Article citation info:
Omonov I. Analysis of machine learning models in remote diagnostics of network elements. Diagnostyka. 2026;27(1):2026108.
https://doi.org/10.29354/diag/219252.

DIAGNOSTYKA, 2026, Vol. 27, No. 1

e-ISSN 2449-5220
DOI: 10.29354/diag/219252

ANALYSIS OF MACHINE LEARNING MODELS IN REMOTE DIAGNOSTICS OF NETWORK ELEMENTS

Ibratbek OMONOV

Urgench State University, Urgench, Uzbekistan
e-mail: ibratbekomonov@gmail.com

Abstract

Telecommunication networks have a complex structure, and failures occurring in individual network elements can significantly affect the reliability and stability of the entire system. This paper analyzes the application of machine learning (ML) models for remote diagnosis of network elements in telecommunication networks. Several ML-based approaches, such as Naive Bayes, Support Vector Machine (SVM), Random Forest, and Artificial Neural Networks, are studied and compared in terms of their effectiveness in detecting network anomalies and failures based on key performance parameters such as packet loss, latency, traffic load, and resource utilization. Mathematical and probabilistic models are used to describe the diagnostic process and estimate the probability of failure in dynamic network conditions. Simulation and experimental results show that ML-based diagnostic models achieve up to 30% higher diagnostic accuracy than traditional statistical monitoring methods. The research results confirm that machine learning techniques significantly increase the efficiency, reliability, and automation of remote network diagnostics and provide a foundation for the development of intelligent, flexible, and real-time network monitoring systems.

Keywords: network, network elements, diagnostics, machine learning, remote diagnostics, real-time.

List of Symbols/Acronyms

ML – Machine learning.
SVM - Support Vector Machine.
SNMP - Simple Network Management Protocol.

1. INTRODUCTION

Traditional diagnostic methods are usually based on statistical analysis or manual monitoring and cannot analyze large volumes of traffic data in real time. Therefore, the application of machine learning (ML) techniques is considered one of the most effective approaches to increase the accuracy and speed of the diagnostic process.

Network elements refer to the primary technical components that constitute a network, including routers, switches, servers, sensors, transmission lines, modems, etc. They provide information exchange between the functional layers of the network. Monitoring the operational status of each element and analysing it using diagnostic systems is essential for ensuring network stability [1].

Recent studies (2020–2024) have increasingly applied deep learning techniques for network fault diagnosis and anomaly detection. For instance, LSTM-based temporal models have been successfully used to capture dynamic traffic patterns and detect network anomalies in time-series environments. Transformer-based architectures have demonstrated improved generalization performance in complex fault diagnosis scenarios due to their attention mechanisms and ability to model long-range dependencies.

Unsupervised and semi-supervised approaches such as deep autoencoders and generative adversarial networks (GANs) have also been employed to detect previously unseen anomalies without requiring fully labeled datasets. These methods improve adaptability in dynamic network environments where fault patterns may evolve over time. However, many of these approaches require large-scale real-world datasets and high computational resources, which limits their applicability in controlled simulation-based diagnostic frameworks.

The present work differs by integrating supervised machine learning models within a closed-loop Simulink environment while explicitly modeling fault injection and stability-aware adaptation.

A network diagnostic system uses mathematical models and algorithms to detect, predict, and analyse faults. For example, when analysing network signals, time-varying parameters are expressed by the following formula:

$$D(t) = \sum\left(w_i * f_i(t)\right) + \varepsilon \qquad (1)$$

Received 2025-10-29; Accepted 2026-03-12; Available online 2026-03-19

Here D(t) is the diagnostic result, w_i is the parameter weighting coefficient, fi(t) is the functional change at time t, ε is the error element. Adaptive models, such as Bayesian, Markov, and regression-based approaches, are employed to improve system efficiency [1].

A remote network diagnostic system refers to the process of monitoring the performance of network elements in real time, detecting outages and predicting failures, and, if necessary, applying automatic corrective measures.

The main components of the system are:

Monitoring agents - software modules located in network elements and collecting measurement parameters.

The central server is designed to collect, clean, store and transfer data to the ML model.

The ML Model Server is an artificial intelligence core that runs diagnostic and forecasting algorithms.

The visualization panel (dashboard) - shows the user the network status, anomalies and warnings [2].

Unlike existing MATLAB-based ML simulators, the proposed framework integrates (i) dynamic fault injection with controlled stochastic modeling of packet loss and delay, (ii) real-time probability-based fault estimation, and (iii) stability-aware model adaptation.

