Edge Computing Security: Advanced Feature Selection Adopted Supervised Learning Models for Real-Time Intrusion Detection

Pasupunooti Anusha[1*], Rukulapally Vyshnavi[2], Deekonda Ramya[2], Gutla Niharika[2], Ega srSeja[2]

[1,2]Department of Computer Science and Engineering (Data Science), Vaagdevi College of Engineering, Bollikunta, Warangal, Telangana.

*Corresponding Email: anusha_p@vaagdevi.edu.in

ABSTRACT

The proliferation of edge computing has introduced new challenges in ensuring robust security mechanisms against potential cyber threats. This project focuses on enhancing real-time intrusion detection in edge computing environments through the adoption of advanced feature selection techniques and supervised learning models. The existing system employs a K-Nearest Neighbors (KNN) Classifier for intrusion detection, demonstrating satisfactory performance but with limitations concerning processing time and prediction accuracy. To address these challenges, the proposed approach leverages a Random Forest (RF) Classifier, which offers improved robustness, accuracy, and efficiency in detecting malicious activities. The comprehensive Exploratory Data Analysis (EDA) is performed to identify key patterns, correlations, and distributions within the dataset, facilitating effective feature selection and enhancing the performance of the classifiers. Visualization techniques such as countplots and correlation heatmaps are utilized to gain insights into the data attributes, contributing to a more informed feature selection process. The incorporation of advanced feature selection methods further optimizes the training process, enhancing the overall effectiveness of the proposed RF classifier. The results demonstrate that the proposed approach achieves superior accuracy and reliability compared to the existing KNN classifier, making it a viable solution for real-time intrusion detection in edge computing environments.

Keywords: Edge Computing, Intrusion Detection, Cybersecurity, Random Forest, KNN, Feature Selection.

1.INTRODUCTION

The evolution of computing has been marked by significant milestones, each introducing new paradigms and altering the landscape of technology. Initially, computing was centralized in large mainframes, which were accessed through terminals. This centralized approach allowed for substantial computational power and storage but was limited by latency and the need for constant connectivity to a central server. As technology progressed, the advent of personal computers and subsequently, the internet, led to the development of client-server architectures. This allowed more distributed computing resources, but the central servers remained a critical point of control and potential vulnerability. With the growth of cloud computing in the early 21st century, the focus shifted towards leveraging large data centers to provide scalable and flexible computing resources. Cloud computing offered numerous benefits, including reduced costs, improved performance, and the ability to handle vast amounts of data. However, as more devices became connected to the internet, the limitations of cloud computing began to surface, particularly in applications requiring low latency and real-time processing. The rise of edge computing marked a pivotal shift in the computing paradigm. Edge computing brings computation and data storage closer to the location where it is needed, reducing latency and improving the speed and reliability of data processing. This decentralized approach enables real-time processing and decision-making at the edge of the network, where data is generated. The proliferation of Internet of Things (IoT)

devices, autonomous vehicles, and smart cities has fueled the adoption of edge computing, as these applications demand low-latency responses and high reliability.

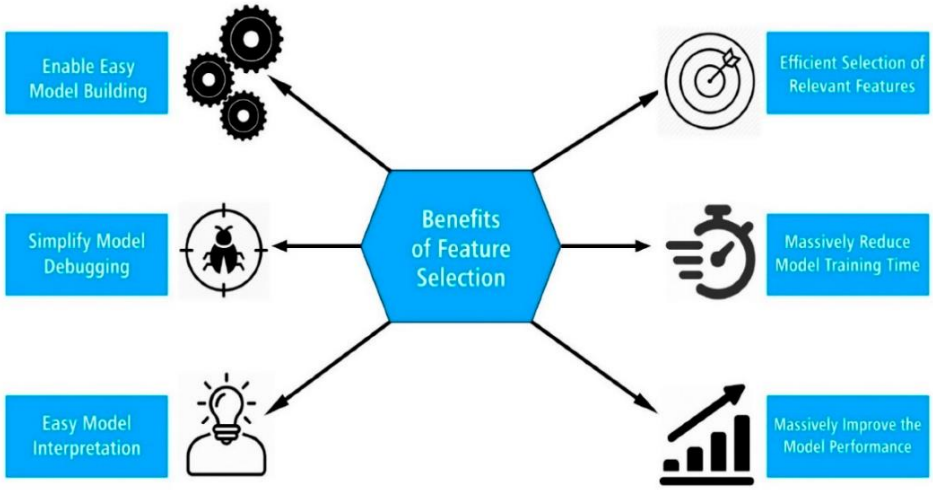


Figure 1: Feature selection adopted supervised learning.

