Intrusion Detection with Supervised Machine Learning using SMOTE for Imbalanced Datasets

Joan Telo

RECEIVED
17 September 2020
REVISED
18 October 2020

ACCEPTED FOR PUBLICATION
20 January 2021
PUBLISHED
21 January 2021

Keywords: Imbalanced datasets, Intrusion detection, Machine learning, NSLKDD dataset, SMOTE algorithm

Abstract

The This research study explores the importance of intrusion detection and the use of machine learning algorithms for detecting intrusions. One of the major issues faced in intrusion detection is the problem of imbalanced datasets, where one class is significantly underrepresented compared to the others. To address this issue, the study uses the NSLKDD dataset, which contains five classes: Benign, DOS, Probe, R2L, and U2R. The study employs three machine learning methods, namely Decision tree, KNN, and Linear SVC, to classify the different classes of intrusions. The error rates of these classifiers were measured before and after applying the Synthetic Minority Over-sampling Technique (SMOTE) algorithm, which is a popular technique for balancing imbalanced datasets. Before applying SMOTE, the Linear SVC algorithm had the highest error rate (0.278389), followed by the Decision tree algorithm (0.237890), and KNN algorithm (0.242060). This indicates that the Decision tree algorithm was the best performing model among the three classifiers before applying SMOTE. However, after applying SMOTE, the KNN algorithm had the lowest error rate (0.231281), followed by the Linear SVC algorithm (0.234608), and the Decision tree algorithm (0.244100). This indicates that the KNN algorithm was the best performing model among the three classifiers after applying SMOTE. This study demonstrates the effectiveness of machine learning algorithms in detecting intrusions, particularly in addressing the issue of imbalanced datasets using SMOTE. The results suggest that the KNN algorithm is the most effective in terms of reducing error rates after applying SMOTE, followed by the Linear SVC algorithm. The findings of this study may have implications for developing more accurate intrusion detection systems in the future.

Introduction

Intrusion is a pervasive and growing problem in the digital age. The rise of the internet and the proliferation of interconnected devices have created new opportunities for malicious actors to exploit vulnerabilities in computer systems and networks. The increasing reliance on technology for critical functions, such as banking, healthcare, transportation, and communication, has made intrusion a significant threat to public safety and national security.

Intrusion is a critical security issue that affects individuals, organizations, and society as a whole. It refers to the unauthorized access, manipulation, or destruction of computer systems, networks, and data by malicious actors. Intrusion can occur through various means, such as

hacking, malware, phishing, social engineering, and physical theft. Its consequences can be severe, ranging from financial loss and reputational damage to legal liabilities and compromised privacy or security.

One of the most common forms of intrusion is hacking, which refers to the use of technical skills and tools to gain unauthorized access to computer systems and networks. Hackers can exploit weaknesses in software, hardware, and network configurations to bypass security controls and infiltrate systems. Once inside, they can steal sensitive information, install malware, alter or delete data, or use the system as a platform for further attacks.

Malware is another prevalent form of intrusion that involves the use of malicious software to compromise computer systems and networks. Malware can take various forms, such as viruses, worms, trojans, spyware, and ransomware. It can be spread through email attachments, malicious websites, infected software, or social engineering. Once installed, malware can perform various malicious activities, such as stealing passwords, recording keystrokes, capturing screenshots, or encrypting files.

Phishing is a form of social engineering that involves the use of deception to trick individuals into divulging sensitive information, such as passwords, credit card numbers, or personal data. Phishing attacks can take many forms, such as fake emails, websites, or phone calls that mimic legitimate sources. The goal of phishing is to obtain sensitive information that can be used for identity theft, financial fraud, or other malicious purposes.

Social engineering is a broader term that encompasses various techniques used to manipulate people into performing actions that benefit the attacker. Social engineering can take many forms, such as pretexting, baiting, quid pro quo, or tailgating. Pretexting involves creating a false identity or story to gain trust and access to sensitive information. Baiting involves offering something of value, such as a free gift or service, to entice people to perform an action, such as clicking on a link or downloading a file. Quid pro quo involves offering a benefit in exchange for a service, such as providing a software update in exchange for a password. Tailgating involves following someone into a secure area without authorization.

Physical theft is a form of intrusion that involves stealing computer systems, devices, or data physically. Physical theft can occur in various settings, such as homes, offices, vehicles, or public places. Physical theft can be carried out by individuals or organized groups, and it can result in the loss of sensitive data, intellectual property, or critical infrastructure.

Intrusion can have severe consequences for individuals, organizations, and society as a whole. For individuals, intrusion can result in identity theft, financial fraud, or loss of privacy. For organizations, intrusion can result in reputational damage, financial loss, legal liabilities, or disrupted operations. For society as a whole, intrusion can result in compromised national security, public safety, or economic stability.

