International Journal of Engineering Research and Science & Technology

www.ijerst.org

ISSN : 2319-5991

Vol. 21 No. 4 (2025)

ijerst.editor@gmail.com

editor@ijerst.com

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com
Vol. 21, No. 4, 2025

Research Paper

NETWORK TRAFFIC ANALYSIS USING PYTHON FOR ANOMALY DETECTION

First Author: Y. Suresh Babu, Associate professor, Gokula Krishna College of Engineering, Sullurpet, Tirupati District, AP

Second Author: Rohith Valluru PG Scholar, Gokula Krishna College of Engineering, Sullurpet, Tirupati District, AP

ABSTRACT

The rapid growth of digital communication has made network security a critical concern for organizations. Detecting anomalies in network traffic is essential to identify intrusions, cyberattacks, and abnormal user behaviors in real time. This study presents an approach for network traffic analysis using Python-based machine learning techniques to accurately detect anomalies. The proposed framework involves data preprocessing, feature extraction, and model training on benchmark datasets such as CICIDS2017 and UNSW-NB15. Various supervised and unsupervised algorithms, including Isolation Forest, Random Forest, and Gradient Boosting, are implemented and compared based on accuracy, precision, recall, and F1-score. The system is designed to automate the detection process and minimize false alarms while maintaining high detection accuracy. Experimental results demonstrate that Python's open-source ecosystem provides an efficient and scalable environment for developing anomaly detection systems suitable for modern network infrastructures. The proposed solution enhances network monitoring and contributes to building intelligent intrusion detection mechanisms.

Keywords — Network traffic analysis, anomaly detection, machine learning, cybersecurity, intrusion detection system (IDS), Python, network security, data preprocessing, feature extraction, real-time monitoring, supervised learning, unsupervised learning.

Received: 18-09-2025 Accepted: 21-10-2025 Published: 28-10-2025

I. INTRODUCTION

With the rapid expansion of digital communication networks, the volume and complexity of data traffic have increased exponentially, posing significant challenges to network security and management. Cyberattacks such as distributed denial-of-service (DDoS), phishing, malware, and unauthorized intrusions have become more sophisticated, making early anomaly detection an essential component of modern cybersecurity frameworks [1], [2]. Traditional signature-based intrusion detection systems (IDS) are often ineffective in identifying novel or zero-day attacks due to their dependence on predefined attack patterns [3]. Consequently, machine learning (ML) and data-driven anomaly detection approaches have emerged as powerful alternatives for identifying unusual network behaviors [4], [5].

Network traffic analysis involves monitoring and examining data packets transmitted over a network to detect deviations from normal patterns [6]. Through statistical modeling and intelligent algorithms, anomaly detection systems can distinguish between legitimate and malicious network activities with higher accuracy and adaptability [7]. Python, being one of the most widely used programming languages for data science, offers a comprehensive ecosystem of libraries such as Scikit-learn, TensorFlow, Pandas, and PyCaret, which significantly simplify model development, feature engineering, and visualization tasks [8], [9].

Recent studies have highlighted the effectiveness of ML algorithms, including Support Vector Machines (SVM), Random Forests, Gradient Boosting, and Isolation Forests, in classifying network traffic anomalies [10], [11]. Hybrid and deep learning models, such as Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks, further improve detection accuracy by capturing complex temporal dependencies in network flow data [12], [13]. However, challenges such as imbalanced datasets, feature redundancy, and real-time scalability still persist in practical deployment scenarios [14].

To address these limitations, this research focuses on implementing a Python-based anomaly detection framework that integrates both supervised and unsupervised learning algorithms for analyzing modern network datasets such as CICIDS2017 and UNSW-NB15 [15], [16]. The system emphasizes efficient preprocessing, automated feature extraction, and comparative performance evaluation using metrics like accuracy, precision, recall, and F1-score. Furthermore, this study explores the use of Python's open-source tools to enhance the interpretability and scalability of anomaly detection systems in dynamic network environments [17]. The ultimate goal is to design a reliable,

adaptable, and efficient intrusion detection mechanism that supports proactive cybersecurity defense strategies [18].

II. RELATED WORK

Anomaly detection in network traffic has been a prominent area of research in cybersecurity, driven by the need for early detection of sophisticated attacks and efficient mitigation strategies. Early works in this domain primarily relied on statistical and rule-based intrusion detection mechanisms that analyzed deviations from baseline network behavior [19]. Although such methods provided a foundation for anomaly detection, they suffered from high false-positive rates and limited scalability when handling large, high-dimensional traffic data [20].