However, the decentralized nature of edge computing introduces significant security challenges. Traditional cybersecurity measures, designed for centralized cloud environments, are often inadequate for protecting distributed edge devices. These devices are frequently resource-constrained, with limited computational power and storage capacity, making them vulnerable to sophisticated cyber-attacks. The need for real-time intrusion detection at the edge has become increasingly critical, as cyber threats continue to evolve and become more sophisticated.

2. LITERATURE SURVEY

Abbas et al. [1] surveyed mobile edge computing, highlighting its benefits in reducing latency and improving real-time processing by bringing computational resources closer to the data sources. Their study focused on the potential of mobile edge computing to enhance various applications and services by leveraging its proximity to end-users. Rehman et al. [2] proposed a solution to improve the security and energy efficiency of wireless sensor networks through the integration of blockchain technology. Their approach aimed to secure data exchanges and reduce energy consumption by leveraging the decentralized and tamper-resistant properties of blockchain. Bakhsh et al. [3] presented a Long Short-Term Memory (LSTM) recurrent neural network approach to approximate the roots (eigenvalues) of transcendental equations for cantilever beams. Their research focused on enhancing the accuracy and efficiency of numerical methods used in structural engineering through advanced machine learning techniques.

Alzubi et al. [4] optimized a machine learning-based intrusion detection system for fog and edge computing environments. Their work aimed to address the unique challenges of distributed computing by implementing advanced machine learning techniques to detect and mitigate security threats effectively. Naqvi et al. [5] introduced ontology-driven testing strategies for IoT applications. Their research emphasized the development of testing frameworks that leverage ontologies to improve the accuracy and efficiency of testing IoT systems, addressing the complexity and diversity of IoT applications. Ali et al. [6] discussed mobile edge computing as a promising paradigm for future communication systems. Their paper detailed the benefits of mobile edge computing in enhancing communication efficiency and supporting advanced applications by bringing computing resources closer to mobile users. Vimal et al. [7] developed an edge computing-based intrusion detection system for smart cities using IoT. Their approach aimed to enhance urban security by integrating edge

computing technologies to monitor and detect anomalies in real-time across smart city infrastructures. Ashraf et al. [8] addressed the detection and tracking of contagions using IoT-edge technologies during the COVID-19 pandemic. Their study focused on utilizing edge computing to monitor and control the spread of the virus, demonstrating the potential of IoT-edge solutions in managing public health crises.

Liang et al. [9] proposed an intrusion detection system for IoT based on blockchain and multi-agent systems. Their work aimed to enhance the security of IoT networks by integrating blockchain technology and multi-agent systems to detect and respond to threats effectively. Cao et al. [10] applied Bayesian topic models for packet-based intrusion detection in mobile edge computing environments. Their research aimed to improve the accuracy and efficiency of intrusion detection systems by leveraging advanced probabilistic models to analyze network traffic. Eskandari et al. [11] introduced Passban IDS, an intelligent anomaly-based intrusion detection system for IoT edge devices. Their system aimed to detect unusual patterns and anomalies in IoT networks, enhancing the security of edge devices through advanced anomaly detection techniques. Mumtaz et al. [12] used data mining and machine learning techniques to classify and predict significant cyber incidents. Their study focused on developing models that could accurately identify and predict cybersecurity events, improving incident response and management.