Preventing and detecting intrusion requires a combination of technical, procedural, and human factors. Technical measures include implementing strong passwords, firewalls, antivirus software, encryption, monitoring tools, and patches. Procedural measures include establishing security policies, procedures, and guidelines; conducting risk assessments, audits, and penetration testing; and performing incident response and disaster recovery. Human factors include raising awareness and training employees, customers, and stakeholders about the risks and best practices of cybersecurity, as well as promoting a culture of security that prioritizes vigilance, accountability, and continuous improvement.

One of the most effective ways to prevent and detect intrusion is to adopt a defense-in-depth approach that combines multiple layers of security controls. This approach recognizes that no single control can provide complete protection against intrusion and that a combination of controls can mitigate various types of threats. Defense-in-depth can include network segmentation, access control, authentication and authorization, encryption, intrusion detection and prevention, and incident response.

Network segmentation involves dividing a network into smaller subnetworks that are isolated from each other, reducing the attack surface and limiting the scope of intrusion. Access control involves restricting access to computer systems, networks, and data based on the principle of least privilege, which grants users only the minimum permissions needed to perform their job duties. Authentication and authorization involve verifying the identity of users and determining their level of access to resources based on their role and credentials. Encryption involves converting plaintext into ciphertext using an algorithm and a key, making it unreadable to unauthorized parties. Intrusion detection and prevention involve monitoring network traffic and system logs for signs of suspicious activity and blocking or alerting on such activity. Incident response involves preparing and executing a plan to contain and mitigate the effects of intrusion, such as isolating compromised systems, collecting evidence, and reporting to authorities.

In addition to technical measures, procedural measures, and human factors, preventing and detecting intrusion also requires collaboration and information sharing among stakeholders. Cybersecurity threats are not limited by geographic, sectoral, or national boundaries, and they require a collective response that involves government agencies, industry associations, academia, and civil society. Information sharing can enable stakeholders to share intelligence, best practices, and resources, as well as to coordinate response efforts and enhance situational awareness.

The challenge of intrusion is not only technical but also ethical, legal, and societal. Intrusion raises complex questions about privacy, security, accountability, and human rights, as well as about the balance between innovation and regulation. The proliferation of surveillance technologies, the monetization of personal data, and the emergence of new threats such as deepfakes, disinformation, and cyberwarfare pose fundamental challenges to democratic values and institutions. Therefore, preventing and detecting intrusion requires not only technical expertise but also ethical leadership, legal frameworks, and societal dialogue that foster trust, transparency, and resilience.

Imbalanced datasets in intrusion detection

Imbalanced datasets are a common issue in machine learning, particularly in supervised learning tasks. An imbalanced dataset is one where the distribution of target classes is uneven, with one class having significantly more or less examples than the other(s). This can lead to biased models, as the classifier will be more likely to predict the majority class, ignoring the minority class. This can be particularly problematic when the minority class is the one of interest, such as in fraud detection, where only a small percentage of transactions are fraudulent.

One potential solution to imbalanced datasets is to use resampling techniques, such as oversampling the minority class or undersampling the majority class. Oversampling involves replicating instances of the minority class, while undersampling involves randomly removing instances of the majority class. However, both methods can lead to overfitting, as the model may become too biased towards the minority class or may miss important information from the

majority class. Careful consideration is needed when choosing a resampling technique, and often a combination of both oversampling and undersampling can be used.

Another solution to imbalanced datasets is to use different evaluation metrics than the standard accuracy, such as precision, recall, and F1 score. These metrics focus on the performance of the model on the minority class, providing a better indication of how well the model is doing overall. For example, recall measures the percentage of positive instances that are correctly identified by the model, while precision measures the percentage of predicted positive instances that are actually positive.

Imbalanced datasets can also be addressed through algorithmic modifications. For example, in decision tree algorithms, weights can be assigned to different classes to give more importance to the minority class. In support vector machines, the cost function can be adjusted to account for class imbalance. However, these modifications may require additional computational resources and may not always improve performance.

Methods

Dataset

This study used NSL-KDD dataset. The NSL-KDD dataset is a benchmark dataset for network intrusion detection systems. It is an improved version of the original KDD Cup 99 dataset, which has several limitations such as a limited number of attacks and a high degree of redundancy. The NSL-KDD dataset addresses these limitations by removing duplicate records and adding new types of attacks, resulting in a more comprehensive and challenging dataset. The dataset contains network traffic data collected from a simulated environment, with five different types of attacks and normal traffic. The dataset has been widely used for evaluating the performance of various intrusion detection techniques and algorithms.

