Cite this research:
Kayode-Ajala, O.,(2021). Anomaly Detection in Network Intrusion Detection Systems using and Machine Learning and Dimensionality Reduction SSRAML SageScience, 4(1), 12–26.

Article history:
Received:
January/11/2021
Accepted:
April/07/2021

.

Anomaly Detection in Network Intrusion Detection Systems Using Machine Learning and Dimensionality Reduction

Olaolu Kayode-Ajala
Independent Researcher

Abstract

A fundamental aspect of cybersecurity is the detection of network intrusions, which pose a significant threat to the confidentiality and integrity of sensitive data. Network Intrusion Detection Systems (NIDS) are crucial tools for identifying and responding to unauthorized access or malicious activities within a network. This study investigates the efficacy of various machine learning algorithms for the classification of network traffic into normal and anomalous categories, employing the NSL-KDD dataset as a benchmark. We apply a rigorous preprocessing pipeline, including feature scaling and dimensionality reduction using Principal Component Analysis (PCA). The dataset contains 122 original features, which are reduced to 20 principal components while preserving meaningful information. To assess the performance of our models, we utilize seven different machine learning algorithms: Logistic Regression, K-Neighbors Classifier, Gaussian Naive Bayes (Gaussian NB), Linear Support Vector Classifier (Linear SVC), Decision Tree Classifier, Random Forest Classifier, and a variant of Random Forest with PCA. The following metrics are employed for evaluation: training and test accuracy, precision, and recall. Logistic Regression exhibits competitive results with a training accuracy of 86.97% and a test accuracy of 86.62%. K-Neighbor Classifier surpasses other models with training accuracy (98.05%) and test accuracy (97.94%). Gaussian NB, Linear SVC, Decision Tree Classifier, and Random Forest Classifier all exhibit good performance, consistently achieving high accuracy, precision, and recall scores. Incorporating PCA into the Random Forest Classifier provides a minimal reduction in performance, ensuring that dimensionality reduction does not compromise the model's effectiveness. The PCA Random Forest demonstrates a training accuracy of 98.99% and a test accuracy of 98.83%. Our findings suggest the suitability of these machine learning algorithms for intrusion detection tasks, with K-Neighbors Classifier standing out as the most robust performer in this study. Dimensionality reduction via PCA found streamlining computation without a significant sacrifice in model accuracy. This came at the expense of a slight reduction in recall, indicating a trade-off between precision and sensitivity to positive instances. The Random Forest analysis identified login attempts as the most crucial feature for network classification in intrusion detection, followed by the rate of contacting different destination hosts for the same service. Moreover, according to the findings of this study, Guest vs. non-guest logins, data volume transfer, service type, and service rate variations were also vital factors for accurate network traffic classification.

Keywords: Network Intrusion Detection, Machine Learning Algorithms, NSL-KDD Dataset, Dimensionality Reduction, Principal Component Analysis (PCA), Cybersecurity

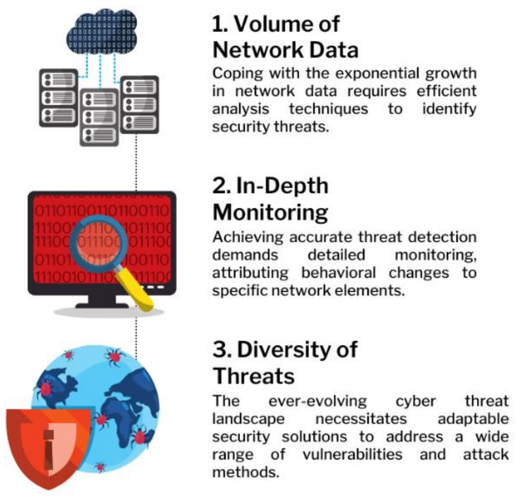
Introduction

An Intrusion Detection System (IDS) is a crucial component within network security infrastructure designed to scrutinize and monitor internet traffic with the primary objective of identifying and preventing unauthorized access or malicious activities [1], [2]. IDSs are employed to safeguard networks from a wide spectrum of security threats, including but not limited to, unauthorized access attempts, malware intrusions, and denial-of-service attacks. One of the distinguishing features of IDSs is their ability to not only detect potential threats but also to take proactive measures, such as blocking traffic originating from unverified or suspicious IP addresses. This capability is instrumental in mitigating security breaches promptly [3], [4].

In recent years, there has been a significant surge in the frequency and complexity of network attacks, necessitating the enhancement and refinement of IDS technology. These attacks encompass a wide range of tactics, from stealthy intrusion attempts to sophisticated malware propagation techniques. Consequently, IDSs have evolved to incorporate advanced algorithms and machine learning models, enabling them to detect subtle anomalies and patterns indicative of a security breach. Additionally, IDSs now operate in real-time, continuously analyzing network traffic to promptly identify and respond to emerging threats, thereby reducing the potential for damage and data loss [5].

The escalation in network attacks underscores the critical importance of IDSs in today's cybersecurity landscape. Organizations must invest in robust and adaptable IDS solutions that can adapt to evolving threats. Moreover, the collaboration between IDSs and other security measures, such as firewalls and intrusion prevention systems (IPS), is essential to create a comprehensive defense strategy. As the threat landscape continues to evolve, IDS technology must remain at the forefront of network security, continuously evolving to meet the challenges posed by sophisticated and relentless adversaries.

Figure 1. challenges of modern network security


1. Volume of Network Data
Coping with the exponential growth in network data requires efficient analysis techniques to identify security threats.
2. In-Depth Monitoring
Achieving accurate threat detection demands detailed monitoring, attributing behavioral changes to specific network elements.
3. Diversity of Threats
The ever-evolving cyber threat landscape necessitates adaptable security solutions to address a wide range of vulnerabilities and attack methods.

