

Novel Approach for Cyber Attack Detection in Network using Machine Learning Techniques

A. Sri Nagesh¹, Samrat Krishna Gaddam², K. Bhuvaneswari³

¹Department of Computer Science and Engineering, RVR & JC College of Engineering, Guntur, Andhra Pradesh, India

²Department of Computer Science, Parvathaneni Brahmayya Siddhartha College of Arts & Science, Vijayawada, Andhra Pradesh, India

³Department of Computer Science, Parvathaneni Brahmayya Siddhartha College of Arts & Science, Vijayawada, Andhra Pradesh, India

¹email: asrinagesh@gmail.com, ²email: samratkrishnagaddam@gmail.com ³email: bhuvikosana437@gmail.com



Open Access Research Article

Received: 09/07/2026

Accepted: 15/08/2026

Published: 24/09/2026

Corresponding author email:

asrinagesh@gmail.com

Citation:

Sri Nagesh A et al., "Quantum Inspired Metaheuristic Routing for Ultra-Dense 6G Networks", *Ci-STEM Journal of Digital Technologies and Expert Systems*, Vol. 3(2), pp. 109-124, 2026

doi: 10.55306/CJDTES.2026.030204

Copyright:

©2026 Sri Nagesh A et al.,

This is an open-access article distributed under the terms of the Creative Commons Attribution License which grants the right to use, distribute, and reproduce the material in any medium, provided that proper attribution is given to the original author and source, in accordance with the terms outlined by the license.

(<https://creativecommons.org/licenses/by/4.0/>)

Published By:

Ci-STEM Global Services Foundation, India.

Abstract:

Cyber-attacks on computer networks are evolving, and signature-based defence systems are struggling to keep pace. One of the more pragmatic answers to this problem has been machine learning, not because it replaces a security analyst, but because it can sift through traffic much faster than a human team while catching patterns that static rules miss. This paper investigates the performance of four supervised classifiers, namely Support Vector Machine (SVM), Decision Tree, Random Forest and an Artificial Neural Network (ANN) on the NSL-KDD dataset, which is an improved, de-duplicated version of the older KDD Cup 1999 data. All four models were run through the same pre-processing, feature engineering and validation steps so there is no skew in the comparison due to inconsistent handling of the data. It is important to note upfront how much reported IDS accuracy depends on the dataset and protocol used. For example, work on evaluating SVM on CICIDS2017 has reported precision as high as 97.80% on one attack category and as low as 69.79% on another, which is a widespread for the same algorithm. In our own runs on NSL-KDD Random Forest scored the best with 99.6% accuracy while ANN came close at 99.11% and SVM and Decision Tree stood behind at 93.29% and 83.4% respectively. The pattern suggests that ensemble tree methods are well suited for the discrete, high-dimensional structure of network connection records. The paper also puts these numbers alongside recent feature-selection- and deep-learning-based intrusion detection works to see where the classical approach still holds its ground and where it doesn't.

Keywords:

Artificial neural network, Cybercrime, Decision tree, Intrusion detection system (IDS), Machine learning, Network security, NSL-KDD, Random forest, Support vector machine.

1. INTRODUCTION

Network security research has picked up considerably over the past decade. A lot of this is simply the growth of Internet usage, and some of it is the steady drumbeat of data breaches that make the news every few months. Whatever the reason, the practical reality is that there is more data available to more people and devices than ever before and keeping that data private and intact is no longer something that can be considered secondary by organizations. An intrusion is when someone accesses or changes a network resource that they were not supposed to touch, according to security practitioners. They are usually classified into four categories:

- 1) Denial-of-Service (DoS) attacks, which try to make a resource unavailable to the people who actually need it;
- 2) Probing attacks, which scan a network looking for weaknesses worth exploiting later;
- 3) User-to-Root (U2R) attacks, where an attacker who already has some foothold escalates it into administrator-level access;
- 4) Remote-to-Local (R2L) attacks, where someone with no account on a machine finds a way to get local access anyway.

Because it is much better to catch these attacks before they do damage than after, an intrusion detection system (IDS) needs to watch network activity continuously and flag anything that looks like trouble. At its core, this is a prediction problem: train a model on records of past attacks, then use it to judge whether a new connection looks malicious. That is the approach taken here, using the NSL-KDD dataset to train and compare several machine learning models. NSL-KDD is a revised version of the older KDD Cup 1999 (KDD99) dataset, with the redundant records stripped out so that classifiers are not simply memorizing the handful of attack patterns that happen to repeat most often [3], [19]. The four algorithms to be considered are Artificial Neural Network (ANN), Support Vector Machine (SVM), Decision Tree and Random Forest, and the goal is simple: to determine which of them can most reliably classify network traffic as normal or malicious. The choice of algorithm is more important than it may seem, as it sets the limits for the trade-off between accuracy, false alarms and computing costs an IDS can actually achieve. The rest of the paper attempts to answer that question with facts and evidence and not with theory.

The paper is organized as follows. Section 2 covers related work in intrusion detection and the machine learning methods used for it. The methodology is described in Section 3: the dataset, the preprocessing steps, feature engineering, the mathematical basis of each classifier and the metrics used to judge them. Section 4 presents Experimental Results. The four classifiers are lined up against each other. Section 5 deals with the implications of these results in the light of some recent published work. Section 6 concludes with some thoughts on what could be done differently next time.

2. RELATED WORK

There is a fair amount of previous work to draw on here, and it falls roughly into five strands: building better benchmark data sets, applying classical classifiers directly, moving to deep architectures, trimming down the feature space before training, and, more broadly, using machine learning on cyber security problems that sit next to intrusion detection rather than inside it. Buczak and Guven [9] provide a survey of data mining and machine learning methods for cyber security. This covers much of the same ground, and is a useful map of the field before getting into specific papers. Below are the five threads that influenced the decisions made in Section 3.

A. Work Built Around the Datasets Themselves

A lot of progress in this area has actually been progress in the datasets, not just the algorithms. Tavallaee et al. [19] studied the original KDD Cup 1999 data and found that it contained a large number of redundant records, enough to bias any classifier toward the attack type that occurred most often. Their correction is the dataset used in this paper, NSL-KDD. A similar point has been made by Sharafaldin et al. [3] about KDD99 in general. KDD99 was put together in 1999 with 41 features and simply does not represent how attacks look today, which is why they built CICIDS2017 from more realistic background traffic and current attack scenarios. Ring et al. [10] took a step back and surveyed the wider landscape of intrusion detection datasets, cataloging what each one is good and bad for. Moustafa and Slay [26] added UNSW-NB15 to the mix, blending synthetic modern attacks with real background traffic. Stiawan et al. [27] performed an information gain analysis on CICIDS2017 in an effort to find the truly relevant features for anomaly detection. Overall, these papers provide a pretty convincing case that reported detection accuracy is tightly coupled to the dataset used to generate it. This is part of the reason this study uses a single, well understood dataset, NSL-KDD, for all experiments, rather than mixing sources.