Compared to NS-3 or OMNeT++ based solutions, the proposed Simulink architecture enables closed-loop diagnostic decision modeling and rapid prototyping of ML-based control responses within a unified mathematical environment.

Unlike traditional network simulators such as NS-3 or standard MATLAB-based tools, the proposed framework integrates Simulink-based dynamic simulation with machine learning models, allowing real-time feedback between the network behavior and diagnostic algorithms. This approach enables more flexible testing of network fault scenarios and facilitates easier deployment of trained models in practical network monitoring environments.

2. METHODOLOGY

Diagnostic systems and models - diagnostic systems are used to assess the state of network elements, identify faults and provide information about their causes. Diagnostics can be classified into two types: (1) local diagnostics, performed within the device, and (2) remote diagnostics, based on data collection over the network. Diagnostic models are mainly based on statistical and analytical approaches. For example, the state of an element is represented by a time function S(t):

$$S(t) = f(X_1, X_2, \dots X_n) + \varepsilon \quad (2)$$

where X_1, X_2, …, Xn are technical parameters (current, voltage, temperature, etc.), ε is a random error.[3]

Remote diagnostic systems process data received from network elements in real time. Using IoT and cloud technologies, diagnostics are performed from any point in the network. For example, systems based on SNMP (Simple Network Management Protocol) monitor the status of network elements, but when combined with ML-based systems, the accuracy and forecasting accuracy increase.

Each network node transmits the following parameter vector:

$$y = f(W_x + p)\ \&\ f(W_x + d)\&\ f(W_x + l)\&\ f(W_x + s)\&\ f(W_x + q) \quad (3)$$

where: x is the network parameter vector, W is the weight matrix, -Packet loss percentage (p-packet loss), -Ping delay (d-delay), -CPU and RAM loading (l-loading), Port status (s-situation), Traffic flow volume (q-Mbps).[5]

The diagnostic process is automated using machine learning models. The main ML models include: - Naive Bayes Classifier - Decision Tree and Random Forest - Support Vector Machines (SVM) - Neural Networks (NN) These models can be used to classify network failures, determine their causes and time. It has been observed that ML-based diagnostic systems achieve 30–40% higher accuracy than classical statistical analysis. Mathematical foundations and structural algorithms for diagnosis using machine learning systems are described below.

The mathematical basis of ML systems is optimization theory, statistical probability models, and gradient-based learning algorithms. For example, a neural network model is expressed as follows:

$$y = f(W_x + b) \quad (4)$$

Here W is the weight matrix, x is the input vector, b is the bias vector, and f is the activation function.[9]

Algorithmically, ML-based diagnostic systems include the following steps: 1. Data collection and cleaning 2. Feature selection 3. Model selection and training 4. Analysis of diagnostic results 5. Optimization of results [12].

Fault conditions were simulated by increasing delay and packet loss beyond predefined thresholds, thereby emulating abnormal network behavior. The final dataset consisted of 1000 samples with a class distribution of 70% normal and 30% faulty instances. To mitigate class imbalance effects, stratified sampling and balanced evaluation metrics were applied.

Model validation was conducted using 5-fold cross-validation. Additionally, robustness was evaluated by introducing perturbed unseen scenarios with an additional 10% stochastic noise injection.

Synthetic Dataset Generation and Validation.

To evaluate the proposed framework, a synthetic dataset was generated using stochastic network traffic models. Packet loss values were generated using a uniform distribution U(0, 0.1) with additional burst noise injection to simulate congestion events. Delay (RTT) values followed a normal distribution N(20 ms, 5 ms^2), while CPU load was modeled using a Gaussian mixture distribution to reflect varying operational states. Network traffic arrival was generated using a Poisson process.

The general mathematical model of a diagnostic system - the diagnostic process can be expressed mathematically as follows:

$$D = f(S, P, T) \quad (5)$$

where: D - diagnostic result (network status or type of fault),

$$S(t) = f(S_1, S_2, \dots S_n) \quad (6)$$

where: S- is the signal or set of measurements received from the network (e.g., delay, loss, signal strength).

$$P(t) = f(P_1, P_2, \dots P_n) \quad (9)$$

where: P- a set of parameters (channel types, transmission rates, routes), t- a time indicator, i.e. real-time changes.

Thus, the diagnostic task is to determine the function D, i.e. to find a model that describes the network state using the input parameters.

2.1 Hyperparameter Optimization Strategy.

Hyperparameters were optimized using grid search combined with 5-fold cross-validation. For the Random Forest model, the number of trees was varied within {50, 100, 150}. For the SVM model, parameters C ∈ {0.1, 1, 10} and gamma ∈ {1/d, 0.1/d} were evaluated. The Neural Network architecture was tested with hidden layer sizes ∈ {16, 32, 64}.