The proliferation of hacker probes and malicious attacks targeting computer networks underscores the growing urgency of effective network security measures. One of the key challenges in this context lies in the differentiation between anomalous network activity and the norm, a task that is not only arduous but also time-consuming. To discern potential threats, human analysts are often required to sift through extensive volumes of network data, seeking out irregular sequences of network connections [6], [7].

The process of distinguishing between legitimate network traffic and potentially malicious activity is indeed a formidable challenge. This complexity arises from the myriad tactics employed by hackers, who continually adapt and refine their techniques to evade detection. As a result, network security analysts are burdened with the responsibility of meticulously examining network logs and data, a task that demands a deep understanding of network protocols and patterns.

Efforts to streamline and enhance this process have led to the development of advanced Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS). These systems leverage sophisticated algorithms, machine learning, and pattern recognition techniques to automate the detection of anomalous behavior within network traffic. By doing so, they alleviate the burden on human analysts, allowing them to focus on investigating and responding to identified threats rather than exhaustively combing through data.

The challenge of network security is compounded by three significant limitations, each of which plays a pivotal role in exacerbating this issue. Firstly, the exponential growth in the volume of network data presents a formidable hurdle that is expected to persist. This surge in data can be primarily attributed to the increasing levels of connectivity, the widespread adoption of Internet of Things (IoT) devices, and the extensive reliance on cloud-based services [8], [9]. Addressing this burgeoning data volume necessitates the development of techniques capable of swiftly, efficiently, and effectively analyzing this data to identify potential security threats.

The second limitation stems from the demand for in-depth monitoring and granularity to enhance the accuracy and effectiveness of Network Intrusion Detection Systems (NIDS). To achieve improved threat detection, NIDS analysis must transition from abstract and high-level observations to a more detailed and contextually-aware approach. This entails the ability to attribute behavioral changes within the network to specific elements, such as individual users, distinct operating system versions, or particular network protocols. Achieving this level of granularity is essential for accurately pinpointing the source and nature of potential security breaches [10]–[12].

The third limitation pertains to the sheer diversity of network threats and attack vectors. The evolving landscape of cyber threats encompasses a multitude of techniques and attack methodologies. Consequently, security systems must contend with a vast array of potential vulnerabilities and malicious activities. This necessitates ongoing research and development efforts to create adaptable and comprehensive security solutions capable of addressing the multitude of evolving threats [13], [14].

There are two primary categories of NIDS, namely Signature-based NIDS (SNIDS) and Anomaly Detection-based NIDS (ADNIDS), each offering distinct approaches to threat detection [15]–[17].

In the case of SNIDS, the system operates by identifying attacks based on predefined rules that have been pre-installed specifically for known attack patterns. Network traffic is continuously compared against an updated database of attack signatures, allowing the system to promptly detect intrusions within the network traffic dataset. This approach is effective in identifying known threats but may struggle with detecting previously unseen attacks or evolving threats [18]–[20].

On the other hand, ADNIDS takes a different approach by classifying unknown or abnormal network behavior. It achieves this by meticulously analyzing the structures of normal network traffic behavior, forming a baseline. Any network traffic that deviates from this established pattern is flagged as a potential intrusion. This unique methodology equips ADNIDS with the capability to detect previously unknown or novel attacks, thereby enhancing the security posture of an organization. One of the notable advantages of ADNIDS is its capacity to predict and detect new and emerging threats, filling the gap left by SNIDS, which relies heavily on established attack signatures. This predictive capability is particularly valuable in the ever-evolving landscape of cyber threats where attackers constantly devise novel strategies to breach network defenses. As a result, ADNIDS has become an indispensable component of modern network security strategies, offering enhanced protection against a broader spectrum of threats, including those that have yet to be documented.

Data and preprocessing

The NSL-KDD dataset comprises records of internet traffic observed by a basic intrusion detection network, where, the traffic encountered by an actual IDS, with only traces of its presence remaining [21]. These records consist of 43 features per entry, where 41 features pertain to the characteristics of the observed traffic, and the remaining two serve as labels. These labels include information on whether the traffic is normal or indicative of an attack, as well as a severity score associated with the observed traffic input [22], [23].

Within this dataset, there are four distinct classes of attacks that can be identified and categorized. Firstly, there is the Denial of Service (DoS) category, which encompasses network connections associated with attacks aimed at overwhelming system resources, rendering them inaccessible to legitimate users [24], [25]. Second is the Probe category, which encompasses network connections related to probing or scanning activities, often initiated by attackers in an attempt to gather information about the target system or network. Thirdly, the Remote-to-Local (R2L) category includes network connections associated with remote-to-local attacks, wherein unauthorized attempts are made to access a system or network remotely, typically by exploiting vulnerabilities to gain access. Lastly, the User-to-Root (U2R) category represents attacks where individuals with limited privileges on a system attempt to escalate their privileges to achieve superuser or root access.

Table 1. Categories in NSL-KDD dataset

Category	Description
Normal	This category represents normal network traffic or connections that are not associated with any intrusion or malicious activity.
Denial-of-Service (DoS)	This category includes network connections that are part of denial-of-service attacks, where an attacker overwhelms a system or network resource to make it unavailable to users [26].
Probe	The Probe category encompasses network connections that involve probing or scanning activities, such as reconnaissance attempts by attackers to gather information about the target system or network [27].
Remote-to-Local (R2L)	The R2L category includes network connections related to remote-to-local attacks. These attacks involve unauthorized attempts to access a system or network remotely, typically exploiting vulnerabilities to gain access [28].
User-to-Root (U2R)	The U2R category represents user-to-root attacks, where an attacker with limited privileges on a system attempts to escalate their privileges to gain superuser or root access.