B. Classical Machine Learning Applied Directly

Much work has just taken a standard classifier and run network traffic through it, measuring the output. Reddy et al. [1] and Uyyala [2] independently studied intrusion detection using machine learning on network traffic and reported results for several classifiers. Uyyala [2] applied SVM on the CICIDS2017 specifically and obtained a precision of 97.80% on one attack class and only 69.79% on another after normalizing the data, and found that replacing SVM with Random Forest, CNN or ANN increased accuracy to 99.93%, 63.52% and 99.11% respectively. Class imbalance is a common problem in this type of work and Ahmed et al. [5] addressed it by combining an oversampling technique (SMOTE) with standard classifiers on UNSW-NB15 to prevent rare attack categories being overwhelmed by the majority during training. Sangkatsanee et al. [8] took a more applied approach,

and built a real-time IDS based on decision-tree classification of flow statistics, achieving a false-alarm rate low enough for practical use. Resende and Drummond [4] combined a genetic algorithm with profiling for an adaptive anomaly-based IDS on CICIDS2017 and achieved a detection rate of 92.85% at 0.69% false positive rate. The work by Farnaaz and Jabbar [20] combined Random Forest with symmetrical-uncertainty feature selection on NSL-KDD and reported 99.67% accuracy, which is higher than a plain J48 decision tree baseline. Going back further, Christopher [7] wrote up port-scanning techniques and how to defend against them, which is really the signature-based ancestor of the probing-detection work that came later. On the survey side, Khraisat et al. [21] and Ozkan-Okay et al. [24] both took stock of the field: the former around techniques, datasets, and open problems, the latter as a full systematic review spanning signature-based, anomaly-based, and hybrid designs. More recently, Waghmode et al. [23] built a quantum-inspired least-squares SVM on top of exhaustive feature selection and reported 99.3%, 99.5%, and 93.3% accuracy on NSL-KDD, CICIDS2017, and UNSW-NB15 — the underlying quantum-inspired SVM formulation itself comes from Ding et al. [32]. Hadem et al. [39] used SVM with selective logging inside an SDN-based IDS for IP traceback, and Kumar et al. [38] reviewed research trends across network-based intrusion detection more broadly.

C. The Move Toward Deep Learning

Deep architectures promise to learn feature representations automatically instead of requiring someone to engineer them by hand, and a number of studies have tested that promise on intrusion data. Javaid et al. [13] used sparse autoencoders in a self-taught learning setup for NSL-KDD, while Tang et al. [14] tried a deep neural network inside a software-defined networking context. Shone et al. [12] went with a non-symmetric deep autoencoder for unsupervised feature learning and then handed the resulting features to a Random Forest, getting strong numbers on both KDD99 and NSL-KDD. Vinayakumar et al. [15] looked at deep learning for network traffic prediction more generally, and Kim and Kim [11] combined an LSTM-based system-call language model with an ensemble strategy for host-based detection. On the review side, Aleesa et al. [25] pulled together a taxonomy of deep-learning IDS approaches along with the open problems still facing the field, and Psychogyios et al. [34] focused specifically on how deep learning handles time-series network data. Das et al. [31] mixed ensemble learning with feature selection for a comparative study, and Yue et al. [33] proposed contrastive learning as a way to improve representation learning when labelled data is scarce. Zoghi and Serpen [22] took a slightly different tack, designing an ensemble classifier tuned to the statistical quirks (class imbalance, class overlap) of the specific dataset it runs on, using UNSW-NB15 as the test case.

D. Cutting Down the Feature Space

Network intrusion datasets tend to be full of correlated or outright redundant features, so it is not surprising that a lot of effort has gone into feature selection as a way to improve both accuracy and speed. LDA and PCA were used by Ibrahimi and Ouaddane [6] on NSL-KDD and received improved classification accuracy and false-alarm rate. Saheed et al. [35] employed a bat-inspired metaheuristic and a residue number system based encoding to discover compact and highly valuable feature subsets. Nabi and Zhou [36] compared different dimensionality reduction techniques for cyber security tasks and concluded that careful feature selection results in more consistent detection across datasets rather than higher accuracy on a single dataset. The symmetrical-uncertainty approach of Farnaaz and Jabbar [20] discussed above is actually of the same story: feature selection improves both detection rate and false-alarm rate of a Random Forest classifier. These results fairly directly informed the assembly of the feature engineering stage of the pipeline of this paper itself, which is described in Section III.

E. Machine Learning Beyond Intrusion Detection

Similar classification ideas are seen in cyber security problems close to network intrusion detection. Bio-inspired metaheuristic optimization was used by Gibson et al. [28] to improve the accuracy of spam-email detection. Ige et al. [29] compared the performance of Naive Bayes and other non-Bayesian classifiers and deep learning models in terms of detecting phishing detection. Their work is relevant in providing context on how a simple probabilistic classifier such as Naive Bayes compares with ensemble or neural methods. This is a reminder that the four classifiers investigated in this paper are not confined to traditional wired networks, as evidenced by Otoum and Nayak [37] who combined anomaly- and signature-based detection in Internet-of-Things settings. A study by Ingre and Yadav

[30] explored the performance of an ANN on NSL-KDD, which serves as a good benchmark for the ANN results discussed in Section IV. Taken as a whole, this body of work suggests that SVM, Decision Tree, Random Forest, and ANN continue to be the workhorses of machine-learning-based cyber defense even as newer methods emerge, which is really the motivation for running all four through the same experimental protocol here rather than picking just one.

3. RESEARCH METHODOLOGY

Fig. 1 lays out the workflow used for this study: data collection and preparation first, then model training, evaluation, and finally the output stage. The subsections below go through each of these in turn.