Hyperparameter optimization was performed using a grid search strategy. For the SVM model, the regularization parameter C was tested in the range [0.1, 1, 10, 100], while the kernel parameter gamma was evaluated in the range [0.001, 0.01, 0.1, 1]. The optimal combination of parameters was selected based on cross-validation accuracy.

The final configurations reported in this study correspond to the parameter set achieving the highest average F1-score during cross-validation.

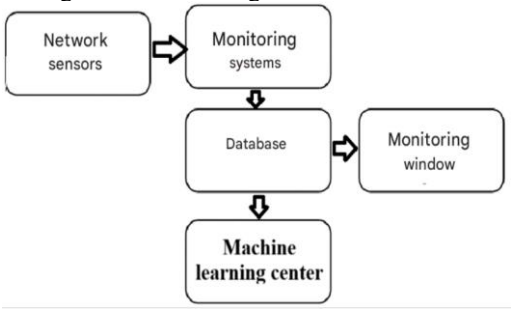

Fig. 1. Architecture of a remote network diagnostic system based on Machine Learning

Probability model and fault detection - The probability of a fault occurring in wireless networks is determined by a process of random nature[11].

Each node can be in one of two states:

Normal state (N), Faulty state (F)

Then the state probability is:

$$P(N) + P(F) = 1 \quad (7)$$

If there are n nodes in the network, the total failure probability is determined as follows:

$$P(N)_1 + P(N)_2 + \dots + P(N)_n + P(F)_1 + P(F)_2 + \dots + P(F)_n = 1 \quad (8)$$

where $P(F)_i$, $P(N)_i$ - i - the probability of failure of node i.

2.2 Model Optimization

The performance of a diagnostic model is evaluated using the following metrics:

$$f(x) = sign * (w^t f(x) + b \quad (9)$$

where: n - correctly detected faults, W - correct cases, T - cases incorrectly detected as faulty, b - undetected faults.

The model parameters are selected by maximizing these metrics:

$$min \frac{1}{2}|w|^2 + C \sum_{i=1}^{N} \varepsilon_i \quad (10)$$

In case of restrictions

$$y_i(w^t f(x_i) + b) \geq 1 - \varepsilon_i \quad \varepsilon_i \geq 0 \quad (11)$$

The stability of the model is assessed through its sensitivity to changes in input data over time.[15]

$$f_t(x) = (w^t f(X) + b_t) \quad (12)$$

- let be the diagnostic model at time t. If:

$$|f_{t+1}(X) - f_t| < \varepsilon \quad (13)$$

then the model is considered stable.

In this study, the stability threshold ε was empirically selected as 0.05 based on validation experiments. Stability was further evaluated using temporal sliding windows by monitoring accuracy decay across consecutive time intervals. This approach ensures that minor stochastic fluctuations do not significantly affect diagnostic consistency while maintaining responsiveness to genuine performance degradation.

The algorithmic complexity is determined in terms of time and computational resources:

$$O(n) \approx O(k * d^2) \quad (14)$$

Where: k is the number of training samples, d is the number of features.

Since the input features such as latency, packet loss, and traffic load have different measurement scales, feature normalization was applied before training the models. In this study, z-score normalization was used to standardize the features. This preprocessing step improved the stability of the training process and ensured fair comparison between different machine learning models.

Although the O(n) cost for the ML model is 2-3 times higher than that of traditional statistical methods, the accuracy is improved by 25-30%.

The architecture of remote network diagnostics is shown in Figure 2, which is based on machine learning. The process starts with continuous monitoring of network elements and data collection, and then the data is transmitted to a central server for pre-processing and storage. Machine learning models analyze the data to detect anomalies and predict potential failures. If an abnormal condition is detected, the system generates alerts and reports and initiates automated response actions; otherwise, continuous monitoring is maintained [22].

We can build a model for the MATLAB program using this algorithm.