The NSL-KDD dataset represents a significant improvement over the original KDD'99 dataset in several key aspects. First and foremost, it addresses the issue of redundancy present in the training set, ensuring that classifiers are not skewed toward more frequently occurring records. By eliminating redundant entries, the dataset promotes fairness and accuracy in the evaluation of machine learning algorithms for intrusion detection. Furthermore, the NSL-KDD dataset eliminates duplicate records in the proposed test sets, eliminating bias that could arise from methods with superior detection rates on common records. This adjustment enhances the objectivity and reliability of performance assessments for various learning algorithms [29].

Another enhancement in the NSL-KDD dataset is the careful selection of records from different difficulty levels, proportionate to their representation in the original KDD data set. This approach widens the range of classification rates achieved by different machine learning methods, facilitating a more precise evaluation of diverse learning techniques and their efficacy in intrusion detection.

Additionally, the NSL-KDD dataset ensures that the number of records in both the training and test sets remains reasonable. This eliminates the need for random sampling of a smaller subset for experiments, ensuring consistent and comparable evaluation results across different research endeavors. These improvements collectively address the critical shortcomings of the original KDD dataset, particularly the problem of bias toward frequent records, and enhance the dataset's suitability for robust intrusion detection research [30].

In the data preprocessing phase, a binary encoding scheme was applied to the 'outcome' column of the dataset. For rows where the 'outcome' is labeled as "normal," a value of 0 was assigned, indicating a normal network activity. Conversely, for rows where the 'outcome' deviates from "normal" and represents an anomaly or intrusion, a value of 1 was assigned. This encoding simplifies the classification task, allowing for the clear differentiation between normal and anomalous network behaviors

We applied Robust Scaler to transform the dataset. Robust Scaler operates by first eliminating the median value from the dataset, which effectively centers the data around zero. This step is crucial in mitigating the influence of outliers on subsequent analysis. Subsequently, Robust Scaler scales the data based on the interquartile range (IQR), which is the range between the 1st quartile (25th quantile) and the 3rd quartile (75th quantile). This scaling procedure ensures that the data is more resistant to the impact of extreme values, making it particularly well-suited for datasets with outliers. By using the IQR for scaling, Robust Scaler effectively transforms the data into a more standardized form, contributing to improved model performance and reliability in various machine learning applications.

Methods

To evaluate the effectiveness of our models, we employ seven different machine learning algorithms: Logistic Regression, K-Neighbors Classifier, Gaussian Naive Bayes (Gaussian NB), Linear Support Vector Classifier (Linear SVC), Decision Tree Classifier, Random Forest Classifier, and a modified version of Random Forest incorporating Principal Component Analysis (PCA).

Logistic Regression is a widely used statistical model for binary classification tasks. It is a linear model that predicts the probability of a binary outcome based on one or more predictor variables. The key idea behind logistic regression is to model the log-odds of the probability of the positive class as a linear combination of the predictor variables. This is achieved through the logistic function, also known as the sigmoid function, which maps any real-valued number into the range [0, 1]. Logistic Regression is computationally efficient and relatively simple to implement, making it a popular choice for various applications, including spam detection, medical diagnosis, and credit risk assessment. However, it assumes a linear relationship between the predictors and the log-odds, which may not hold in all cases, and it is sensitive to outliers.

The K-Neighbors Classifier is a non-parametric, instance-based machine learning algorithm used for both classification and regression tasks. In the context of classification, it predicts the class of a data point by considering the classes of its k-nearest neighbors in the training dataset. The choice of 'k' determines the level of smoothing and model complexity [31], [32]. Smaller values of 'k' result in more complex and potentially noisy decision boundaries, while larger values of 'k' result in smoother decision boundaries. K-Neighbors Classifier is simple to understand and implement, and it can handle complex decision boundaries and non-linear relationships in the data. However, it can be sensitive to the choice of 'k' and is computationally expensive for large datasets since it requires calculating distances between the query point and all training data points.

Gaussian Naive Bayes is a probabilistic classification algorithm based on Bayes' theorem. It assumes that the features are continuous and follow a Gaussian (normal) distribution. Despite its simplicity and the "naive" assumption that features are conditionally independent given the class label, Gaussian NB often performs surprisingly well in practice, especially when the independence assumption is reasonably satisfied. It is particularly suited for text classification tasks, such as spam email detection and sentiment analysis. Gaussian NB calculates the likelihood and prior probabilities from the training data and then uses Bayes' theorem to estimate the posterior probabilities of different class labels for a given input. While Gaussian NB is computationally efficient and works well with high-dimensional data, it may not perform as well when the independence assumption is strongly violated or when dealing with categorical or discrete data, where other variants of Naive Bayes, such as Multinomial or Bernoulli NB, may be more appropriate [33], [34].

The Linear Support Vector Classifier, often referred to as Linear SVC, is a supervised machine learning algorithm used primarily for binary classification tasks. It belongs to the family of Support Vector Machines (SVMs) and is specifically designed for linearly separable datasets. Linear SVC works by finding the optimal hyperplane that best separates the two classes while maximizing the margin between them. This hyperplane is determined based on a subset of training data points called support vectors. Linear SVC is particularly effective when dealing with high-dimensional data and datasets with a large number of features. It can also be extended to multiclass classification using strategies like one-vs-all. However, Linear SVC may not perform well on datasets with complex, non-linear decision boundaries, for which kernelized SVMs or other non-linear models might be more suitable.

