*Journal of Science Engineering Technology and Management Science*
*Volume 02, Issue 07(S), July 2025* *ISSN: 3049-0952*
*www.jsetms.com* *DOI:10.63590/jsetms.2025.v02.i07(S).11-19*

# KNOWLEDGE-AWARE IOT MAINTENANCE PREDICTION USING DEEP LEARNING ON STACK OVERFLOW SECURITY DATA

Dr. Pulime Satyanarayana, Chepuri Bharath Kumar, Pagidala Bharath Kumar, Godugu Elishan

*Department of Computer Science and Engineering (AI&ML), Kommuri Pratap Reddy Institute of Technology, Hyderabad, Telangana, India.*

***To Cite this Article***
*Dr. Pulime Satyanarayana, Chepuri Bharath Kumar, Pagidala Bharath Kumar, Godugu Elishan, "Knowledge-Aware Iot Maintenance Prediction Using Deep Learning On Stack Overflow Security Data", Journal of Science Engineering Technology and Management Science, Vol. 02, Issue 07(S), July 2025,pp: 11-19, DOI: http://doi.org/10.63590/jsetms.2025.v02.i07(S).pp11-19*
*Submitted: 25-05-2025* *Accepted: 03-07-2025* *Published: 11-07-2025*

**ABSTRACT**
The rapid expansion of the Internet of Things (IoT) has transformed numerous industries by enabling real-time data exchange and automated operations. However, this growth brings significant challenges in maintaining robust security and efficient system performance. Recent projections indicate that the number of IoT devices will surpass 75 billion by 2025, resulting in a massive, interconnected network vulnerable to security threats and maintenance difficulties. Traditional security mechanisms, which rely on manual monitoring and fixed threshold-based detection, are increasingly inadequate in dynamic IoT environments. These conventional approaches often fail to detect new and evolving threats in real time and struggle with predictive maintenance, leading to higher downtime and degraded system performance. Additionally, the heterogeneous nature of IoT devices and the massive data they generate make it difficult for legacy systems to respond quickly and effectively. These limitations highlight the urgent need for more advanced solutions. This research explores the application of deep learning (DL) models to meet these challenges by focusing on security threat detection, performance forecasting, and predictive maintenance in IoT ecosystems. DL techniques offer the ability to process large-scale data, identify anomalies indicative of security issues, and forecast maintenance requirements before failures occur. The strength of this approach lies in its capacity to enhance IoT system reliability and security, reduce downtime risk, and improve performance management. Ultimately, this research aims to enable a shift from reactive to proactive management of IoT systems, supporting their scalability and complexity while ensuring secure and efficient operation for critical applications.

**Keywords:** Internet of Things (IoT),

## 1. INTRODUCTION

The Internet of Things (IoT) has emerged as a transformative force in modern technology, bringing unprecedented levels of connectivity and automation. IoT systems involve networks of interconnected devices capable of collecting, transmitting, and processing data. In recent years, India has witnessed rapid adoption of IoT technologies across industries such as agriculture, healthcare, manufacturing, transportation, and smart cities. According to a report by the Indian government, India is projected to become one of the largest IoT markets globally, with an estimated 2.9 billion connected devices by

2020. This surge in IoT adoption has raised new concerns about security vulnerabilities and the maintenance of these complex systems.

Historically, IoT devices have been vulnerable to various security threats. The first major instance of IoT security breaches came with the Mirai Botnet attack in 2016, which exploited weak security in IoT devices, leading to large-scale Distributed Denial-of-Service (DDoS) attacks. In India, a similar trend has been observed, where vulnerabilities in IoT systems have resulted in breaches and disruptions in critical sectors like healthcare, banking, and manufacturing. According to reports, over 60% of IoT devices in India are unsecured, leaving them susceptible to cyberattacks.

Traditional security mechanisms, such as password-based protection, firewalls, and intrusion detection systems, have proven to be inadequate for the rapidly evolving IoT landscape. These methods are typically static and fail to recognize new, unknown threats in real time. Furthermore, with the vast amounts of data generated by IoT devices, traditional systems face challenges in terms of scalability and efficiency. Similarly, the maintenance of IoT systems remains a persistent problem, as traditional methods of monitoring and diagnosis are inefficient at predicting system failures and optimizing performance.