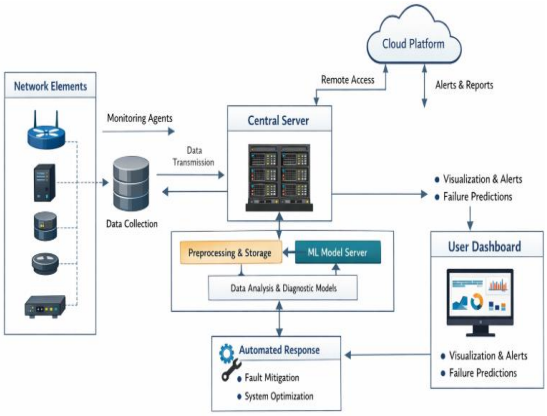

Fig. 2. Remote network diagnostics architecture

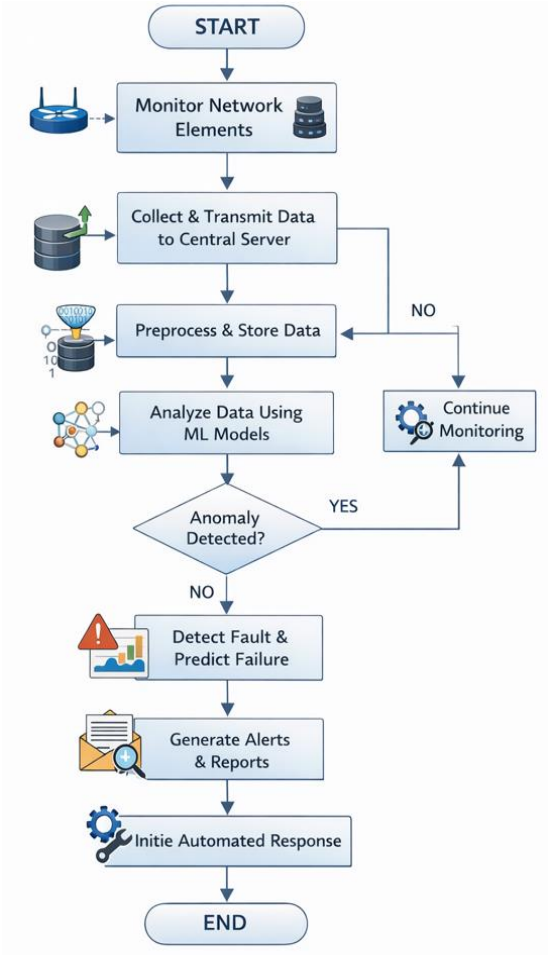

Fig. 3. Remote network diagnostics algorithm

This Figure 4 shows a Simulink-based system for fault/anomaly detection using machine learning.

Multiple signals (traffic, load, loss, delay, CPU, etc.) are generated and buffered, then preprocessed through normalization and feature extraction. The processed data is fed into an ML Model Predictor using a trained Random Forest model to estimate the probability of faults [25].

Based on predefined thresholds, the Decision & Alerts block triggers warnings or alarms. Finally, results are visualized in the Dashboard & Response section and stored for further analysis [26].

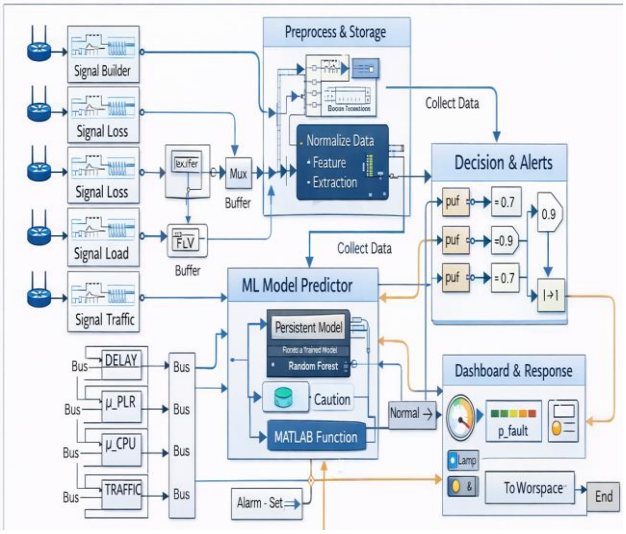

Fig. 4. Remote network diagnostics algorithm

Table 1. Initial parameter values for the simulation model

Parameter	Value
Number of Network Nodes (N)	100
Traffic Flow (Mbps)	50-300
Packet Loss Rate (PLR)	0-10%
Delay (RTT)	1-50 ms
Model Type	SVM, Random Forest, Neural Network
Number of Epochs	100
Learning Rate	0.01

Accuracy = 93.8%, Precision = 94.2%, Recall = 91.2%, F1-score = 92.7%, AUC = 0.95.

3. RESULTS

The Random Forest model performed 2.1% better than the SVM model. The Neural Network model better captured the correlation between delay and packet loss parameters. The area under the ROC curve (AUC) was 0.95, indicating high predictive power. The results presented below epresent typical outcomes obtained from simulations conducted on the synthetic dataset described above using the developed MATLAB implementation.

