Abnormal Detection in Network Traffic based on Machine Learning Methodologies

Nadinti Nagoor Vali[1], Munagala Rajeswari[2], Muchalapuri Sai Teja[3]
[1,2,3]*Dept. of CSE (Data Science), Rajeev Gandhi Memorial College of Engineering and Technology, Nandyal, India*

***Abstract*—Internet tech grows fast, so does danger from online attacks. Because companies need safe connections to share info, protection matters a lot. Older threat scanners work by spotting familiar signs, yet they miss anything unfamiliar. New risks slip through easily - that is why finding odd behaviors draws serious attention now.**

What shows up out of place in network flow often breaks typical usage trends. Odd behaviors might point to digital dangers like flood assaults, scanning intrusions, or login breaches. Instead of fixed checklists, smart systems now catch oddities by studying tons of activity records on their own. Learning from real examples helps them spot red flags hidden within regular operations.

This project explores how machines can spot odd patterns in web activity using smart software tricks. Instead of relying on one method, it tests multiple ways to see what works best. One popular collection of internet data helps check if these tools catch strange actions correctly. Outliers pop up when something feels off compared to normal flow. Among those tested, some guess better than others by noticing hidden clues. Each technique handles signals differently - some pay attention to rare events while others weigh frequent signs more heavily. Results show not every model sees threats the same way. What slips past one might get caught by another. Performance shifts depending on how messy or clean the information appears. Spotting trouble early depends less on complexity and more on fit. Some respond quickly; others need time to adjust before catching subtle changes. How they learn matters just as much as what they find.

Missing pieces get filled first. Then duplicates vanish one by one. Categorical items turn into numbers next. Numerical inputs shrink down to a common scale after that. Features thin out through smart filtering along the way. Models learn only once the data settles. Effectiveness shows up in scores like accuracy and precision later. Recall joins F1-score to confirm results stand firm. Each metric tells part of the story during checks.

Outcomes from tests show XG Boost and Light GBM outper- form many alternatives when it comes to prediction accuracy. Because they handle messy, real-world data well, spotting odd behavior in network flows becomes more reliable. When patterns shift over time, these methods adapt without constant reworking by hand. Instead of relying on rigid rules, the approach learns from examples - making detection sharper. Even with noisy inputs, useful signals emerge through repeated modeling steps. One big plus? It works across different kinds of networks without major tweaks. As attacks evolve, so does its ability to tell normal from suspicious. Behind the scenes, decision layers stack up like filters catching subtle clues. Not every model manages this balance between speed and precision - but these do. So, when traffic speeds increase, response times stay low. Detection isn't perfect, yet improvement trends point upward steadily. For now, leaning on smart grouping beats going solo with one algorithm.

***Index Terms*—Network Security, Anomaly Detection, Machine Learning, Intrusion Detection System, KDDCup99 Dataset**

I. INTRODUCTION

Fast internet growth changed how people and groups share info. Because of this shift, computer connections now support daily tasks like talking online, moving files, using web apps, plus buying things digitally. Still, more connectivity brings bigger dangers. Attacks including DDoS floods, system scans, break-ins, along with stolen records threaten systems con- stantly. With these issues growing, staying safe on networks demands close attention from experts who study digital pro- tection methods.

Firewalls and older threat scanners often guard networks, using fixed rules to spot digital dangers. These tools hunt for familiar attack shapes, matching live traffic against stored blueprints of past breaches.

When an assault fits one of those old templates, alarms go off. Yet unfamiliar moves slip through quiet, because nothing in their code rings a bell. As hackers grow smarter, shifting tactics faster than rulebooks update, gaps widen. Smarter defenses must learn on the fly, catching odd behavior before harm spreads.

Now here's a different way it happens: spotting oddities in data flow helps catch sneaky moves inside digital networks. Instead of matching known codes, these tools study how information usually travels and notice when something shifts. When regular rhythms break, trouble might be near - maybe hackers, maybe leaks. Watching nonstop, they raise quiet alarms if things feel off, giving time to step in before harm spreads.