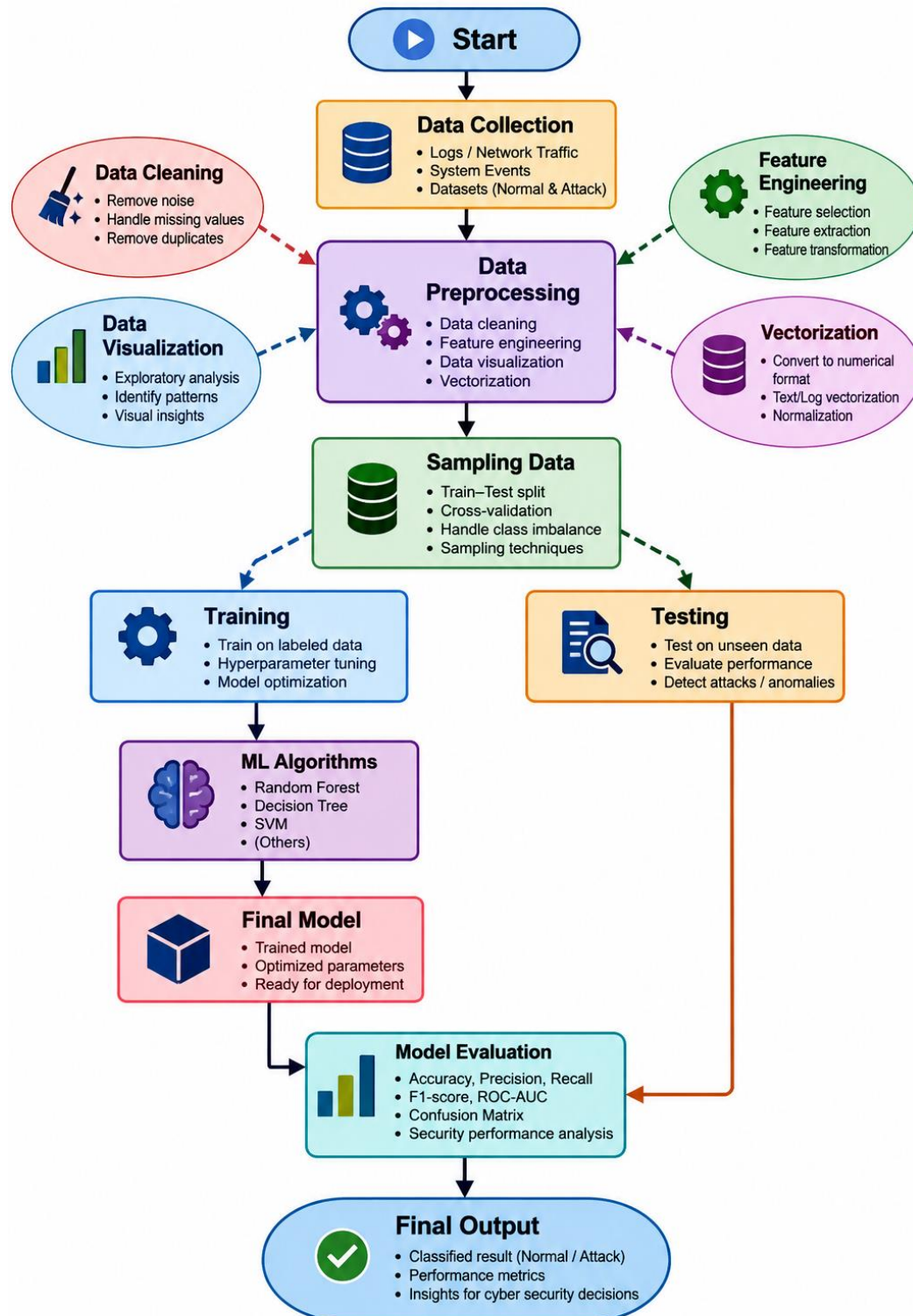


Fig. 1. Workflow of the proposed intrusion detection system.

A. Dataset

The first step is simply getting the data in. NSL-KDD was used throughout, which, as noted earlier, is a cleaned-up version of KDD99 with the duplicate and redundant records taken out [3], [19]. The copy used here has 43 fields, 125,973 records for training, and 22,544 held out for testing. A few things make NSL-KDD a better benchmark than its predecessor:

- no duplicate records in the training set, so classifiers are not pushed toward whatever attack category happens to be over-represented;
- no null values anywhere in the fields;
- no unnecessary duplicate records at all, meaning a classifier does not end up training on the same example over and over without realizing it.

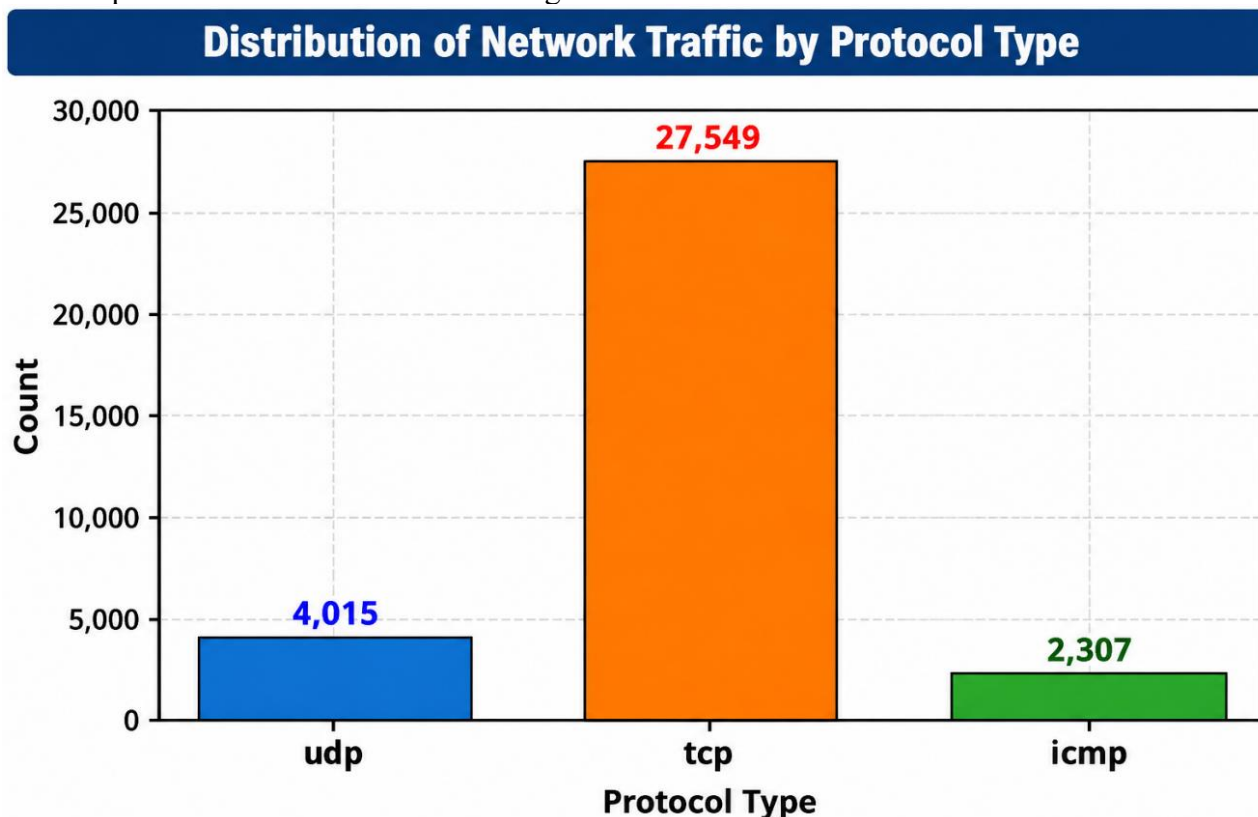


Fig. 2. Distribution of connection records across protocol types (TCP, UDP, and ICMP) in the NSL-KDD training partition.

B. Data Validation Process

Validation is how an estimate of a model's error rate on unseen data is produced, and it matters most exactly when a data sample might not represent the underlying population well — which, in practice, is most of the time. Every attribute, whether it is a float or an integer, needs to be checked for missing values, duplicates, and whether its type and format actually match what is expected. A held-out validation subset is what gets used to see how well a model fits the training data while its hyperparameters are being tuned, though it is worth being a little careful here: the more that subset gets consulted, the more information leaks from it into the model configuration, and the more optimistic (and less trustworthy) the evaluation on it becomes. Regression is the right tool when the data looks like a time series; classification, which is what this paper uses throughout, fits the discrete attack-category labels that NSL-KDD provides.

Besides looking for missing values, duplicates, and wrong data types, the dataset needs to be checked for unequal numbers of classes and any problems with how the features are shown. Keeping the training and testing data apart is important to understand the efficiency of model working. In order to completely understand and make valid comparisons all the models should follow the same preprocessing steps. Applying validation metrics like F1-score, accuracy, precision and recall makes the model understanding easy.