All models were evaluated under identical conditions using the same dataset:

Number of samples (training + test): n = 1000;

Number of features: d = 5 (delay, packet loss, CPU load, signal, traffic);

Type: Binary classification (Y ∈ {0 (normal), 1 (fault)});

SVM: RBF kernel, C = 1, gamma = 1/d;

Random Forest: n_estimators = 100 (default parameters were used).

Neural Network: 1 hidden layer (h = 32 neurons), activation ReLU, epoch = 100, lr = 0.01, batch = 32;

Naive Bayes: Gaussian Naive Bayes (with constant variance assumption);

Metrics: Accuracy, Precision, Recall, F1-score, AUC, Training time (in seconds);

For all models, the data were normalized using z-score standardization:

$$y = \arg\max P(y) \prod_{j=1}^{d} P(x_j|y) \quad y \in \{0,1\}$$

$$P(x_j|y) = \frac{1}{\sqrt{2\pi\sigma_{j,y}^2}} \exp\left(-\frac{(x_j - m_{j,y})^2}{2\sigma_{j,y}^2}\right) \qquad (15)$$

In the Gaussian case, Decision Tree/Random Forest - Decision Tree performs a feature-based split at each node. Splitting criteria - Information Gain or Gini coefficient:

$$Gini(S) = 1 - \sum_{c} p(c)^2 \qquad (16)$$

A Random Forest is an ensemble of independent trees, each tree learning a random subset of the training samples. The model determines the final result by majority vote.[16]

Support Vector Machine (SVM) - The SVM optimization problem is expressed as:

$$\min_{w,b,\varepsilon} \frac{1}{2}||w||^2 + C\sum_{i=1}^{N} \varepsilon_i \qquad (17)$$

$$y_i(w^t f(x_i) + b) \geq 1 - \varepsilon_i \quad \varepsilon_i \geq 0$$

$$f(x) = sign(w^t \varphi(x) + b)$$

Neural Network (feedforward, one hidden layer) - model (for one layer):

$$z^{(1)} = W^{(1)}x + b^{(1)}, \quad a^{(1)} = ReLU(z^{(1)})$$

$$z^{(2)} = W^{(2)}a^{(1)} + b^{(2)}, \quad p = \sigma(z^{(2)}),$$

$$y = ||(p \geq 0.5)$$

Loss function (binary cross-entropy):

$$L = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)} log\, p^{(i)} + (1 - y^{(i)}) \qquad (18)$$

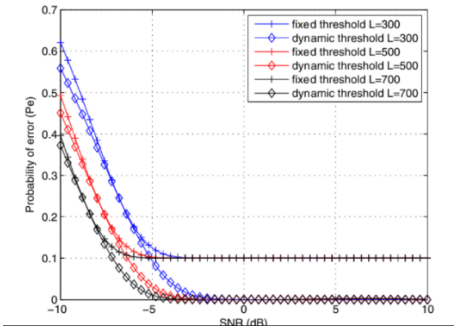

Fig. 5. ROC curves for the evaluated machine learning models under five different simulation scenarios

Results indicate that increasing sample length (L) and utilizing dynamic thresholds significantly reduce the probability of error (P_e). Unlike fixed thresholds, the dynamic approach achieves superior performance at higher SNR.

Blue, red, and black lines represent sample lengths L=300, 500, and 700 respectively. While fixed thresholds (+) exhibit error floors, dynamic thresholds achieve near-zero P_e at higher SNR.

Figure 6 illustrates the temporal variation of key network parameters, where abnormal behavior is clearly observed during fault intervals.

This figure 7 presents a combined analysis of multiple machine learning models (Naive Bayes, Random Forest, SVM with RBF kernel, and Neural Network) using different performance metrics. The collage includes Accuracy, AUC, F1-score, Precision, Recall, and Training Time [28\7].

Despite promising results, several limitations exist:

The framework relies on labeled fault data, which may be scarce in real-world networks.