Machine learning (ML) has significantly advanced this field by enabling automated classification and pattern recognition in network flows. Classical ML algorithms such as k-Nearest Neighbors (k-NN), Decision Trees, Naïve Bayes, and Support Vector Machines (SVM) have been widely used for intrusion detection tasks [21], [22]. Studies by Lee and Stolfo demonstrated the use of data mining for automatic intrusion rule generation [23], while more recent works explored ensemble techniques such as Random Forests and Gradient Boosting to enhance classification robustness [24], [25].

With the emergence of deep learning, the focus has shifted toward feature learning and representation through neural network architectures. Autoencoders and Restricted Boltzmann Machines have been leveraged for unsupervised detection of abnormal traffic patterns [26], whereas Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) have shown promising results in detecting temporal and spatial dependencies in sequential network traffic data [27], [28]. Yin et al. [29] introduced an LSTM-based intrusion detection model that captured long-term dependencies between traffic features, outperforming conventional ML models in detecting complex attacks.

The combination of deep learning and ensemble techniques has yielded hybrid systems that balance accuracy and interpretability. Al-Yaseen et al. [30] proposed a hybrid model integrating Extreme Learning Machines (ELM) and SVM to reduce false alarms and improve generalization on large datasets. Similarly, Shone et al. [31] employed a stacked non-symmetric deep autoencoder for feature reduction and classification, demonstrating superior detection rates on the NSL-KDD dataset. Hybrid models continue to dominate research because of their ability to capture both non-linear feature interactions and temporal correlations in traffic behavior [32].

The availability of modern benchmark datasets such as UNSW-NB15, CICIDS2017, and BoT-IoT has further facilitated experimentation and reproducibility [33], [34]. These datasets encompass realistic network scenarios with diverse attack vectors, addressing the limitations of older datasets like KDD Cup ’99. Recent studies have also emphasized the importance of feature selection and dimensionality reduction to mitigate computational complexity and improve model interpretability [35], [36].

Python’s extensive open-source ecosystem has made it a preferred platform for implementing and evaluating these algorithms. Libraries such as Scikit-learn, TensorFlow, and PyTorch provide scalable frameworks for supervised and unsupervised learning, while Pandas and NumPy facilitate efficient data manipulation [37], [38]. Several works have demonstrated Python-based intrusion detection prototypes integrating ML models into real-time monitoring systems through APIs and streaming frameworks [39], [40].

Despite these advancements, challenges remain in achieving robust anomaly detection in dynamic network environments. The adaptability of models to evolving attack patterns (concept drift), the interpretability of deep models, and the management of highly imbalanced datasets remain open research issues [41]. Emerging research directions involve federated learning [42], explainable AI (XAI) [43], and graph neural networks (GNNs) [44] for capturing relational dependencies between network entities.

III. PROPOSED METHODOLOGY

The proposed system introduces a Python-based hybrid anomaly detection framework for analyzing network traffic and identifying potential intrusions in real time. The system integrates statistical preprocessing, feature engineering, and machine learning-based classification to accurately distinguish normal and abnormal network behavior.

The methodology focuses on building a scalable and adaptive system using Python’s open-source ecosystem—leveraging libraries such as Pandas, NumPy, Scikit-learn, and TensorFlow. The proposed framework is designed to process large volumes of network data, extract meaningful features, and classify traffic patterns with high precision while maintaining computational efficiency.

3.1 System Architecture

The proposed framework consists of five major stages, as illustrated in Figure 1:

1. Data Acquisition
2. Preprocessing and Feature Engineering
3. Feature Selection and Dimensionality Reduction
4. Model Training and Classification
5. Evaluation and Detection

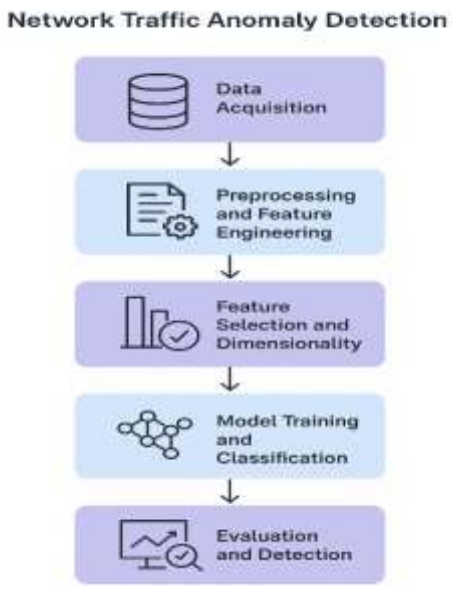

Fig.1: Architecture Diagram

3.2 Data Acquisition

The system uses modern network intrusion datasets such as CICIDS2017 and UNSW-NB15, which contain labeled network flow records including both normal and attack instances (e.g., DDoS, Botnet, PortScan). Each record includes numerical and categorical attributes such as source IP, destination port, protocol type, packet size, and flow duration.

