ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VII, July 2026

Machine Learning Based Network Issues Classifier, Recommender and Threat Prediction for OSI Model Layers

C.A. Nnalue, C .I. Ugwu, U. K. Ome, S.O Aneke, C.I. Egbu, J. Onyima

Department of Computer Science, University of Nigeria, Nsukka

DOI: https://doi.org/10.51583/IJLTEMAS.2026.150700146

Received: 08 August 2026; Accepted: 13 August 2026; Published: 24 August 2026

ABSTRACT

Computer networking creates remarkable impacts in solving modern activities, but as they have evolved to become larger and more complex, they also present challenges when it comes to finding problems, identifying issues, and examining potential security problems. Most network management systems offer a single focus point, or a technical solution that is not easy for ordinary person to see, leading to time delays, increased risk, and additional vulnerabilities. The goal of this study is to allow for better identification of problems, propose a set of applicable solutions, and find possible threats that may arise through analyzing traffic on the network using an intelligent, machine-learning based model for classifying network problems based on the OSI model. To identify and classify problems, two sets of data were used; real-world database of network problems received from the Office of Accountant General for the Federation (OAGF), and the well-known NSL-KDD datasets for conducting research on intrusion detection. Machine learning algorithms have been examined for each component of the research, with the Naïve Bayes algorithm giving the highest classification accuracy, while Adaboost gave the most accurate classification of malicious traffic. The three models were integrated into a single web application using the Python Flask, HTML and CSS languages, and were designed with a three-layer architecture which splits the interface, processing, and storage components. The full System offers users information about the problem they are having. It also gives them information about the layer on the OSI model that contains the problem as well as recommendations on how to fix it. The system checks if the specific traffic has evidence of an attack. In this paper, showed how machine learning can assist with network troubleshooting, increase security awareness among users, decrease dependence on subject matter experts, and ultimately accelerate and increase accuracy in decision making.

Keywords: OSI model, Network issues classification, Solution recommendation, Threats prediction, Machine learning.

INTRODUCTION

Computer networks have become ubiquitous in modern society; computer networks underpin the day-to-day operations of virtually every industry, and are utilized by individuals as part of their daily activities, such as browsing the internet or controlling the appliances in their smart home. The ongoing expansion of network infrastructure, and therefore complexity, has led to manual network management and incident response no longer meeting the needs of modern networks [1], which negatively affects an organization's ability to troubleshoot and identify abnormal/illicit behaviours on their networks. The heavy reliance of network administrators on troubleshooting tools such as ping, traceroute, netstat, and log file analysis to troubleshoot and identify problems within the networks, results in delays in responding to problems and greater vulnerability to human error [2]. The ever-increasing complexity of networks has resulted in the emergence of machine learning techniques and AI-based network management that can solve the challenges associated with complex networks. The introduction of AI and ML into network management has led to advancements such as predictive analytics, anomaly detection, and automation of configuration; as a result, many organizations are increasingly using unsupervised machine learning in unstructured raw network data to improve network performance and support use cases, such as traffic engineering, anomaly detection, Internet traffic classification, and quality of service

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VII, July 2026

optimization [3]. While there have been many advancements in current network management systems, limitations still exist.

The majority of the existing systems work reactively to threats rather than proactively as they only recognise the threats after they have already caused harm. This causes loss of productivity. The complexity of these systems also poses a problem for users who are not qualified experts in using the systems effectively in terms of maintaining and keeping their networks safe; hence, technicians need to provide assistance at a very high cost. Current systems do not provide a means for classifying the problems in the network and therefore make it difficult for administrators to find resolutions to these problems. Also, current systems do not provide any means of recommending solutions to detected problems.

Therefore, this paper presents an example of a Machine Learning model based system capable of classifying problems in a network based on the OSI model layers allowing for the isolation or localisation of problems and providing solutions to the located problems. The system is designed to operate proactively, detecting threats before any relevant damage occurs, and will assist in providing a safer and more secure environment. The system also includes an easy-to-use graphic user interface (GUI) that allows for an effective means of consulting with the system by both expert and non-expert users.

Objective of Study

To collect and prepare real world datasets including network fault type/solution and network intrusion detection dataset (for predicting network threats).

To build and train machine learning classifiers to classify network issues according to the layers of the OSI model, recommend solution, and predict network threats.

To classify network issues, recommend a solution for those classified network issues, and predict network threats.

To evaluate the performance of the model based on the standard classification metrics of accuracy, precision, recall and F1-score used to measure performance

LITERATURE REVIEW

The OSI Model is a logical, layered architecture model that define the necessary functionality for establishing System-to-System communications [4]. The OSI Model can be used as a telecommunications/commonality model to assist in identifying services, software, processing and hardware requirements for transferring data between computing devices over a network [5]. The OSI Model is broken into seven different layers, which are: Application, Presentation, Session, Transport, Network, Data Link and Physical. The Physical layer defines connector and interface specifications necessary to connect to a network as well as, the medium (cable) to be utilized for transmitting a bit stream over the network [6]. Based upon the data link layer, the OSI Model provides reliable transmission of frames or data using adjacent nodes; it is built upon the non-reliable service of the physical layer. This is accomplished through the utilization of error detection and error control typically by using a Cyclic Redundancy Check (CRC). According to [7], The Network Layer is responsible for data routing, forwarding and addressing packets of data. The Transport Layer is responsible for transferring or communicating data between hosts. The Transport Layer is comprised of two protocols – TCP (Transmission Control Protocol) which provides for reliable communication and UDP (User Datagram Protocol).

Session Layers will manage the established connections; as well manage the port as well as session connections. The Presentation Layer's job is to manage how data is formatted so that the Session Layer can represent the data in a way that meets the standards required for encapsulating, and will also provide the appropriate encoding so that the receiver can interpret or execute the data correctly. The Application Layer is the layer in which the user application runs; therefore this is the layer that directly works with and manipulates data being accessed by the

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VII, July 2026

user. This will be the Layer that provides the user the interface to the network when communicating with your application.

Computer Network Issue

Interconnectivity between two or more computers that can communicate with each other via a networked data transfer is referred to as a computer network [8]. When computers are linked, they are able to send and receive messages from one another over the same network. There are always issues or problems associated with networks, which affect their ability to function correctly and provide reliable and secure services to users [10]. A problem with a network is defined as either a hardware or software problem that causes the network not to function correctly or no longer provide an adequate level of connectivity [11]. These types of problems are caused when there is an improperly configured computer system and/or insufficient computer processing (capacity), memory (storage), etc. Network security threats are defined as any threat that can compromise the integrity, confidentiality, or availability of a network (CIA) [12]. Network security threats take on many different forms, such as malware, phishing attempts, ransomware, and Distributed Denial of Service (DDOS) attacks [13]. All of these can have a negative impact on people and businesses/organizations by causing financial loss, identity theft, and disruption of mission-critical operations.

Machine Learning

By continually improving the performance of machine learning algorithms based on previous experiences, it has become a powerful tool for accurately predicting outcomes. Generally speaking, statistical, mathematical, and numerical techniques are used to generate some type of knowledge from datasets that can result in a summary, a visual representation, a grouping, or even predicting an outcome based on the information contained within the dataset.

Machine learning algorithms are typically classified into three main categories: supervised learning (in which the algorithm performs a task such as classifying or predicting by learning from datasets that contain example inputs with assigned desired output), unsupervised learning (in which the algorithm does not learn from labeled data sets but rather acts on unlabeled data sets) and reinforcement learning (in which the algorithm is not supervised and learns solely through trial and error).