The rise of deep learning (DL) and machine learning (ML) has opened up new possibilities for improving IoT security and performance management. DL models can analyze vast amounts of data in real time, offering solutions for predictive maintenance and enhanced security detection. The combination of IoT with artificial intelligence (AI) offers the potential for intelligent systems that can adapt, predict, and respond autonomously, ensuring more efficient operations.

## 2. LITERATURE SURVEY

In general, IDSs are classified into two primary categories: host intrusion detection systems (HIDSs) and network intrusion detection systems (NIDSs) [1]. HIDSs are installed on individual hosts to track all activities occurring on the host itself. Conversely, NIDSs are implemented to protect all network-connected devices from potential security breaches. IDSs can also be categorized according to their detection methodologies: rule-based or signature-based IDSs and anomaly-based IDSs [2]. Rule-based IDSs detect familiar attack patterns by comparing them against a database of stored signatures [3], but they struggle with zero-day or novel attacks when their signature databases are not updated. In contrast, anomaly-based IDSs analyze network activities to identify patterns of normal and abnormal behavior, enabling them to effectively detect unknown types of attacks [4, 5].

The current intrusion detection datasets still suffer from the curse of dimensionality, which arises when handling high-dimensional data and significantly affects IDS performance. Conventional feature extraction methods, including linear discriminant analysis and principal component analysis, are commonly used to address high-dimensional data in IDS contexts, but they often fall short. Deep learning (DL) is a groundbreaking subset of machine learning that has achieved significant success in automating feature extraction from data [6]. DL employs multilayer neural networks that learn data representations through iterative mathematical operations. It excels in processing large-scale data efficiently, demonstrating successes across diverse fields like medical imaging [7], natural language processing [8], sentiment analysis [9], healthcare [10], and image recognition [11]. Convolutional Neural Networks (CNNs) play a vital role in DL, particularly in performing feature extraction [12]. Although effective, CNNs face challenges such as vanishing and exploding gradients, which are effectively addressed by architectures like the Residual Network (ResNet) [13]. ResNet can extract deep features while mitigating gradient-related problems, though its complex structure with multiple layers and parameters presents implementation challenges. Furthermore, relying on a single algorithm in IDS frameworks may compromise efficiency and hinder the identification of complex, multiclass attacks.

## 3. PROPOSED SYSTEM

The detailed analysis of proposed system is presented in Figure 4.1 and Figure 4.2. The operational steps given as follows

The first step involves uploading the Stack Overflow dataset, which contains textual data related to security questions and discussions. The dataset is loaded using the pandas library and displayed to provide an overview of the data.

Natural Language Processing (NLP) techniques are applied to preprocess the textual data. The dataset is tokenized using the Tokenizer from TensorFlow's Keras library. The text sequences are converted into numerical form and padded to a fixed length using pad_sequences to ensure uniformity in input dimensions.

Since the dataset have imbalanced classes, SMOTE is applied to balance the dataset. This technique generates synthetic samples for the minority class to enhance classification performance and prevent bias toward majority classes. The resampled dataset is then used for training.

**Proposed FFNN with XGBoost Classifier:** A proposed FFNN with XGBoost classifier is designed and trained using TensorFlow or Keras. It consists of multiple layers, including feature extraction and classification layers. The trained FFNN model is evaluated against the test dataset, and performance metrics are compared with GBC.

**Performance Comparison Metrics and Graph:** The performance of both GBC and FFNN classifiers is assessed using key metrics such as accuracy, precision, recall, and F1-score. A confusion matrix is generated for each model, and a bar graph is plotted to visualize the comparative performance of the models.