Sample Records from Network Intrusion Dataset														
#	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_same_srv_rate	dst_host_diff_srv_rate	dst_host_same_src_port_rate
0	0	udp	other	SF	146	0	0	0	0	0	...	0.00	0.60	0.88
1	0	tcp	private	S0	0	0	0	0	0	0	...	0.10	0.05	0.00
2	0	tcp	http	SF	232	8153	0	0	0	0	...	1.00	0.00	0.03
3	0	tcp	http	SF	199	420	0	0	0	0	...	1.00	0.00	0.00
4	0	tcp	private	REJ	0	0	0	0	0	0	...	0.07	0.07	0.00

Fig. 3. Sample view of the NSL-KDD data frame after loading, showing categorical fields (protocol type, service, flag) alongside numeric traffic statistics.

C. Data Preprocessing

But, a machine learning model cannot take the raw traffic logs directly. So, there is a need of preprocessing to convert them into a usable format. The majority of this step is data cleaning: removing duplicate or suspicious records, which would otherwise impact accuracy and reliability. For categorical fields (protocol type, service and flag), they are encoded numerically. Continuous fields are scaled so that a distance- or margin-based classifier such as SVM does not end up weighting one feature more heavily simply because its raw numbers happen to be larger. All of this is to catch and eliminate the kinds of errors and anomalies that would otherwise silently degrade the value of the data further down the pipeline.

D. Exploratory Data Analysis

Exploratory data analysis (EDA) is really just getting a feel for the dataset before doing anything else to it. When analyzing class balance, distribution of protocol types, and frequency of each attack category, it usually shows trends, corrupted records, and outliers long before they can do any damage to model training. Figure 4 shows the breakdown of normal vs attack traffic by protocol type for NSL-KDD and it is immediately clear that ICMP traffic is heavily skewed toward attack labeled records.

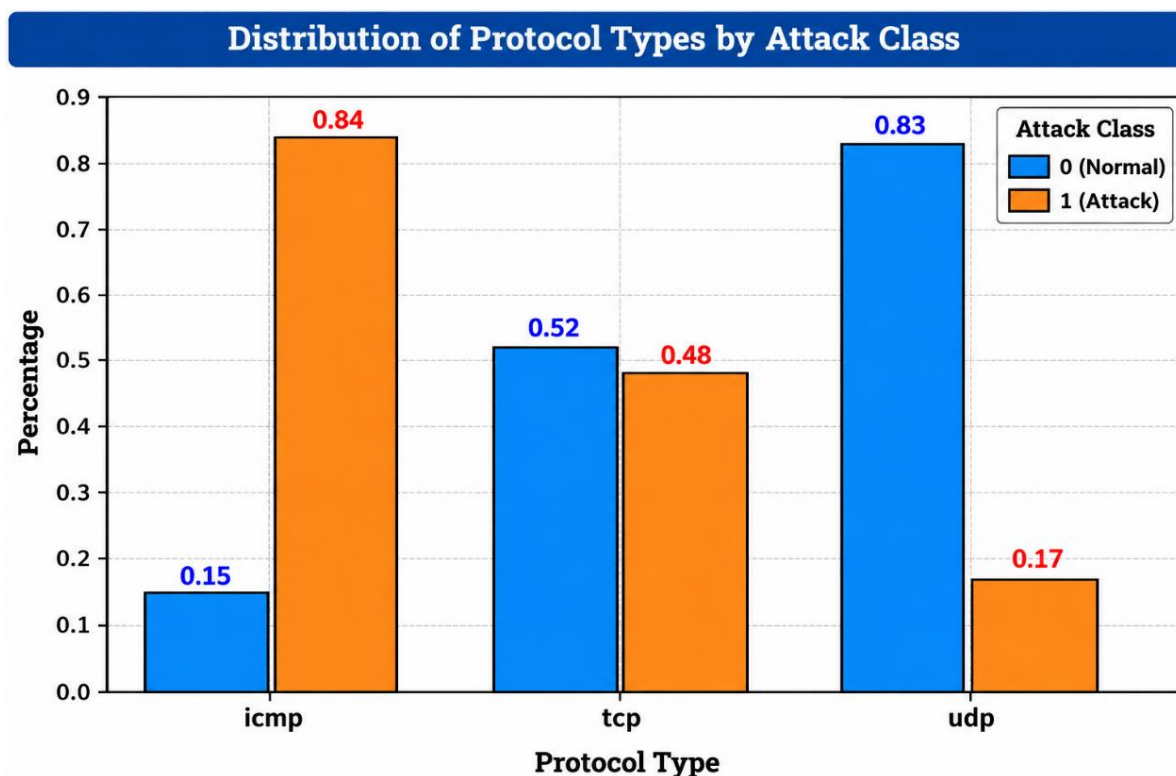


Fig. 4. Percentage of normal (class 0) versus attack (class 1) traffic across protocol types.

E. Feature Engineering

Feature engineering is the process of taking raw fields and transforming them into something that is more useful for a classifier to learn from — selecting, transforming and combining as needed. In this work we dropped the redundant and near-constant fields, label-encoded the categorical attributes to fit each classifier and normalized the numeric attributes to a common scale prior to any training.

F. Machine Learning Algorithms and Their Mathematical Formulation

Four classifiers were trained on the pre-processed data which were Random Forest, SVM, Decision Tree and an ANN. Here is a description of each, with the mathematics involved.

1) Random Forest:

Random forest is an ensemble of decision trees that are standard. Each tree is trained on a bootstrapped sample of the training data and each split considers a random subset of features [16]. It's popular for good reasons: accurate, not too hard to use, and flexible enough to attack both classification and regression thanks to its non-linear decision boundaries. In a classification task the prediction of the forest on the input x is a simple vote from the B trees:

$$f(x) = \frac{1}{B} \sum_{b=1}^B f_b(x) \quad (1)$$

Here $\hat{f}(x)$ is the class the forest lands on, B is how many trees are in the forest, and f_b is what the b -th tree individually predicts. The trees are decorrelated with each other, so averaging their votes reduces the variance compared to trusting any one tree by itself [16].

2) Support Vector Machine (SVM):

SVM works by finding the optimal hyperplane to separate classes in an N -dimensional feature space [17]. That separating hyperplane is written as:

$$w \cdot x + b = 0 \quad (2)$$

where w is the weight vector perpendicular to the hyperplane and b is the bias term. What SVM does is actually to optimize for the largest possible margin between the closest points of each class, the support vectors. This is the constrained optimization problem:

$$\min \frac{1}{2} \|w\|^2 \quad \text{s.t.} \quad y_i(w \cdot x_i + b) \geq 1 \quad (3)$$

with x_i as the i -th training sample and $y_i \in \{-1, +1\}$ its label. This is common for traffic patterns that are not linearly separable. Kernel functions implicitly deal with it by transforming the input to a higher dimensional space where it can be separated.

3) Decision Tree:

Decision tree models can be used for classification and regression, but classification is far more common. It functions as a tree-based classifier where internal nodes examine some feature of the dataset and the branches represent the outcome of that test. Each leaf culminates with a predicted class label. The idea is to learn a compact set of decision rules from training data, which still generalizes to connection records which the model has not seen before. Splits are chosen to minimize the impurity of the resulting partitions, commonly measured by entropy:

$$H(S) = - \sum_{i=1}^c p_i \log_2 p_i \quad (4)$$

where p_i is the share of subset S belonging to class i , and c is the number of classes. Whichever attribute cuts entropy down the most — the information gain — gets chosen at each split:

$$IG(S, A) = H(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} H(S_v) \quad (5)$$

where S_v is the subset of S where attribute A takes value v .