Supervised learning uses training to establish relationships between input and output data. A key advantage of using supervised learning as opposed to unsupervised and reinforcement learning methods is that the machine learning algorithm has already established a means of relating input and output datasets, thereby enabling it to improve its performance as more data becomes available.

In contrast, unsupervised learning is where the algorithm is trained using only unlabeled datasets. The machine learning algorithm is trained without the benefit of a pre-existing relationship between inputs and outputs and thus learns to interact with data independently of the human user. Reinforcement learning is an aspect of machine learning that allows an AI-driven system (agent) to learn through trial and error using feedback from its actions [18].

Review of Related Works

The purpose of the study conducted by Mohammed et al. [19] was to develop an ML-based network status detection and fault localization system aimed specifically at ISPs. Traditionally, ISPs have performed service assurance and maintenance through manual methods; for example, if a network or a service fails, an alarm is triggered, and NOC technicians (available 24/7) receive alerts to locate and fix the problem. This method of operation is inefficient. The researchers noted the necessity of creating a system that would automatically discover and localise possible faults on the network as soon as possible in order to limit the negative impact of such problems on customers' experiences. They argued that many of the challenges associated with fault localisation and network state classification are primarily related to large amounts of data and are analytically difficult to solve (especially in large and/or complex networks), but ML algorithms will enable such data to be

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VII, July 2026

processed more effectively. The researchers also introduced a multiclass classification scheme for classifying network state detection; network state was classified into three categories: normal state, congested state, and physical network problem. The congested state and physical network problem states were further subdivided into sub-states in order to facilitate localising the fault(s); thus, the system produced a total of eight classes for their testbed. The process used by GNS3 to emulate a topography similar to an ISP's genuine network has been used to gather and generate the data needed to create this classifying ensemble algorithm for determining network status. Overall, using this algorithm produced accuracies between 97% and 99%. Its inputs consisted of Quality of Service (QoS) and Quality of Experience (QoE) metrics collected from the ISP's testbed. The testbed was established based on the lack of existing data sets needed for this system. The QoS metrics consisted of per-path and per-device measurements; for per-path measurements, the parameters measured were end-to-end delay, jitter, packet loss, out-of-sequence packets, and discarded packets; in other words, service level agreement (SLA) measurements were obtained for each of the three paths. With respect to per-device measurements, the parameters measured included the number of packets sent and received by ports for certain routers. QoE was indirectly specified by the file download speed in an application that transfers files through the network. Although this study and the present study address similar concepts of multiclass classification of networks, the importance of this research is that the present study will identify network problems based on the OSI layer.

In reference to [20], researchers investigated via Deep Learning approaches how faults could be detected in communication networks. The authors noted that modern networks produce massive and complex data sets, which complicates the task of identifying faults by traditional rule-based methods quickly and accurately. To address this limitation, a Deep Learning model was developed to learn by example, both normal and abnormal traffic patterns, directly from the data. This allows the model to create its own features as well as identify fault behaviours on the network, therefore improving detection accuracy, especially with respect to subtle faults that might otherwise go undetected due to their proximity to normal activity on the network. The research results reported within the paper demonstrate that Deep Learning is capable of identifying relationship patterns within network behaviour, the recognition of which had eluded previous generations of techniques. The effort provided by Feng and Zhao provides an adequate basis for understanding how machine learning supports detecting network faults. However, it is dissimilar to the objectives of this paper, as it focusses solely on identifying and classifying faults within communication networks, and does not provide a defined breakdown of fault types by layer by OSI (Open Systems Interconnection Model). Additionally, there is no area of research that addresses recommendations for action after a fault has been identified. This research aims to classify network problems based on the OSI model more clearly showing where the problem lies within the layers of a network. The paper will also attempt to look at ways to determine potential future threats to a network, something not addressed in the previous work. In addition to using pre-existing datasets like Feng and Zhao did for training their models; this work is being developed to allow users to enter information about the network so they can get an immediate diagnosis of their problem(s). Overall, Feng and Zhao's research shows how deep learning can be used to identify faults in networks, and this current research project will extend that use of deep learning by adding OSI-layer mapping of faults, recommending solutions for those faults, providing users with the ability to work directly with the system and by forecasting possible threats against a network.

A robust Network Intrusion Detection System (NIDS) using both machine learning and deep learning is discussed in [21]. The exponential growth of network data is attributed to various factors such as the Internet of Things (IoT), cloud-based services, and an enormous amount of devices that connect to the Internet. The increasing number of attacks poses a serious risk to the security of networks, creating the need for a network security solution that can identify existing and future threats. Existing NIDS use multiple layers of defense to protect networks to ensure that if one layer of defense fails, multiple other layers will prevent the attack. Most existing NIDS are either signature-based or anomaly-based detection systems; Signature-based systems detect only those threats that are specifically mentioned in a threat database, whereas anomaly detection will identify new malicious activity that deviates from normal activity, such as zero-day attacks. Machine learning, deep learning, and both models were examined for their effectiveness in detecting intrusions to a NIDS. According to the authors, NIDS based on ML builds classification models through training with a large and diverse number of network example data points to maximize accuracy in classifying attacks into their respective categories. Furthermore, the authors state that Deep Learning Models help NIDS learn from attack behaviour based upon

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VII, July 2026

the features of the networks themselves, thus removing the necessity to correlate, select, and represent the features utilized for classification by NIDS via traditional means; rather, these NIDS can learn the underlying behaviour of the network itself to classify the behaviour exhibited by the attacker with a minimum number of false positives. The authors then identify three examples of network attacks (i.e., Fuzzers, DoS, Backdoor) and describe their proposal and experimentation of the UNSW-NB15 Dataset developed by the Australian Centre for Cyber Security and the binary classification models that were used to identify two attacks to those examples, e.g., the use of SVM, Adaboost, XGBoost, Random Forest, K-Nearest Neighbors (KNN), Decision Tree (DT), Multi-Layer Perceptron (MLP), and Deep Multi-Layer Perceptron (Deep MLP). The Deep MLPs consist of 2 dense layers utilizing a Relu activation function and an additional dense layer utilizing a sigmoid activation function, which generates a well-mixed sampling of the input-generated outputs, resulting in the successful prediction of the target class based upon the outputs from the previous layers. The metrics that were used for evaluating the system included accuracy, precision, recall, F1-Measure, Receiver Operating Characteristic Curve (ROC), and Area under ROC (AUC). The network intrusion detection system was implemented on Ubuntu 20.04.4 running on an Intel Core 11th generation i7 processor with a RAM size of 16GB and a hard disk drive size of 1TB. The models were implemented using the programming language python utilizing Keras 2.3.1 and TensorFlow 2.2.0 libraries for machine learning and deep learning. After testing a number of different models for comparison, the authors concluded that decision tree (0.99, 99.05), support vector machine (0.94, 95.17), and K-nearest neighbor (0.959 if K=7), were among the highest performing methods for detecting network attacks. In addition, they also identified that the features used to train the models were all very important and were able to produce significant differences between attack and legitimate network flow. For the deep learning methods, multi-layer perceptron demonstrated the highest level of accuracy (0.9844) using the Adam optimizer and 80:20 Train-Test ratio. The objective of this study as well as the present one is the development of a model to help solve the issues associated with network security; however, the focus of this study does not include overall management of the network nor does it include consideration of other types of network failures.

