



A COMPARATIVE ANALYSIS OF DEEP LEARNING TECHNIQUES FOR INSIDER THREAT DETECTION



^{*1} A. O. Akinlolu, ²E. R. Jimoh, ³I. A. Alabere and ⁴O. Ekundayo

¹ Department of Mathematical and Computer Sciences, Fountain University, Osogbo, Nigeria

² Department of Physical & Mathematical Sciences, Ojaja University, Eiyekorin, Kwara State.

³ Department of Computer Science, University of Ilorin, Ilorin, Nigeria.

⁴ Department of Computer Science, Faculty of Information and Communication Technology, Kwara State University Malete.

Received: April 16, 2024 Accepted: July 28, 2024

Abstract:

An insider threat refers to any malicious activities carried out by employees, contractors, or vendors who have authorised access to an organisation's IT assets, resulting in significant negative impacts on its data and information resources. Existing literature reviews have identified shortcomings in utilizing diverse domains (such as system log files, file processes, logon records, HTTP, email, external drives, and the Lightweight Directory Access Protocol (LDAP)) to develop techniques capable of identifying insiders posing a threat to an organisation with minimal false positives. In contrast, this study opts for a more robust domain, specifically LAN data packets, to assess the activities of users within the organization's Local Area Network. It monitors deviations from the normal flow of data packets on the network, thereby classifying these anomalies as either malicious or benign. The synthetic dataset KDDCUP'99, evaluated in this study, is widely recognized as one of the few publicly accessible datasets for network-based anomaly detection systems. The proposed stacked ensemble model demonstrates superior predictive performance, achieving an accuracy of 98%, compared to the 91.88% and 98.58% accuracy of the individual classifiers Naïve Bayes and KNN, respectively. The model can be improved by incorporating additional user behaviours such as email communication, browser activity, and file access to enhance accuracy and applicability.

Keywords:

Deep Learning Techniques, Insider Threat Detection, LAN Data Packets, Network-Based Anomaly Detection, KDDCUP'99 Dataset, Stacked Ensemble Model.

Introduction

In today's digital era, safeguarding sensitive information and fortifying defences against security breaches stand as top priorities for organizations across diverse industries. While external threats frequently garner significant attention, the risk posed by insider threats cannot be underestimated. Insider threats encompass a spectrum of malicious actions carried out by individuals within an organization, including employees, contractors, or business associates, who misuse their authorized access to compromise the integrity, confidentiality, or availability of data.

Detecting and mitigating insider threats demand sophisticated and proactive approaches that surpass conventional security measures. The emergence of insider threats highlights the necessity for advanced techniques and technologies capable of pinpointing suspicious behaviours and anomalous activities within organizational networks. Traditional security methods often concentrate on defending against external threats at the perimeter, leaving vulnerabilities within the organization unattended. Furthermore, insider threats present a greater challenge in detection due to the perpetrators' legitimate access to systems and data, making it harder to distinguish their malicious actions from normal behaviour.

Conventional methods for detecting insider threats primarily depend on rule-based systems and predefined thresholds, which may fail to adapt to evolving attack strategies and subtle deviations from typical behaviour patterns. Consequently, there's an increasing acknowledgement of the significance of harnessing machine learning and artificial

intelligence (AI) to augment insider threat detection capabilities. These technologies present the opportunity to analyse extensive datasets in real-time, pinpoint suspicious patterns, and promptly identify potential insider threats with higher accuracy and efficiency.

One promising approach to insider threat detection involves the use of ensemble model classification, which combines multiple machine learning algorithms to improve prediction accuracy and robustness. Ensemble methods leverage the diversity of individual models to mitigate biases and errors inherent in any single classifier, thereby enhancing the overall performance of the system. By integrating diverse perspectives and learning mechanisms, ensemble models can capture complex relationships and nuances in the data, making them well-suited for detecting subtle anomalies indicative of insider threats.

This paper focuses on developing an ensemble model for insider threat detection, specifically targeting the identification of malicious insiders within an enterprise's Local Area Network. The study utilizes machine learning algorithms such as KNN, Naive Bayes, and XGBoost, along with Recursive Feature Elimination (RFE) for data preprocessing with the aim of addressing the challenges faced in treating different domains (logon, HTTPS, LDAP domain) as one entity due to data inhomogeneity and the need for special data preprocessing techniques.