4) Artificial Neural Network (ANN):

The ANN here is a Multilayer Perceptron (MLP), a feed-forward network where each neuron takes a weighted sum of its inputs, adds a bias, and pushes through a nonlinear activation function ϕ :

$$y = \varphi\left(\sum_{i=1}^n w_i x_i + b\right) \quad (6)$$

An MLP has three types of layers, input, hidden, and output. They are loosely modeled on the way biological neurons pass signals along. Training is done by back-propagation which calculates the gradient of a loss function in relation to each of the weights in the network and update them to minimize prediction error [18]. The logistic sigmoid is a popular choice for activation:

$$\varphi(z) = \frac{1}{1 + e^{-z}} \quad (7)$$

that squeezes its input to the range (0, 1) — convenient for producing something that looks like a probability, whether the classification task is binary or multi-class.

G. Proposed Approach

Table I shows the whole experimental procedure followed for each of the classifiers in this work from beginning to End.

Table 1. Steps followed for training and evaluation of model

Step	Description
Step 1	Import the required libraries that will be the foundation of the project (data processing, visualization, and machine learning libraries).
Step 2	Load the NSL-KDD dataset and normalize the numeric features to ensure better and consistent performance of the algorithm.
Step 3	Run the pre-processing pipeline on the data including cleaning suspicious or missing values.
Step 4	The dataset is divided into training and testing part to train and evaluate the model.
Step 5	Build candidate IDS models by training SVM, Random Forest, Decision Tree and ANN classifiers on the training partition.
Step 6	Assess the performance of each trained model on the held-out testing partition.
Step 7	Each model is evaluated by the evaluation metrics defined in Section III-H.
Step 8	Determine and compare the accuracy of each model on both the training and testing datasets.

H. Evaluation Metrics

We evaluate the performance using four standard metrics for classification based on the confusion matrix. TP, TN, FP and FN denote true positive, true negative, false positive and false negative respectively:

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

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

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

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (11)$$

Accuracy is simply the percentage of all connection records that are correctly classified. Precision and recall break this down a bit: precision asks how many of the predicted attacks were actual attacks, recall asks how many actual attacks were actually caught. F1-score is a measure that combines precision and recall into a single number, and it is more important than it might sound when normal and attack classes are imbalanced — as they generally are in NSL-KDD.

4. EXPERIMENTAL RESULTS

All four algorithms of SVM, Random Forest, Decision Tree and ANN were trained and tested on NSL-KDD with the pipeline in place from Section III and identical preprocessing and validation steps throughout. The best performance was achieved by Random Forest in the training and testing partitions, with ANN not far behind. Fig. 5 shows all four side by side and Table II gives the exact numbers.

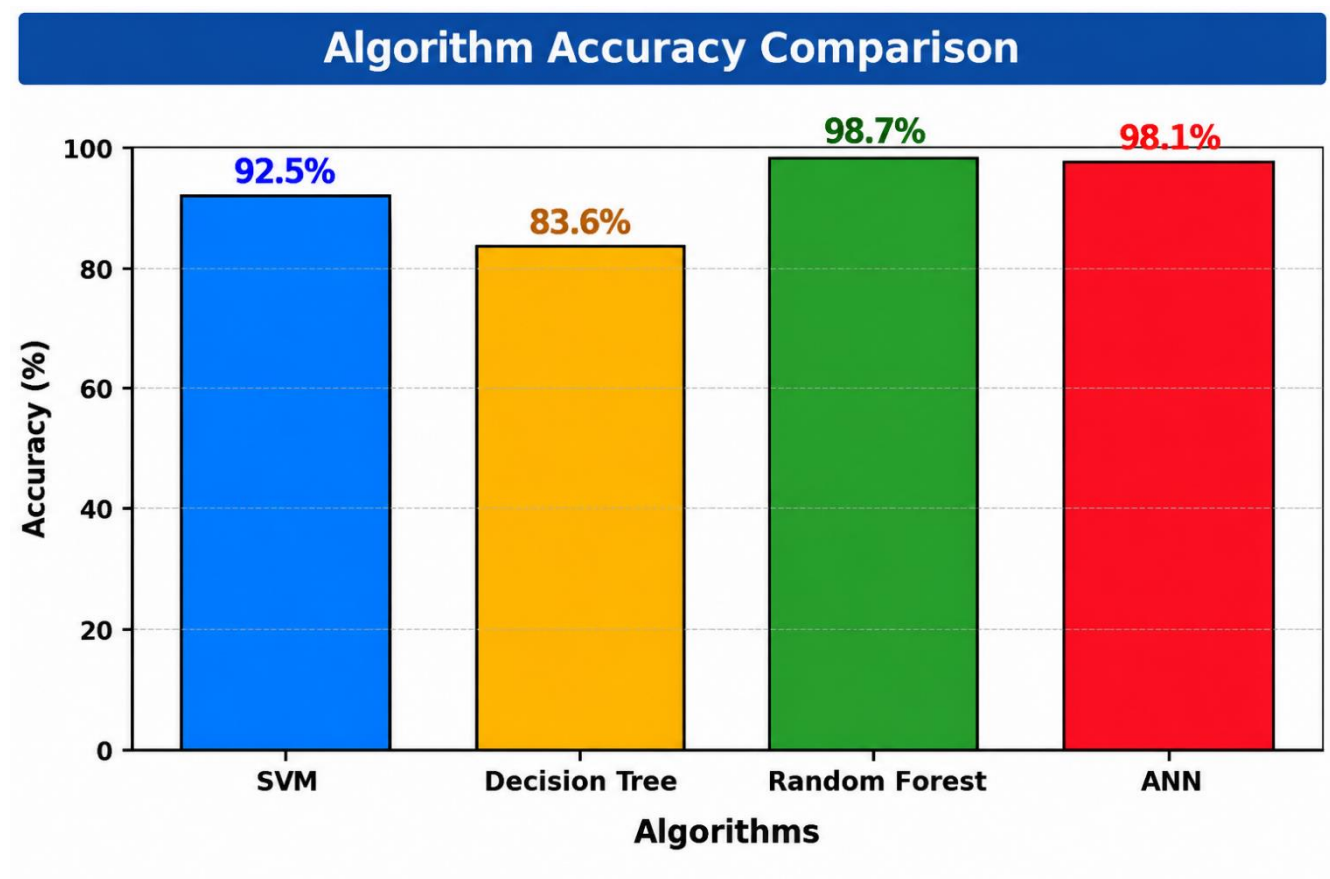


Fig. 5. Comparison of classification accuracy of SVM, Decision Tree, Random Forest and ANN on NSL-KDD test partition.

Table 2. Comparison of Machine-Learning Models and Their Classification Accuracy

Model	Accuracy (%)
Random Forest	99.60
Neural Network (ANN)	99.11
Support Vector Machine	93.29
Decision Tree	83.40

The Random Forest was 99.6% accurate, Artificial Neural Network (ANN) was 99.11%, Support Vector Machine (SVM) was 93.29% and Decision Tree was the lowest at 83.4%. This is not too surprising: ensemble tree methods are supposed to reduce variance relative to a single tree, but still be able to deal with the nonlinear, mostly categorical structure that network connection records tend to have. The difference between a single, independent Decision Tree and a Random Forest of many Trees of Decisions is a good example of why it is better to combine trees than to use a single tree that may be overfit.