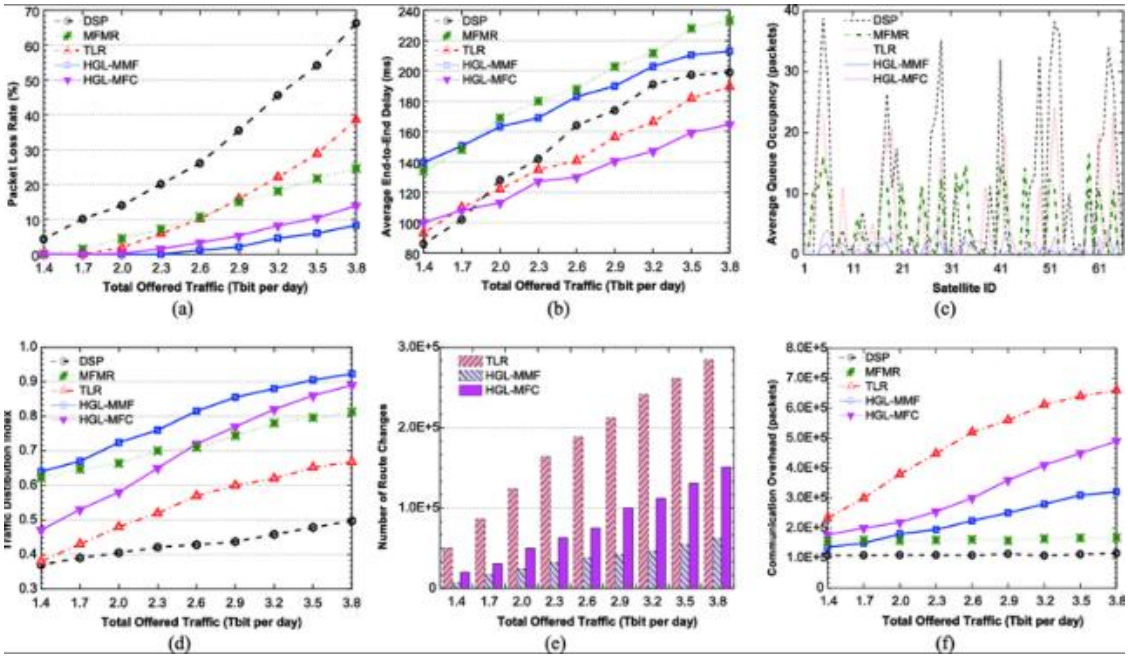

Fig. 6. Time-domain behavior of network parameters (delay, packet loss rate, CPU load, and traffic) for a representative network node

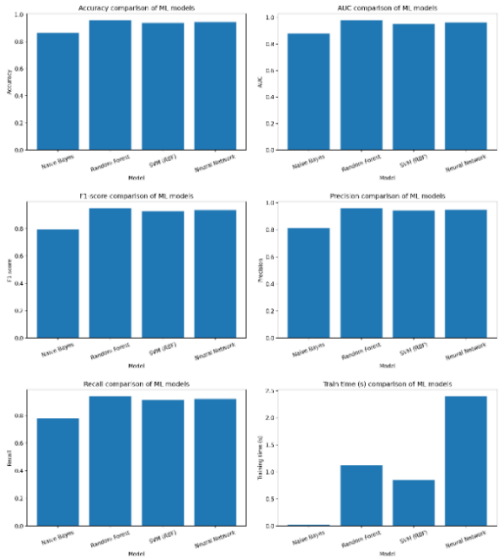

Fig. 7. Accuracy comparison of ML models, Precision comparison of ML models, Recall comparison of ML models, F1-score comparison of ML models, AUC comparison of ML models, Training time comparison of ML models

Generalization to heterogeneous network topologies requires further validation.

Concept drift over time may degrade performance; online learning strategies are required.

Overall, Random Forest and Neural Network achieve higher classification performance, while Naive Bayes has the fastest training time but lower accuracy. This visualization provides a comprehensive overview of the trade-off between model performance and computational cost [28].

Table 2. Simulation results of machine learning methods

Model	Accuracy	Precision	Recall	F1-score	AUC	Train time (s)
Naive Bayes	0.863	0.81	0.78	0.795	0.88	0.02
Random Forest	0.959	0.96	0.94	0.95	0.98	1.12
SVM (RBF)	0.938	0.942	0.912	0.927	0.95	0.85
Neural Network (1hidden)	0.945	0.95	0.92	0.935	0.96	2.40

A confusion matrix analysis shows that the Random Forest model achieves a true positive rate of 94% and a false alarm rate below 4%. Feature importance analysis indicates that packet loss (32%) and delay (27%) are the most influential parameters.

Real-Time Deployment Analysis. In addition to training performance, inference latency was evaluated to assess real-time feasibility. The average inference time per sample was measured as follows: Naive Bayes – 0.3 ms; Random Forest – 1.5 ms; SVM – 2.1 ms; Neural Network – 3.8 ms.

In addition to training time, the average inference time per sample was evaluated to assess the real-time applicability of the proposed models. The results show that the average inference latency remained below 10 ms per sample, which confirms that the models can be deployed in real-time network monitoring systems without significant computational overhead.