Literature Review

The literature presents various approaches to insider threat detection, each leveraging different methodologies and techniques to identify malicious activities within organizational networks. Aldairi, Karimi & Joshi (2019) proposed a machine learning-based approach using unsupervised learning algorithms like Isolation Forest and One-class SVM on system logs, demonstrating superior efficiency in detecting insider attacks compared to traditional methods. While the proposed approach showed effectiveness in detecting insider attacks, its reliance on system logs may limit its applicability in scenarios where comprehensive logging is not available or feasible. Additionally, the unsupervised learning algorithms used may struggle with detecting nuanced insider threats that exhibit complex behaviours.

Similarly, Wei, Chow & Yiu (2021) proposed an unsupervised anomaly detection scheme utilizing Cascaded Autoencoders (CAEs) and Bidirectional Long Short-Term Memory (BiLSTM) to achieve superior insider threat prediction performance. While the unsupervised anomaly detection scheme showed promising results, the performance may be influenced by the quality and representativeness of the training data. Additionally, the reliance on pre-processed behaviour sequences may introduce noise or bias, impacting the model's effectiveness in real-world settings.

Pantelidis, Bendiab, Shiaeles & Kolokotronis (2021) explored the effectiveness of deep learning algorithms, particularly Autoencoder and Variational Autoencoder, in detecting insider attacks without human intervention, showcasing remarkable performance on public datasets. The evaluation of deep learning algorithms on public datasets may not fully capture the complexity and diversity of insider threat scenarios encountered in practice. Moreover, the effectiveness of the proposed models may be influenced by factors such as data imbalance and domain-specific characteristics.

Le & Zincir-Heywood (2019) introduced a user-centered machine learning model that utilizes supervised algorithms such as Logistic Regression, Random Forest, and Artificial Neural Network to effectively identify malicious insiders. Although the supervised machine learning model showcased remarkable accuracy in detecting malicious insiders, its dependency on labelled training data could present obstacles in situations where acquiring an ample amount of labelled data is arduous or expensive. Moreover, the model's effectiveness might diminish when confronted with novel or unfamiliar patterns of insider threats. Furthermore, Nicolaou, Shiaeles & Savage (2020) developed a machine learning system utilizing the Local Outlier Factor (LOF) to distinguish malicious activities from legitimate ones based on user-centered analysis, achieving notable success in outlier detection. The use of Local Outlier Factor (LOF) for outlier detection may be sensitive to the choice of parameters and the distribution of data. Additionally, the performance of the model may vary across different datasets and organizational contexts, limiting its generalizability. Nasir, Afzal, Latif & Iqbal (2021) proposed a user behaviour-based detection technique using Long-Short Term

Memory (LSTM) Autoencoder, demonstrating relatively good accuracy compared to other known methods. While the LSTM Autoencoder-based approach showed promising results in detecting unusual user behaviours, the performance may be influenced by factors such as the length and granularity of the behaviour sequences. Moreover, the approach may struggle with detecting subtle insider threats that do not exhibit drastic deviations from normal behaviour. Furthermore, Faisal, Asif, Haider & Imran (2020) conducted a study where they employed linguistic analysis to construct predictive models for assessing employee risk levels through email communication analysis. Their work demonstrated the potential of supervised machine learning techniques in identifying anomalous emails. However, the effectiveness of supervised machine learning techniques for email anomaly detection might be constrained by the availability and quality of labelled training data. Additionally, the performance of these models could vary depending on the specific features and characteristics of the email communication data.

In another study, Saadi, Al-Ibadi, Tong & Farkas (2018) proposed a hybrid model that integrates Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) to effectively model user actions and detect abnormal behaviours, thereby enhancing the detection of insider threats. Despite the efficiency demonstrated by the hybrid CNN-LSTM model in detecting abnormal behaviours, its performance could be influenced by factors such as the granularity of the input data and the complexity of user actions. Moreover, the model's effectiveness might diminish in scenarios involving noisy or incomplete data.

Additionally, Paul & Mishra (2020) developed an unsupervised deep neural network model utilizing LSTM Autoencoder to forecast threat scenarios based on employees' routine activities. Showcasing its effectiveness on real-world datasets. The unsupervised deep neural network model may struggle with detecting subtle anomalies or deviations from normal behaviour patterns, particularly in scenarios with high variability and complexity. Additionally, the model's performance may be influenced by factors such as the choice of hyperparameters and the representation of user activities.