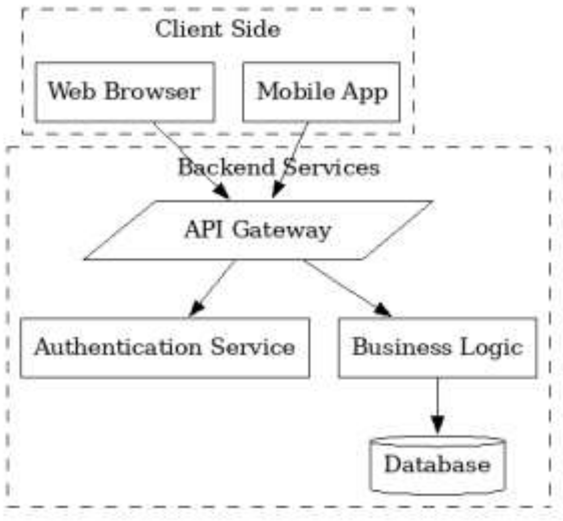


Fig. 1: Application Scenario.

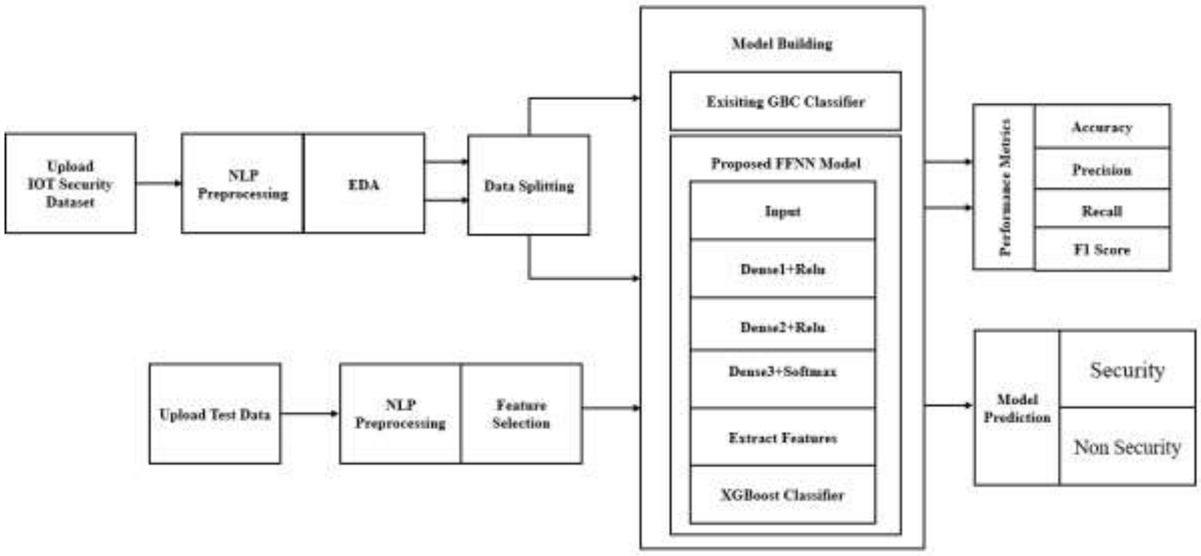


Fig. 2: Architectural Block Diagram of the Proposed System.

Once the FFNN model is trained and evaluated, it is used to predict the output for new test data. The test data is pre-processed in the same way as training data, and predictions are made using the trained FFNN model. The predicted labels are then displayed along with the input test samples for final analysis.

**Proposed FFNN Feature Extraction**

The Feed Forward Neural Network (FFNN) is a type of artificial neural network where information flows in one direction from input to output without cycles or loops. It is the most basic form of deep learning model used for both classification and regression tasks.