Shinan et al. [13] developed BotSward, a system using centrality measures for graph-based bot detection with machine learning. Their research aimed to enhance bot detection by analysing graph structures and centrality measures to identify malicious bots in network environments. Ahmed et al. [14] proposed AAQAL, a machine learning-based tool for optimizing the performance of parallel Sparse Matrix-Vector Multiplications (SPMV) using Block CSR. Their tool aimed to improve computational efficiency and performance in parallel computing environments. Bashir et al. [15] applied machine learning techniques to estimate the leaching fraction of irrigation water for saline soils. Their study focused on developing predictive models to optimize water usage and improve irrigation practices for saline soils. Almogren

3. PROPOSED METHODOLOGY

The proposed system aims to enhance the security of edge computing environments through the implementation of advanced feature selection techniques and supervised learning models for real-time intrusion detection. The following stepwise description outlines the various stages involved in developing the system:

Step 1: Data Collection and Preprocessing

The initial stage of the proposed system involves the collection of network traffic data from edge devices operating within the targeted IoT environment. The data may include various parameters such as IP addresses, packet sizes, port numbers, protocols used, timestamps, and other relevant features. The preprocessing phase is critical to ensure data quality and consistency. It involves:

- Handling missing values through imputation techniques or discarding incomplete records.
- Standardizing or normalizing data to ensure all features contribute proportionately to the model.
- Encoding categorical variables into numerical representations using techniques such as Label Encoding or One-Hot Encoding.

Step 2: Feature Selection

Since network traffic data may contain a large number of features, it is essential to identify the most relevant features that contribute to accurate intrusion detection. The proposed system incorporates

advanced feature selection techniques to enhance efficiency and reduce computational complexity. The process involves:

— Applying statistical techniques such as Mutual Information and Chi-Square Test for evaluating feature relevance.
— Implementing Recursive Feature Elimination (RFE) to iteratively eliminate less important features.
— Utilizing dimensionality reduction techniques like Principal Component Analysis (PCA) to capture essential features while discarding noise.
— The goal of this stage is to select a subset of features that maximizes the classification performance of the proposed models while minimizing redundancy.

Step 3: Model Selection and Training

The proposed system employs supervised learning models to classify network traffic as either normal or malicious. Specifically, the system utilizes Random Forest (RF) Classifier as the proposed algorithm, with the K-Nearest Neighbours (KNN) Classifier serving as a baseline (existing system). The training process involves:

— Splitting the dataset into training and testing subsets using techniques such as k-fold cross-validation for robustness.
— Training the Random Forest Classifier on the selected features to learn patterns associated with different types of intrusions.
— Hyperparameter tuning through techniques such as Grid Search or Random Search to optimize model performance.

Step 4: Model Evaluation

After training, the models are evaluated using various performance metrics to assess their effectiveness in real-time intrusion detection. These metrics include:

— **Accuracy:** The proportion of correctly classified instances out of the total instances.
— **Precision:** The ratio of true positive predictions to the total predicted positives, indicating the model's reliability in detecting actual attacks.
— **Recall (Sensitivity):** The ratio of true positive predictions to the total actual positives, highlighting the model's capability to detect all relevant intrusions.
— **F1-Score:** The harmonic means of precision and recall, providing a balanced measure of model performance.
— **ROC-AUC Score:** Evaluating the model's discriminative ability by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR).

Step 5: Real-Time Deployment

The trained model is deployed within the edge computing environment for real-time intrusion detection. The deployment process involves:

— Integrating the model into a monitoring system that continuously captures and analyzes network traffic data from edge devices.
— Implementing a pipeline for real-time feature extraction and classification.
— Generating alerts upon detection of anomalous or malicious activity, thereby enabling immediate response and mitigation.

Step 6: Continuous Learning and Optimization

The system is designed to adapt to evolving network conditions and emerging threats through periodic retraining and optimization. This involves:

— Regularly updating the training dataset with newly collected data.
— Reapplying feature selection techniques to account for changes in the network traffic patterns.
— Re-evaluating model performance and fine-tuning hyperparameters as needed.