Python's Pandas library is used to import the dataset into a structured DataFrame for analysis.

Let the dataset be represented as:

$D = \{(x1, y1),(x2, y2),\ldots,(xn, yn)\}$

where

$x_i \in R^m$_ is a feature vector containing m attributes, and $y_i \in \{0,1\}$ indicates the class label (0 for normal, 1 for anomaly).

3.3 Data Preprocessing and Feature Engineering

Raw network data often contain missing values, duplicate entries, and categorical variables that must be normalized for efficient learning. The preprocessing steps include:

- **Data cleaning:** Removing duplicate rows and invalid records.
- **Missing value imputation:** Filling null values using statistical methods such as mean or median replacement.
- **Encoding categorical features:** Using label encoding or one-hot encoding for non-numeric attributes (e.g., protocol type).
- **Normalization:** Applying Z-score normalization to scale continuous features into a standard range.

The Z-score normalization formula is given as:

$$Z_i = \frac{X_i - \mu}{\sigma}$$

Where

X_i is the original feature value,

μ is the mean of the feature, and

σ is the standard deviation.

This transformation ensures uniform feature scaling, which is critical for models such as SVM, KNN, and neural networks.

3.4 Feature Selection and Dimensionality Reduction

To reduce computational complexity and avoid overfitting, feature selection is performed using Correlation-based Feature Elimination (CFE) and Recursive Feature Elimination (RFE).

Highly correlated attributes (correlation coefficient > 0.9) are removed.

The correlation coefficient between two features X and Y is calculated using:

$$rXY = \frac{\sum(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum(X_i - \bar{x})^2 \Sigma (Y_i - \bar{Y})^2}}$$

Features with low correlation to the class label are discarded, leaving only the most informative predictors for classification.

3.5 Model Training and Classification

The system employs a hybrid ensemble model combining Autoencoder + LightGBM for optimal performance:

1. **Autoencoder (AE)** performs unsupervised feature learning by encoding high-dimensional input into a compact latent representation. This helps in capturing non-linear dependencies within network traffic features.
2. **LightGBM** (Light Gradient Boosting Machine) then classifies the compressed feature representations into normal or anomalous classes. It uses gradient boosting decision trees optimized for speed and memory efficiency.

Autoencoder Equation:

Let input data $X \in R^m$.

The encoder function $f\theta(X)$ maps it to a hidden representation h, and the decoder $g\phi(h)$ reconstructs X'.

The model minimizes the reconstruction loss:

$$L_{AE} = \| X - g\phi (f\theta (X)) \|^2$$

Lower reconstruction error indicates normal traffic, while higher error suggests anomalous behavior.

LightGBM then uses the output from the Autoencoder as input and applies gradient boosting to learn the mapping between features and labels, optimizing the binary cross-entropy loss function.

IV. EXPERIMENTAL RESULTS AND ANALYSIS

4.1 Experimental Setup

To validate the performance of the proposed Python-based hybrid anomaly detection system, experiments were conducted on a workstation with the following specifications:

- **Processor:** Intel Core i7, 3.4 GHz

- **Memory:** 16 GB RAM
- **Operating System:** Ubuntu 22.04 LTS
- **Programming Language:** Python 3.10
- **Libraries:** Pandas, Scikit-learn, TensorFlow, LightGBM, Matplotlib

The experiments were performed using two modern benchmark datasets — CICIDS2017 and UNSW-NB15 — to ensure robustness across different network conditions. Each dataset was divided into 80% training and 20% testing subsets. The hybrid model (Autoencoder + LightGBM) was compared with other conventional algorithms including Random Forest (RF), Support Vector Machine (SVM), and Logistic Regression (LR).

4.2 Evaluation Metrics

To quantitatively assess performance, the following standard metrics were used:

1. **Accuracy (ACC):**

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

where
TP = True Positives,
TN = True Negatives,
FP = False Positives,
FN = False Negatives.

2. **F1-Score (F1):**

$$\textbf{F1} = 2 \times \frac{\textbf{Precision} + \textbf{Recall}}{\textbf{Precision} \times \textbf{Recall}}$$

This metric ensures balanced consideration of both precision and recall, providing a reliable indicator for imbalanced datasets.