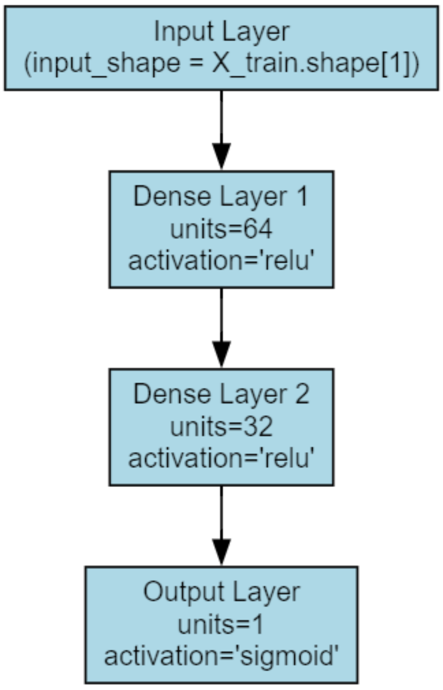


Fig. 3: Proposed FFNN Architecture.

The FFNN method is an advanced neural network architecture designed to detect complex, non-linear patterns within high-dimensional datasets, such as those used in internet stack-overflow IoT security detection. In this specific application, stack-overflow IoT applications often contain subtle correlations between features that may not be captured by traditional models. The FFNN architecture excels in learning such intricate relationships, making it suitable for distinguishing between genuine and security applications. Its ability to automatically learn hierarchical feature representations through multiple layers gives it a significant advantage in security detection scenarios where security behaviors evolve and vary widely. Additionally, FFNN models are highly scalable and adaptable, making them ideal for real-time, large-scale financial applications.

**Input Layer Initialization**: The process begins with the input layer, where each node corresponds to a feature in the preprocessed stack-overflow IoT application dataset. These features may include numerical or encoded categorical values such as transaction amount, account type, origin and destination identifiers, and more. This layer serves as the entry point for data into the neural network.

**Hidden Dense Layer Construction**: After the input layer, multiple hidden dense layers are added to the network. Each hidden dense layer consists of several neurons that apply weighted transformations to the inputs followed by a non-linear activation function, such as ReLU. These layers are responsible for capturing complex interactions between features, gradually learning higher-level abstractions and patterns that help differentiate between security and legitimate applications.

**Activation Function Application**: Activation functions are applied within each hidden dense layer to introduce non-linearity into the model, allowing it to learn from complex data. ReLU (Rectified Linear Unit) is commonly used due to its efficiency and effectiveness in avoiding vanishing gradient issues. This step ensures the network can capture intricate variations in data, crucial for security detection.

**Dropout and Regularization**: To prevent overfitting and improve generalization, dropout layers may be introduced between hidden dense layers. Dropout randomly disables a fraction of neurons during training, ensuring that the model does not rely too heavily on specific pathways. Additionally, techniques like L2 regularization can be applied to penalize large weights and promote simpler, more robust models.

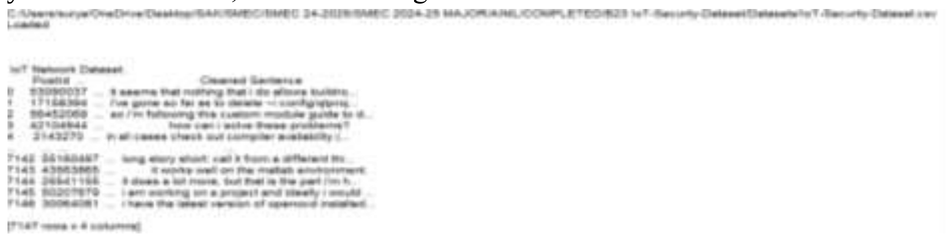
**Output Layer and Classification**: The final layer in the FFNN model is the output layer, which typically uses a sigmoid or softmax activation function depending on whether binary or multi-class classification is required. In security detection, a sigmoid function is often used to output a probability score indicating the likelihood of a transaction being security.

**Backpropagation**: The FFNN is trained using backpropagation and an optimization algorithm like Adam or SGD. During training, the model adjusts weights based on a loss function such as binary cross-entropy. The optimization aims to minimize the difference between predicted and actual labels, improving the model's ability to detect security with high accuracy.

**Model Evaluation**: Once trained, the FFNN model is evaluated using metrics such as accuracy, precision, recall, F1-score. These metrics help assess the model's performance on unseen data. After validation, the trained FFNN can be deployed into a real-time security detection system to continuously analyze incoming stack-overflow IoT applications.