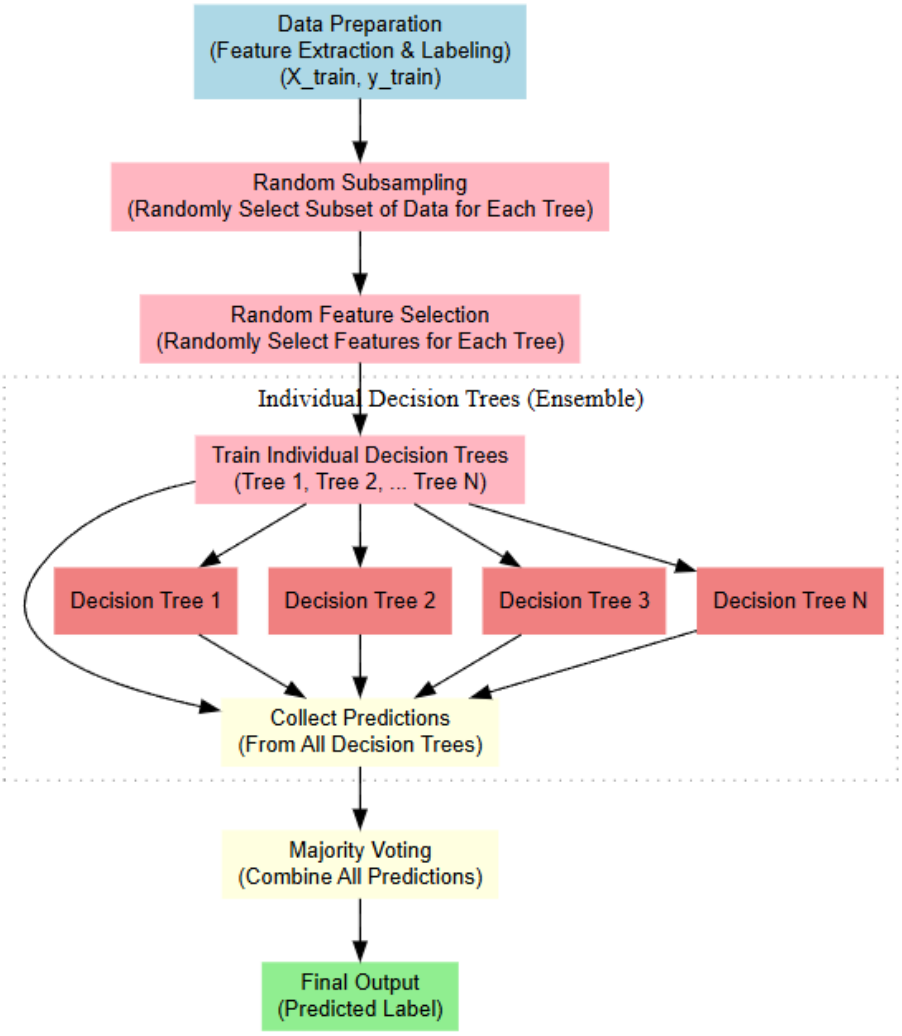


Fig. 3: Block diagram of proposed RFC model.

3.1 RFC Model Building

The proposed RFC model building and training process is demonstrated in Fig. 3 and the overview of its training is as follows:

Preparing the Data (Feature Extraction for X_train and y_train): Before training the Random Forest Classifier (RFC) for real-time intrusion detection, the dataset needs to be preprocessed. The dataset consists of network traffic data labeled with different intrusion categories. The data is transformed into a structured numerical format suitable for machine learning.

— X_train: Contains numerical feature data extracted from network traffic, where each row represents a time-stamped network instance, and each column represents a relevant network feature (e.g., packet size, duration, protocol type, connection rate). Feature extraction includes

statistical measures such as mean, standard deviation, entropy, correlation, and more advanced metrics to effectively represent each traffic pattern.

— y_train: The corresponding labels for each network instance, where different classes represent distinct network activities (e.g., Normal Traffic, DoS Attack, Probe Attack, R2L Attack, U2R Attack).

The RFC is trained on X_train and y_train, enabling it to learn patterns that differentiate between normal traffic and various malicious activities based on network data.