The Decision Tree Classifier is a widely used machine learning algorithm for both classification and regression tasks. It creates a hierarchical structure resembling a tree to make decisions based on the input features. At each internal node of the tree, the algorithm selects the feature that best splits the data into subsets with the highest homogeneity, typically using measures like Gini impurity or entropy. The decision tree continues to split the data until it reaches a stopping criterion, such as a maximum tree depth or a minimum number of samples per leaf. Decision trees are interpretable, easy to visualize, and can handle both categorical and numerical features. However, they are prone to overfitting, especially when the tree is deep and complex. Techniques like pruning and using ensemble methods like Random Forest can mitigate overfitting issues [35].

The Random Forest Classifier is an ensemble learning method that combines multiple decision trees to improve the accuracy and robustness of the classification model. It operates by creating a collection of decision trees, each trained on a random subset of the training data and using random subsets of the features (bagging). The final prediction is made by aggregating the predictions of individual trees, typically by a majority vote for classification tasks. Random Forests are known for their ability to handle high-dimensional data, reduce overfitting, and provide feature importance rankings. They are less sensitive to hyperparameter tuning compared to individual decision trees and are robust to outliers and noisy data. Random Forests can handle both classification and regression problems effectively and have found applications in various domains, including image recognition, bioinformatics, and finance. However, they may not perform well on extremely imbalanced datasets, and their interpretability is lower compared to single decision trees.

Dimensionality reduction is a fundamental technique in machine learning and data analysis that aims to reduce the number of features or dimensions in a dataset while preserving its essential information [36]. The primary motivation behind dimensionality reduction is to alleviate the curse of dimensionality, which can lead to increased computational complexity, overfitting, and difficulties in visualizing and interpreting data. Dimensionality reduction methods can broadly be categorized into two types: feature selection and feature extraction. Feature selection involves choosing a subset of the original features, while feature extraction creates new features that are combinations of the original ones [37].

Principal Component Analysis, commonly referred to as PCA, is a widely used dimensionality reduction technique. PCA falls under the category of feature extraction and aims to transform the original data into a new set of uncorrelated variables called principal component [38], [39]s. These components are ordered by their variance, with the first principal component capturing the most variance in the data, the second capturing the second most, and so on. By selecting a subset of these principal components, one can effectively reduce the dimensionality of the data while retaining a significant portion of the information.

PCA works by finding the orthogonal linear transformations of the original features that maximize the variance in the data. It has numerous applications, including data compression, noise reduction, and visualization. PCA is particularly useful in scenarios where multicollinearity exists among the original features, as it helps in decorrelating them [40], [41]. However, PCA assumes that the data is centered (i.e., it has zero mean), and it may not perform well when the relationship between variables is non-linear. In such cases, non-linear dimensionality reduction techniques like t-Distributed Stochastic Neighbor Embedding (t-SNE) or Isomap may be more appropriate.

Results

Logistic Regression achieved a training accuracy of 86.97% and a test accuracy of 86.62%. The precision values for training and testing were 82.81% and 82.57%, respectively, while the recall values were 90.86% for training and 90.61% for testing. These results suggest that the Logistic Regression model has a good balance between accuracy and precision, with high recall indicating its ability to correctly identify positive instances.

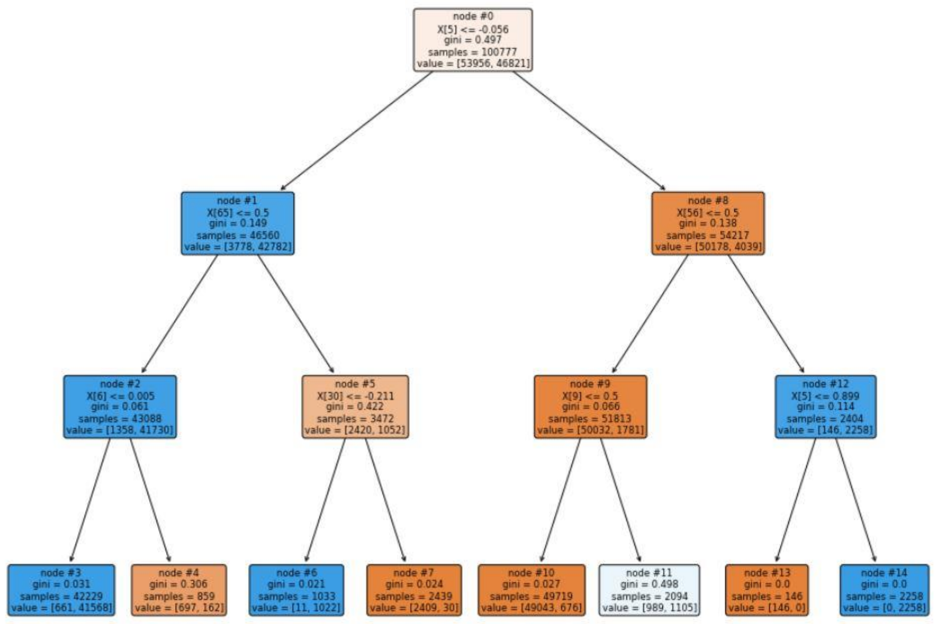
Table 2. Evaluation of the models

Methods/Matrices	Training Accuracy	Test Accuracy	Training Precision	Test Precision	Training Recall	Test Recall
Logistic Regression	86.97	86.62	82.81	82.57	90.86	90.61
KNeighbors Classifier	98.05	97.94	98.23	98.06	97.73	97.67
Gaussian NB	90.8	90.61	91.63	91.53	88.48	88.3
Linear SVC	96.15	95.95	95.25	94.91	96.68	96.65
Decision Tree Classifier	98.99	98.87	99	98.85	98.99	98.87
Random Forest Classifier	98.99	98.89	98.99	98.94	98.99	98.82
PCA Random Forest	98.99	98.83	98.99	98.93	98.99	98.7

In contrast, the KNeighbors Classifier demonstrated superior performance with a training accuracy of 98.05% and a test accuracy of 97.94%. It exhibited high precision for both training and testing, with values of 98.23% and 98.06%, respectively. The recall values were also strong, at 97.73% for training and 97.67% for testing. These results highlight the KNeighbors Classifier's capability to accurately classify instances, particularly evident in its high precision and recall rates.