Lately, tools that learn from data have become popular for spotting odd network activity. Instead of following fixed rules, these models pick up on hidden trends by studying tons of past examples. Because they adapt over time, surprises like new attack types might still get caught. Handling massive flows of information feels natural to them, which helps when networks grow busier. Accuracy often goes up since decisions come from patterns rather than guesses. Some well-known approaches - like SVM, Na¨ıve Bayes, tree-based splits, forest ensembles, or boosted sequences - have already found a place inside systems built to flag intrusions. What once felt slow now responds quicker, shaped by how real traffic behaves.

Looking at network traffic, this work tests how well various machine learning methods spot odd patterns. Built around the KDDCup99 collection, known for testing security systems, the analysis digs into loads of connection logs. These logs mix ev- eryday actions with harmful ones - like overwhelming servers, scanning networks, gaining distant access, or escalating local privileges. Instead of just flagging known threats, the models pick up rhythms in regular usage to catch anything offbeat.

This project tests a few machine learning tools - Isolation Forest, Na¨ıve Bayes, SVM, XGBoost, and Light GBM - to spot unusual activity. While one method might struggle with messy inputs, another finds subtle trends in network behavior more easily. XGBoost along with Light GBM stands out because they combine many small models to grasp intricate data links. Their ability to refine predictions comes from layering decisions rather than relying on single rules. What matters most is how well each handles real-world complexity without breaking down.

First off, raw data gets cleaned up before anything else happens. Moving on, labels turn into numbers so computers can understand them. Later, values adjust to fit a common scale, making comparisons fairer across features. Sometimes, less useful columns get removed entirely during analysis prep. With everything set, one portion trains the model while another checks its predictions later. To judge results, people look at correct guesses alongside missed ones carefully each time. Each method shows strength through scores like precision or how often it catches real cases right.

One goal drives this work: building a machine learning tool that spots odd behavior in data flow across networks. Instead of guessing, it uses patterns to catch harmful actions fast. Different algorithms take turns being tested side by side. Each one gets checked for speed, precision, accuracy. Some stumble on false alarms. Others miss subtle threats. This comparison reveals which method handles real-world chaos best. Outcomes show were machine learning fits neatly into safety setups. Not every model adapts well. A few stand out under pressure. Their strengths highlight paths forward. Security teams might use these findings to upgrade defenses quietly. Learning from data beats rigid rules when surprises happen often. Hidden flaws emerge only through repeated trials. Success hides in details others overlook. Progress comes not from theory alone but how tools behave when stressed. The main objectives of this study are:

- To analyze network traffic data and identify abnormal behavior
- To apply machine learning algorithms for anomaly detec- tion
- To compare the performance of different machine, learn- ing models
- To improve the accuracy of intrusion detection systems

II. RELATED WORK

These days, spotting odd behavior in networks gets a lot of attention because hackers are getting more active. Because of rising digital dangers, experts look into different ways to catch suspicious activity fast. One way uses math-based models; another leans on smart software that learns over time. Some tests even try brain-like computing structures to track down strange data flows. The goal stays clear: find intrusions

quicker without missing new kinds of break-in attempts.

Early attempts at spotting network intrusions used statistics alongside preset rules. Traffic behavior got examined then matched against known patterns to flag odd actions. Yet these methods struggled when facing fresh or changing attack types due to reliance on human-built definitions. With digital dangers growing sharper, experts turned toward models that adapt by learning from real-world data instead. Such tools started recognizing unusual activity without needing every threat spelled out ahead of time. Lately, more folks have turned to machine learning for spotting odd behavior in network data simply because it handles massive amounts at once. Instead of relying on traditional rules, models like Decision Trees sort through information by splitting choices step by step. Even though they're basic, these trees often miss subtle signs unless grouped together - Random Forest does exactly that by combining many. Support Vector Machines stand out when dealing with tangled, multi-layered data, drawing sharp boundaries others can't. They thrive where dimensions pile up, making them a go-to for messy real-world traffic patterns. On the flip side, Na¨ıve Bayes takes a guess based on odds, using past likelihoods to label new cases fast. Its speed comes from skipping complex checks, assuming each clue acts alone - even if that's rarely true. Despite that shortcut, it still performs well across different setups without slowing down.

