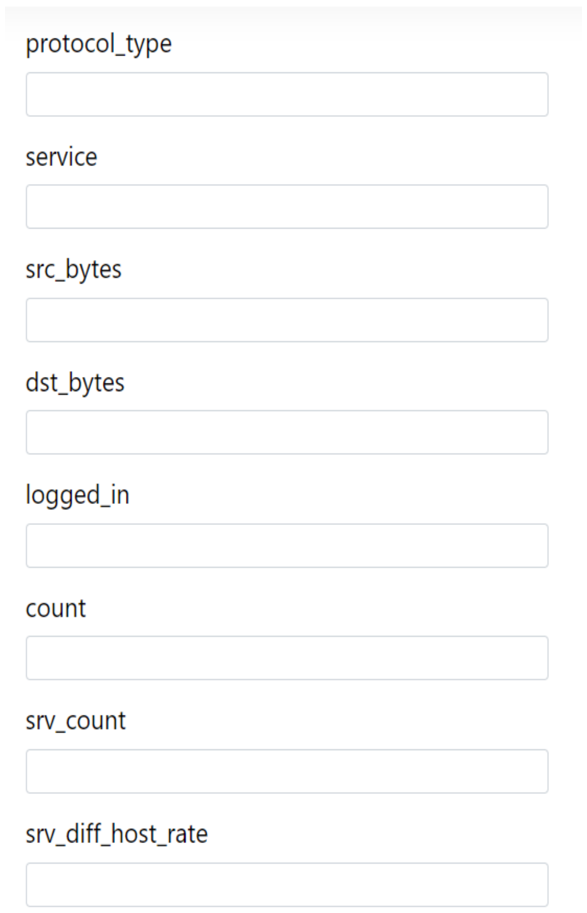
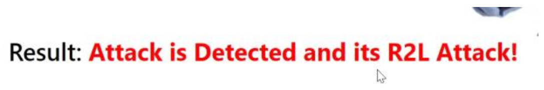


Fig 24User input



Result: Attack is Detected and its R2L Attack!

Fig 25 Predict result for given input

5. CONCLUSION

The project strategically incorporates advanced deep learning models, including Long Short-Term Memory (LSTM), Recurrent Neural Network (RNN), and Gated Recurrent Unit (GRU), to bolster network attack detection [27, 29]. By leveraging the capabilities of these models, the project aims to enhance computer network security, offering a robust defense mechanism against various forms of network attacks. The project's framework is rigorously evaluated using the NSL-KDD dataset, showcasing its effectiveness in detecting network attacks with high accuracy. This empirical validation demonstrates the framework's practical utility and its capability to accurately identify and respond to malicious activities within computer networks. The inclusion of deep learning models and ensemble techniques provides the framework with adaptability, allowing it to learn and adjust to evolving attack patterns. This adaptability is crucial for enhancing the network's resilience against emerging threats, ensuring that the framework remains effective in addressing the dynamic nature of cybersecurity challenges. The project extends its capabilities by employing ensemble techniques like Voting Classifier and Stacking Classifier, contributing to heightened accuracy in network attack detection. Additionally, the integration of a user-friendly Flask interface with secure authentication enhances the overall user experience during system testing. This interface facilitates the input of data for performance evaluation, making the framework accessible and practical for cybersecurity professionals. The combination of ensemble techniques and user-friendly features enhances the framework's effectiveness and usability in real-world scenarios.

6. FUTURE SCOPE

Future work aims to optimize the computational efficiency of the system by delving into the architecture of individual deep learning models. This optimization will lead to more efficient network attack detection, making the system both powerful and cost-effective. The focus on system scalability will make it more practical for deployment in real-world network security solutions. This enhancement will not only ensure that the system can handle larger and more complex networks but also reduce operational costs. The project's future work includes developing mechanisms for real-time network attack detection [19]. This real-time capability will allow the system to swiftly respond and mitigate threats as they occur, reducing potential damage and vulnerability. As network data volumes continue to grow, the project will adapt the framework to handle large-scale network data. This adaptation is essential to accommodate the increasing complexity and volume of network traffic, ensuring the system remains effective. Extending the framework to include anomaly detection capabilities is a critical aspect of future work. This expansion will enable the system to identify novel and evolving attack patterns, offering a proactive approach to network security and adapting to emerging threats.