The potential of AI-based approaches being used to improve accuracy in traffic classifications within IOT has been established by this research paper [22], showing how intelligent systems can be used to enhance the security of IOT networks through the use of advanced technologies. It also highlights both a definition of the IOT as well as identifying the IOT devices as opportunistic targets for cyber-attack, since they tend to have limited processing power, memory, and bandwidth that present limitations to implementing traditional security.

The focus of this paper was to measure the level of malicious activity within IOT networks utilizing various types of machine learning techniques for detecting malicious activity against a benchmark data set. It quantified the performance of each of those techniques on both binary and multi-class classification tasks in an attempt to improve both the reliability and security of IOT networks. The methodology of this research was outlined by starting from Data Acquisition and Data Processing which included; Data Collection (obtaining actual network traffic data that reflected actual IOT traffic being observed), Data Cleansing (to ensure the data being analyzed was of high quality and consistent with one another), Feature Extraction, Data Transformation, Feature Selection and Dimensionality Reduction. After that, there was Model Selection and Model Training which included; Algorithm Selection, Model Training and Model Evaluation. The proposed framework will be developed to provide the ability for network administrators to have a valuable resource to support their security monitoring efforts, as well as optimizing how to allocate resources and ultimately manage their overall network.

Research Gap

Upon reviewing the relevant literature for this study, it became clear that very few of the studies made attempts to incorporate both types of network management as an approach to enhance overall network management. Most of the studies either focus on network threats or on network faults, but not both strategies have been considered. Only a very small number of studies focused on both issues, but offered no solutions that would help reduce the time required to resolve network faults. In the study described in this paper, the use of the OSI Reference Model to group network faults provides a means of establishing a general standard for network failures. Moreover, no system was developed as a result of reviewing the literature for developing a network management system to represent the grouping of the faults into categories, but only diagrams were created.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VII, July 2026

METHODOLOGY

Technology Used

When deploying machine learning algorithms/systems, there are various tools/technologies that must be utilized; these tools consist of programming languages, libraries, etc., all required to create and deploy machine learning algorithms. To develop the proposed system, the primary programming language used is Python since it is a high-level general-purpose programming language with a rich toolbox (libraries & frameworks) [24]. The backend of the machine learning system being developed in this study will utilize Flask, which is a lightweight framework for Python. The machine learning system development technology will use Python libraries. The proposed machine learning system will be developed with the following Python frameworks: Scikit-learn (used for training and predicting), Pandas (for pre-processing/manipulating data), etc. Scikit-learn is an efficient and straightforward tool for mining and analyzing data and is based on Numpy, Scipy, and Matplotlib (all of which are Python libraries) [25]. Scikit-learn employs Python's rich environment to provide extensive implementations of many established algorithms while also maintaining user-friendliness by being closely integrated with Python [26].

Pandas is a python package built for a broad range of data analysis and manipulation including tabular data, time series etc.; it can be used for data wrangling in order to transform data into a representation more suitable for analytics in different scenarios [27].

Data Collection

To develop training and testing datasets for this model solution, we collected two datasets: (1) NSL-KDD - A modified version of the KDD'99 dataset used as a standard for intrusion detection and classification of the model of the OSI model. It was retrieved from the Kaggle website and was modified to correct data collection errors. The dataset will be used for the threat detection portion of the model. (2) Network Issues - This set of data from AGF's ICT staff will be used to accurately classify network issues and determine the solutions for specific network issues as outlined in the OSI Model.

Table 1 Features of the NSL-KDD dataset used for network threat detection

S/N	Feature name	Description
1	Duration	The length of time taken (in seconds) during a connection
2	Protocol_type	The type of protocol used
3	Service	Destination network service (e.g., ftp, telnet, http)
4	Flag	status of connection
5	Src_bytes	Number of bytes transferred from source to destination
6	dst_bytes	Number of bytes transferred from destination to source
7	Land	If the source and destination IP addresses are equal, it is 1 otherwise 0
8	wrong_fragment	Number of wrong fragments
9	Urgent	Number of urgent packets in the connection
10	Hot	Number of hot indicators
11	num_failed_logins	Number of unsuccessful attempts at login
12	logged_in	If logged in 1, otherwise 0
13	num_compromised	Number of compromised states

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VII, July 2026

14	root_shell	if a command interpreter with a root account is running rootshell it is equal 1, otherwise 0
15	su_attempted	if an su command was attempted it is equal to 1, otherwise 0. (Temporary login to the system with other user credentials)
16	num_root	Number of root access
17	num_file_creations	Number of file creation operations
18	num_shells	Number of active command interpreters (shell prompts)
19	num_access_files	Number of operations on access control file
20	num_outbound_cmds	Number of outbound commands in an ftp session
21	is_host_login	Is host login = 1 if the login is on the host login list , else it is 0
22	is_guest_login	if guest login it is equal to 1, otherwise it is 0
23	Count	Number of connections to the same destination host as the current connection in a given interval
24	srv_count	Number of connections to the same service as the current connection in a given interval
25	serror_rate	percentage of connections with SYN errors among the connections aggregated in count
26	srv_serror_rate	percentage of connections with SYN errors among the connections aggregated in srv_count
27	rerror_rate	percentage of connections with REJ errors among the connections aggregated in count
28	srv_rerror_rate	percentage of connections with REJ errors among the connections aggregated in srv_count
29	same_srv_rate	Number of connections to the same service among connections aggregated in count
30	diff_srv_rate	Number of connections to the different service among connections aggregated in count
31	srv_diff_host_rate	percentage of connections to different destination host among the connections aggregated in srv_count
32	dst_host_count	Number of connections that have same destination host IP address
33	dst_host_srv_count	Number of connections to the same destination host that use the same service (same port number)
34	dst_host_same_srv_rate	Percentage of connections to the same destination that use the same service
35	dst_host_diff_srv_rate	Percentage of connections to the same destination that use different service
36	dst_host_same_src_port_rate	Percentage of connections to the same destination host that were to the same source port
37	dst_host_srv_diff_host_rate	percentage of connections to different destination host among the connections aggregated in dst_host_srv_count

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VII, July 2026

38	dst_host_serror_rate	percentage of connections with SYN errors among the connections aggregated in dst_host_count
39	dst_host_srv_serror_rate	percentage of connections with SYN errors among the connections aggregated in dst_host_srv_count
40	dst_host_rerror_rate	percentage of connections with REJ errors among the connections aggregated in dst_host_count
41	dst_host_srv_rerror_rate	percentage of connections with REJ errors among the connections aggregated in dst_host_srv_count
42	Attack	Indicates if there is a malicious attack and the type. If there is no attack, it is equal to 1. Other values indicates an attack
43	Lastflag	The difficulty level (of prediction)

Table 2 Features of the dataset used for network fault troubleshooting

S/N	Feature name	Description
1	OSI_Layer	It ranges from 1 to 7. This represents the layers the network issue belongs to.
2	OSI_Layer_name	Name of the layer the network issues belong to.
3	Device_Type	The type of device experiencing the network issue
4	Symptom	The network issue being experienced
5	Solution	Solution (troubleshooting tip) to the symptom or network issue
6	Use_Case	Situations where such issue can occur (or related issues)

