Don Bosco Institute of Technology Delhi Journal of Research
ISSN: 3048-9288

RESEARCH ARTICLE
Vol.3(1) (2026)

An Intelligent Approach to Network Intrusion Detection using Machine Learning

Poonam Rani*

[1]Ph.D Scholar, MVN University, Haryana

***Correspondence:** <mahitaneja0505@gmail.com>

Published: 2026

DOI: https://doi.org/10.48165/dbitdjr.2026.3.01.05

ABSTRACT

Keeping networks secure is a big challenge for today's organizations, especially with cyber threats growing more sophisticated all the time. Traditional signature-based intrusion detection systems aren't enough anymore—they rely on known patterns, so they often miss new or complex attacks. This study introduces a machine learning-based intrusion detection system that spots threats using data-driven analysis instead of just matching signatures. The system sorts network traffic into safe or malicious categories by using multiple machine learning models, like Random Forest, Support Vector Machine, Decision Tree, and K-Nearest Neighbor. These models were trained on well-known benchmarking datasets (CICIDS 2017 and CICIDS 2019), which gives them a solid foundation. To improve accuracy and cut down on false alarms, the system goes through thorough data preparation, careful feature selection, and fine-tuning during training. Plus, the Streamlit platform adds interactive visualizations and motion, making it easier to explore and understand what's happening across the network.

Keywords: Machine Learning; Intrusion Detection Systems; Supervised Learning

INTRODUCTION

The digital world isn't just growing—it's exploded. The internet isn't this extra tool sitting in the corner anymore; it's right at the center of how we live. Think about it: smart cities, cars that drive themselves, wristbands checking your heart rate, your bank living in your pocket. We're plugged in around the clock, and let's face it, screens eat up more of our day than ever before. Sure, all these upgrades make life smoother and people more productive, but they come with a cost. Hackers see opportunity everywhere. They poke around for weak links and go after them, causing chaos for banks, hospitals, really anyone. And when a system gets hit, it doesn't stay local—the mess fans out. Money gets lost fast, and if a company's reputation takes a nosedive, good luck crawling back.

That's why network security isn't optional anymore. It's at the top of everyone's list. Most companies run the basics: set up firewalls, use encryption, keep antivirus humming in the background. It helps, but honestly, none of it is unbreakable. Some bad actors still manage to sneak in.

That's where Intrusion Detection Systems, or IDS, make a difference. Active IDS doesn't just sit there watching; it steps in and blocks suspicious traffic on the spot. Passive IDS is more of a silent observer—it keeps an eye on things, alerts someone if it notices trouble, and then lets the IT team decide what to do. Both have their place. Both are part of the defense. But battling cyberattacks is never easy. There's always a new trick around the corner.

LITERATURE REVIEW

Patcha and Park (2007) [8] carried out a thorough analysis of anomaly detection techniques to spot malicious network activity. According to their findings, anomaly-based intrusion detection systems have the potential to identify attacks that were previously undiscovered by spotting departures from accepted patterns of normal behavior. The study did, however, also note some significant drawbacks, most notably the high false positive rates and the difficulties in accurately simulating typical network behavior. These drawbacks demonstrate the need for more intelligent and adaptable detection techniques, which strongly supports the incorporation of machine learning methods into modern IDS frameworks.

Zhang et al. In-depth Survey on Network Anomaly Detection Techniques for Malicious Network Activity

by (2008) [9]. The work done by them shows that anomaly detection based intrusion detection systems can detect abnormal behavior or deviation from the learned normal patterns, which may lead to the discovery of new attacks. They did also highlight quite serious limitations, most significantly that false positive rates were unacceptably high and we had difficulty modelling normal network behaviour accurately. These limitations provide strong grounds for supplementing modern IDS frameworks with machine learning techniques in order to evolve towards more intelligent and adaptive detection methodologies.

Garcia-Teodoro et al. A comprehensive review of anomaly-based intrusion detection techniques and challenges in implementation (2009) [10]. The authors highlighted some of the major challenges that hinder economics of large-scale network environments in order to implement these IDS including scalability limitations, quality of data and real time processing challenge. They proposed that optimization is a fundamental requirement in constructing effective IDS solutions, with their findings emphasizing the importance of balancing computational efficiency and detection accuracy.