Along with the raw accuracy numbers, there was also a small prototype interface built to show what it might actually look like to use these models in practice. Figure 6 shows the input form an operator would use to hand off connection attributes to the trained model, such as connection count, last flag, and connection status.

Network Intrusion Detection System

Attack:
 Gate

Number of connections to the same destination host as the current connection in the past two seconds :
 10

The percentage of connections that were to different services, among the connections aggregated as dst_host_count :
 88.1

The percentage of connections that were to the same source port, among the connections aggregated as dst_host_srv_count :
 87.4

The percentage of connections that were to the same service, among the connections aggregated as dst_host_count :
 93.0

Number of connections having the same port number :
 1

Same as the connection <rsh/ssh or Zero> :
 50

Last Flag :
 1

If successfully logged in, 0 otherwise :
 1

The percentage of connections that were to the same service, among the connections aggregated as count :
 24

The percentage of connections that have activated the flag <S0, S1, S2 or S3>, among the connections aggregated as count :
 38

Destination network service used http or not :
 Yes

Submit

Fig. 6. Sample user-input form used to submit connection attributes to the trained detection model.

Once those parameters are submitted, the model processes the record and returns a predicted class — normal, or whichever attack category the connection most resembles. Fig. 7 shows what that output screen looks like for one such submission.

Random Forest leads the ANN by only 0.49 percentage points, so the two are close. ICMP traffic is heavily skewed towards attacks (Fig. 4), which likely helps tree-based models. Table II reports accuracy only, so per-class precision, recall and F1 would give a fuller picture. The ranking applies to NSL-KDD only and may not hold on datasets like CICIDS2017. Random Forest worked well compared to Decision tree because it have used multiple trees to give the final result which is ensembling.

Network Intrusion Detection System

Attack:

Other ▼

Number of connections to the same destination host as the current connection in the past two seconds :

1

The percentage of connections that were to different services, among the connections aggregated as dst_host_count :

1

The percentage of connections that were to the same source port, among the connections aggregated as dst_host_srv_count :

1

The percentage of connections that were to the same service, among the connections aggregated as dst_host_count :

1

Number of connections having the same port number :

1

State of the connection <Normal or Error> :

SF

Last Flag :

1

If successfully logged in, 0 otherwise :

1

The percentage of connections that were to the same service, among the connections aggregated as count :

1

The percentage of connections that have activated the flag <S0, S1, S2 or S3>, among the connections aggregated as count :

1

Destination network service used http or not :

No

Fig. 7. Sample output screen showing the predicted traffic classification for a submitted connection record.

Fig. 8 shows a second run with a different set of inputs, along with the attack class the system predicted for it.

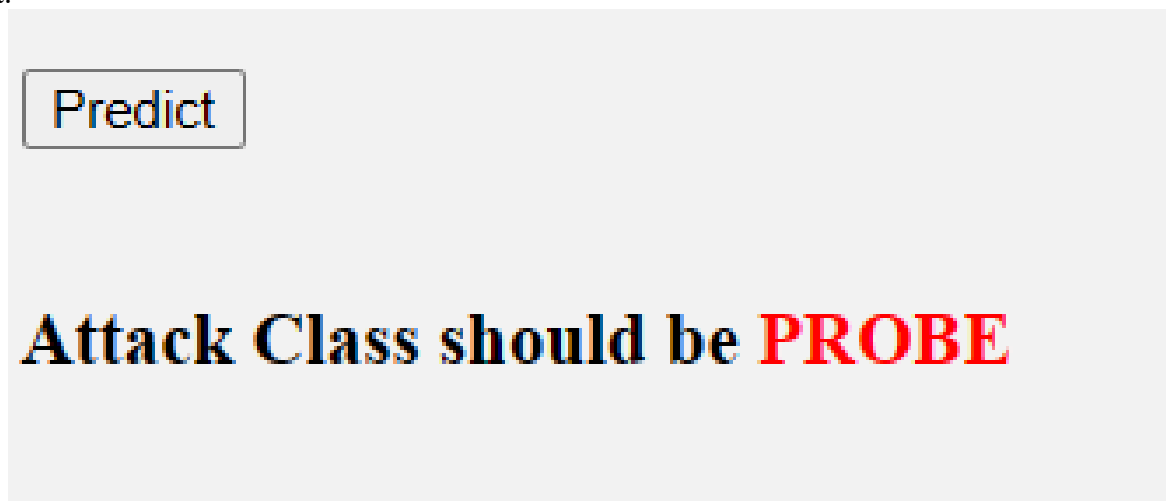


Fig. 8. Predicted attack class returned by the system for the input parameters shown in Fig. 6.

Fig. 9 and Fig. 10 show one more pair like this, again with a different combination of connection attributes producing its own predicted category.

Network Intrusion Detection System

Attack:

Number of connections to the same destination host as the current connection in the past two seconds :

The percentage of connections that were to different services, among the connections aggregated as dst_host_count :

The percentage of connections that were to the same source port, among the connections aggregated as dst_host_srv_count :

The percentage of connections that were to the same service, among the connections aggregated as dst_host_count :

Number of connections having the same port number :

State of the connection <Normal or Error> :

Last Flag :

If successfully logged in, 0 otherwise :

The percentage of connections that were to the same service, among the connections aggregated as count :

The percentage of connections that have activated the flag <S0, S1, S2 or S3>, among the connections aggregated as count :

Destination network service used http or not :

Fig. 9. Output screen obtained when a different combination of input parameters is supplied to the detection system.

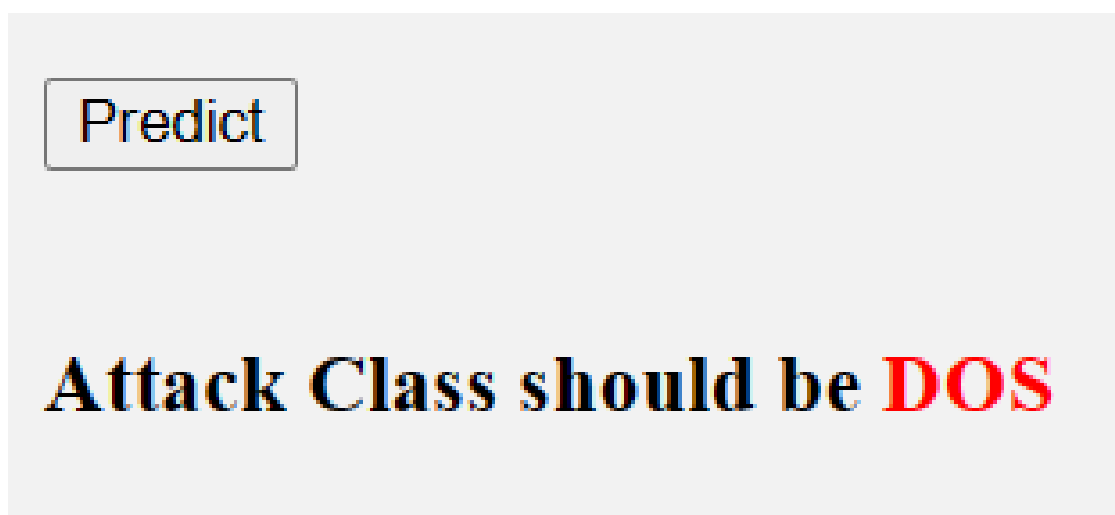


Fig. 10. Predicted attack class corresponding to the input parameters shown in Fig. 9.