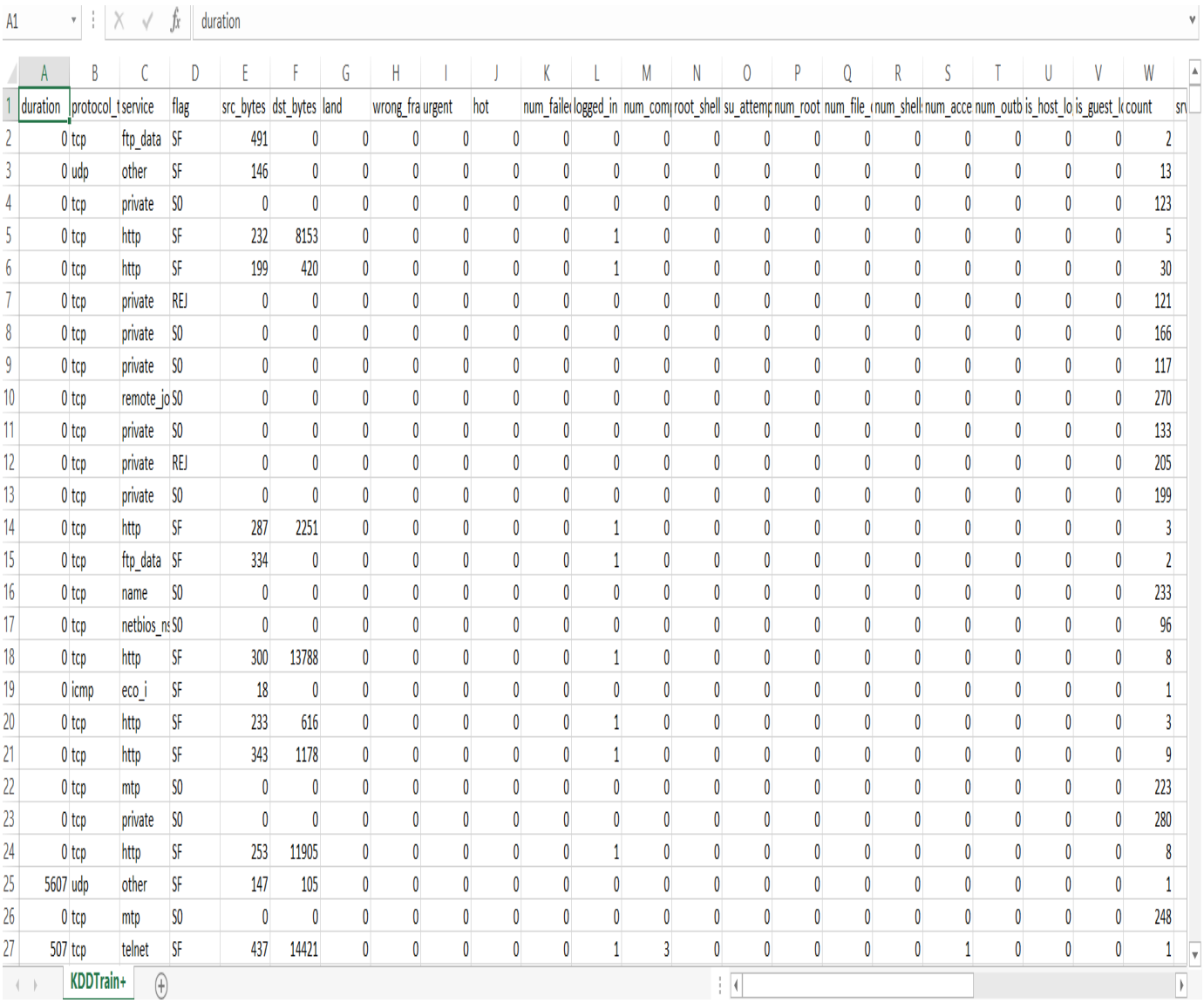
Table 3 Sample of dataset used for network fault troubleshooting

OSILayer	OSI Layer name	Device Type	Symptom	Solution	Use Case
1	Physical	SAN	Intermittent link loss due to damaged fiber cables	Replace damaged cables; perform regular physical inspections	Detecting Layer 1 failures via log patterns
1	Physical	Routers	Duplex mismatch causing late collisions	Configure both ends to full-duplex manually	Classifying Layer 1 vs Layer 2 errors
1	Physical	IoT Devices	Power over Ethernet (PoE) failure	Verify PoE configuration and cable quality	Identifying hardware faults from logs
2	Data Link	Mobile Phones	Bluetooth pairing keeps failing	Re-pair and reset session manager	Troubleshooting wireless session issues
2	Data Link	Switches	Unicast flooding observed	Check CAM table overflow and port security	Layer 2 forwarding anomalies
2	Data Link	Printers	Printer appears offline despite connectivity	Verify LLDP/CDP neighbor discovery	Device discovery and inventory management
3	Network	Routers	Subnet mask misconfiguration causing limited scope	Correct subnet mask and verify gateway	Detecting IP range overlaps

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VII, July 2026

4	Transport	NAS	File copy stalls	Check SMB over TCP behavior and retransmission rates	Diagnosing transport-layer file transfer issues
4	Transport	Switches	Telnet/SSH timeouts	Ensure TCP 23/22 is open and unblocked	Monitoring Layer 4 service availability
5	Session	APs	Client disassociates unexpectedly	Investigate 802.11r or PMK caching	Session mobility Layer 5 analysis
5	Session	IoT Devices	Session timeout too short	Extend keepalive interval	Session lifetime tuning
6	Presentation	Servers	Encryption mismatch during DC promotion	Reset KRBTGT password and validate encryption types	Detecting presentation layer mismatches
6	Presentation	IoT Devices	Incompatible SSL/TLS versions	Upgrade firmware to support modern TLS	Detecting crypto compatibility issues
7	Application	Laptops	Outlook wonâ€™t connect to Exchange	Check Autodiscover and mailbox permissions	Diagnosing email client failures
7	Application	NAS	Web UI inaccessible	Restart web service or check HTTPS binding	Diagnosing app-level UI failures

Table 4: Sample of NSL-KDD dataset used for network threat detection



duration	protocol_t	service	flag	src_bytes	dst_bytes	land	wrong_fra	urgent	hot	num_faile	logged_in	num_com	root_shell	su_attemp	num_root	num_file_	num_shell	num_acce	num_outb	is_host_lo	is_guest_l	count	srv
0	tcp	ftp_data	SF	491	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	
0	udp	other	SF	146	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	
0	tcp	private	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	123	
0	tcp	http	SF	232	8153	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	5	
0	tcp	http	SF	199	420	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	30	
0	tcp	private	REJ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	121	
0	tcp	private	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	166	
0	tcp	private	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	117	
0	tcp	remote_jo	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	270	
0	tcp	private	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	133	
0	tcp	private	REJ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	205	
0	tcp	private	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	199	
0	tcp	http	SF	287	2251	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	3	
0	tcp	ftp_data	SF	334	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	2	
0	tcp	name	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	233	
0	tcp	netbios_ns	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	96	
0	tcp	http	SF	300	13788	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	8	
0	icmp	eco_i	SF	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
0	tcp	http	SF	233	616	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	3	
0	tcp	http	SF	343	1178	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	9	
0	tcp	mtp	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	223	
0	tcp	private	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	280	
0	tcp	http	SF	253	11905	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	8	
5607	udp	other	SF	147	105	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
0	tcp	mtp	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	248	
507	tcp	telnet	SF	437	14421	0	0	0	0	0	1	3	0	0	0	0	0	1	0	0	0	1	

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VII, July 2026

Instantiation of the Model

The term “model instantiation" refers to putting a designed machine-learning framework into place that performs all functions of classifying network problems, providing corrective solution(s), and predicting otherwise possible future network threats. There are essentially two functions to be carried out by the machine-learning solution built for this study. Thus, we will discuss 2 modules: a module for classifying network problems and providing recommendation(s) for solutions, and a module for predicting or detecting possible future network threats.