A comprehensive study on anomaly based intrusion detection techniques and their practical issues in deployment can be found by Sommer and Paxson (2010) [11]. The authors pointed out several critical challenges preventing the implementation of intrusion detection systems (IDS) in large Internet environments, such as scalability limitations, data quality issues and real-time processing difficulties. Through their research, they underscored an efficient balance of computational efficiency and detection accuracy, which led to the conclusion that optimization is a fundamental requirement for developing effective IDS solutions.

Buczak and Guven looked at how machine learning is used in intrusion detection systems. They grouped the different methods into supervised, unsupervised, and hybrid categories. Their research shows that supervised learning works well and labels data correctly when there is a lot of training data to work with. They also pointed out that these models are more accurate and efficient if you pick the right features and prepare the data ahead of time. This approach also saves time and cuts down on the processing power the computer needs. Basically, building a good system depends on how well you organize the data and choosing the right algorithm for the job.

In 2016, Kim and his team looked into how deep learning could be used for network intrusion detection. They found that deep neural networks are better than older methods at spotting complex patterns in network traffic, which makes them more accurate. However, they also pointed out several practical problems. These models need a lot of computing power and massive amounts of training data, and it is often hard to understand how they actually make their decisions. Because of these issues, it is difficult to use deep learning in systems that need to work in real time or on hardware with limited resources. This highlights the need for security tools that are both more efficient and easier to explain.

Sharafaldin and his team released the CICIDS2017 dataset in 2018 to help people study intrusion detection. This dataset is a popular choice for training and testing machine learning models because it includes both regular network activity and many different types of modern cyberattacks. Researchers use it often because the traffic is realistic and covers a wide range of scenarios, which makes it easier to get reliable results when evaluating how well detection systems actually work.

Abdulhammed and the research team used the AWID dataset to develop a machine learning system for detecting threats in wireless networks. They found that preprocessing the data—things like scaling and normalizing attributes—was essential for getting the model to perform well. To keep the system fast but accurate, they tested several classifiers and focused on selecting the right features. Their results showed that the Random Forest model worked best when using 32 selected features, achieving an accuracy of 99.64 percent along with high precision and recall scores. When compared to other methods, Random Forest proved to be the most effective option. Ultimately, the study shows that ensemble learning techniques work very well for wireless security, especially when you optimize the features being used.

Ahmad et al. (2021) [16] focused on real-world applications. Instead of traditional rule based models, their findings revealed that machine

learning boosts detection precision while cutting down on false alarms. Yet challenges remain

uneven data distribution continues to skew outcomes, processing demands rise over time, besides

needing constant adjustments as new threats emerge. Because of this, future work must tackle

scalability and flexibility if these systems are to stay effective across evolving environments.

Behind every improvement lies the need for smarter adaptation in the face of shifting attack pattern

METHODOLOGY

To boost network safety using smart traffic checks plus instant threat spotting, the proposed setup came alive as a Machine Learning-powered Intrusion Detection System

(ML-IDS). It sorts online behavior - harmless or harmful - with ease, blending ML tricks and deep traffic scans. From gathering raw inputs through cleaning steps, picking key traits, teaching models, catching intrusions, right down to showing outcomes - the framework moves stepwise in clean blocks.

Data Acquisition

Starting off, the designed setup pulls from standard network attack records

CICIDS2017 along with CICIDS2019 - to cover a broad mix of old and new hacking attempts. Flow length shows up here,packet numbers tag along, different protocols appear, while traffic stats help draw lines between normal activity and harmful actions across those sets.

I. Data Preprocessing

From raw network traffic, data gets shaped so machines can learn from it. Missing pieces fixed, odd entries corrected - this keeps things reliable. Values adjusted, numbers resized to fit a standard range. Categories turned into numbers, making them usable. Extra columns dropped when they add no value. Once cleaned, the set splits - one part trains models, another checks how well they work.

II. Feature Selection and Extraction

Picking the right traits helps spot intrusions more clearly. Some data from network traffic repeats itself, adds little value, or sticks too close to other signals - this can slow models down. Instead of using everything, a tighter set focuses on what matters. Think Flow Duration, how many packets move forward or backward, average packet size, times SYN flags show up, and bytes flowing each second. Behavior clues hidden in normal and harmful traffic come through these numbers. What stands out shapes how well detection work.

III. Model Training

Instead of relying on just one method, four common techniques - Random Forest, SVM, Decision Tree, plus KNN - learn from cleaned-up data to catch intrusions. Because they depend on labeled examples, these models study past network behavior, spotting differences between harmful actions and normal ones through repeated exposure.