Lastly, Fagade & Tryfonas (2017) simulated a model considering different risk situational factors across various domains, aiming to aggregate risk scores from each domain to predict malicious insider activities accurately. However, existing techniques often analyse users' behaviour independently, leading to high false positive rates. Therefore, there is a need for more holistic approaches considering users' behavioural patterns over time to improve detection accuracy and reduce false positives. While the model considers different risk situational factors across various domains, the aggregation of risk scores may oversimplify the complexity of insider threat scenarios. Moreover, the effectiveness of the model may be limited by the availability and reliability of data from each domain, as well as the assumptions underlying the risk aggregation process.

To address the gaps identified in the literature on insider threat detection, a comprehensive approach will be adopted, focusing on integrating multiple methodologies and techniques while mitigating their respective limitations. Firstly, to overcome the reliance on system logs and pre-processed behaviour sequences, a hybrid approach combining both supervised and unsupervised learning methods will be devised, leveraging the strengths of each to enhance detection accuracy across diverse insider threat scenarios. This hybrid model will incorporate machine learning algorithms such as LSTM Autoencoders and graph-based techniques to capture both nuanced insider behaviours and structural anomalies within organizational networks. Finally, to enhance detection accuracy and reduce false positives, a holistic approach considering users' behavioural patterns over time will be implemented, incorporating temporal analysis and context-aware risk assessment to better characterize evolving insider threat scenarios. Through these concerted efforts, the proposed framework aims to advance the state-of-the-art in insider threat detection by bridging existing gaps in the literature and providing a more comprehensive and effective solution for safeguarding organizational security.

MATERIAL AND METHODS

Data Acquisition

This study evaluates the synthetic dataset utilized in the Third International Knowledge Discovery and Data Mining Tools Competition (KDDCUP '99), which was generated by the Defense Advanced Research Projects Agency (DARPA) in 1999. This dataset was constructed using recorded network traffic from the 1998 dataset and pre-processed to contain 41 features per network connection. These features, delineated in Tables 1 through 4, are organized into four distinct groups.

The KDD'99 dataset, sourced from the UCI machine learning repository, has been widely employed as one of the limited publicly accessible datasets for network-based anomaly detection systems. It comprises raw TCP dump data from a local-area network (LAN) that simulates a typical U.S. Air Force LAN.

Each TCP/IP connection in the dataset yields 41 features, encompassing both quantitative and qualitative aspects from both normal and attack data (3 qualitative and 38 quantitative features). Each connection is labelled as either normal or anomalous, with only one type of attack attributed to it. The dataset is divided into two portions: 80% for training purposes and 20% for testing the model.

Table 1: Basic features of individual TCP connections

Feature no	Feature name	Description	Type
1	protocol_type	type of the protocol, e.g. tcp, udp, etc.	discrete
2	duration	length (number of seconds) of the connection	continuous
3	dst_bytes	number of data bytes from destination to source	continuous
4	service	network service on the destination, e.g., http, telnet, etc.	discrete
5	urgent	number of urgent packets	continuous
6	flag	normal or error status of the connection	discrete
7	src_bytes	number of data bytes from source to destination	continuous
8	wrong_fragment	number of "wrong" fragments	continuous
9	land	1 if connection is from/to the same host/port; 0 otherwise	discrete

Table 2: Content features within a connection suggested by domain knowledge

Feature no	Feature name	Description	Type
10	num_compromised	number of "compromised" conditions	continuous
11	num_failed_logins	number of failed login attempts	continuous
12	hot	number of "hot" indicators	continuous
13	is_guest_login	1 if the login is a "guest" login; 0 otherwise	discrete
14	num_outbound_cmds	number of outbound commands in an ftp session	continuous
15	num_file_creations	number of file creation operations	continuous
16	su_attempted	1 if "su root" command attempted; 0 otherwise	discrete
17	num_root	number of "root" accesses	continuous
18	is_hot_login	1 if the login belongs to the "hot" list; 0 otherwise	discrete
19	num_shells	number of shell prompts	continuous
20	root_shell	1 if root shell is obtained; 0 otherwise	discrete
21	logged_in	1 if successfully logged in; 0 otherwise	discrete
22	num_access_files	number of operations on access control files	continuous