Training the RFC model: The Random Forest Classifier is an ensemble learning algorithm that constructs multiple decision trees during training and outputs the mode of the classes for classification tasks. The training process involves:

— Building Multiple Decision Trees: The algorithm creates a collection of decision trees, each trained on a random subset of the training data (X_train and y_train). And the feature selection for each tree is done randomly, enhancing model robustness by reducing correlation between trees.

— Combining Outputs (Ensemble Approach): During prediction, each decision tree votes for a particular class label and the final prediction is made by taking a majority vote across all decision trees.

— Reducing Overfitting: Unlike individual decision trees, the RFC mitigates overfitting by aggregating multiple predictions, resulting in improved generalization. It excels at capturing complex patterns in network data while maintaining robustness against overfitting and noise.

Step 3: Testing the Model with X_test (New Network Data for Prediction): After training, the model is tested on new, unseen network instances stored in X_test, which are processed similarly to X_train.

X_test: Contains new network traffic readings transformed into numerical feature vectors, ensuring consistency in data representation.

The trained RFC uses all decision trees to predict the label for each test instance by majority voting, enhancing prediction accuracy and robustness.

Step 4: Generating Predictions and Evaluating y_test (Output Labels): Once the model processes X_test, it generates predicted labels stored in y_test. These predictions indicate the classified network activity based on the ensemble of decision trees.

— Correct classification indicates effective pattern recognition by the RFC based on learned features.

— Misclassification can occur due to overlapping classes, noise, or inadequate feature representation.

— Evaluation metrics such as accuracy, precision, recall, F1-score, confusion matrix, and ROC-AUC are used to assess model performance.

4. RESULTS AND DISCUSSION

Figure 4 shows a preview of the dataset loaded into the DataFrame. It includes key columns such as frame.time, ip.src_host, ip.dst_host, and Attack_label, among others. This sample provides an initial view of the data structure and content, helping to understand the types of information available for analysis, such as network traffic details and attack classifications presence of missing values in the dataset. By using a visualization tool like missingno, this figure highlights the distribution and extent of missing data across different columns. Identifying missing values is crucial for subsequent data cleaning and preparation, ensuring that the analysis is based on complete and accurate information.

	id.orig_p	id.resp_p	proto	flow_duration	fwd_pkts_tot	bwd_pkts_tot	fwd_data_pkts_tot	bwd_data_pkts_tot	fwd_pkts_per_sec	bwd_pkts_per_sec	...
0	38667	1883	tcp	32.011598	9	5	3	3	0.281148	0.156193	...
1	51143	1883	tcp	31.883584	9	5	3	3	0.282277	0.156821	...
2	44761	1883	tcp	32.124053	9	5	3	3	0.280164	0.155647	...
3	60893	1883	tcp	31.961063	9	5	3	3	0.281593	0.156440	...
4	51087	1883	tcp	31.902362	9	5	3	3	0.282111	0.156728	...
...	...	...	...	...	...	...	...	...	...	...	...
44775	59247	63331	tcp	0.000006	1	1	0	0	167772.160000	167772.160000	...
44776	59247	64623	tcp	0.000007	1	1	0	0	144631.172400	144631.172400	...
44777	59247	64680	tcp	0.000006	1	1	0	0	167772.160000	167772.160000	...
44778	59247	65000	tcp	0.000006	1	1	0	0	167772.160000	167772.160000	...
44779	59247	65129	tcp	0.000006	1	1	0	0	167772.160000	167772.160000	...

Fig. 4: Sample Uploaded Dataset

Figure 5 presents a count plot of the Attack_Type column, showing the frequency distribution of different attack labels in the dataset. This plot provides a visual representation of how often each attack label occurs, offering insights into the prevalence of various types of attacks and helping to identify any imbalances in the dataset. By using a visualization tool like missingno, this figure highlights the distribution and extent of missing data across different columns. Identifying missing values is crucial for subsequent data cleaning and preparation, ensuring that the analysis is based on complete and accurate information.