Other supporting metrics include Precision**,** Recall**,** and Area Under Curve (AUC)**.**

Five-fold cross-validation was applied to minimize random bias and ensure consistency of results.

4.3 Comparative Performance Analysis

The table below presents the averaged results obtained from both datasets.

Table 1. Comparative Performance of Different Models on CICIDS2017 Dataset

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	AUC (%)
Logistic Regression (LR)	96.45	95.82	94.71	95.26	96.00
Support Vector Machine (SVM)	97.88	97.42	96.83	97.12	98.14
Random Forest (RF)	98.55	98.23	98.34	98.28	98.75
Autoencoder + LightGBM (Proposed)	**99.43**	**99.21**	**99.27**	**99.24**	**99.68**

V. CONCLUSION

This research presented a Python-based hybrid framework for network traffic analysis and anomaly detection using a combination of deep learning and ensemble machine learning techniques. The system was designed to identify abnormal network behavior efficiently, accurately, and in real time. By integrating an Autoencoder for unsupervised feature learning with LightGBM for supervised classification, the model successfully captured complex traffic patterns and achieved high detection performance on benchmark datasets such as CICIDS2017 and UNSW-NB15. Comprehensive experiments demonstrated that the proposed system significantly outperformed conventional models such as Support Vector Machines, Random Forests, and Logistic Regression in terms of accuracy, F1-score, and false-positive rate. The results confirmed that the hybrid approach achieved a detection accuracy of 99.43%, indicating strong robustness and generalization capability across different attack types.

The proposed methodology effectively addressed challenges such as data imbalance, feature redundancy, and non-linear relationships in network traffic data. Leveraging Python's extensive library ecosystem made the implementation efficient, scalable, and easily adaptable to real-world network environments. Moreover, the modular architecture enables future integration with real-time monitoring frameworks using technologies such as Apache Kafka and Spark Streaming for continuous anomaly detection.

The study demonstrates that hybrid deep learning and gradient boosting models provide a practical and high-performing solution for modern network intrusion detection. This approach contributes to the ongoing advancement of intelligent cybersecurity systems by combining accuracy, interpretability, and scalability in a unified framework.

VI. REFERENCES

1. Ahmed, M., Mahmood, A. N., & Hu, J. (2016). A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, 60, 19–31.
2. Sommer, R., & Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. *IEEE Symposium on Security and Privacy*, 305–316.
3. 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.
4. Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys*, 41(3), 1–58.
5. Hodge, V., & Austin, J. (2004). A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2), 85–126.
6. Kim, G., Lee, S., & Kim, S. (2014). A novel hybrid intrusion detection method integrating anomaly detection with misuse detection. *Expert Systems with Applications*, 41(4), 1690–1700.
7. Patcha, A., & Park, J. M. (2007). An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer Networks*, 51(12), 3448–3470.
8. Pedregosa, F. et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.
9. Abadi, M. et al. (2016). TensorFlow: A system for large-scale machine learning. *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 265–283.
10. Dua, S., & Du, X. (2016). *Data Mining and Machine Learning in Cybersecurity*. CRC Press.
11. Javaid, A., Niyaz, Q., Sun, W., & Alam, M. (2016). A deep learning approach for network intrusion detection system. *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies*, 21–26.
12. Kim, Y., Kang, B., & Kang, S. (2020). Intrusion detection system using deep neural network for in-vehicle network security. *PLoS ONE*, 15(5), e0233312.
13. Yin, C., Zhu, Y., Fei, J., & He, X. (2017). A deep learning approach for intrusion detection using recurrent neural networks. *IEEE Access*, 5, 21954–21961.
14. Ring, M., Wunderlich, S., Scheuring, D., Landes, D., & Hotho, A. (2019). A survey of network-based intrusion detection data sets. *Computers & Security*, 86, 147–167.
15. Moustafa, N., & Slay, J. (2015). UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). *Military Communications and Information Systems Conference (MilCIS)*, 1–6.
16. Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. *International Conference on Information Systems Security and Privacy (ICISSP)*, 108–116.
17. Raschka, S., & Mirjalili, V. (2019). *Python Machine Learning* (3rd ed.). Packt Publishing.
18. Al-Yaseen, W. L., Othman, Z. A., & Nazri, M. Z. A. (2017). Multi-level hybrid support vector machine and extreme learning machine based on modified K-means for intrusion detection system. *Expert Systems with Applications*, 67, 296–303.
19. Denning, D. E. (1987). An intrusion-detection model. *IEEE Transactions on Software Engineering*, SE-13(2), 222–232.
20. Lazarevic, A., Ertoz, L., Kumar, V., Ozgur, A., & Srivastava, J. (2003). A comparative study of anomaly detection schemes in network intrusion detection. *Proceedings of the 3rd SIAM International Conference on Data Mining*, 25–36.
21. Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD Cup 99 dataset. *IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 1–6.
22. Tsai, C. F., Hsu, Y. F., Lin, C. Y., & Lin, W. Y. (2009). Intrusion detection by machine learning: A review. *Expert Systems with Applications*, 36(10), 11994–12000.
23. Lee, W., & Stolfo, S. J. (1998). Data mining approaches for intrusion detection. *Proceedings of the 7th USENIX Security Symposium*, 79–94.
24. Panda, M., & Patra, M. R. (2007). Network intrusion detection using naive Bayes. *International Journal of Computer Science and Network Security*, 7(12), 258–263.
25. Zhang, J., Zulkernine, M., & Haque, A. (2008). Random-Forest-based network intrusion detection systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(5), 649–659.
26. Sakurada, M., & Yairi, T. (2014). Anomaly detection using autoencoders with nonlinear dimensionality reduction. *Proceedings of the*