In the following sections, the process of designing each of the 2 modules will be addressed with a discussion of the various methods/algorithms utilized for design.

Network Issue Classification & Solution Recommendation: The Python tools required for their loading, preprocessing, analysing of data, training and testing of the model(s) were loaded, imported. The next step in this module’s implementation involves the loading of the dataset sourced from an ICT staff member at OAGF that contains network problems mapped to an OSI layer as well as suggested solutions, which is used to train the machine learning system. At this stage, (when we load the dataset) the dataset needs to be preprocessed. This includes converting text to lower case, normalising, tokenising and lemmatisation so that the model trains on clean, reliable data. Outlined below are brief explanations of the steps:

Splitting the dataset: A preprocessed dataset was split into 80% training and 20% testing using train_test_split. The input feature was “symptom” (X value), and the output feature was “OSI_LAYER” (Y value).

The Model Selection and Hyperparameter Optimization Process: A total of four algorithms used to create a model (Naïve Bayes, Logistic, Random Forest, Linear SVM) were created through a hyperparameter search for each algorithm. A pipeline that used a TF-IDF vectorizer was constructed for each algorithm/final model. The previously defined text data was then numerically transformed using the pipeline and used to fit the individual algorithms which were then searched to produce optimal hyperparameters via RandomizedSearchCV.

Network Threat Prediction: Installing Importing Python Data Science Stack & Necessary Tools: Installed and imported the Python data science stack and tools needed to load, pre-process and analyze data, train and evaluate models.

Loading & Cleaning Dataset: The NSL-KDD dataset was loaded into the Python environment from which model builds were created. Errors/inconsistencies in the dataset were removed so that the dataset quality will be improved. Duplicate record(s) were removed and missing/null records were handled accordingly.

Feature Engineering: Features from the dataset were scaled and encoded to create new features to improve model performance; at the same time, VIF (Variance Inflation Factor) was used during feature engineering to check correlation as well as reduce multicollinearity in the dataset.

Exploratory Data Analysis: Activities performed to uncover patterns, identify anomalous data, and analyze data included statistical measurement, visualizing features in the dataset, detecting outliers (which were identified as anomalous data) and visualizing correlation patterns.

Scaling and Normalization: To ensure that all numerical features are scaled correctly, as the first step in preparing machine learning models for use, we applied scaling to standardize all numerical features. The purpose of scaling was to adjust every feature so that they all fit into a common range and, therefore, allow them to all operate overall the same way concerning the model, so as to not bias the model, and to allow me to normalize the entire dataset.

Stratified Train and Test Split: For my two-part dataset, I first divided the data into 80 percent training data and 20 percent testing data, and then used stratification to ensure that the same proportion of classes (attack vs. normal) was approximately equal in both the training and testing set.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VII, July 2026

Anomaly Detection with Unsupervised Models: We trained several unsupervised anomaly detection algorithms on the training dataset as independent models. Each unsupervised algorithm was run on the training dataset to create an unsupervised anomaly detection model for each algorithm. After creating the model for each algorithm, we then extracted an outlier score for each record from each algorithm and added the score as a new feature to the dataset. An example of unsupervised algorithms used as part of this work include Isolation Forest, DBSCAN, Gaussian Mixture Model, Local Outlier Factor (LOF), Elliptic Envelope and One-Class SVM and KNN. UMAP was used to visualize the distribution (anomolous vs. normal) of the data within the dataset due to the high dimensionality of the dataset.

Managing Class Imbalance: The Synthetic Minority Over-sampling Technique (SMOTE) was employed to create a synthetic sample of the minority class to create a balanced dataset for binary classification.

Supervised Models were trained: A variety of supervised classification algorithms were trained on both balanced and unbalanced datasets; hyper-parameter tuning was also performed using Logistic Regression, KNN, Decision Tree, and Adaboost classification techniques.

Evaluation of Models was conducted: Model Evaluation Metrics (Classification) included Accuracy, Precision, Recall, and F1 Score to assess the performance of each of the supervised classification algorithms.

Model Selection: Of the supervised classification algorithms, Adaboost produced the best results based on the results of the evaluation of the performance from each of the supervised classification algorithms. Adaboost was chosen as the algorithm used to perform the binary classification of the data.

Model Serialization: The models developed, and their resulting model was serialized (converted to an appropriate format for use in production), which are then subject to pickle serialization for the purpose of deploying the chosen supervised classification model.

System Software Implementation Processes

To ensure the development and performance of a machine learning system can be carried out effectively, the appropriate development environment was selected. This environment consisted of a series of tools and technologies that enable the development and implementation phases of the project to be as flexible and efficient as possible. The development and implementation of the machine learning system took place in a Windows 11 environment, with the primary development environment being Visual Studio Code (VS Code), an open-source text editor. The VS Code application provides an integrated workspace to execute scripts, scripts written in .html, .css, python (.py), and Jupyter Notebooks (.ipynb) formats, allowing for the completion of all phases of data preprocessing, model training, and implementing the machine learning system in one environment. VS code is cross-platform, and due to its lightweight design, only requires minimal system resources, allowing developers to install extensions as needed, based on their individual projects. Additionally, VS Code has built-in features such as syntax highlighting, code intelligence, and semantic features such as intelligent code completion, function and variable suggestions, and real-time error reporting to assist with code development, maintain logical consistency, and help to eliminate bugs. [28]

Python was selected as the programming language due to its versatility, ease of use, and extensive library ecosystem [29], allowing machine learning developers to access libraries that contain the necessary tools and functionality to manipulate and analyze/understand the data to create machine learning model. The readability of the Python programming language, along with its ability to run on multiple platforms, provides an important advantage for this project [30]. The web application was created using Flask, which is a lightweight and fast web framework that allows developers to easily set up their applications according to their requirements. Flask is built around the use of technologies such as the Python programming language, and as such, it integrates seamlessly with other Python libraries and frameworks [28]. This combination of the underlying operating system, the development environment and the programming tools has created a stable and productive environment for developing and evaluating the final product being proposed.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VII, July 2026

RESULT AND DISCUSSION

Software Testing

Software testing was an essential stage in the development of the machine learning system to ensure its accuracy, stability and overall performance. The testing strategy(s) adopted focused on verifying that each module performed its intended function and that the integrated system met both technical and functional expectations/requirements. The testing process was conducted in two main stages: "Testing during development" and "Local host testing".

Testing During Development: At the development stage, each module of the system was individually examined as it was built to validate its performance and detect possible defects. The network issue classification and solution recommendation model was trained on a dataset containing various network faults, their OSI Layers, and the corresponding solution(s) amongst other features.

Testing involved checking for run-time errors, checking whether the model(s) could correctly identify the OSI layer of each fault and suggest the right solution(s). Metrics like accuracy, precision, recall and F1-score were used to evaluate its performance. Naïve Bayes had the best performance, below are the values of the evaluation: **Accuracy:** 0.8388, **Precision:** 0.8407, **Recall:** 0.8388, **F1-Score:** 0.8385.