REFERENCES

[1] F. Tang, X. Chen, M. Zhao, and N. Kato, “The roadmap of communication and networking in 6g for the metaverse,” IEEE Wireless Communications, 2022.

[2] H. Guo, X. Zhou, J. Liu, and Y. Zhang, “Vehicular intelligence in 6g: Networking, communications, and computing,” Vehicular Communications, vol. 33, p. 100399, 2022.

[3] P. L. Indrasiri, E. Lee, V. Rupapara, F. Rustam, and I. Ashraf, “Malicious traffic detection in iot and local networks using stacked ensemble classifier,” Computers, Materials and Continua, vol. 71, no. 1, pp. 489– 515, 2022.

[4] Y. Maleh, Y. Qasmaoui, K. El Gholami, Y. Sadqi, and S. Mounir, “A comprehensive survey on sdn security: threats, mitigations, and future directions,” Journal of Reliable Intelligent Environments, pp. 1–39, 2022.

[5] J. Wang, J. Liu, J. Li, and N. Kato, “Artificial intelligence-assisted network slicing: Network assurance and service provisioning in 6g,” IEEE Vehicular Technology Magazine, 2023.

[6] M. A. Talukder, K. F. Hasan, M. M. Islam, M. A. Uddin, A. Akhter, M. A. Yousuf, F. Alharbi, and M. A. Moni, “A dependable hybrid machine learning model for network intrusion detection,” Journal of Information Security and Applications, vol. 72, p. 103405, 2023.

[7] J. Liu, B. Kantarci, and C. Adams, “Machine learning-driven intrusion detection for contiki-ng-based iot networks exposed to nsl-kdd dataset,” in Proceedings of the 2nd ACM workshop on wireless security and machine learning, 2020, pp. 25–30.

[8] T. Su, H. Sun, J. Zhu, S. Wang, and Y. Li, “Bat: Deep learning methods on network intrusion detection using nsl-kdd dataset,” IEEE Access, vol. 8, pp. 29 575–29 585, 2020.

[9] G. C. Amaizu, C. I. Nwakanma, J.-M. Lee, and D.-S. Kim, “Investigating network intrusion detection datasets using machine learning,” in 2020 International Conference on Information and Communication Technology Convergence (ICTC). IEEE, 2020, pp. 1325–1328.

[10] M. Esmaeili, S. H. Goki, B. H. K. Masjidi, M. Sameh, H. Gharagozlou, and A. S. Mohammed, “Ml-ddosnet: Iot intrusion detection based on denial-of-service attacks using machine learning methods and nsl-kdd,” Wireless Communications and Mobile Computing, vol. 2022, 2022.

[11] A. K. Balyan, S. Ahuja, U. K. Lilhore, S. K. Sharma, P. Manoharan, A. D. Algarni, H. Elmannai, and K. Raahemifar, “A hybrid intrusion detection model using ega-pso and improved random forest method,” Sensors, vol. 22, no. 16, p. 5986, 2022.

[12] K. Jiang, W. Wang, A. Wang, and H. Wu, “Network intrusion detection combined hybrid sampling with deep hierarchical network,” IEEE access, vol. 8, pp. 32 464–32 476, 2020.

[13] C. Liu, Z. Gu, and J. Wang, “A hybrid intrusion detection system based on scalable k-means+ random forest and deep learning,” Ieee Access, vol. 9, pp. 75 729–75 740, 2021.

[14] S. Cherfi, A. Boulaiche, and A. Lemouari, “Multi-layer perceptron for intrusion detection using simulated annealing,” in Modelling and Implementation of Complex Systems: Proceedings of the 7th International Symposium, MISC 2022, Mostaganem, Algeria, October 30-31, 2022. Springer, 2022, pp. 31–45.

[15] A. O. Alzahrani and M. J. Alenazi, “Designing a network intrusion detection system based on machine learning for software defined networks,” Future Internet, vol. 13, no. 5, p. 111, 2021.