IV. Intrusion Detection and Prediction

Once trained, the model spots intrusions and predicts threats as they happen. Incoming network data, along with live warnings from Snort, gets cleaned and shaped before analysis. It pulls key traits from the flow of information, then decides if activity is harmful or safe. This automatic sorting helps catch digital dangers early. Spotting risks quickly strengthens defenses and supports faster reactions when trouble appears.

V. Visualization and Deployment

A web dashboard built with Streamlit shows intrusion detection predictions clearly, updating as things happen. Starting live views of network flow, it moves into showing what each event was classified as. Suspicious patterns trigger automatic warnings - drawing attention without delay. Features unfold one after another: visual tracking comes first, then decisions appear, followed by instant signals when risk appears.

Comparative Model Evaluation

To check how well the proposed system works, several machine learning algorithms were used and compared. These include:

Random Forest –

One way to predict outcomes involves building multiple decision trees instead of just one. Sometimes each tree studies its own chunk of information before sharing conclusions. After every tree has had a turn, they either vote on categories or average out numbers depending on the task. Because everyone contributes separately, errors tend to cancel out over time.

Support Vector Machine –

Most folks call it SVM, even if they're sorting stuff into categories or guessing numbers. Picture a scribble on paper dividing two crowds - that's what this method builds. Distance matters, because wider gaps mean fewer mistakes later on. Performance climbs when the split pushes outward, staying clear of both sides. Better spacing helps it handle surprises down the road.

Decision Tree –

A branching path grows from one choice to another, shaped by data patterns. Starting at the top, a single point splits into directions based on conditions. Each split leads further inward, guided by rules drawn from examples. Paths stretch outward until they stop at endpoints holding answers. Structure unfolds downward, mirroring how people weigh options slowly. Every turn depends on what came just before it. Nodes inside pass judgment before sending flow onward. Final stops give results without going farther. Checks on features show up inside the structure.

•Branches represent values of attributes

•Leaf nodes show final decisions or predictions

K–Nearest Neighbors (KNN) –

K-Nearest Neighbor works by looking at nearby examples when making guesses. Instead of building a model fast, it

waits until it must decide something. When faced with a new case, it checks which past cases are nearest. The prediction comes from those neighbors, counting votes for categories or averaging numbers. This method does not rush to learn - its learning happens late. Memory matters here, since every old point sticks around just in case .It uses every part of the data at once, running numbers just during forecasts instead of adjusting early from examples.

One after another, each model learned from identical data - neatly prepared - to keep results level. Testing followed the same path, so differences showed only what mattered. Accuracy, precision, recall - these shaped how each model was judged through common metrics. F1-score came into play alongside them during evaluation. One way to spot the top algorithm for catching intrusions lies in its classification skill - its reliability matters just as much. What counts? The ability to separate regular traffic from threats stands out clearly. Dependability shifts into focus when results stay consistent. Effectiveness shows up through sharp distinction, not guesswork. A closer look at performance reveals which method handles real-world patterns without failing. Classification accuracy becomes key only if trust holds steady. Clear separation of safe versus risky data defines strong tools.

RESULTS AND DISCUSSION

From start to finish, tests ran on standard data sets - CICIDS2017 along with CICIDS2019 - to check how well the machine learning intrusion detector works. Instead of guessing, it sorted web activity into safe or dangerous by relying on the Random Forest method. Through repeated trials, its skill grew at spotting attack signs while telling apart regular actions from hidden digital risks.

Evaluation Metrics

1. Accuracy

What you get when counting right guesses for regular and harmful web activity together shows accuracy. This number comes from a specific math setup.

Accuracy = (TP + TN) / (TP + TN + FP + FN)

2. Precision

What you get when right guesses about attacks are weighed against total predictions. This number comes from a specific calculation method

3. Recall

Precision = TP / (TP + FP)

What recall shows is how well a model spots real attacks when they happen. This measure comes from applying the following calculation:

4. F1-Score

Recall = TP / (TP + FN)

One way to think about model performance? The F1score blends precision with recall into a

single measure.When data splits unevenly across classes, this score often tells a clearer story than accuracy alone. It comes from a specific calculation - harmonic mean of those two values

Where,

. TP = True Positives,

. TN = True Negatives,

. FP = False Positives,

. FN = False Negatives

F1 = 2 × (Precision × Recall) / (Precision + Recall)

Table 1: Comparison Table of the Algorithms

Algorithm	Metrics			
	Accuracy	Precision	Recall	F1-Score
Random Forest	98.5%	96.2%	94.0%	94.5%
SVM	97.8%	95.5%	90%	89.5%
Decision Tree	96.9%	94.8%	92.5%	86%
KNN	97.3%	95.1%	93.2%	90.5%