The threat detection model which was trained on the NSL-KDD dataset also went through the phase of testing during development. Throughout this phase, debugging was done continuously. While analyzing the data and preparing it for training (also during the process of training), issues were identified and fixed, parameters were adjusted, some features were dropped, amongst other implementations to achieve efficiency.

Metrics like accuracy, precision, recall, F1-score, ROC-AUC curve, confusion matrix were also used here to evaluate the performance of the model. Adaboost on the imbalanced test set produced the best performance based on the evaluation metrics, the result are as follows: An accuracy of 0.9928, precision of 0.9952, recall value of 0.9898, and F1-score of 0.9925.

The Flask backend was tested to confirm that it correctly handled HTTP requests, processed inputs from the web interface and communicated effectively with the MySQL database. Flask's built-in debugger, python's logging functions were used extensively to trace and fix errors in routing, data handling, and API responses.

1) Testing During Development: After the development testing phase, the fully integrated system was deployed on a local host server for functionality and performance evaluation. This phase simulated how end-users would interact with the application through a web browser.

2) The focus of this phase was observing how all components/modules worked together and whether the system could handle real input data effectively. The network classification and solution recommendation model was tested by providing sample network faults to ensure they were correctly classified based on the OSI Layer and the corresponding solution(s) were generated. The threat detection model was also tested to verify it identified whether a network traffic was malicious or not based on the inputs from the user interface. The details of the network issue/problem were entered into the input box provided in fig. 1, the submit button was clicked on, and the OSI Layer and recommended solution displayed as shown in fig.2. In fig.3, the appropriate values for all the network traffic details in the form displayed were selected and submitted. On clicking on the predict button, the result of the prediction was displayed as shown in fig.5.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VII, July 2026