The Gaussian Naive Bayes (NB) model achieved a training accuracy of 90.80% and a test accuracy of 90.61%. It demonstrated relatively balanced precision values, with 91.63% for training and 91.53% for testing. However, its recall values were slightly lower, with 88.48% for training and 88.30% for testing. This suggests that the Gaussian NB model is robust but may not perform as well as the KNeighbors Classifier in identifying positive instances.

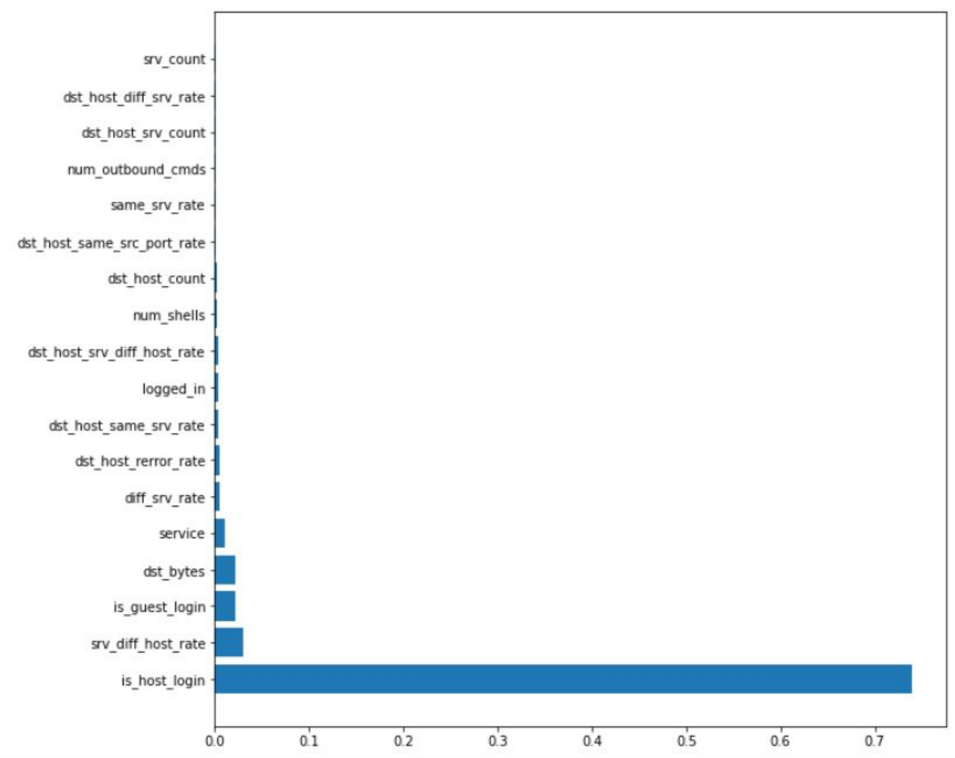
Figure 2. Decision tree



The Linear SVC exhibited a relatively high training accuracy of 96.15%, demonstrating its ability to fit the training data well. The test accuracy of 95.95% indicates that the model generalizes effectively to unseen data. In terms of precision, the training and test precision scores of 95.25% and 94.91%, respectively, reflect the model's ability to make accurate positive predictions. The recall scores, both in training (96.68%) and testing (96.65%), highlight the model's capability to identify a substantial proportion of true positive instances while maintaining a good balance with precision.

Decision Tree model achieved good results with a training accuracy of 98.99% and a test accuracy of 98.87%, demonstrating remarkable performance in capturing the underlying patterns in the training and test datasets. The precision scores, both in training (99.00%) and testing (98.85%), indicate that the model excels in making precise positive predictions. The recall scores, identical in both training and testing at 98.99%, underscore the model's proficiency in identifying true positive instances without compromising precision.

Figure 3. Feature importance random forest.