Table 3: Traffic features computed using a two-second time window

Feature no	Feature name	Description	Type
23	count	number of connections to the same host as the current connection in the past two seconds Note: The following features refer to these same-host connections.	continuous
24	serror_rate	% of connections that have ``SYN" errors	continuous
25	error_rate	% of connections that have ``REJ" errors	continuous
26	same_srv_rate	% of connections to the same service	continuous
27	diff_srv_rate	% of connections to different services	continuous
28	srv_count	number of connections to the same service as the current connection in the past two seconds Note: The following features refer to these same-service connections.	continuous
29	srv_error_rate	% of connections that have ``SYN" errors	continuous
30	srv_error_rate	% of connections that have ``REJ" errors	continuous
31	srv_diff_host_rate	% of connections to different hosts	continuous

Table 4: Host-based traffic features

Feature no	Feature name	Description	Type
32	dst_host_srv_error_rate	% of connections to the current host and specified service that have an RST error	continuous
33	dst_host_srv_count	Count of connections having same dst host and using same service	
34	dst_host_srv_diff_host_rate	% of connections to current host having same src port	continuous
35	dst_host_srv_error_rate	% of connections to the current host and specified service that have an S0 error	continuous
36	dst_host_same_src_port_rate	% of different services on current host	continuous
37	dst_host_count	Count of connections having same dst host	continuous
38	dst_host_error_rate	% of connections to the current host that have an S0 error	continuous
39	dst_host_diff_srv_rate	% of connections having same dst host and using same service	continuous
40	dst_host_error_rate	% of connections to the current host that have an RST error	continuous
41	dst_host_same_srv_rate	count of connections having same dst host and using same service	continuous

A connection refers to a series of TCP packets initiated and terminated at distinct times. During this period, data moves between a source IP address and a target IP address, adhering to a specified protocol. Each connection is categorized as either normal or an attack, with precisely one designated attack type. Individual connection records are approximately 100 bytes in size.

Attacks can be classified into four primary groups:

1. DOS (Denial-of-Service): Examples include SYN floods.
2. R2L (Remote-to-Local): Involves unauthorized access from a remote machine, such as password guessing.
3. U2R (User-to-Root): Unauthorized attempts to gain local superuser (root) privileges, like various "buffer overflow" attacks.
4. Probing: Activities related to surveillance and probing, such as port scanning.

Data Normalization

Normalization, as defined by Aldairi et al. (2019), involves processing raw data and extracting essential attributes to construct features. In this study, Recursive Feature Elimination (RFE) was employed to eliminate inconsistent

data, alongside the transformation of categorical variables into numeric representations.

Process Flow

Figure 1 illustrates the Process Flow Diagram (PFD), which provides a visual depiction of the sequential steps undertaken in this study. Initiated by the collection of datasets, the process advances through phases such as data normalization, feature engineering selection, training deep learning algorithms, classification/prediction, and evaluation of results and performance. Each stage is meticulously organized and implemented with the objective of revealing insights, fostering innovative solutions, and propelling advancements in the domain of insider threat detection.