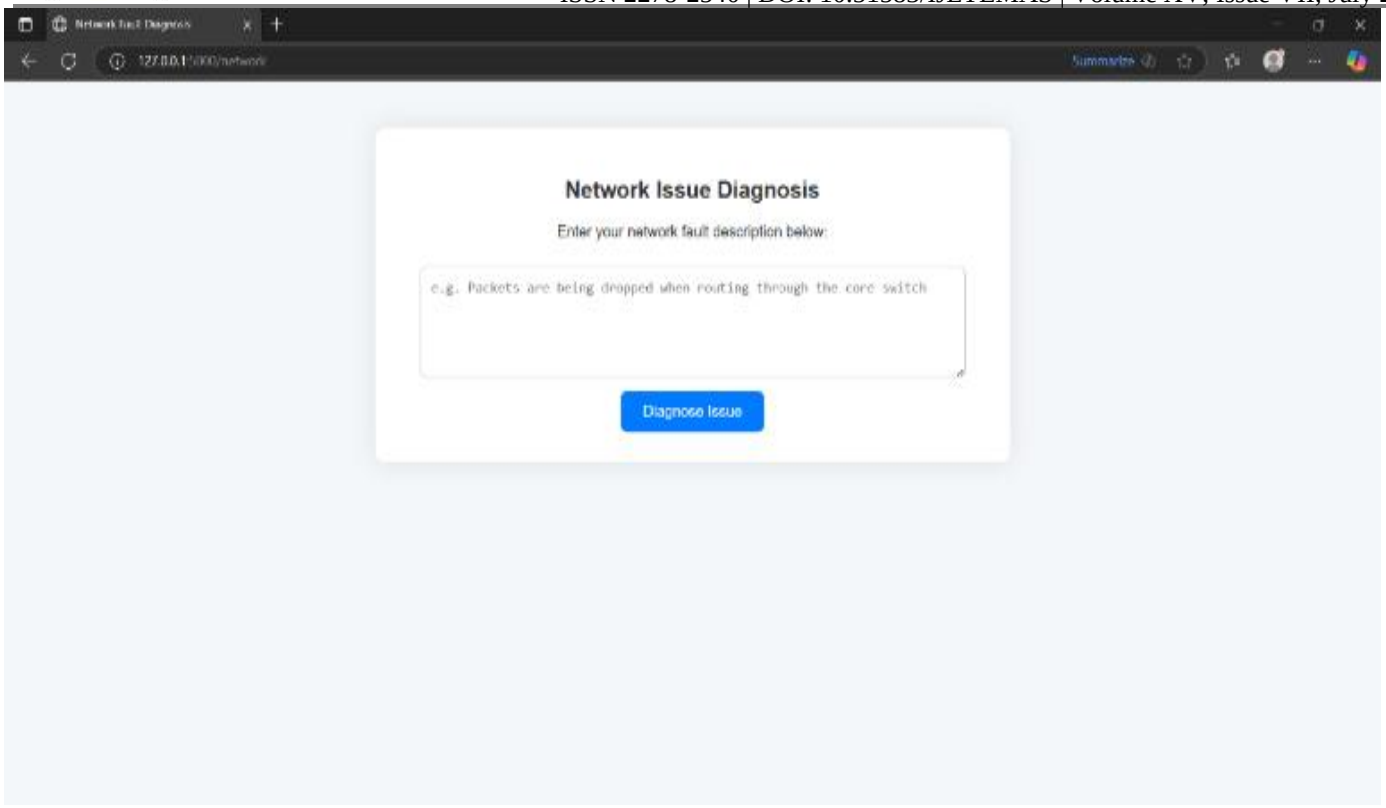


Fig.1. Network Issue classification/Recommendation Input Interface

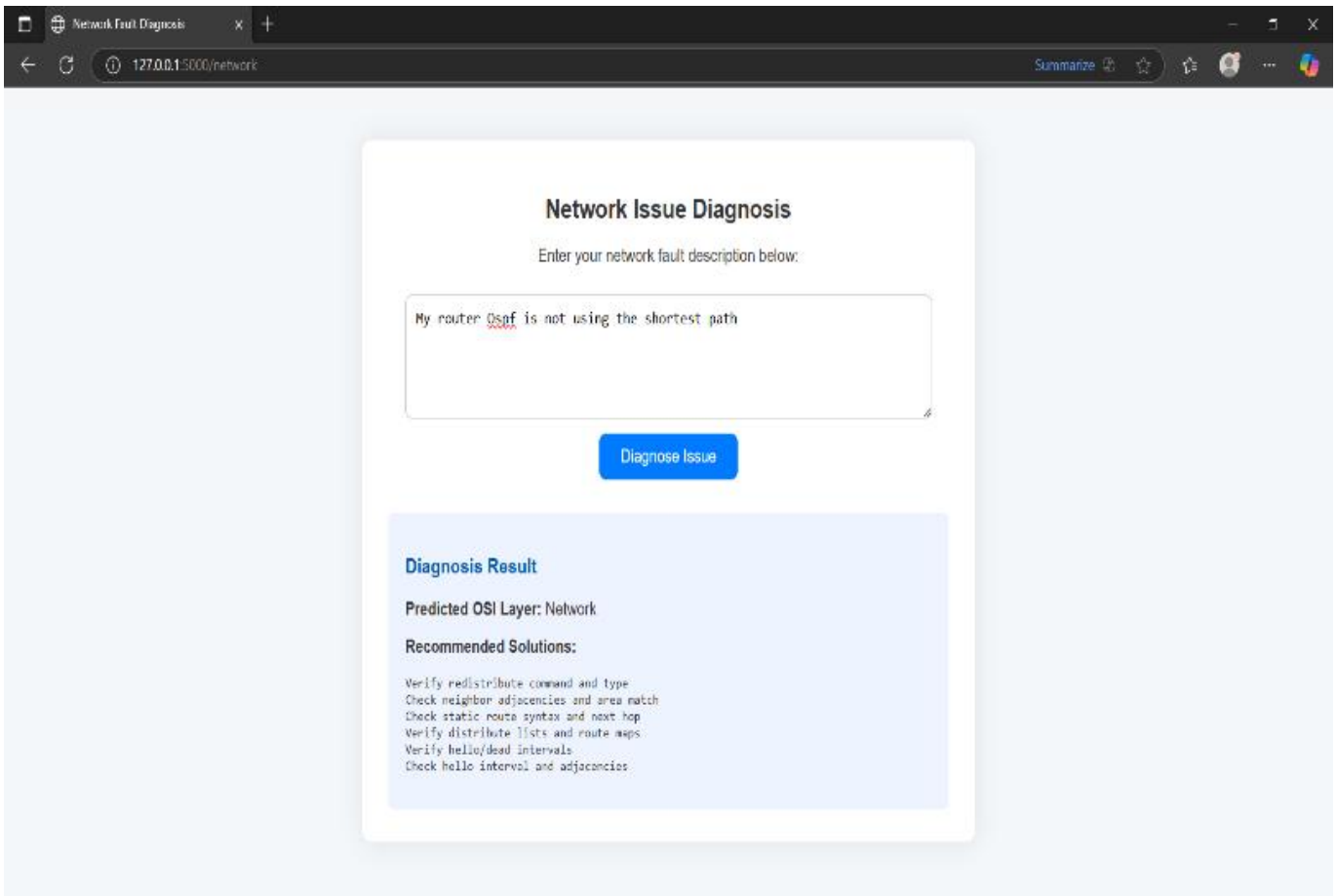


Fig. 2. Network Issue classification/Recommendation Result

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VII, July 2026

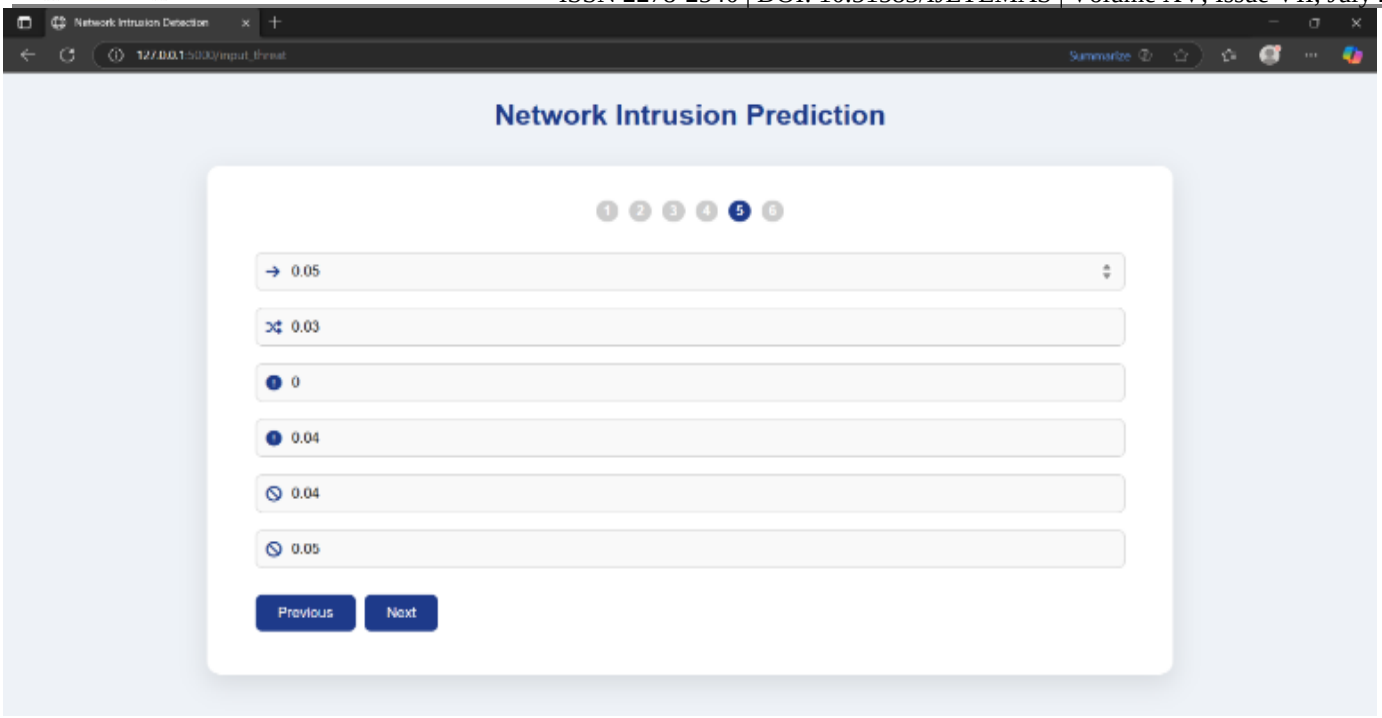


Fig. 3. Network Threat Prediction Interface

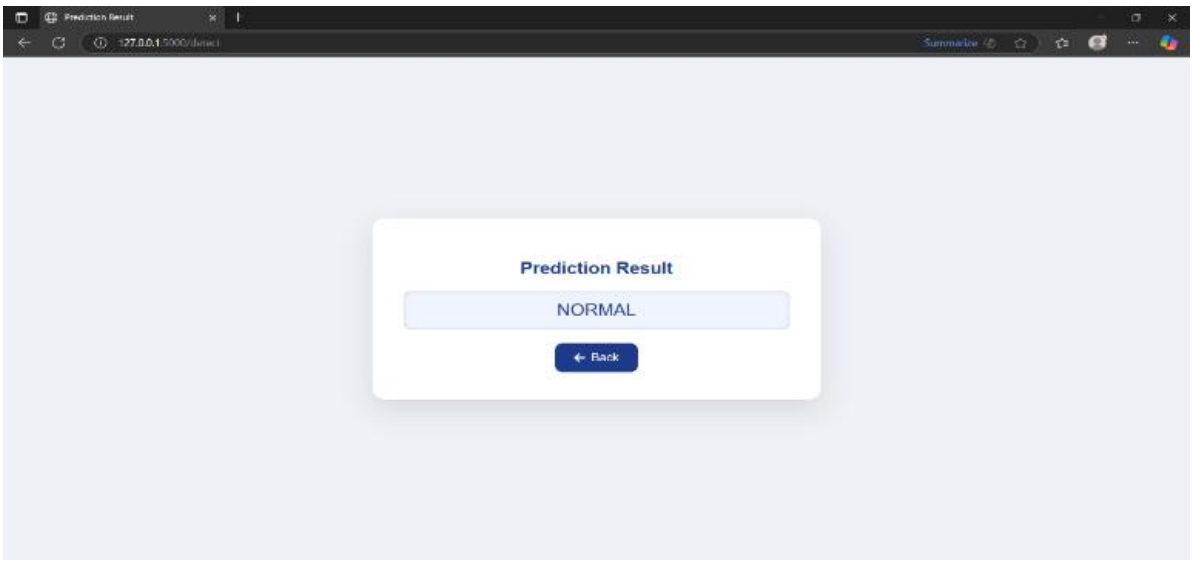


Fig. 4. Network Threat Prediction Result

CONCLUSION

This paper develop a system that enhances network management by localizing network issue while providing solution(s) and predicting malicious activities on the network (threats) that could be exploited. Through careful data collection, data analysis, system design, implementation and testing, it has been demonstrated that machine learning can play a significant role in supporting network operations.