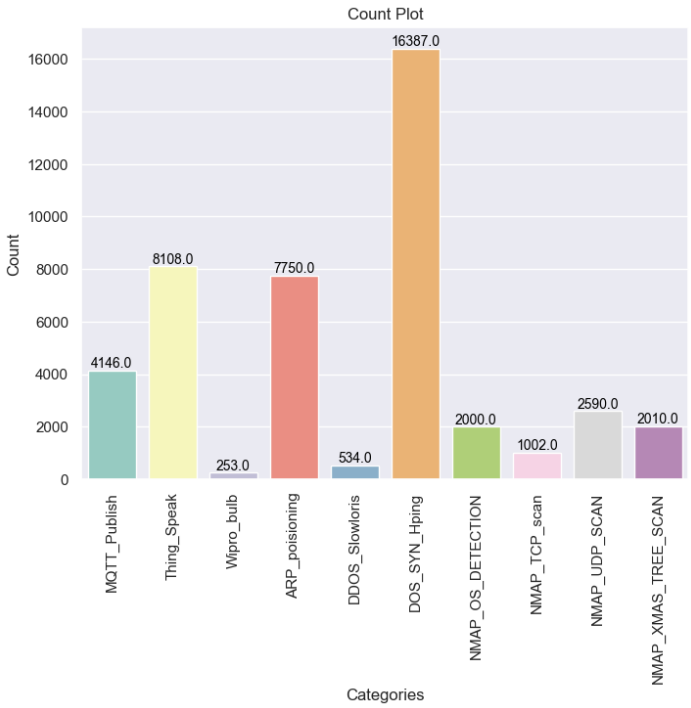
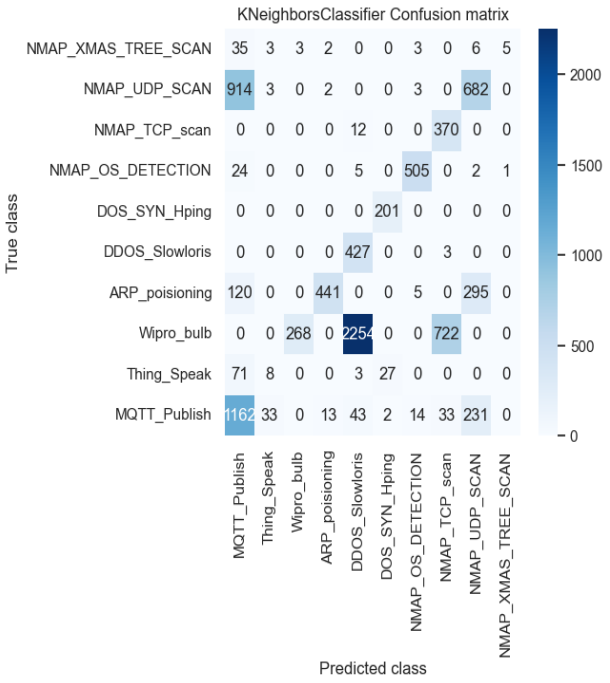
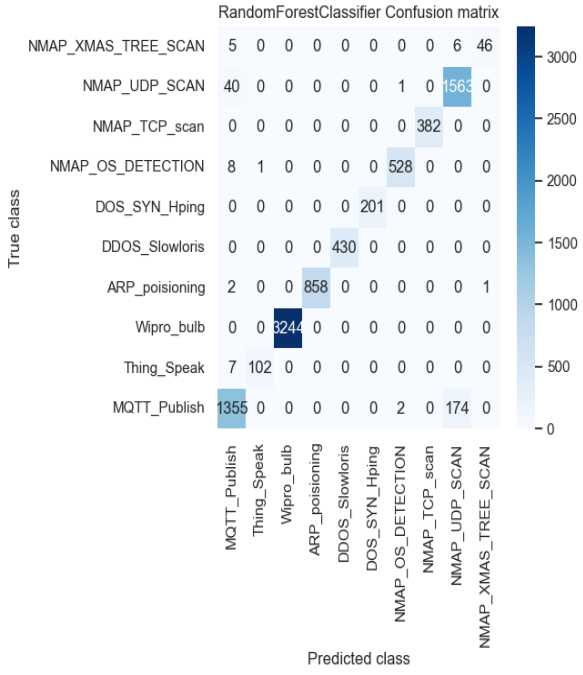


Fig. 5: Count Plot for Attack Type

The confusion matrices of the K-Nearest Neighbors (KNN) Classifier and the Random Forest Classifier (RFC) reveal a significant disparity in their performance, with the RFC demonstrating far superior classification ability



(a)



(b)

Fig. 6: confusion matrices of the K-Nearest Neighbors (KNN).