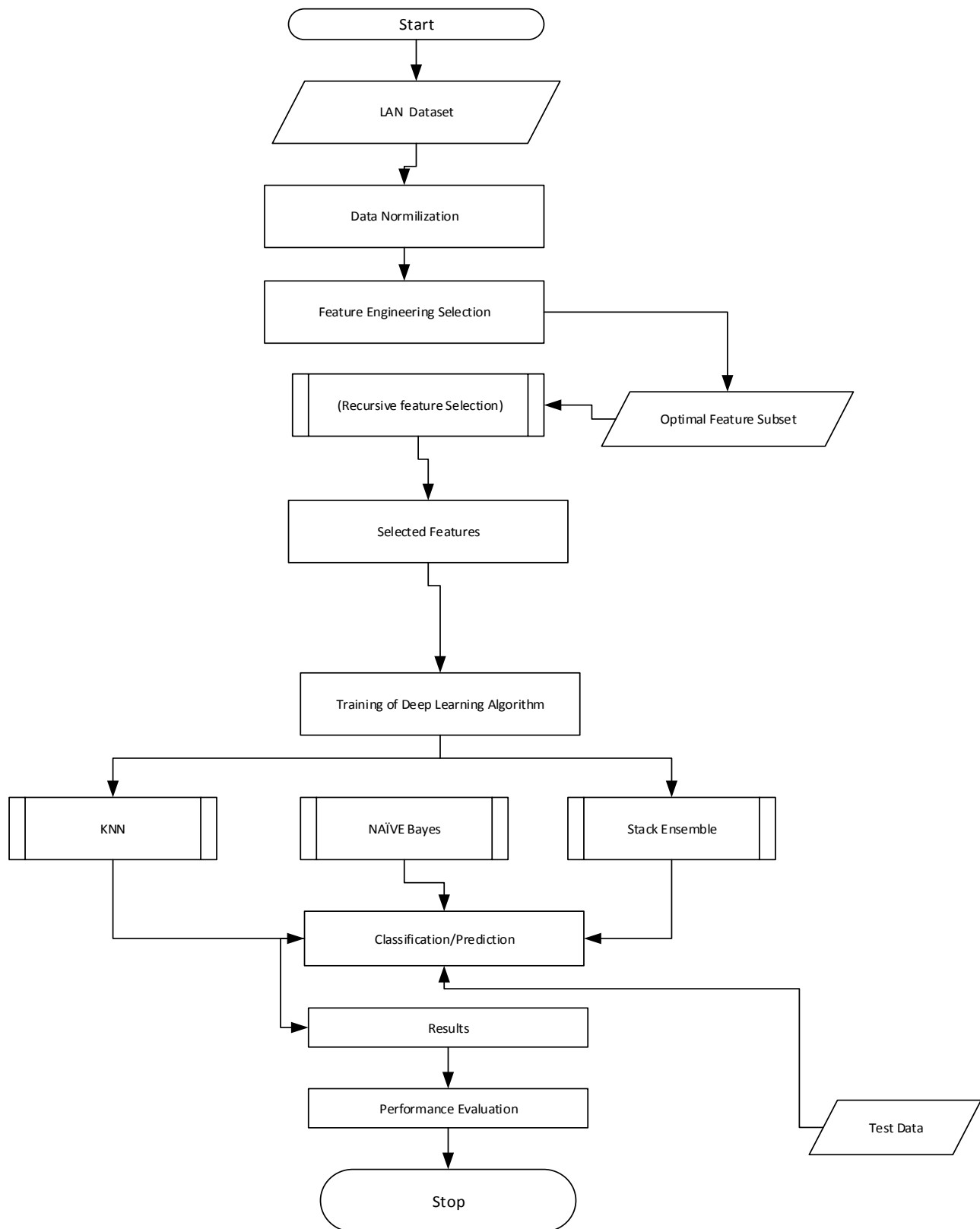


Figure 1: Process Flow Diagram (PFD)

Recursive Feature Elimination

Working with a subset of characteristics rather than all of them reduces the danger of over-fitting, improves accuracy, and cuts training time in half. The RFE was used to select optimal features from the acquired dataset, it is a feature selection method that fits a model and removes the weakest feature (or features) until the desired amount of features is reached. As shown in Table 5, a total of 14 out of the 41 features (column) were selected from the training dataset to be optimally suitable or most relevant in predicting the target variable.

Table 5: Pre-processed Features

S/No	Selected Features
1	'src_bytes',
2	'dst_bytes',
3	'hot',
4	'logged_in',
5	'count',
6	'srv_count',
7	'diff_srv_rate',
8	'dst_host_srv_count',
9	'dst_host_same_srv_rate',
10	'dst_host_diff_srv_rate',
11	'dst_host_same_src_port_rate',
12	'dst_host_srv_diff_host_rate',
13	'protocol_type',
14	'service',

Classifiers

The goal of this study is to classify deviation in user's behaviour on the network as normal or malicious, to achieve this, the study uses KNN, Naïve Bayes and XGBoost algorithms in a stacked ensemble model. The classification stage are formulated into three experiments as represented below:

1. Experiment A: Reduced dataset + KNN.
2. Experiment B: Reduced dataset + NAÏVE BAYES.
3. Experiment C (Ensemble): Experiment ((A and B) + XGBOOST).

Performance Evaluation

The evaluation parameter shows the result of the each of the classifiers used in this study. The testing (probing) evaluation was achieved using the True Positive rate (TP), False Positive (FP), True Negative (TN) and, False Negative (FN), Error rate, Classification Accuracy, sensitivity, Specificity, ROC and Error Rate.