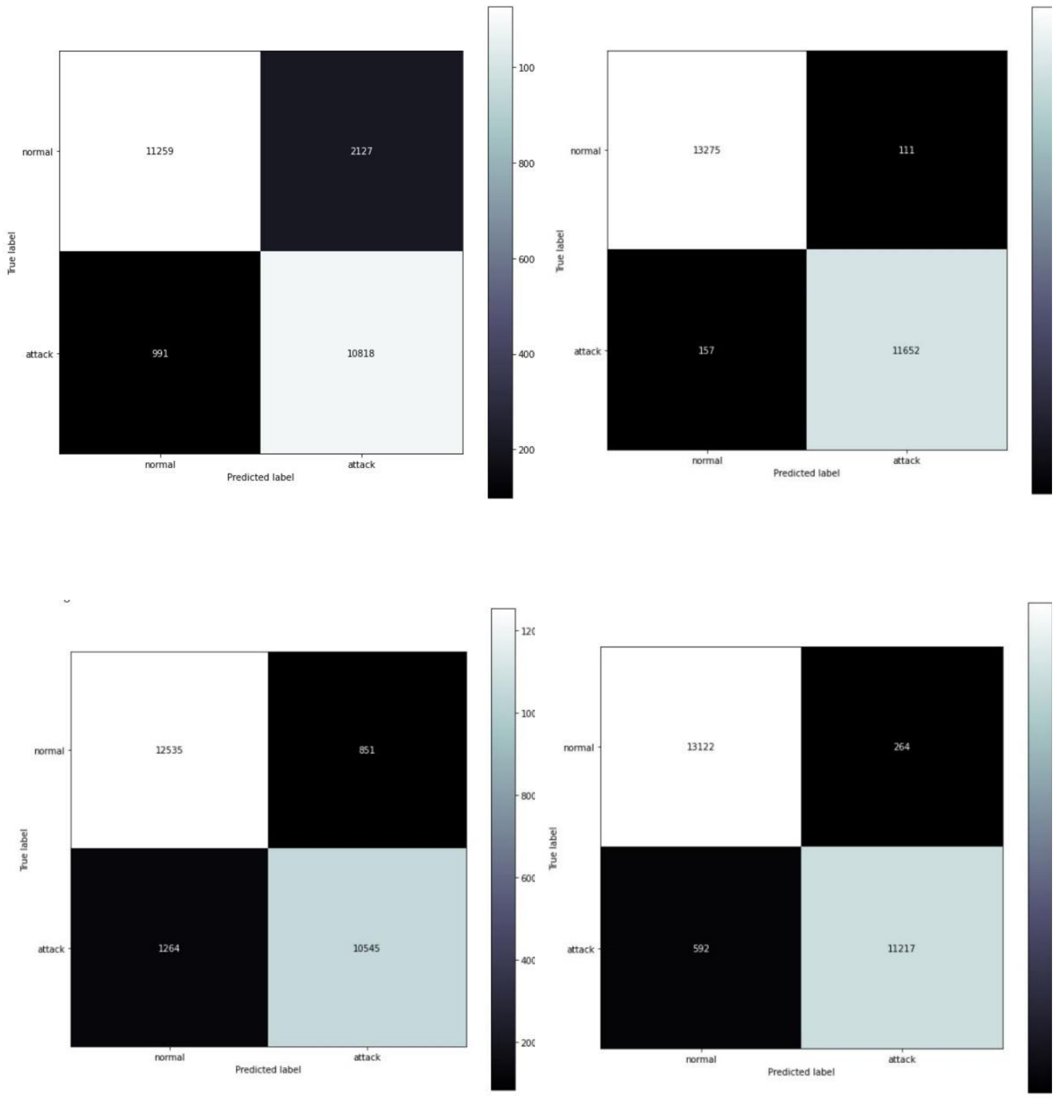
The Random Forest Classifier also delivered strong performance with a training accuracy of 98.99% and a test accuracy of 98.89%. Its precision scores, in both training (98.99%) and testing (98.94%), demonstrate a high level of accuracy in classifying positive instances. While the training recall remained at 98.99%, the test recall slightly decreased to 98.82%, indicating a slight trade-off between recall and precision on the test dataset.

The PCA Random Forest model exhibited commendable results with a training accuracy of 98.99% and a test accuracy of 98.83%. It maintained high precision in both training (98.99%) and testing (98.93%), indicating its ability to make accurate positive predictions. The recall scores, while strong in both training (98.99%) and testing (98.70%).

PCA appeared to enhance precision, as evidenced by high precision scores in both training and testing. However, it also introduced a slight trade-off between precision and recall on the test dataset, as the recall value decreased slightly. This suggests that PCA led to a more precise but somewhat less sensitive model when identifying positive instances.

On the other hand, the PCA Random Forest model exhibited commendable results with high training and testing accuracy and maintained high precision values, similar to the non-PCA Random Forest model. However, the impact of PCA was more pronounced in terms of recall. The recall scores, while still strong, suggested a slightly greater trade-off between recall and precision compared to the non-PCA Random Forest model. This indicates that PCA reduced the model's sensitivity to positive instances to some extent.

Figure 4. (counter-clockwise) Logistic regression, KNN, Gaussian NB, and Linear SVC



The result obtained from feature importance analysis using the Random Forest algorithm provides valuable insights into the significance of different features in a dataset, particularly in the context of intrusion detection or any other classification task. In this case, it's intriguing to observe that the 'is_host_login' feature emerges as the most important one. This suggests that whether a login attempt is made using the host's account or not carries substantial weight in determining the outcome of the classification. This could be indicative of the system's vulnerability to insider attacks, making it a crucial factor to consider when building an intrusion detection model.

Following 'is_host_login,' the 'srv_diff_host_rate' feature is ranked second in importance. This ranking implies that the rate at which different destination hosts are contacted for the same service within a short time frame is a key determinant in classifying network connections. High values in this feature could signify potential scanning or probing activities, which are common precursors to various network attacks. Thus, it underlines the significance of monitoring and analyzing such rates in network traffic for effective intrusion detection.

The third-ranking feature, 'is_guest_login,' adds another layer of context to the analysis. Its high importance suggests that distinguishing between guest and non-guest logins is critical for understanding network behavior. Guest logins often come with limited privileges, and detecting them accurately can help identify potential vulnerabilities or unauthorized access attempts within a network.

Table 3. Important Features

Feature Name	Definition
is_host_login	This binary feature indicates whether the login attempt was made using the host's account or not.
srv_diff_host_rate	This feature represents the rate of different destination hosts that were contacted for the same service as the current connection in the past two seconds.
is_guest_login	This binary feature indicates whether the login attempt was made as a guest or not.
dst_bytes	This feature represents the number of data bytes sent by the destination host during the connection.