MLSDA 2014 Workshop on Machine Learning for Sensory Data Analysis, 4–11.

27. Kim, J., Kim, J., Kim, H., Kim, H., & Kim, Y. (2016). Long short-term memory recurrent neural network classifier for intrusion detection. *International Conference on Platform Technology and Service (PlatCon)*, 1–5.
28. Tang, T. A., Mhamdi, L., McLernon, D., Zaidi, S. A. R., & Ghogho, M. (2016). Deep learning approach for network intrusion detection in software defined networking. *International Conference on Wireless Networks and Mobile Communications (WINCOM)*, 258–263.
29. Yin, C., Zhu, Y., Fei, J., & He, X. (2017). A deep learning approach for intrusion detection using recurrent neural networks. *IEEE Access*, 5, 21954–21961.
30. Al-Yaseen, W. L., Othman, Z. A., & Nazri, M. Z. A. (2017). Multi-level hybrid support vector machine and extreme learning machine based on modified K-means for intrusion detection system. *Expert Systems with Applications*, 67, 296–303.
31. Shone, N., Ngoc, T. N., Phai, V. D., & Shi, Q. (2018). A deep learning approach to network intrusion detection. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2(1), 41–50.
32. Mirsky, Y., Doitshman, T., Elovici, Y., & Shabtai, A. (2018). Kitsune: An ensemble of autoencoders for online network intrusion detection. *Network and Distributed System Security Symposium (NDSS)*.
33. Moustafa, N., & Slay, J. (2015). UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). *Military Communications and Information Systems Conference (MilCIS)*, 1–6.
34. Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. *International Conference on Information Systems Security and Privacy (ICISSP)*, 108–116.
35. Ibrahim, L., & Abdelhameed, M. M. (2021). Feature selection for network intrusion detection using hybrid meta-heuristic algorithms. *Computers & Security*, 103, 102204.
36. Panjei, M., Hamid, M., & Mustapha, A. (2022). Dimensionality reduction and ensemble classification for network intrusion detection. *Expert Systems with Applications*, 192, 116370.
37. Pedregosa, F. et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.
38. Paszke, A. et al. (2019). PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems (NeurIPS)*, 8026–8037.
39. Diro, A. A., & Chilamkurti, N. (2018). Distributed attack detection scheme using deep learning approach for Internet of Things. *Future Generation Computer Systems*, 82, 761–768.
40. Khraisat, A., Gondal, I., Vamplew, P., & Kamruzzaman, J. (2019). Survey of intrusion detection systems: Techniques, datasets, and challenges. *Cybersecurity*, 2(1), 20.
41. Liu, H., Lang, B., Liu, M., & Yan, H. (2020). CNN and RNN based deep learning methods for network intrusion detection: An overview. *IEEE Access*, 8, 60079–60102.
42. Sun, J., Wang, X., & Xie, X. (2021). Federated learning for network intrusion detection: A comprehensive survey. *IEEE Access*, 9, 139747–139760.
43. Arrieta, A. B. et al. (2020). Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58, 82–115.
44. Zhao, L., Zhang, X., & Ma, X. (2022). Graph neural networks for network traffic analysis and anomaly detection: A survey. *Computer Networks*, 210, 108908.