The NSL-KDD dataset has become a popular choice for researchers in the field of network security due to its realistic nature and comprehensive coverage of attacks. The dataset has been used for a variety of research tasks such as classification, clustering, and anomaly detection. Researchers have also proposed various machine learning algorithms and techniques for intrusion detection using this dataset, including decision trees, support vector machines, neural networks, and ensemble methods. The NSL-KDD dataset has contributed significantly to the development of intrusion detection systems and remains a valuable resource for researchers in the field.

Classes

Table 1 presents the five classes in the NSL-KDD dataset along with their descriptions. The first class is "Benign", which represents normal network traffic that is not malicious. This class is used as a baseline for comparison with the other classes. The second class is "DoS", which refers to Denial of Service attacks that try to overwhelm a target system with traffic. The third class is "Probe", which consists of reconnaissance or probing attacks that gather information about a target system for future exploitation. The fourth class is "R2L", which stands for Remote-to-Local attacks that exploit vulnerabilities in remote services to gain unauthorized access to a target system. The final class is "U2R", which represents User-to-Root attacks that attempt to gain superuser privileges on a target system. These classes provide a comprehensive and diverse set of attack types for evaluating the performance of intrusion detection systems and other security techniques.

Table 1.	
Class	Description
Benign	Normal network traffic that does not exhibit any signs of intrusion or attack. Used as a baseline for comparison.
DoS	Denial of Service attacks that attempt to overwhelm a target system with a flood of traffic, making it unresponsive.
Probe	Reconnaissance or probing attacks that gather information about a target network or system for future exploitation.
R2L	Remote-to-Local attacks that exploit vulnerabilities in remote services to gain unauthorized access to a target system.
U2R	User-to-Root attacks that attempt to gain superuser privileges on a target system.

SMOTE

The Synthetic Minority Over-sampling Technique (SMOTE) is a powerful method for handling class imbalance problems in machine learning. In many real-world scenarios, the data is highly skewed towards one class, resulting in models that perform poorly on the minority class. SMOTE is a data augmentation technique that helps to address this issue by generating synthetic samples of the minority class, thus increasing the representation of that class in the dataset. The technique works by selecting a random sample from the minority class and generating new samples by interpolating between the chosen sample and its k-nearest neighbors in the feature space. By increasing the number of samples in the minority class, SMOTE helps to balance the class distribution, which leads to better performance of classifiers and other machine learning algorithms.

One of the main advantages of SMOTE is that it is a simple and effective method that can be easily implemented in most machine learning frameworks. It does not require any additional data collection or labeling efforts, making it a cost-effective solution for class imbalance problems. Moreover, SMOTE is a non-parametric technique that does not make any assumptions about the underlying data distribution, which makes it robust to a wide range of data types and problem domains. However, it is worth noting that SMOTE may not always work well for highly complex datasets, where the underlying data distribution is not well-defined, or for problems where the minority class is inherently difficult to model.

In addition to SMOTE, there are two other popular techniques for handling class imbalance problems in machine learning: over-sampling and under-sampling. Over-sampling involves increasing the number of samples in the minority class by generating new samples through techniques such as SMOTE or simply duplicating existing samples. This method helps to balance the class distribution, but it can lead to overfitting, especially when the new samples are too similar to the original ones. Over-sampling is most effective when the dataset is small, and the minority class has a low variance. Under-sampling, on the other hand, involves reducing the number of samples in the majority class by randomly selecting a subset of the data. This method helps to balance the class distribution, but it may result in the loss of important information, especially when the majority class has a high variance. Under-sampling is most effective when the dataset is large, and the minority class has a high variance.

SMOTE is a popular over-sampling technique because it generates synthetic samples that are similar but not identical to the original ones, reducing the risk of overfitting. However, it may

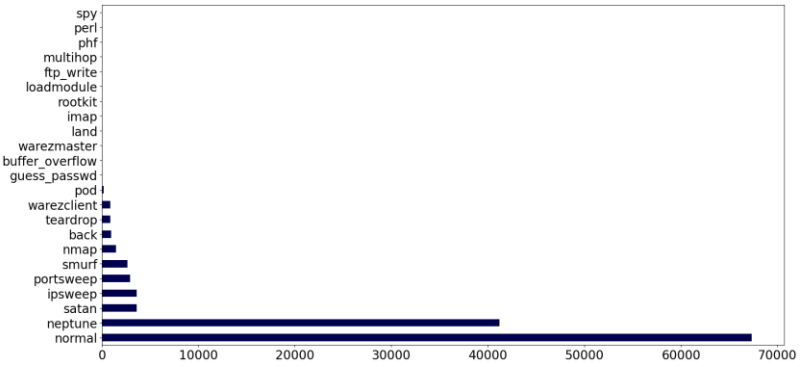
not always work well for highly complex datasets. Under-sampling can also be effective in certain situations, but it should be used with caution to avoid the loss of important information. Ultimately, a combination of over-sampling, under-sampling, and SMOTE may be the best solution to address class imbalance problems in machine learning.