'Dst_bytes,' ranked fourth, measures the volume of data sent by the destination host during a connection. Its importance underscores the role of data transfer in characterizing network connections. Unusually high or low data transfer volumes can be indicative of different network activities, including potentially malicious ones such as data exfiltration or denial of service attacks. The importance of the 'service' feature is also found to be significant. It implies that the type of service being accessed plays a significant role in determining the nature of a network connection. Different services have varying levels of security requirements and vulnerabilities, making this an essential feature for understanding and classifying network traffic accurately. The feature 'diff_srv_rate' ranks sixth in importance, indicating that variations in service rates are also relevant for intrusion detection. Differences in service rates can be indicative of suspicious or anomalous behavior and warrant closer scrutiny.

Conclusion

The field of cybersecurity faces ongoing challenges in safeguarding confidential data, necessitating precise methods for detecting network intrusions. This research focuses on the application of machine learning techniques in Network Intrusion Detection Systems (NIDS) and investigates the effectiveness of dimensionality reduction using Principal Component Analysis (PCA) to enhance the efficiency of intrusion detection models. Seven distinct machine learning algorithms are evaluated in this research: Logistic Regression, K-Neighbors Classifier, Gaussian Naive Bayes (Gaussian NB), Linear Support Vector Classifier (Linear SVC), Decision Tree Classifier, Random Forest Classifier, and a variant of Random Forest integrated with PCA. These models are assessed using key performance metrics, including training and test accuracy, precision, and recall.

Logistic Regression demonstrated a good balance between accuracy and precision, with high recall rates, indicating its potential to correctly identify positive instances. On the other hand, the KNeighbors Classifier exhibited superior performance with exceptional accuracy, precision, and recall values, showcasing its ability to accurately classify instances.

The Gaussian Naive Bayes model, while robust, fell slightly behind the KNeighbors Classifier in identifying positive instances. The Linear SVC showed a strong capacity to fit the training data and generalize effectively to unseen data while maintaining a good balance between precision and recall. The Decision Tree model excelled in capturing underlying patterns and making precise positive predictions, with identical high recall scores in both training and testing. Similarly, the Random Forest Classifier displayed strong performance, albeit with a slight trade-off between recall and precision on the test dataset. The PCA Random Forest model, while commendable, exhibited a slightly greater trade-off between recall and precision compared to its non-PCA counterpart. while dimensionality reduction using PCA improved computational efficiency without significant performance degradation, it is essential to acknowledge that this reduction process may discard some nuanced features that could be valuable for detecting anomalies.

The analysis using the Random Forest algorithm revealed the importance of various features in classifying network connections for intrusion detection. The most significant feature was related to login attempts, highlighting its crucial role in determining network behavior. Following that, the rate at which different destination hosts were contacted for the same service emerged as the second most important factor. The distinction between guest and non-guest logins also played a key role in classification. Additionally, the volume of data transferred during a connection, the type of service accessed, and variations in service rates can be essential factors in understanding and classifying network traffic in intrusion detection systems.

References

[1] D. S. Kim, H.-N. Nguyen, and J. S. Park, “Genetic algorithm to improve SVM based network intrusion detection system,” in *19th International Conference on Advanced Information Networking and Applications (AINA’05) Volume 1 (AINA papers)*, 2005, vol. 2, pp. 155–158 vol.2.

[2] R. Roman, J. Zhou, and J. Lopez, “Applying intrusion detection systems to wireless sensor networks,” in *CCNC 2006. 2006 3rd IEEE Consumer Communications and Networking Conference, 2006*, Las Vegas, NV, USA, 2006.

[3] M. S. Hoque, M. A. Mukit, and M. A. N. Bikas, “An implementation of intrusion detection system using genetic algorithm,” *arXiv preprint arXiv:1204.1336*, 2012.

[4] R. Panigrahi and S. Borah, “A detailed analysis of CICIDS2017 dataset for designing Intrusion Detection Systems,” *Int. J. Eng.*, 2018.

[5] M. Almseidin, M. Alzubi, S. Kovacs, and M. Alkasassbeh, “Evaluation of machine learning algorithms for intrusion detection system,” in *2017 IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY)*, 2017, pp. 000277–000282.

[6] N. Farnaaz and M. A. Jabbar, “Random Forest Modeling for Network Intrusion Detection System,” *Procedia Comput. Sci.*, vol. 89, pp. 213–217, Jan. 2016.

[7] M. K. Putchala, “Deep Learning Approach for Intrusion Detection System (IDS) in the Internet of Things (IoT) Network using Gated Recurrent Neural Networks (GRU),” Wright State University, 2017.

[8] S. Sharma and R. K. Gupta, “Intrusion detection system: A review,” *International Journal of Security and Its*, 2015.

[9] F. Sabahi and A. Movaghar, “Intrusion Detection: A Survey,” in *2008 Third International Conference on Systems and Networks Communications*, 2008, pp. 23–26.

[10] M. A. M. Hasan, M. Nasser, B. Pal, and S. Ahmad, “Support vector machine and random forest modeling for intrusion detection system (IDS),” *J. Intell. Learn. Syst. Appl.*, vol. 06, no. 01, pp. 45–52, 2014.

[11] Z. M. Fadlullah, H. Nishiyama, N. Kato, and M. M. Fouda, “Intrusion detection system (IDS) for combating attacks against cognitive radio networks,” *IEEE Netw.*, vol. 27, no. 3, pp. 51–56, May 2013.

[12] J. Jabez and B. Muthukumar, “Intrusion Detection System (IDS): Anomaly Detection Using Outlier Detection Approach,” *Procedia Comput. Sci.*, vol. 48, pp. 338–346, Jan. 2015.

[13] K. A. Jackson, “Intrusion detection system (ids) product survey,” *Los Alamos National Laboratory*, 1999.

[14] R. Vinayakumar and K. P. Soman, “Evaluation of recurrent neural network and its variants for intrusion detection system (IDS),” *of Information System …*, 2017.