Looking at how each model did, Random Forest came out ahead when judging by accuracy and sorting ability. Because it combines many decision trees, it handles tricky network traffic patterns well. This setup reduces the chance of memorizing noise, which helps it work better on new data. Stronger predictions come from its way of balancing detail with simplicity

Though the Support Vector Machine handled classification well, it used more computing power, making it less ideal for instant processing tasks. Faster than others, the Decision Tree came at a price - reduced accuracy because it often learned too much detail from training examples. K-Nearest Neighbors managed decent results yet struggled when numbers grew, demanding more resources and reacting sharply to how values were scaled. From start to finish ,RandomForest handled every measure with steady results, standing out as the go-to choice for the planned security monitoring setup - not flashy, just solid where it counts.

Confusion Matrix Analysis

Looking at how well the top model does, its predictions are laid out in a confusion matrix. This breakdown shows exactly where it gets things right or mixes up categories.

The confusion matrix represents the classification results in terms of True Positives, True Negatives, False Positives,

and False Negatives.

	Predicted Normal	Predicted Attack
Actual Normal	TN = 950	FP = 20
Actual Attack	FN = 30	TP = 900

. **True Positives (TP).** When the system spots harmful actions online and flags them right, those count as true positives. Spotting many means it does well at catching threats.

. **True Negatives (TN):** Most of the time, regular data flow gets labeled just right - harmless activity stays under the radar. When true negatives are high, it means fewer false alarms pop up during routine operations.

. **FalsePositives(FP):** Wrong signals happen if regular activity gets labeled as harmful. Fewer mistakes like this mean better performance, because too many alerts make systems less trustworthy. A smaller number here helps keep things running smoothly instead of triggering unnecessary warnings.

. **False Negatives (FN):**

Failing to catch real breaches shows up here. Missed threats matter because they slip through defenses, putting systems at risk.

Most guesses made by thesystem line up right with reality - plenty oftrue hits on both safe and harmful activity. Mistakes happen, yet they show up less often than accurate calls, which

keeps errors like false alarms or missed threats in check. What stands out is how well it tells

regular behavior apart from attacks - not perfectly, but close enough to matter.Most of the

time, few false alarms mean less busywork for staff. When errors that miss threats stay rare, it show the tool actually catches breakin tries. Taken together, what we saw proves this detection method works well, hits targets consistently, fits smoothly into live networks.What stands out is how well the suggested setup spots network breaches without slowing down. Because it uses TheRandomForest method, sorting through tangled streams of connection data becomes more reliable.Its strength comes from combining many decision paths, which handles messy inputs better thanSinglemodel approaches. Performance jumps when diverse data patterns are processed together instead of in isolation.

Even though results were solid, some flaws showed up. False alarms popped up a bit when data was uneven, possibly muddying alerts. Performance still leans heavily on how good, varied, and fair the training data is. The model only knows what it has seen before.

Even so, this setup could work well right away in today's security setups because it catches

threats without constant fixes.Its strength shows up when attacks shift fast and older tools fall

short.

CONCLUSION

Looking into smarter ways to protect networks, this work builds a system that spots intrusions by studying traffic patterns with machine learning. Using well-known data collections - CICIDS2017 and CICIDS2019 - the setup also pulls in live alerts from Snort to mirror how it might run in the real world. Instead of sticking to one method, several approaches were tested: Random Forest stands out when stacked against others like Support Vector Machine, Decision Tree, or K-Nearest Neighbors. Performance wasn't judged lightly; confusion matrices helped confirm where each model succeeded or stumbled. Accuracy stayed highest, errors lowest, across tests - credit goes to Random Forest for holding steady.

It turns out machine learning pushes intrusion detection much further than older techniques ever did. Because data gets cleaned first, features are fine-tuned, models tested side by side, results become sharper, tougher to break, more dependable. A live view of traffic shows up instantly thanks to a dashboard built with Streamlit - watching threats unfold just got clearer. What stands now is something that works well today, grows when needed, runs without heavy demands.