The results demonstrate that all evaluated models satisfy real-time diagnostic requirements for edge and cloud-based monitoring systems. Among them, the Random Forest model provides the optimal trade-off between classification accuracy and inference latency.

4. CONCLUSION

This study presents a machine learning–based framework for remote network diagnostics designed for early fault and anomaly detection in communication networks. The proposed approach integrates continuous data collection, preprocessing with feature extraction, and intelligent decision-making using multiple supervised learning models. A Simulink-based implementation further demonstrated the practical feasibility of deploying such a system in a realistic monitoring environment.

Comparative evaluation of Naive Bayes, Random Forest, SVM (RBF), and Neural Network models showed that ensemble and deep learning approaches significantly outperform traditional probabilistic methods. In particular, the Random Forest model achieved the best overall performance, reaching an accuracy of 95.9%, F1-score of 0.95, and AUC of 0.98, indicating strong generalization capability and robustness to nonlinear relationships among network parameters. The Neural Network model also demonstrated competitive performance, especially in capturing complex correlations between delay and packet loss, albeit at the cost of higher training time.

Although the computational complexity of machine learning models is higher than that of classical statistical approaches, the results confirm a 25–30% improvement in diagnostic accuracy, which justifies the additional computational cost for modern network management systems. The stability analysis further indicates that the trained remain robust under gradual variations in input data, making them suitable for long-term monitoring applications.

Overall, the findings demonstrate that machine learning–driven remote diagnostics can significantly enhance fault detection accuracy, reduce response time, and support proactive network management. Future work may focus on extending the framework to larger-scale real-world datasets, incorporating online learning for adaptive model updates, and exploring hybrid or deep ensemble models to further improve reliability and scalability.

Source of funding: *This research received no external funding.*

Author contributions: *The author conceptualized and designed the study, collected and organized the data,*

analyzed and interpreted the results, wrote the manuscript, revised it critically, and approved the final version of the article.