[16] T. Wisanwanichthan and M. Thammawichai, “A double-layered hybrid approach for network intrusion detection system using combined naive bayes and svm,” IEEE Access, vol. 9, pp. 138 432–138 450, 2021.

[17] N. Sahar, R. Mishra, and S. Kalam, “Deep learning approach-based network intrusion detection system for fog-assisted iot,” in Proceedings of international conference on big data, machine learning and their applications: ICBMA 2019. Springer, 2021, pp. 39–50.

[18] F. Z. Belgrana, N. Benamrane, M. A. Hamaida, A. M. Chaabani, and A. Taleb-Ahmed, “Network intrusion detection system using neural network and condensed nearest neighbors with selection of nsl-kdd influencing features,” in 2020 IEEE International Conference on Internet of Things and Intelligence System (IoTaIS). IEEE, 2021, pp. 23–29.

[19] M HASSAN ZAIB, “NSL-KDD — Kaggle.” [Online]. Available: https://www.kaggle.com/datasets/hassan06/nslkdd

[20] E. Bisong and E. Bisong, “Introduction to scikit-learn,” Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners, pp. 215–229, 2019.

[21] A. Pashamokhtari, G. Batista, and H. H. Gharakheili, “Adiotack: Quantifying and refining resilience of decision tree ensemble inference models against adversarial volumetric attacks on iot networks,” Computers & Security, vol. 120, p. 102801, 2022.

[22] S. Tufail, S. Batool, and A. I. Sarwat, “A comparative study of binary class logistic regression and shallow neural network for ddos attack prediction,” in SoutheastCon 2022. IEEE, 2022, pp. 310–315.

[23] A. Raza, H. U. R. Siddiqui, K. Munir, M. Almutairi, F. Rustam, and I. Ashraf, “Ensemble learning-based feature engineering to analyze maternal health during pregnancy and health risk prediction,” Plos one, vol. 17, no. 11, p. e0276525, 2022.

[24] S. Ismail and H. Reza, “Evaluation of na¨ıve bayesian algorithms for cyber-attacks detection in wireless sensor networks,” in 2022 IEEE World AI IoT Congress (AIIoT). IEEE, 2022, pp. 283–289.

[25] T. Wu, H. Fan, H. Zhu, C. You, H. Zhou, and X. Huang, “Intrusion detection system combined enhanced random forest with smote algorithm,” EURASIP Journal on Advances in Signal Processing, vol. 2022, no. 1, pp. 1–20, 2022.

[26] F. Rustam, M. F. Mushtaq, A. Hamza, M. S. Farooq, A. D. Jurcut, and I. Ashraf, “Denial of service attack classification using machine learning with multi-features,” Electronics, vol. 11, no. 22, p. 3817, 2022.

[27] S. Kaur and M. Singh, “Hybrid intrusion detection and signature generation using deep recurrent neural networks,” Neural Computing and Applications, vol. 32, pp. 7859–7877, 2020.

[28] S. M. Kasongo and Y. Sun, “A deep gated recurrent unit based model for wireless intrusion detection system,” ICT Express, vol. 7, no. 1, pp. 81–87, 2021.

[29] F. Laghrissi, S. Douzi, K. Douzi, and B. Hssina, “Intrusion detection systems using long short-term memory (lstm),” Journal of Big Data, vol. 8, no. 1, p. 65, 2021.

[30] Y. Lin, H. Zhao, X. Ma, Y. Tu, and M. Wang, “Adversarial attacks in modulation recognition with convolutional neural networks,” IEEE Transactions on Reliability, vol. 70, no. 1, pp. 389–401, 2020.

[31] S. Zargar, “Introduction to sequence learning models: Rnn, lstm, gru,” no. April, 2021.

[32] L. van der Maaten and G. Hinton, “Visualizing data using t-sne,” Journal of Machine Learning Research, vol. 9, no. 86, pp. 2579–2605, 2008. [Online]. Available: http://jmlr.org/papers/v9/vandermaaten08a.html