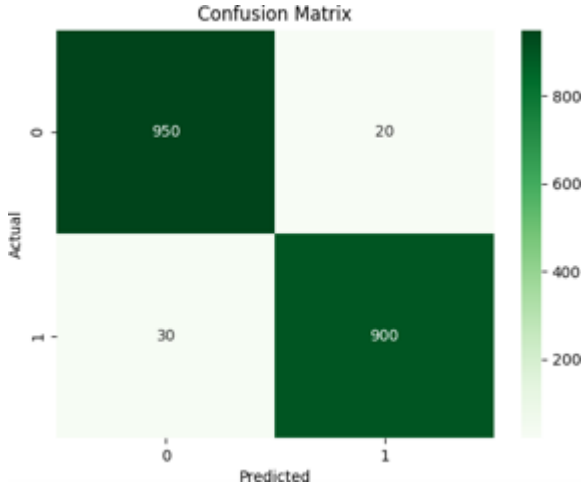

Figure 1. Confusion Matrix

REFERENCES

Levin, I., & Mamlok, D. (2021). Culture and society in the digital age. *Information*, *12*(2), 68. https://doi.org/10.3390/info12020068

Usmani, N. A., Ahmed, T., & Faisal, M. (2022). An IoT-based framework toward a feasible safe and smart city using drone surveillance. In K. Kumar, G. Saini, D. M. Nguyen, N. Kumar,

& R. Shah (Eds.), *Smart cities* (pp. 97–112). CRC Press.

Steinmetz, K. F., Pimentel, A., & Goe, W. R. (2021). Performing social engineering: A qualitative study of information security deceptions. *Computers in Human Behavior, 124*, 106930. https://doi.org/10.1016/j.chb.2021.106930

Ahmad, Z., Khan, A. S., Shiang, C. W., Abdullah, J., & Ahmad, F. (2021). Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Transactions on Emerging Telecommunications Technologies, 32*, e4150. https://doi.org/10.1002/ett.4150

Sarhan, M., Layeghy, S., & Portmann, M. (2022). Towards a standard feature set for network intrusion detection system datasets. *Mobile Networks and Applications, 27*, 357–370. https://doi.org/10.1007/s11036-021-01843-0

Thakkar, A., & Lohiya, R. (2022). A survey on intrusion detection system: Feature selection, model, performance measures, application perspective, challenges, and future research directions. *Artificial Intelligence Review, 55*, 453–563. https://doi.org/10.1007/s10462-021-10037-9

Perháč, J., Novitzká, V., Steingartner, W., & Bilanová, Z. (2021). Formal model of IDS based on BDI logic. *Mathematics, 9*(18), 2290. https://doi.org/10.3390/math9182290

Patcha, A., & Park, J.-M. (2007). Network anomaly detection with incomplete audit data. *Computer Networks, 51*(13), 3935–3955. https://doi.org/10.1016/j.comnet.2007.04.017

Zhang, J., Zulkernine, M., & Haque, A. (2008). Random-forests-based network intrusion detection systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 38*(5), 649–659. https://doi.org/10.1109/TSMCC.2008.923876

García-Teodoro, P., Díaz-Verdejo, J., Maciá-Fernández, G., & Vázquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security, 28*(1–2), 18–28. https://doi.org/10.1016/j.cose.2008.08.003

Sommer, R., & Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. In *2010 IEEE Symposium on Security and Privacy* (pp. 305–316). https://doi.org/10.1109/SP.2010.25

Buczak, A. L., & Guven, E. (2016). A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials, 18*(2), 1153–1176. https://doi.org/10.1109/COMST.2015.2494502

Aminanto, E., & Kim, K. (2016). Deep learning in intrusion detection system: An overview. In *2016 International Research Conference on Engineering and Technology (IRCET 2016)*. Higher Education Forum.

Sharafaldin, I., Habibi Lashkari, A., & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *Proceedings of the International Conference on Information Systems Security and Privacy*.

Abdulhammed, R., Faezipour, M., Abuzneid, A., & Alessa, A. (2018). Effective features selection and machine learning classifiers for improved wireless intrusion detection. In *2018 International Symposium on Networks, Computers and Communications (ISNCC)* (pp. 1–6). https://doi.org/10.1109/ISNCC.2018.8530969

Tait, K., Khan, J. S., Alqahtani, F., Shah, A. A., Khan, F. A., Rehman, M. U., Boulila, W., & Ahmad, J. (2021). Intrusion detection using machine learning techniques: An experimental comparison. In *2021 International Congress of Advanced Technology and Engineering (ICOTEN)* (pp. 1–10).

How to Cite this Article: Rani P.. (2026). An Intelligent Approach to Network Intrusion Detection using Machine Learning. *DBITDJR*, ***3***(1), 44-49. https://doi.org/10.48165/dbitdjr.2026.3.01.05