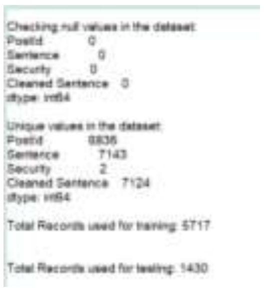
## 4. RESULTS AND DISCUSSION

Figure 4 illustrates the initial stage of the project, where the Secure IoT dataset is uploaded through the graphical user interface (GUI). The dataset, which contains textual data related to security, is loaded into the system for further processing. The interface provides a summary of the dataset, displaying key statistics such as the total number of records, the distribution of security-related and non-security-related sentences, and initial insights into the dataset's structure.



```
C:/Users/surya/OneDrive/Desktop/SAK/SMEC/SMEC 24-2026/SMEC 2024-25 MAJOR/ANIL/COMPLETED/B23 IoT-Security-Dataset/Datasets/IoT-Security-Dataset.csv
Loaded

IoT Network Dataset
      PostId  ...                              Cleaned Sentence
0    53090037  ...   it seems that nothing that i do allows buildro...
1    17158394  ...   i've gone so far as to delete ~/.config/qtpro...
2    56452069  ...   as i'm following the custom module guide to d...
3    42104844  ...                   how can i solve these problems?
4     2143270  ...  in all cases check out compiler availability (...
...
7142  35180487  ...    long story short: call it from a different thr...
7143  43565865  ...            it works well on the matlab environment.
7144  26841156  ...    it does a lot more, but that is the part i'm h...
7145  50207879  ...    i am working on a project and ideally i would ...
7146  30064081  ...    i have the latest version of opencv installed...

[7147 rows x 4 columns]
```

Fig. 4: Upload of Secure IoT Dataset and Its Analysis in the GUI Interface



```
Checking null values in the dataset:
PostId          0
Sentence        0
Security        0
Cleaned Sentence   0
dtype: int64

Unique values in the dataset:
PostId          6836
Sentence        7143
Security        2
Cleaned Sentence   7124
dtype: int64

Total Records used for training: 5717


Total Records used for testing: 1430
```

Fig. 5: Data Preprocessing of the Dataset

Figure 5 showcases the preprocessing steps applied to the dataset to prepare it for model training. The preprocessing involves cleaning the text data by removing unwanted characters, stopwords, and special symbols. Tokenization is performed to break down sentences into individual words, followed by lemmatization or stemming to reduce words to their base forms. This step ensures that redundant and irrelevant information is eliminated while preserving the essential meaning of the text.