Some methods that mix many smart guesses work well for spotting odd network behavior. Instead of relying on just one tree-shaped model, systems like Random Forest pile them up to get sharper answers while staying balanced. One after another, they learn from mistakes, making each new guess a little wiser than before. What stands out is how tools like XG Boost dig deep into messy data without getting confused by noise. Even when patterns hide behind layers of normal activity, Light GBM finds faint trails others miss.

Lately, some researchers have turned to deep learning to boost how well anomalies are caught. Instead of simpler methods, networks like CNNs pick up layout-based clues in data flows. Temporal trends? RNNs handle those, noticing sequences across time slices. By building intricate internal views of incoming information, these models spot oddities more precisely. Yet heavy computation loads loom training eats both time and hardware strength. That weight sometimes blocks smooth adoption when instant threat spotting matters most.

Researchers often turn to the KDDCup99 data when testing how well new ways can spot odd behavior online. While some work shows certain tools handle patterns better than others on this set, results differ across experiments. Though created years ago, its mix of attack examples still serves a purpose in model trials today. Not every digital threat is covered; yet coverage includes floods that overwhelm systems, stealthy scans, unauthorized access attempts, and privilege escalations. Because these cases appear together, judging if a system catches intrusions becomes possible under varied conditions. Even though work on spotting odd behavior has moved forward, catching tricky and shifting digital dangers remains tough. When one group of examples shows up way more than others, it throws off how well systems learn. Data that carries too many traits adds confusion, just like bits that carry useless or messy signals. Picking the right method matters - so does cleaning things up before feeding them into a model. Getting these steps right make spotting problems more reliable.

A fresh look at spotting odd behavior in data flow tests several smart methods - Isolation Forest, Na¨ıve Bayes, SVM, XGBoost, LightGBM - not lined up side by side but run through real checks. Performance judged on common scoring rules reveals which one handles strange patterns best over time. Instead of guessing, results point toward stronger choices for catching unusual signals across networks.

III. PROPOSED METHODOLOGY

The proposed system aims to detect anomalies in network traffic using machine learning algorithms. The main objective is to identify abnormal behavior within network flows effi- ciently and accurately. The overall process begins with collect- ing raw network data, followed by cleaning and preprocessing the data before extracting useful features. Machine learning models are then trained on the processed dataset to identify patterns that represent normal and abnormal behavior.

The workflow is designed to ensure efficient processing of network traffic data while maintaining high detection accu- racy. The major stages involved

in the proposed methodology include data collection, data preprocessing, feature selection, model training, and model evaluation.

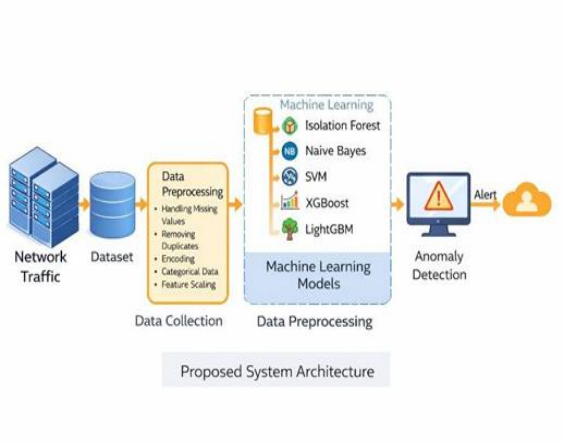

Fig. 1. Proposed methodology for anomaly detection in network traffic

A. Data Collection

The dataset used in this research is the KDDCup99 dataset, which is widely used for intrusion detection research. The dataset contains network traffic records representing both nor- mal activities and different types of cyber-attacks. Each record contains multiple attributes describing network connections such as protocol type, service type, source and destination bytes, and other network characteristics.

The dataset includes various attack categories including Denial of Service (DoS), Probe attacks, Remote to Local (R2L), and User to Root (U2R) attacks. These categories help evaluate the effectiveness of machine learning algorithms in identifying abnormal network behavior.