Summarily, the paper successfully achieved its objectives, demonstrating how intelligent systems can improve efficiency and reliability in network environments. It shows how data-driven approaches can complement traditional network management practices. This work provides a basis for future improvements and for building more efficient, automated network support systems. Future work could be on the implementation of real-time

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VII, July 2026

monitoring to provide continuous insights and faster detection of faults and threats while still incorporating an interactive interface with users for further understanding or support.

REFERENCES

1. Smith, J. A. (2020.) "Automation of network management and incident response," International Journal of Artificial Intelligence and Machine Learning in Engineering, 15(16). Department of Information Technology.
2. Kirch, O., & Dawson, T. (2000). "Linux Network Administrator's Guide," 2nd ed. The Linux Documentation Project. [Online]. Available: https://tldp.org/LDP/nag2/index.html
3. Usama, M., Qadir, J., Raza, A., Arif, H., Yau, K.-L. A., Elkhatib, Y., Hussain, A., & Al-Fuqaha, A. (2019). "Unsupervised machine learning for networking: Techniques, applications and research challenges," IEEE ACCESS, 7, 65579-65615.. https://doi.org/10.1109/ACCESS.2019.2916648
4. Miller, R.L. (2025). "The OSI model: An overview," SANS Institute, GSEC Practical Assignment Version 1.2e. [Online]. Available: https://www.giac.org/paper/gsec/1417/odi-model-overview/102634
5. kanade, V. (2022). "What is the OSI model? Definition, layers, and importance," Spiceworks, Nov. 1. [Online]. Available: https://www.spiceworks.com/tech/networking/articles/what-is-osi-model/
6. Simoneau, P (2026). "The OSI Model: Understanding the seven Layers of Computer Networks [White Paper]." Global Knowledge Training LLC. Retrieved from https://faculty.sfcc.spokane.edu/Rudlock/files/WP_Simoneau_OSIModel.pdf
7. Kumar, S., Dalal, S., & Dixit, V. (2014). "The OSI model: Overview on the seven layers of computer networks," International Journal of Computer Science and Information Technology Research, vol. 2, no. 3, pp. 461–466. [Online]. Available: https://www.researchpublish.com/upload/book/THE%20OSI%20MODEL%20OVERVIEW%20ON%20THE%20SEVEN%20LAYERS-607.PDF
8. Terra, J. (2025). "What is a computer network? Definition, components, objectives, and best practices.," Simplilearn. Retrieved from https://www.simplilearn.com/what-is-computer-network-article
9. Tanenbaum, A. S., & Wetherall, D. J. (2011). "Computer networks (5th ed.)," Pearson Education.. Retrieved from https://csc-knu.github.io/sys-prog/books/Andrew%20S.%20Tanenbaum%20-%20Computer%20Networks.pdf
10. Buddha, D. "7 common network issues and how to solve them," LinkedIn, Nov. 22, 2024. [Online]. Available: https://www.linkedin.com/pulse/7-common-network-issues-how-solve-them-deepanshu-budhija-bqecc
11. UMBOSS, "Network fault management," Retrieved from https://umboss.com/blog/network-fault-management/
12. Fortinet, "Network security threats," Accessed: May 13, 2026. [Online]. Available: https://www.fortinet.com/tw/resources/cyberglossary/network-security-threats
13. Darktrace, "Network security threats," Accessed: May 13, 2026. [Online]. Available: https://www.darktrace.com/cyber-ai-glossary/network-security-threats
14. Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2018) "Foundations of Machine Learning," 2nd ed. Cambridge, MA, USA: MIT Press. [Online]. Available: MIT Press Book Page. [Accessed: May 19, 2026].
15. Deuschle, William J. (2019). "Undergraduate Fundamentals of Machine Learning," Bachelor's thesis, Harvard College. [Online]. Available: https://nrs.harvard.edu/URN-3:HUL.INSTREPOS:37364585
16. Tagliaferri, L., Morales, M., Birbeck, E.,& Wan, A. (2019). "Python Machine Learning Projects," New York, NY, USA: DigitalOcean. [Online]. Available: DigitalOcean PDF.
17. Malla Reddy College of Engineering & Technology (2020). Machine Learning [R17A0534]: Lecture Notes, Dept. Computer Science and Engineering. [Online]. Available: https://mrcet.com/downloads/digital_notes/CSE/iv%20Year/MACHINE%20LEARNING(R17A0534).pdf
18. University of York, "What is reinforcement learning?," University of York Online, n.d. [Online]. Available: https://online.york.ac.uk/resources/what-is-reinforcement-learning/

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VII, July 2026

19. Mohammed, A. R., Mohammed, S. A., Côté, D., & Shirmohammadi, S (2021). "Machine learning-based network status detection and fault localization," IEEE Transactions on Instrumentation and Measurement, vol. 70, Art. no. 3521710, doi: 10.1109/TIM.2021.3094223.
20. Feng, Caiying, and Yan Zhao (2022). "Fault identification and analysis of communication network based on deep learning," Mobile information systems 1: 1456425.
21. Dhanya, K. A., Vajipayajula, S., Srinivasan, K., Tibrewal, A., Kumar, T. S., & Kumar, T. G. (2023), "Detection of network attacks using machine learning and deep learning models," Procedia Computer Science, vol. 218, pp. 57–66. doi: 10.1016/j.procs.2022.12.401.
22. Priya, C. S., Somjiyani, G., Thulasi, B., Yashaswini, V., & Shivani, P. (2025). "IoT network traffic classification using machine learning algorithms," International Journal of Engineering.
23. Crabtree, M. (2024). "What is machine learning? DataCamp. Retrieved from https://www.datacamp.com/blog/what-is-machine-learning
24. Muller, A.C., & Guido, S. (2026). "Introduction to machine learning with Python: A guide for data scientists"' O'Reilly Media. Retrieved from https://www.nrigroupindia.com/e-book/Introduction%20to%20Machine%20Learning%20with%20Python%20%28%20PDFDrive.com%20%29-min.pdf
25. Gupta, A. (2025). "An introduction to Scikit-learn: Machine learning in Python (Lesson 28 of 52," Simplilearn. Retrieved from https://www.simplilearn.com/tutorials/python-tutorial/scikit-learn
26. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). "Scikit-learn: Machine learning in Python." Journal of Machine Learning Research, 12, 2825–2830. Retrieved from https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf
27. Andersen, N.B. (2023), "What is Pandas in Python?. Retrieved from https://builtin.com/data-science/pandas
28. Microsoft Corporation. (2025). "Intellisense – Visual Studio Code documentation." Retrieved from https://code.visualstudio.com/docs/editing/intellisense
29. Netguru. (2025). "Python and machine learning: Why Python is great for ML projects. Netguru." Retrieved from https://www.netguru.com/blog/python-machine-learning
30. GeeksforGeeks. (2025). "Machine learning with Python," Retrieved from https://www.geeksforgeeks.org/machine-learning/machine-learning-with-python/