Evaluation Metrics

1. Accuracy: this is the percentage of the instances, which are classified correctly by the classifier

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

2. True positive (TP) rate: TP

is the number of fault-prone modules accurately classified. The TP rate assesses how well a classifier recognizes fault-prone modules. It is also known as a sensitivity measure.

$$\text{TP rate} = \frac{TP}{TP + TN}$$

3. False positive (FP) rate: The FP rate is the percentage of non-fault-prone modules that were classified wrongly.

$$\text{FP rate} = \frac{FP}{FP + TN}$$

4. Precision is the number of classified fault-prone modules that are genuinely fault-prone modules.

$$\text{Precision} = \frac{TP}{TP + FP}$$

5. Recall: This is the percentage of fault-prone modules that are correctly classified.

$$\text{Recall} = \frac{TP}{TP + FN}$$

6. F-measure: It is the harmonic mean of precision and recall

$$\text{F-measure} = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

Implementation Platform

This study utilized the Python programming language to develop the proposed model. While other platforms exist, Python was chosen for its extensive collection of libraries, frameworks, and extensions, facilitating the implementation of various functionalities. For this study, the following libraries and frameworks were employed:

1. Keras, TensorFlow, and Scikit-learn for machine learning.
2. NumPy for high-performance scientific computing and data analysis.
3. SciPy for advanced computing.
4. Pandas for general-purpose data analysis.
5. Seaborn for data visualization.

Results

The results obtained from the implementation of the proposed models as shown in Table 6. Weighted precision, recall, F1-score and accuracy attained were calculated using the evaluation metrics highlighted above

Table 6: Experimental Results

Technique	F1-score	Recall	Precision	Specificity	Sensitivity	Accuracy (%)	Error rate
KNN	0.99	0.99	0.99	0.98407	0.9878	98.58	0.0142
NB	0.92	0.93602	0.92	0.93118	0.90505	91.88	0.0812
Stack Ensemble	0.99	0.98653	0.99	0.98617	0.98653	98.63	0.0137

Computational Time

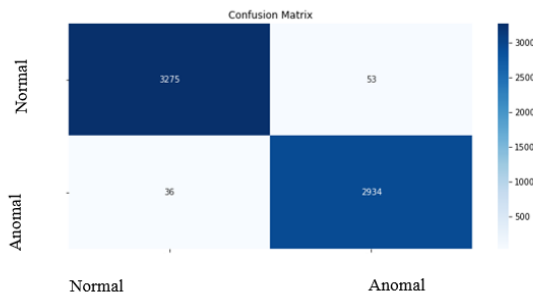
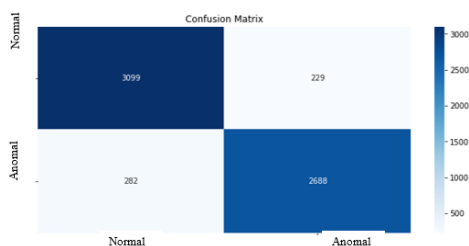
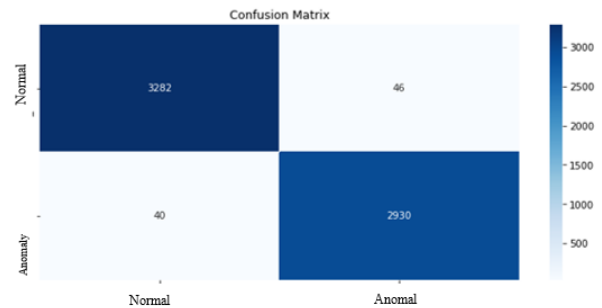
The actual time taken for the implementation of each of the proposed models i.e. the duration for training and testing the models is displayed in Table 7.

Table 7: Execution Time

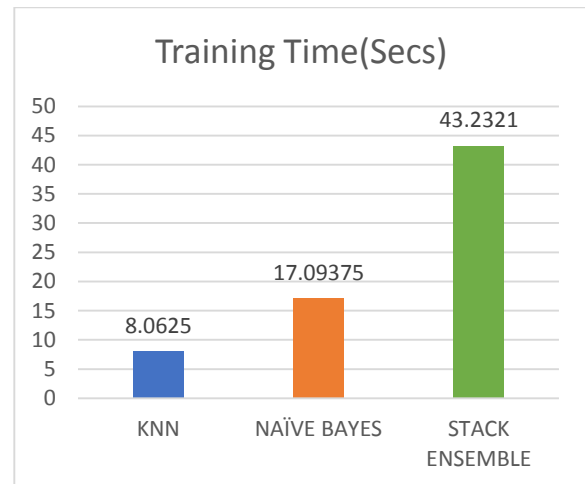
Technique	Training Time (Secs)
KNN	8.0625
NB	17.09375
Stack Ensemble	43.2321

Confusion Matrix

Based on the dataset, Confusion Matrix highlighted the correctly and incorrectly classified instances for each of the proposed models, these are depicted in Figures 2, 3, and 4 respectively.