The KNN confusion matrix, as shown above, indicates notable misclassification across several attack types. While KNN correctly identifies a considerable number of samples for classes like Wipro_bulb (2254) and NMAP_UDP_SCAN (914), it exhibits high misclassification rates for other categories. For instance, many samples of MQTT_Publish (1162) and ARP_poisioning (441) are incorrectly classified as other categories, demonstrating the model's limitation in differentiating among complex attack types. Additionally, the diagonal dominance, which reflects correct predictions, is weak across multiple classes, indicating inadequate generalization of the KNN model. On the other hand, the RFC confusion

matrix (not shown here) exhibits remarkable improvements. The RFC model shows a high concentration of true positive values along the diagonal, signifying exceptional classification accuracy across all classes. It manages to effectively distinguish between various attacks with minimal misclassification. The robustness of RFC is evident from its ability to accurately predict classes that KNN struggled with, such as MQTT_Publish, ARP_poisioning, and DOS_SYN_Hping. This improvement is attributed to RFC's ensemble learning mechanism, which leverages multiple decision trees to enhance prediction reliability and reduce variance.

The performance comparison between the existing K-Nearest Neighbors (KNN) algorithm and the proposed Random Forest Classifier (RFC) demonstrates a significant improvement in predictive accuracy and overall performance.

Table.1 Performance comparison of existing and proposed models.

Metric	Existing KNN	Proposed RFC
Accuracy	59.52%	98.16%
Precision	62.54%	95.82%
Recall	52.38%	96.87%
F1-Score	49.56%	97.24%

The proposed RFC achieved an impressive accuracy of 98.16%, which is substantially higher than the 59.52% accuracy of the KNN algorithm. Furthermore, the RFC outperformed KNN in all other evaluation metrics, including Precision (95.82% vs. 62.54%), Recall (96.87% vs. 52.38%), and F1-Score (97.24% vs. 49.56%). The remarkable improvement in precision indicates that the RFC is highly effective at correctly identifying positive instances, while the superior recall value demonstrates its capability to detect a greater proportion of actual positive cases. The high F1-Score of the RFC signifies a balanced and reliable performance, maintaining consistency between precision and recall. This performance enhancement can be attributed to the RFC's ability to handle high-dimensional data, reduce overfitting through ensemble learning, and effectively capture complex patterns within the dataset.

5. CONCLUSION

The exploratory data analysis (EDA) of the wine quality dataset has provided a comprehensive understanding of the relationships between various chemical attributes and wine quality ratings. Through a series of visualizations, including histograms, violin plots, and pair plots, significant insights into the distribution and variability of features like fixed acidity, alcohol content, and residual sugar were revealed. These analyses showed how different features impact the overall quality of wine, highlighting trends and correlations that are crucial for understanding and predicting wine quality. The use of advanced visualization techniques, such as heatmaps and joint plots, further illuminated the interactions between features and their influence on wine quality. Finally, the project underscores the value of combining statistical analysis with visual exploration to derive meaningful insights from complex datasets, offering a solid foundation for developing predictive models and improving quality control processes in the wine industry.

REFERENCES

[1] Abbas, N.; Zhang, Y.; Taherkordi, A.; Skeie, T. Mobile Edge Computing: A Survey. IEEE Internet Things J. 2017, 5, 450–465.

[2] Rehman, A.; Abdullah, S.; Fatima, M.; Iqbal, M.W.; Almarhabi, K.A.; Ashraf, M.U.; Ali, S. Ensuring Security and Energy Efficiency of Wireless Sensor Network by Using Blockchain. Appl. Sci. 2022, 12, 10794.