[15] W. Lee and S. J. Stolfo, “A framework for constructing features and models for intrusion detection systems,” *ACM Trans. Inf. Syst. Secur.*, 2000.

[16] C. F. Tsai, Y. F. Hsu, C. Y. Lin, and W. Y. Lin, “Intrusion detection by machine learning: A review,” *Expert Syst. Appl.*, 2009.

[17] S. Chebrolu, A. Abraham, and J. P. Thomas, “Feature deduction and ensemble design of intrusion detection systems,” *Comput. Secur.*, vol. 24, no. 4, pp. 295–307, Jun. 2005.

[18] T.-T.-H. Le, J. Kim, and H. Kim, “An Effective Intrusion Detection Classifier Using Long Short-Term Memory with Gradient Descent Optimization,” in *2017 International Conference on Platform Technology and Service (PlatCon)*, 2017, pp. 1–6.

[19] L. Dhanabal and S. P. Shantharajah, “A study on NSL-KDD dataset for intrusion detection system based on classification algorithms,” *International journal of advanced*. e-tarjome.com, 2015.

[20] H. G. Kayacik and A. N. Zincir-Heywood, “Selecting features for intrusion detection: A feature relevance analysis on KDD 99 intrusion detection datasets,” *on privacy, security and …*, 2005.

[21] S. Duque and M. N. B. Omar, “Using Data Mining Algorithms for Developing a Model for Intrusion Detection System (IDS),” *Procedia Comput. Sci.*, vol. 61, pp. 46–51, Jan. 2015.

[22] S. Revathi and A. Malathi, “A detailed analysis on NSL-KDD dataset using various machine learning techniques for intrusion detection,” *International Journal of Engineering Research & Technology (IJERT)*, vol. 2, no. 12, pp. 1848–1853, 2013.

[23] L. M. Ibrahim and D. T. Basheer, “A comparison study for intrusion database (Kdd99, Nsl-Kdd) based on self organization map (SOM) artificial neural network,” *Journal of Engineering*, 2013.

[24] A. K. Shrivas and A. K. Dewangan, “An ensemble model for classification of attacks with feature selection based on KDD99 and NSL-KDD data set,” *International Journal of computer*, 2014.

[25] B. Ingre and A. Yadav, “Performance analysis of NSL-KDD dataset using ANN,” in *2015 International Conference on Signal Processing and Communication Engineering Systems*, 2015, pp. 92–96.

[26] M. S. Pervez and D. M. Farid, “Feature selection and intrusion classification in NSL-KDD cup 99 dataset employing SVMs,” in *The 8th International Conference on*

Software, Knowledge, Information Management and Applications (SKIMA 2014), 2014, pp. 1–6.

[27] D. Protić D., “Review of KDD Cup ’99, NSL-KDD and Kyoto 2006+ datasets,” *Vojnoteh. Glas.*, vol. 66, no. 3, pp. 580–596, Apr. 2018.

[28] R. Thomas and D. Pavithran, “A Survey of Intrusion Detection Models based on NSL-KDD Data Set,” in *2018 Fifth HCT Information Technology Trends (ITT)*, 2018, pp. 286–291.

[29] H.-S. Chae, B.-O. Jo, S.-H. Choi, and T.-K. Park, “Feature Selection for Intrusion Detection using NSL-KDD,” *Recent advances in computer*, 2013.

[30] G. Meena and R. R. Choudhary, “A review paper on IDS classification using KDD 99 and NSL KDD dataset in WEKA,” in *2017 International Conference on Computer, Communications and Electronics (Comptelix)*, 2017, pp. 553–558.

[31] V. Mitra, C.-J. Wang, and S. Banerjee, “Text classification: A least square support vector machine approach,” *Appl. Soft Comput.*, vol. 7, no. 3, pp. 908–914, Jun. 2007.

[32] L. R. Hazım, “Four classification methods Naïve Bayesian, support vector machine, K-nearest neighbors and random forest are tested for credit card fraud detection,” Altınbaş Üniversitesi, 2018.

[33] D. A. D. D. Agrawal, “COMPARISONS OF CLASSIFICATION ALGORITHMS ON SEEDS DATASET USING MACHINE LEARNING ALGORITHM,” *Compusoft*, vol. 7, no. 5, pp. 2760–2765, May 2018.

[34] C. Rahmad, R. Ariyanto, and D. Rizky, “Brain signal classification using genetic algorithm for right-left motion pattern,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 9, no. 11, 2018.

[35] R. Vinayakumar, K. P. Soman, and P. Poornachandran, “Applying convolutional neural network for network intrusion detection,” in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2017, pp. 1222–1228.

[36] M. E. Tipping and C. M. Bishop, “Probabilistic principal component analysis,” *J. R. Stat. Soc.*, 1999.

[37] S.-H. Kim and G. H. Dunteman, “Principal Components Analysis,” *J. Educ. Stat.*, vol. 16, no. 2, p. 141, 1991.

[38] J. Shlens, “A tutorial on principal component analysis,” *arXiv preprint arXiv:1404.1100*, 2014.

[39] S. Wold, K. Esbensen, and P. Geladi, “Principal component analysis,” *Chemometrics Intellig. Lab. Syst.*, vol. 2, no. 1, pp. 37–52, Aug. 1987.

[40] C. R. Rao, “The Use and Interpretation of Principal Component Analysis in Applied Research,” *Sankhyā: The Indian Journal of Statistics, Series A (1961-2002)*, vol. 26, no. 4, pp. 329–358, 1964.

[41] R. Bro and A. K. Smilde, “Principal component analysis,” *Anal. Methods*, 2014.