B. Data Preprocessing

Before applying machine learning algorithms, the dataset undergoes several preprocessing steps to improve data quality and model performance. Missing values are handled and duplicate records are removed from the dataset. Categorical attributes are converted into numerical values using encoding techniques so that machine learning models can process them effectively.

In addition, numerical features are normalized to ensure that all attributes contribute equally during model training. Proper preprocessing improves the reliability of the data and enhances the performance of the anomaly detection system.

Fig. 2. Data preprocessing pipeline

C. Feature Selection

Network datasets often contain a large number of features, many of which may not significantly contribute to anomaly detection. Feature selection techniques are used to identify the most relevant features that influence classification.

Dimensionality reduction techniques such as Principal Com- ponent Analysis (PCA) are applied to reduce the number of features while preserving important information within the dataset. Reducing the feature space improves computational efficiency and minimizes the risk of model overfitting.

D. Model Training

After preprocessing and feature selection, multiple machine learning models are trained to detect anomalies in network traffic. The models used in this study include Isolation Forest, Naive Bayes, Support Vector Machine (SVM), XGBoost, and LightGBM.

Isolation Forest is an unsupervised algorithm designed to isolate anomalies by randomly partitioning the dataset. Na¨ıve Bayes uses probabilistic classification based on Bayes' theo- rem. Support Vector Machine identifies the optimal decision boundary that separates normal and abnormal network behav- ior

Ensemble learning algorithms such as XGBoost and Light- GBM are also used because they combine multiple decision trees to improve prediction accuracy. These algorithms are highly effective in handling large datasets and capturing com- plex relationships within network traffic data

The dataset is divided into training and testing sets, where models

are trained using the training data and evaluated using unseen testing data

E. Model Evaluation

The performance of machine learning models is evaluated using common evaluation metrics such as

accuracy, precision, recall, and F1-score. Accuracy measures the overall correct- ness of the model in classifying network traffic. Precision evaluates how many predicted anomalies are actually correct. Recall measures the ability of the model to detect actual attacks in the dataset.

The F1-score provides a balanced evaluation by combin- ing precision and recall into a single metric. By comparing these metrics across different algorithms, the proposed system identifies the most effective model for detecting anomalies in network traffic and improving network security monitoring.

The following machine learning models are used for anomaly detection:

- Isolation Forest
- Naive Bayes
- Support Vector Machine
- XGBoost
- LightGBM

Each model is trained using the processed dataset and evaluated using multiple performance metrics.

IV. EXPERIMENTAL SETUP

The experiments were conducted using Python program- ming language with libraries such as NumPy, Pandas, Scikit- learn, XGBoost, and LightGBM. The dataset was divided into training and testing sets using an 80:20 ratio.

Cross-validation was used to ensure the reliability of the models. Hyperparameter tuning was performed to optimize model performance.

V. EVALUATION METRICS

The performance of the models is evaluated using the following metrics.

A. Accuracy

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

B. Precision

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

C. Recall

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

D. F1 Score

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (4)$$

VI. RESULTS AND DISCUSSION

To check how well the new anomaly detection method works, different machine learning approaches took part - Isola- tion Forest stepped in, followed by Naive Bayes joining later. SVM showed up alongside XGBoost, while LightGBM com- pleted the group. Testing happened on real network activity logs pulled from the KDDCup99 collection, filled with regular actions plus signs of attacks. Splits between practice rounds and final checks kept things balanced across experiments. Earlier steps shaped the raw information first: messy entries got fixed before labels turned into numbers. Values lined up algorithms.

Table I Model Performance Comparison

Model	Accuracy	Precision	Recall	F1 Score
Isolation Forest	0.40	0.21	0.40	0.28
Naive Bayes	0.81	0.83	0.81	0.81
XGBoost	0.83	0.87	0.84	0.84
LightGBM	0.85	0.88	0.85	0.85
SVM	0.85	0.88	0.86	0.86

VIII. CONCLUSION

through scaling procedures. Important traits stood out after sorting features carefully. Each step aimed at boosting what models could catch once running.