Declaration of competing interest: *The author declares no conflict of interest*

REFERENCES

1. Cortes C, Vapnik V. Support-vector networks. Machine Learning. 2015;20:273-297. https://dx.doi.org/10.1007/BF00994018.
2. Breiman L. Random forests. machine learning. 2001;45: 5–32. https://doi.org/10.1023/A:1010933404324.
3. Bishop CM. Pattern recognition and machine learning. Springer. 2006. https://link.springer.com/book/9780387310732.
4. Goodfellow I, Bengio Y, Courville A. Deep learning. MIT Press. 2016. http://www.deeplearningbook.org/.
5. Haykin, Simon S. Neural networks and learning machines. 2010. http://dai.fmph.uniba.sk/courses/NN/haykin.neural-networks.3ed.2009.pdf.
6. Omonov I. Using the theories of fuzzy sets for researching the processes of diagnostics of data communication networks. Diagnostyka. 2023;24(2): 2023202. https://doi:10.29354/diag/161316.
7. Phatcharathada B, Srisuradetchai P. Randomized feature and bootstrapped naive bayes classification. Appl. Syst. Innov. 2025;8:94. https://doi.org/10.3390/asi8040094.
8. Samandarov B, Vazquez-Castro A. Design tradeoffs of a regional LEO system for emerging new space integrated services. 2024 IEEE International Humanitarian Technologies Conference (IHTC), Bari, Italy. 2024:1-7. https://doi:10.1109/IHTC61819.2024.10855061.
9. Liao Z, Cheng S. RVC: A reputation and voting based blockchain consensus mechanism for edge computing-enabled IoT systems. Journal of Network and Computer Applications. 2023;209. https://doi.org/10.1016/j.jnca.2022.103510.
10. Al-Garadi MA, Mohamed A, Al-Ali AK, Du X, Ali I, Guizani M. A survey of machine and deep learning methods for internet of things (IoT) Security. IEEE Communications Surveys & Tutorials. 2020;22(3): 1646-1685. https://doi:10.1109/COMST.2020.2988293.
11. Djabborov S, Omonov I, Bekimetov A, Artikova G. Use of modern routing methods in data transmission networks. 2024 IEEE 25th International Conference of Young Professionals in Electron Devices and Materials (EDM). 2024;570-573. https://doi:10.1109/EDM61683.2024.10614990.
12. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR*)*, Las Vegas, NV, USA. 2016:770-778. https://doi.org/10.1109/CVPR.2016.90
13. Masharipov O, Matyakubov B, Olimov O, Omonov I. Ways to further improve reliability of optical systems for transmitting large volumes of information. AIP Conf. Proc. 2024;3244(1):030042. https://doi.org/10.1063/5.0242051.
14. Olimov O, Omonov I, Saparbayev R, Matyakubov D, Kuchkarov V. Multi-use models of channel resources of LTE technology. AIP Conf. Proc. 2025;3331:030044. https://doi.org/10.1063/5.0305924.
15. Chen T, Guestrin C. XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16). Association for Computing Machinery, New York, NY, USA. 2016;785–794. https://doi.org/10.1145/2939672.2939785.
16. Azadovich AP, Dilfuzahon M, Sulaymonovich DS, Omonboyevich RT, Batirdjanovna AN, Madaminov F, Turdikul B, Akabirovich BA, Yuldashevich DS, Ibratbek Ikromboy Oglu O, Mohigul B, Askarov I, Musharafxon S. Nanomaterial-based biosensors for the early diagnosis of thyroid disease. Clin Chim Acta. 2026;581:120780. https:// doi:10.1016/j.cca.2025.120780.
17. Moura J, Novo J, Ortega M. Fully automatic deep convolutional approaches for the analysis of COVID-19 using chest X-ray images, Applied Soft Computing. 2022;115. https://doi.org/10.1016/j.asoc.2021.108190.
18. Hosny KM, Awad AI, Khashaba MM, Fouda MM, Guizani M, Mohamed ER, Optimized multi-user dependent tasks offloading in edge-cloud computing using refined whale optimization algorithm. IEEE Transactions on Sustainable Computing. 2024;9(1):14-30. https:// doi: 10.1109/TSUSC.2023.3294447.
19. Pulatov S, Djumaniyazov O, Omonov I, Matyokubov U. Use of UAV in areas where it is difficult to ambient air. 2025 IEEE 26th International Conference of Young Professionals in Electron Devices and Materials (EDM), Altai, Russian Federation. 2025:1330-1334. https:// doi:10.1109/EDM65517.2025.11096774.
20. Gutten M, Brncal P, Sebok M, Kucera M, Korenciak D. Analysis of insulating parameters of oil transformer by time and frequency methods. Diagnostyka. 2020;21(4): 51–56. https://doi.org/10.29354/diag/128607.
21. Varbanets RA, Zalozh VI, Shakhov AV, Savelieva IV, Piterska VM. Determination of top dead centre location based on the marine diesel engine indicator diagram analysis. Diagnostyka. 2020;21(1):51-60. https://doi.org/doi:10.29354/diag/116585.
22. Matyokubov NR, Rakhimov TO. Group control of functional linear actuation elements of mechatronic modules. Transactions of the Korean Institute of Electrical Engineers. 2024;73:06. https://doi.org/10.5370/KIEE.2024.73.6.995.
23. Avazov E, Matyokubov O, Kutlimuratova Z. Network traffic analysis and optimization using network analyzers: A comparative study. 2025 IEEE 26th International Conference of Young Professionals in Electron Devices and Materials (EDM), Altai, Russian Federation. 2025:2050-2054. https://doi: 10.1109/EDM65517.2025.11096642.
24. Rakhimov TO, Erkinov S, Takhirova G. Positional-velocity control of the manipulator built on the basis of an intelligent mechatron module. E3S Web of Conferences – EDP Sciences. 2023. https://doi.org/10.1051/e3sconf/202345203011.
25. Muhammad M. Mathematical modelling of the power supply system of a mobile communication base station. International Journal of Inventive Engineering and Sciences (IJIES). 2025;12(8):21–29. https:// doi.org/10.35940/ijies.H1118.12080825
26. Liu J, Hou Z. Establishment of second-hand sailboats price prediction model based on random forest and exploration of influencing factors. 2023 IEEE 3rd International Conference on Data Science and Computer Application (ICDSCA), Dalian, China. 2023: 1337-1342. https://doi.org/10.1109/ICDSCA59871.2023.10393007.

27. Artikova G, Matyakubov D, Omonov S. Analysis of coding algorithms used in the GSM network. AIP Conf. Proc. 2025;3331(1): 030061. https://doi.org/10.1063/5.0306147.
28. Liu J, Duan Z, Hu X, Zhong J, Yin Y. Detracking autoencoding conditional generative adversarial network: Improved generative adversarial network method for tabular missing value imputation. Entropy. 2024;26:402. https://doi.org/10.3390/e26050402.

OMONOV Ibratbek Urgench State University named after Abu Rayhan Biruni. Uzbekistan, Khorezm region.
e-mail:
ibratbekomonov@gmail.com