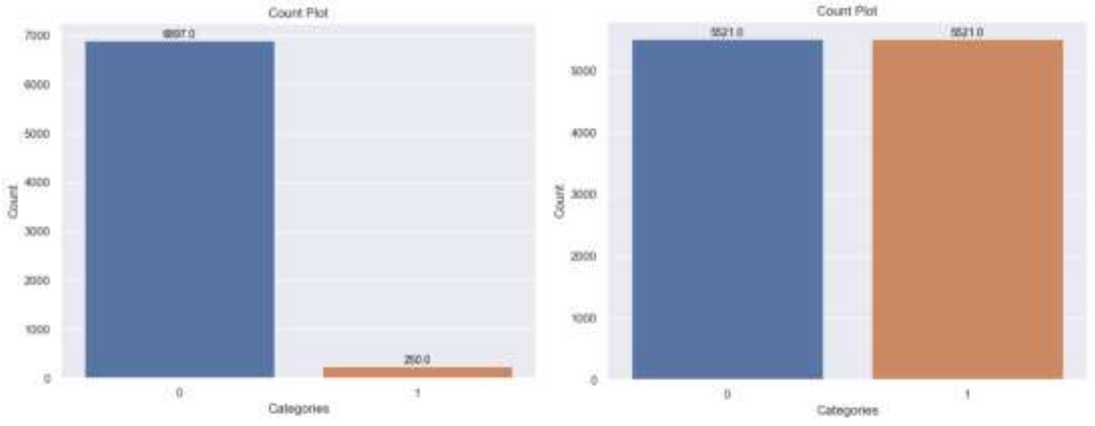
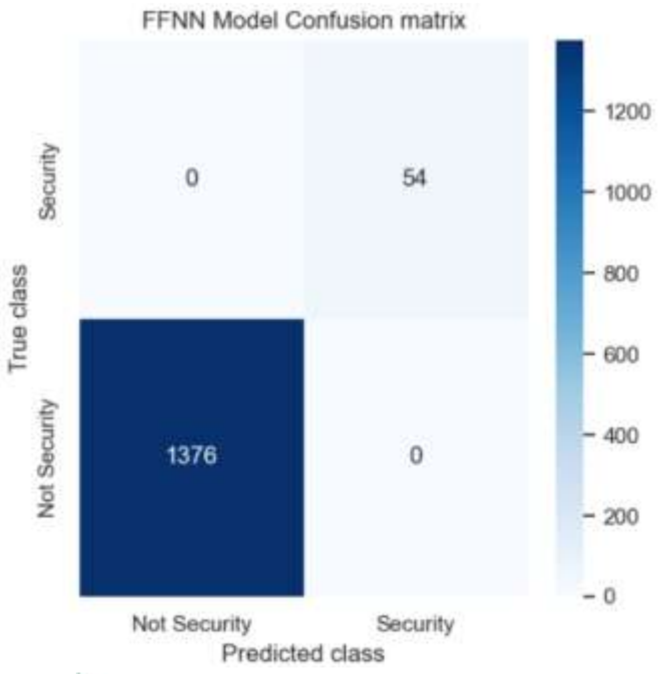
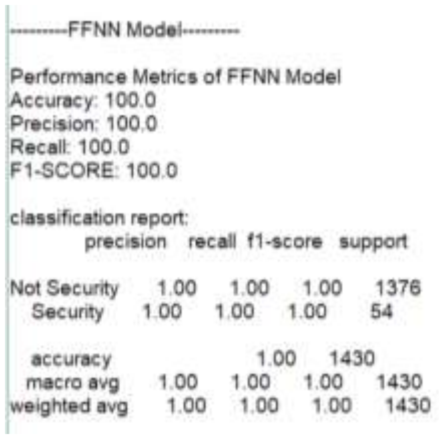


Fig. 6: SMOTE EDA

Figure 6 presents the SMOTE based exploratory data analysis (EDA) conducted on the dataset through the GUI. It includes visualizations such as word frequency distributions, class imbalances, and other statistical summaries. Graphs and charts provide insights into the dataset’s characteristics, helping to understand patterns and trends in the textual data before training the machine learning models.

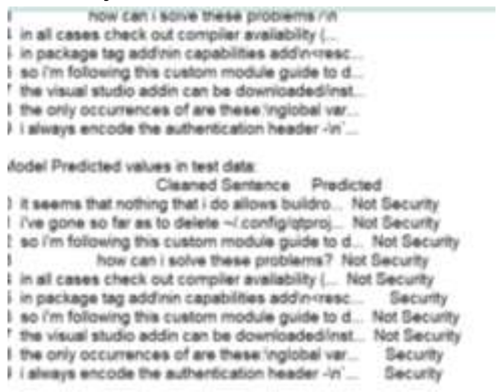




```
---------FFNN Model---------

Performance Metrics of FFNN Model
Accuracy: 100.0
Precision: 100.0
Recall: 100.0
F1-SCORE: 100.0

classification report:
          precision   recall  f1-score   support

Not Security     1.00     1.00     1.00     1376
   Security    1.00     1.00     1.00      54

   accuracy                       1.00     1430
  macro avg     1.00     1.00     1.00     1430
weighted avg     1.00     1.00     1.00     1430
```

Fig. 7: Performance Metrics and Confusion Matrix Plot of the FFNN Classifier Model