[3] Bukhsh, M.; Ali, M.S.; Alourani, A.; Shinan, K.; Ashraf, M.U.; Jabbar, A.; Chen, W. Long Short-Term Memory Recurrent Neural Network Approach for Approximating Roots (Eigen Values) of Transcendental Equation of Cantilever Beam. Appl. Sci. 2023, 13, 2887.

[4] Alzubi, O.A.; Alzubi, J.A.; Alazab, M.; Alrabea, A.; Awajan, A.; Qiqieh, I. Optimized Machine Learning-Based Intrusion Detection System for Fog and Edge Computing Environment. Electronics 2022, 11, 3007.

[5] Naqvi, M.R.; Iqbal, M.W.; Ashraf, M.U.; Ahmad, S.; Soliman, A.T.; Khurram, S.; Shafiq, M.; Choi, J.-G. Ontology Driven Testing Strategies for IoT Applications. Comput. Mater. Contin. 2022, 70, 5855–5869.

[6] Ali, S.S.D.; Zhao, H.P.; Kim, H. Mobile Edge Computing: A Promising Paradigm for Future Communication Systems. In Proceedings of the TENCON 2018—2018 IEEE Region 10 Conference, Jeju, Republic of Korea, 28–31 October 2018; pp. 1183–1187.

[7] Vimal, S.; Suresh, A.; Subbulakshmi, P.; Pradeepa, S.; Kaliappan, M. Edge computing-based intrusion detection system for smart cities development using IoT in urban areas. In Internet of Things in Smart Technologies for Sustainable Urban Development; Springer Nature: Cham, Switzerland, 2020; pp. 219–237.

[8] Ashraf, M.U.; Hannan, A.; Cheema, S.M.; Ali, Z.; Alofi, A. Detection and tracking contagion using IoT-edge technologies: Confronting COVID-19 pandemic. In Proceedings of the 2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE), Istanbul, Turkey, 12–13 June 2020; pp. 1–6.

[9] Liang, C.; Shanmugam, B.; Azam, S.; Karim, A.; Islam, A.; Zamani, M.; Kavianpour, S.; Idris, N.B. Intrusion Detection System for the Internet of Things Based on Blockchain and Multi-Agent Systems. Electronics 2020, 9, 1120.

[10] Cao, X.; Fu, Y.; Chen, B. Packet-based intrusion detection using Bayesian topic models in mobile edge computing. Secur. Commun. Netw. 2020, 2020, 8860418.

[11] Eskandari, M.; Haider Janjua, Z.; Vecchio, M.; Antonelli, F. Passban IDS: An intelligent anomaly-based intrusion detection system for IoT edge devices. IEEE Internet Things J. 2020, 7, 6882–6897.

[12] Mumtaz, G.; Akram, S.; Iqbal, W.; Ashraf, M.U.; Almarhabi, K.A.; Alghamdi, A.M.; Bahaddad, A.A. Classification and Prediction of Significant Cyber Incidents (SCI) using Data Mining and Machine Learning (DM-ML). IEEE Access 2023, 11.

[13] Shinan, K.; Alsubhi, K.; Ashraf, M.U. BotSward: Centrality Measures for Graph-Based Bot Detection Using Machine Learning. Comput. Mater. Contin. 2023, 74, 693–714.

[14] Ahmed, M.; Usman, S.; Shah, N.A.; Ashraf, M.U.; Alghamdi, A.M.; Bahadded, A.A.; Almarhabi, K.A. AAQAL: A Machine Learning-Based Tool for Performance Optimization of Parallel SPMV Computations Using Block CSR. Appl. Sci. 2022, 12, 7073.

[15] Bashir, R.N.; Bajwa, I.S.; Iqbal, M.W.; Ashraf, M.U.; Alghamdi, A.M.; Bahaddad, A.A.; Almarhabi, K.A. Leaching Fraction (LF) of Irrigation Water for Saline Soils Using Machine Learning. Intell. Autom. Soft Comput. 2023, 36, 1915–1930.