What this prototype really demonstrates is that the trained Random Forest model is not just an accuracy number on a slide, but an interactive tool that can be wired up to take a few connection-level attributes and return a specific predicted attack category instead of a plain yes/no alert, which is a much more useful starting point for anyone triaging incidents in a security-operations setting.

5. DISCUSSION

The figures from this study are in fair agreement (not perfect) with what other work using the same or similar data sets has reported. Farnaaz and Jabbar [20] achieved 99.67% accuracy on NSL-KDD using Random Forest after incorporating symmetrical-uncertainty feature selection, which is close enough to the 99.6% obtained here without any explicit feature-selection step to suggest that Random Forest is not that sensitive to redundant features in the first place. Similarly, Ingre and Yadav [30] trained an

ANN on NSL-KDD and achieved a high 99s range comparable to the 99.11% result presented in this work, indicating that feed forward networks are a good choice for this type of classification problem. SVM paints a more complicated picture: the work at CICIDS2017 with this classifier reached a precision as high as 97.80% for one attack category and as low as 69.79% for another [2], a much wider spectrum than the single 93.29% accuracy figure obtained here for NSL-KDD. That gap is actually just the point of Section II-A unfolding in the numbers. Reported IDS performance is heavily dependent on the dataset, attack categories being tested and evaluation protocol being used so comparison of figures across studies needs some caution [10].

Recent work indicates that there is still a lot of room for improvement, primarily by combining classical classifiers and dedicated feature selection or kernel acceleration, rather than replacing the base algorithm. Waghmode et al. [23] combined exhaustive feature selection with a least-squares SVM formulation and got 99.3%, 99.5%, and 93.3% on NSL-KDD, CICIDS2017, and UNSW-NB15 — noticeably higher than the plain SVM result here, which hints that feature selection is often doing more work than the choice of algorithm itself when it comes to squeezing more accuracy out of SVM. Saheed et al. [35] and Ibrahim and Ouaddane [6] make a similar case from different angles: a bio-inspired hybrid method in one case, LDA/PCA in the other, both improving accuracy and lowering false-alarm rate compared to the raw feature set. Taken together, this points to a fairly natural next step for the present work — adding a dedicated feature-selection stage ahead of SVM and Decision Tree training specifically, since Random Forest and ANN already do well enough with the current feature set that there is less obvious upside there.

It is also worth stepping back and asking how these classical algorithms hold up against the growing pile of deep-learning-based IDS work. Shone et al. [12] and Javaid et al. [13] both cleared 97% accuracy with autoencoder-based architectures on KDD99/NSL-KDD, and Aleesa et al. [25] and Psychogios et al. [34] both make the point that deep models can pick up on temporal and sequential attack patterns that a feature-based classifier like the ones used here simply cannot see. That said, deep architectures generally need a lot more training data and compute, and they are harder to interpret than a Random Forest or Decision Tree. Random Forest is already at 99.6% on NSL-KDD with little computational overhead, so it's not clear a deep model would gain much here. That calculus might very well change on larger, messier traffic captures like CICIDS2017 or UNSW-NB15 where the added complexity might actually pay off for itself.

6. CONCLUSION

This paper evaluates the performance of four supervised algorithms such as Random Forest, ANN, SVM and Decision Tree for network intrusion detection on NSL-KDD. All four algorithms were run through the same pipeline: data collection and pre-processing, missing-values handling, exploratory analysis and subsequently training and evaluation. In this way accuracy of each algorithm on the public test partition could be fairly compared. The comparison gives a fair indication of which classifier is best placed for future predictions, in terms of capturing the strongest relationships between connection-level features and attack categories and provides some practical insight into diagnosing what kind of attack a given connection actually represents.

The larger point is that the combination of exploratory data analysis and machine learning classification is a viable approach to building predictive models that reduce the time network operators spend on incident diagnosis and also removes much of the inconsistency inherent in manual inspection. Using NSL-KDD with the feature-engineering and classification pipeline described in Section III, one can reliably identify the type of attack associated with a given connection record, and ensemble methods in particular seem to be good at this.

Out of the four, Random Forest was the best with an accuracy of 99.6% and ANN was not far behind with 99.11%. SVM and Decision Tree had an accuracy of 93.29% and 83.4% respectively. This is a fairly strong case for Random Forest as the most suitable of these four models for detecting cyber-attacks in network traffic. A well-tuned ANN is clearly a competitive alternative rather than a distant runner-up. Following on from the discussion in Section V, there are a few directions it would be worth following up on next: Applying dedicated feature-selection techniques such as those from Ibrahim and Ouaddane [6] and Saheed et al. [35] to see how much they help SVM and Decision Tree

specifically. Extending the evaluation to bigger and newer datasets like CICIDS2017 and UNSW-NB15 [3], [26]. We test hybrid or deep-learning architectures such as autoencoder- and LSTM-based models [11]–[13], [25] to see whether the added complexity is actually justified by better accuracy in a real operational intrusion-detection setting, or simply adds cost with little practical return.

REFERENCES

- [1] D. Reddy, B. T. Sajjan, M. Anusha, S. J. Sadiq, and H. S. Shambulingappa, "Detection of cyber attack in network using machine learning techniques," *International Journal of Advanced Scientific Innovation*, vol. 1, no. 2, pp. 21-25, 2021.
- [2] P. Uyyala, "Detection of cyber attack in network using machine learning techniques," *Journal of Interdisciplinary Cycle Research*, vol. 14, no. 3, pp. 1903-1913, 2022.
- [3] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. Int. Conf. Inf. Syst. Secur. Privacy (ICISSP)*, 2018, pp. 108-116.
- [4] P. A. A. Resende and A. C. Drummond, "Adaptive anomaly-based intrusion detection system using genetic algorithm and profiling," *Security and Privacy*, vol. 1, no. 4, p. e36, 2018.
- [5] H. A. Ahmed, A. Hameed, and N. Z. Bawany, "Network intrusion detection using oversampling technique and machine learning algorithms," *PeerJ Computer Science*, vol. 8, art. e820, 2022.
- [6] K. Ibrahim and M. Ouaddane, "Management of intrusion detection systems based-KDD99: Analysis with LDA and PCA," in *Proc. Int. Conf. Wireless Netw. Mobile Commun. (WINCOM)*, 2017, pp. 1-6.
- [7] R. Christopher, "Port scanning techniques and the defense against them," SANS Institute InfoSec Reading Room, 2001.
- [8] P. Sangkatsanee, N. Wattanapongsakorn, and C. Charnsripinyo, "Practical real-time intrusion detection using machine learning approaches," *Computer Communications*, vol. 34, no. 18, pp. 2227-2235, 2011.
- [9] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153-1176, 2016.
- [10] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Computers & Security*, vol. 86, pp. 147-167, 2019.
- [11] S. Kim and H. K. Kim, "LSTM-based system-call language modeling and robust ensemble method for designing host-based intrusion detection systems," *Pattern Recognition*, vol. 83, pp. 205-217, 2017.
- [12] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41-50, 2018.
- [13] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," in *Proc. 9th EAI Int. Conf. Bio-inspired Inf. Commun. Technol. (BIONETICS)*, 2016, pp. 21-26.
- [14] T. A. Tang, D. McLernon, L. Mhamdi, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," in *Proc. Int. Conf. Wireless Netw. Mobile Commun. (WINCOM)*, 2016, pp. 258-263.
- [15] R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Applying deep learning approaches for network traffic prediction," in *Proc. Int. Conf. Adv. Comput. Commun. Inform. (ICACCI)*, 2017, pp. 2357-2364.
- [16] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5-32, 2001.
- [17] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273-297, 1995.
- [18] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533-536, 1986.
- [19] M. Tavallaei, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symp. Comput. Intell. Secur. Defense Appl. (CISDA)*, 2009, pp. 1-6.