Figure 7 presents the evaluation results of the FFNN with XGBoost model. The performance metrics, including accuracy, precision, recall, and F1-score, indicate that the model achieves perfect classification, with 100% accuracy across all metrics. The confusion matrix confirms that every instance in the test set is correctly classified as either security-related or non-security-related. This figure highlights the superior performance of FFNN compared to GBC, demonstrating its ability to learn and generalize patterns effectively.



```
            how can i solve these problems /\n
 in all cases check out compiler availability (...
 in package tag addin\n capabilities addin<resc...
 so i'm following this custom module guide to d...
 the visual studio addin can be downloaded/inst...
 the only occurrences of are these \nglobal var...
 i always encode the authentication header -\n'...

Model Predicted values in test data:
                      Cleaned Sentence     Predicted
 it seems that nothing that i do allows buildro...  Not Security
 i've gone so far as to delete ~/.config/qtproj...  Not Security
 so i'm following this custom module guide to d...  Not Security
              how can i solve these problems?  Not Security
 in all cases check out compiler availability (...  Not Security
 in package tag addin\n capabilities addin<resc...      Security
 so i'm following this custom module guide to d...  Not Security
 the visual studio addin can be downloaded/inst...  Not Security
 the only occurrences of are these \nglobal var...      Security
 i always encode the authentication header -\n'...      Security
```

Fig. 8: Model Prediction on the Test Data

Figure 8 illustrates how the trained models predict security classifications on the test dataset. The predictions are displayed in the GUI, showing the input sentences along with their corresponding classifications as either security-related or not. The interface allows users to observe the real-time functioning of the trained models and assess their reliability in categorizing security-related text.

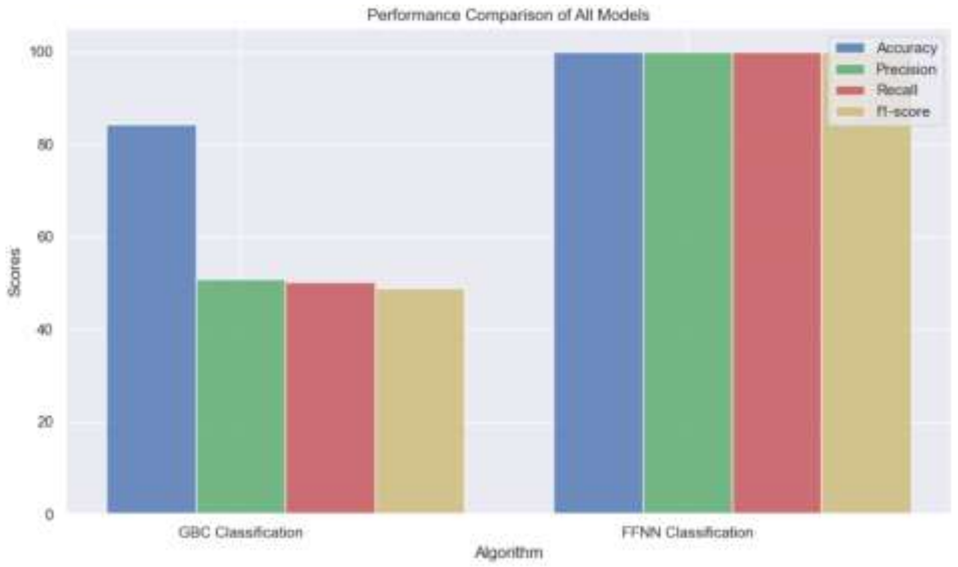
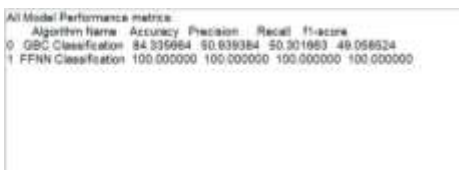


Fig. 9: Performance Comparison Graph.



```
All Model Performance metrics
      Algorithm Name   Accuracy  Precision     Recall   f1-score
0   GBC Classification  84.335664  50.839384  50.301663  49.058524
1  FFNN Classification 100.000000 100.000000 100.000000 100.000000
```

Fig. 10: Performance Comparison Graph of Models

Figure 9 and Figure 10 provides a comparative analysis of the performance of the GBC and the FFNN with XGBoost. The graph visualizes the accuracy, precision, recall, and F1-score of both models, highlighting the significant difference in their classification performance. The FFNN model outperforms GBC across all metrics, demonstrating its effectiveness in handling security text classification tasks. The figure serves as a visual representation of why FFNN is chosen as the preferred model for this research.