Not every test shows the whole picture, yet these numbers give solid clues about how well the system spots unusual activity. What matters most is whether it flags real threats without flooding results with false alarms. One number track total right answer across everything tried. Other zooms in on only the flagged risks - asking if those were truly dangerous. Sometimes a model misses hidden dangers lurking below the surface, which the detection rate exposes clearly. Combining two key views gives one score meant to balance sensitivity against reliability. Each figure adds context where others fall short.

One standout was how ensemble approaches like XGBoost and LightGBM outperformed older techniques across tests. Not limited by simpler patterns, these models handled tangled data structures

without slowing down. Top marks went to LightGBM - its precision led the pack when spotting odd behavior in network flow. Close behind came XGBoost, holding steady with solid scores and sharp identification skills. Still, both proved stronger than conventional options under similar conditions.

A strong ability to spot curved trends marked the Support Vector Machine's work, delivering consistent labels across the data while relying on complex boundaries. Probability drives Na¨ıve Bayes, its straightforward design allowing quick decisions without heavy computation slowing it down. Unusual entries stood out clearly when Isolation Forest was applied, even though oversight during training limited how close predictions came to actual outcomes.

Looking at everything together, it seems combining sev- eral machine learning methods works well for spotting odd behavior in network activity. Because they work as a group, these models notice subtle clues others might miss. Instead of acting alone, each part adds something useful to finding digital risks faster. Picking one method over another makes a difference when setting up tools meant to catch break-ins. Some approaches handle real-world complexity better than simpler ones do. When accuracy matters, choosing wisely shapes how trustworthy the system becomes. Table I shows the comparison of different models.

The results show that ensemble models such as XGBoost and LightGBM provide better performance compared to other

This work introduced a method using machine learning to spot unusual behavior in network activity. Instead of just tracking normal flow, it looked closely at digital traffic to catch signs of possible attacks. Rather than relying on one model, several were tested - Isolation Forest, Na¨ıve Bayes, SVM, XGBoost, and LightGBM among them. Performance checks happened with the well-known KDDCup99 collection of network records.

Out of raw numbers came cleaner patterns once scrubbing, labeling, scaling, also picking key traits shaped the set. Once fixed up, algorithms learned on splits then checked their work through right guesses, tight hits, caught misses, even balance scores.

Out of all tested methods, XGBoost and LightGBM stood out when spotting odd patterns in network data. What made them work well was their knack for handling tangled con- nections inside information flows. Though less flashy, the Support Vector Machine still held its ground when sorting normal from suspicious activity. Meanwhile, Na¨ıve Bayes kept things moving fast without demanding heavy computing power. Performance gains came not just from precision but how smoothly each model ran under real conditions.

It turns out these results show how machine learning boosts anomaly detection in network safety. Looking at data flows helps spot odd actions before harm spreads. Early warnings come alive when models learn normal rhythms across digital pathways. Strange signals get flagged faster than traditional methods manage. Cyber dangers face stronger resistance where algorithms adapt continuously. Detection grows sharper with each pattern shift observed. Networks gain resilience through constant behavioral tracking. Learning curves rise as systems absorb new threat signs.

Finding better ways later might mean trying smart algo- rithms that learn patterns on their own, while watching live data flows closely. Another path opens when fresh and varied network examples are used instead of old ones. Stronger defenses could grow from mixing wide-ranging information sources into the training process. Modern networks demand systems that handle growth without breaking down easily.

ACKNOWLEDGMENT

The authors would like to thank Rajeev Gandhi Memorial College of Engineering and Technology for providing support and resources for this research work.

REFERENCES

[1] T. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD Cup 99 dataset," in *Proc. IEEE Symp. Computational Intelligence for Security and Defense Applications (CISDA)*, 2009, pp. 1–6.

[2] Dua and C. Graff, "UCI machine learning repository," University of California, Irvine, CA, USA, 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[3] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[4] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. ACM*

SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD), 2016, pp. 785–794.

[5] Ke *et al*., "LightGBM: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 3146–3154.