- [20] N. Farnaaz and M. A. Jabbar, "Random Forest modeling for network intrusion detection system," *Procedia Computer Science*, vol. 89, pp. 213-217, 2016.
- [21] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, p. 20, 2019.
- [22] Z. Zoghi and G. Serpen, "Ensemble classifier design tuned to dataset characteristics for network intrusion detection," *arXiv preprint arXiv:2205.06177*, 2024.
- [23] P. Waghmode, M. Kanumuri, H. El-Ocla, and T. Boyle, "Intrusion detection system based on machine learning using least square support vector machine," *Scientific Reports*, vol. 15, art. 12066, 2025.
- [24] M. Ozkan-Okay, R. Samet, O. Aslan, and D. Gupta, "A comprehensive systematic literature review on intrusion detection systems," *IEEE Access*, vol. 9, pp. 157727-157760, 2021.
- [25] A. M. Aleesa, B. B. Zaidan, A. A. Zaidan, and N. M. Sahar, "Review of intrusion detection systems based on deep learning techniques: Coherent taxonomy, challenges, motivations, recommendations, substantial analysis and future directions," *Neural Computing and Applications*, vol. 32, no. 14, pp. 9827-9858, 2020.
- [26] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems," in *Proc. Military Commun. Inf. Syst. Conf. (MilCIS)*, 2015, pp. 1-6.
- [27] D. Stiawan et al., "CICIDS-2017 dataset feature analysis with information gain for anomaly detection," *IEEE Access*, vol. 8, pp. 132911-132921, 2020.
- [28] S. Gibson, B. Issac, L. Zhang, and S. M. Jacob, "Detecting spam email with machine learning optimized with bio-inspired metaheuristic algorithms," *IEEE Access*, vol. 8, pp. 187914-187932, 2020.
- [29] T. Ige, C. Kiekintveld, A. Piplai, A. Zhou, S. Singh, and P. Chowdhury, "An investigation into the performances of the current state-of-the-art naive Bayes, non-Bayesian and deep learning based classifiers for phishing detection: A survey," *arXiv preprint arXiv:2411.16751*, 2024.
- [30] B. Ingre and A. Yadav, "Performance analysis of NSL-KDD dataset using ANN," in *Proc. Int. Conf. Signal Process. Commun. Eng. Syst. (SPACES)*, 2015, pp. 92-96.
- [31] S. Das et al., "Network intrusion detection and comparative analysis using ensemble machine learning and feature selection," *IEEE Transactions on Network and Service Management*, vol. 19, no. 1, pp. 4821-4833, 2021.
- [32] C. Ding, T.-Y. Bao, and H.-L. Huang, "Quantum-inspired support vector machine," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 12, pp. 7210-7222, 2022.
- [33] Y. Yue, X. Chen, Z. Han, X. Zeng, and Y. Zhu, "Contrastive learning enhanced intrusion detection," *IEEE Transactions on Network and Service Management*, vol. 20, no. 2, pp. 1-13, 2022.
- [34] K. Psychogyios, A. Papadakis, S. Bourou, N. Nikolaou, A. Maniatis, and T. Zahariadis, "Deep learning for intrusion detection systems (IDSs) in time series data," *Future Internet*, vol. 16, no. 3, p. 73, 2024.
- [35] Y. K. Saheed, T. O. Kehinde, M. Ayobami Raji, and U. A. Baba, "Feature selection in intrusion detection systems: A new hybrid fusion of bat algorithm and residue number system," *Journal of Information and Telecommunication*, vol. 8, no. 2, pp. 189-207, 2023.
- [36] F. Nabi and X. Zhou, "Enhancing intrusion detection systems through dimensionality reduction: A comparative study of machine learning techniques for cyber security," *Cyber Security and Applications*, vol. 2, art. 100033, 2024.
- [37] Y. Otoum and A. Nayak, "AS-IDS: Anomaly and signature based IDS for the internet of things," *Journal of Network and Systems Management*, vol. 29, art. 6, 2021.
- [38] S. Kumar, S. Gupta, and S. Arora, "Research trends in network-based intrusion detection systems: A review," *IEEE Access*, vol. 9, pp. 157761-157779, 2021.
- [39] P. Hadem, D. K. Saikia, and S. Moulik, "An SDN-based intrusion detection system using SVM with selective logging for IP traceback," *Computer Networks*, vol. 191, art. 108015, 2021.

Authors' Profiles



Dr. A. Sri Nagesh is a Professor at R.V.R. & J.C. College of Engineering, Guntur, India. His research interests include Artificial Intelligence, Quantum Computing, Machine Learning, Image Processing, and Quantum Cryptography. He has authored **36 research publications, including 8 conference papers**, 2 books, and 2 book chapters, and holds 1 granted patent with 6 published patents. He serves on the editorial boards of 5 international journals and has chaired sessions at Faculty Development Programmes. He is a professional member of CSI, ACM, ISTE, and IAENG, contributing to research, academic innovation, and emerging computing technologies.



Samrat Krishna Gaddam is an Assistant Professor in the Department of Computer Science at Parvathaneni Brahmayya Siddhartha College of Arts & Science, Vijayawada, Andhra Pradesh. He is also a Research Scholar at Acharya Nagarjuna University, pursuing doctoral research in Cloud Security under the guidance of Dr. A. Sri Nagesh. His research interests include Cloud Computing, Cyber Security, Network Security, Internet of Things (IoT), Machine Learning, and Blockchain. He has authored several technical textbooks, published research papers, and contributed to innovations in AI-based cyber threat detection. He also serves as the Publisher of the *International Journal of Intelligent Computing Systems* and contributes to digital education through online courses in Prompt Engineering.



K. Bhuvaneswari joined the college in August 2023 and has three years of teaching experience. She completed her M.Sc. in Computer Science from the same college and holds a B.Sc. (Honours) degree. She completed her M.Tech in Computer Science and Engineering (CSE) from RK College of Engineering in June 2026. Her teaching interests include postgraduate courses such as Data Mining Techniques, Web Technologies, Python Programming, Computer Networks, Cloud Computing, and Cryptography & Network Security. She has presented two international and three national conference papers and has participated in five Faculty Development Programmes (FDPs), two international conferences, and two national conferences.