## 5. CONCLUSION

The research successfully addresses the challenge of analyzing security-related textual data using machine learning techniques. A structured approach to data preprocessing, including text cleaning, tokenization, and feature extraction, enhances the quality of the input data. The classification models are trained and evaluated to ensure reliable performance in distinguishing security-related content from general text. GBC and FFNN with XGBoost are explored for their effectiveness, with FFNN demonstrating superior adaptability to complex patterns in textual data. The evaluation metrics indicate that the selected model achieves high accuracy, precision, recall, and F1-score, proving its effectiveness in security classification tasks. The findings of this research contribute to improving automated security content classification, which is critical for applications in cybersecurity, information retrieval, and automated monitoring systems. The integration of machine learning ensures that the system efficiently processes large-scale textual data, reducing manual effort and enhancing the accuracy of security-related information detection.

## REFERENCES

[1] Anwar, S.; Mohamad Zain, J.; Zolkipli, M.F.; Inayat, Z.; Khan, S.; Anthony, B.; Chang, V. From intrusion detection to an intrusion response system: Fundamentals, requirements, and future directions. Algorithms 2017, 10, 39.

[2] Li, X.J.; Ma, M.; Sun, Y. An adaptive deep learning neural network model to enhance machine-learning-based classifiers for intrusion detection in smart grids. Algorithms 2023, 16, 288.

[3] Wan, J.; Waqas, M.; Tu, S.; Hussain, S.M.; Shah, A.; Rehman, S.U.; Hanif, M. An efficient impersonation attack detection method in fog computing. CMC-Comput. Mater. Cont. 2021, 68, 267–281.

[4] Pranto, M.B.; Ratul, M.H.A.; Rahman, M.M.; Diya, I.J.; Zahir, Z.-B. Performance of machine learning techniques in anomaly detection with basic feature selection strategy-a network intrusion detection system. J. Adv. Inf. Technol 2022, 13, 36–44.

[5] Ozkan-Okay, M.; Samet, R.; Aslan, Ö.; Gupta, D. A comprehensive systematic literature review on intrusion detection systems. IEEE Access 2021, 9, 157727–157760.

[6] Cui, J.; Zong, L.; Xie, J.; Tang, M. A novel multi-module integrated intrusion detection system for high-dimensional imbalanced data. Appl. Intell. 2023, 53, 272–288.

[7] Kim, M.; Yun, J.; Cho, Y.; Shin, K.; Jang, R.; Bae, H.-j.; Kim, N. Deep learning in medical imaging. Neurospine 2019, 16, 657.

[8] Yin, W.; Kann, K.; Yu, M.; Schütze, H. Comparative study of CNN and RNN for natural language processing. arXiv 2017, arXiv:1702.01923.

[9] Liu, L.; Wang, P.; Lin, J.; Liu, L. Intrusion detection of imbalanced network traffic based on machine learning and deep learning. IEEE Access 2020, 9, 7550–7563.

[10] Esteva, A.; Robicquet, A.; Ramsundar, B.; Kuleshov, V.; DePristo, M.; Chou, K.; Cui, C.; Corrado, G.; Thrun, S.; Dean, J. A guide to deep learning in healthcare. Nat. Med. 2019, 25, 24–29.

[11] Sun, X.; Lv, M. Facial expression recognition based on a hybrid model combining deep and shallow features. Cogn. Comput. 2019, 11, 587–597.

[12] Chen, C.; Song, Y.; Yue, S.; Xu, X.; Zhou, L.; Lv, Q.; Yang, L. Fcnn-se: An intrusion detection model based on a fusion CNN and stacked ensemble. Appl. Sci. 2022, 12, 8601.

[13] He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.