**Figure 2: KNN Confusion Matrix****Figure 3: Naïve Bayes Confusion Matrix****Figure 4: Stacked Ensemble Confusion Matrix****Graphical Analysis**

The graphical analysis presents a comparison of the training time, classification accuracy, specificity, sensitivity, and error across the proposed models. The graphical illustration is shown in Figure 5.

**Figure 5: Execution Time****Classification Accuracy**

The stacked ensemble model achieved the highest classification accuracy of 98.63%, surpassing other models. This result validates the superior enhancement capability of the ensemble model. Figure 6 detailed the accuracy of the proposed models.

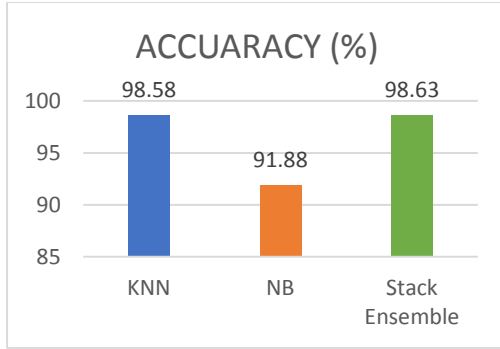


Figure 6: Classification Accuracy

Sensitivity and Specificity

Sensitivity (SN) is determined by dividing the number of accurate positive predictions by the total number of positives, while Specificity (SP) is calculated by dividing the number of accurate negative predictions by the total number of negatives. Optimal sensitivity and specificity reach a value of 1. The results demonstrate that both sensitivity and specificity rates are close to 1, indicating a strong predictive capability. The stacked ensemble model achieved the highest rates for both sensitivity and specificity, as depicted in Figure 7.

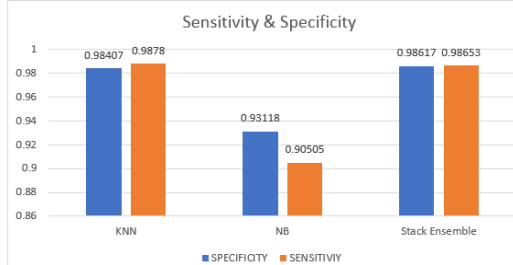


Figure 7: Sensitivity and Specificity Chart

Comparative Analysis for Error rate

The error rate represents the minimum possible error across all classifiers in a random classification scenario. The stacked ensemble achieves an error rate of 0.0137, indicating its superior performance over the two individual base classifiers. The graphical analysis of the error rate is illustrated in Figure 8.

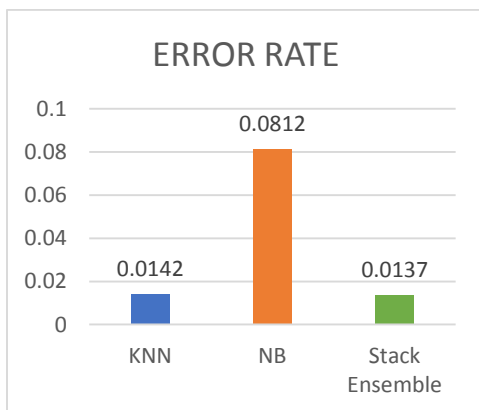


Figure 8: Error Rate

Discussion

Results obtainable from reviewed literature revealed successes recorded in the design and implementation of techniques to curb the menace of Insider Threats even though the techniques were far from perfect. (Aldairi *et al.*, 2019; Eldardiry, Bart, Juan, Hanley *et al.*, 2013; Saadi *et al.*, 2018) made use of system log files (logon, https, LDAP, external drive usage) but had difficulty treating these domains as one entity, this is due to data inhomogeneity. Also, numeric representations of these domains purposely for discretization algorithms that relies on digits for its implementation, this became a difficult task and requires the introduction of special data preprocessing technique. However, a high level of false positives marred the technique implemented by (Le & Zincir-Heywood, 2019; Wei *et al.*, 2021), these techniques proved too risky to call a perfect success, if practically implemented it will many times suspects the right person for wrong doings.

In view of the foregoing, this study focuses on limitations of the aforementioned literature thus make use of Network analysis procedures to detect insiders within an organization's Local Area Network. The proposed ensemble model recorded a 98% near perfect accuracy in the detection of threats/attacks within an office network environment, the model if practically implemented reduces the false alarm, and at the same cut the cost of investigation and saves the organization funds and resources.

In conclusion, the time taken for the implementation of each of the proposed techniques i.e. predictions of the base classifiers (KNN and Naïve Bayes) as well as the ensemble technique as shown in Table 7 shows that ensemble technique executed faster than individual implementations of the base classifiers.

Conclusion

The study showcases the effectiveness of an ensemble model for insider threat detection within an office network environment, emphasizing its superior predictive performance over individual classifiers like KNN and Naïve Bayes. By employing a Meta classifier (XGBoost Stacking) that integrates predictions from the two base classifiers, the model achieves near-perfect accuracy (98%). However, the study is limited to detection and does not address mitigation or remediation strategies. Future research should focus on enhancing the model by incorporating additional user behaviours such as email communication patterns, browser activity, file access and modification to further improve its accuracy and practical applicability.

References

- Aldairi, M., Karimi, L., & Joshi, J. (2019). A Trust Aware Unsupervised Learning Approach for Insider Threat Detection. *IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI)*, 89–98. <https://doi.org/10.1109/IRI.2019.00027>
- Eldardiry, H., Bart, E., Juan Liu, Hanley, J., Price, B., & Brdiczka, O. (2013). Multi-Domain Information Fusion for Insider Threat Detection. *IEEE Security*

- and *Privacy Workshops*, 45–51.
<https://doi.org/10.1109/SPW.2013.14>
- Fagade, T., & Tryfonas, T. (2017). Malicious Insider Threat Detection: A Conceptual Model. *Security and Protection of Information* 15.
- Faisal Janjua, Asif Masood, Haider Abbas, and Imran Rashid (2020). Handling Insider Threat through Supervised Machine Learning Techniques. *Proceeding of Computer Science* 177 (2020) 64-71.
- Le, D. C., & Zincir-Heywood, A. N. (2019). Machine learning based Insider Threat Modelling and Detection. *IFIP/IEEE Symposium on Integrated Network and Service Management (IM)* pp 1-6.
- Nasir, R., Afzal, M., Latif, R., & Iqbal, W. (2021). Behavioral Based Insider Threat Detection Using Deep Learning. *IEEE Access*, 9, 143266–143274.
<https://doi.org/10.1109/ACCESS.2021.3118297>
- Nicolaou, A., Shiaeles, S., & Savage, N. (2020). Mitigating Insider Threats Using Bio-Inspired Models. *Applied Sciences*, 10(15), 5046.
<https://doi.org/10.3390/app10155046>.
- Pantelidis, E., Bendiab, G., Shiaeles, S., & Kolokotronis, N. (2021). Insider Threat Detection using Deep Autoencoder and Variational Autoencoder Neural Networks. *IEEE International Conference on Cyber Security and Resilience (CSR)*, 129–134.
<https://doi.org/10.1109/CSR51186.2021.9527925>
- Paul, S., & Mishra, S. (2020). LAC: LSTM AUTOENCODER with Community for Insider Threat Detection. *4th International Conference on Big Data Research (ICBDR'20)*, 71–77.
<https://doi.org/10.1145/3445945.3445958>
- Saadi, A., Al-Ibadi, Z., Tong, Y., & Farkas, C. (2018). Insider Threats Detection Using CNN-LSTM Model. *International Conference on Computational Science and Computational Intelligence (CSCI)*, 94–99.
<https://doi.org/10.1109/CSCI46756.2018.00025>.
- Wei, Y., Chow, K.-P., & Yiu, S.-M. (2021). Insider threat prediction based on unsupervised anomaly detection scheme for proactive forensic investigation. *Forensic Science International: Digital Investigation*, 38, 301126. <https://doi.org/10.1016/j.fsidi.2021.301126>.