Results

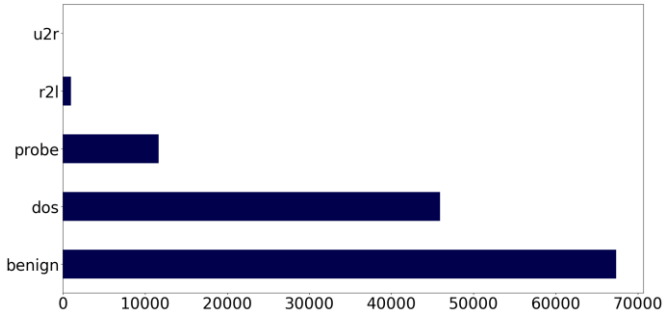
The task at hand is to create a classifier that can accurately categorize individual samples into one of five classes: benign, dos, r2l, u2r, or probe. In order to train this classifier, a dataset has been provided that contains samples which have been labeled with specific attacks. It is important to note that these attacks correspond to certain categories: ftp_write and guess_passwd attacks are classified as r2l (remote-to-local) attacks, while smurf and udpstorm are classified as dos (denial-of-service) attacks. The u2r (user-to-root) category refers to attacks where a regular user attempts to gain root access, while the probe category refers to attempts to gather information about a network or system. Therefore, in order to train the classifier, the labeled samples can be used as examples of the different attack types. By analyzing the features of these samples, the classifier can learn to identify patterns and characteristics that are associated with each type of attack. Once trained, the classifier can then be used to categorize new, unlabeled samples based on these learned patterns and characteristics.

Figure 1. Distributions of attack type

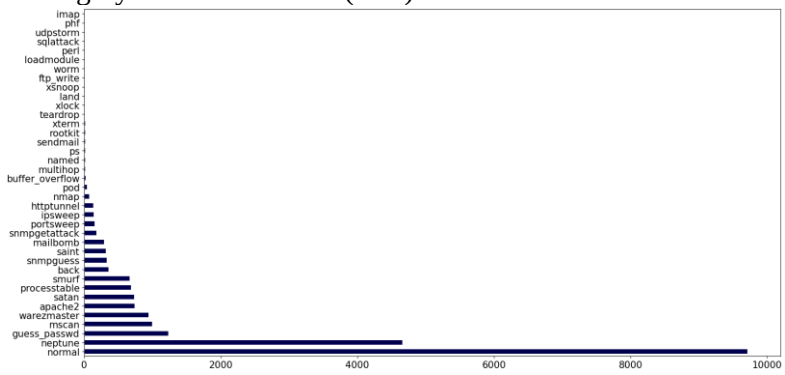
22-category class distribution (Train)



5-category class distribution (Train)



22-category class distribution (Test)



5-category class distribution (Test)

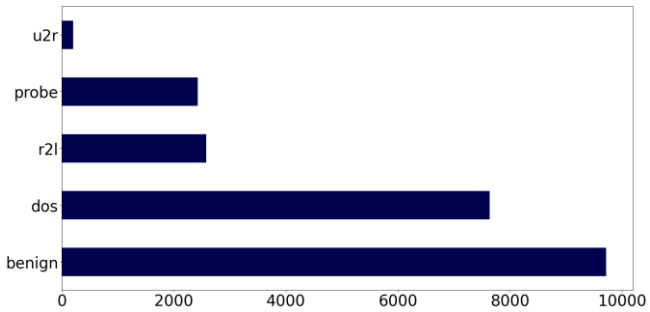
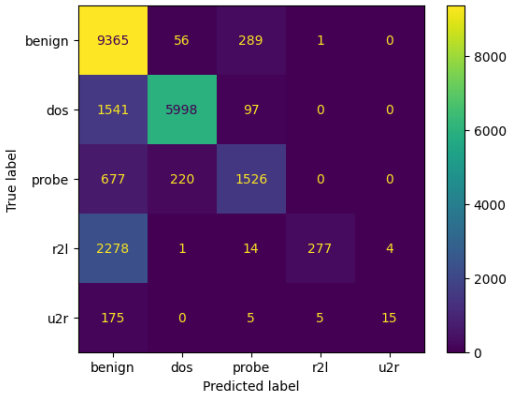


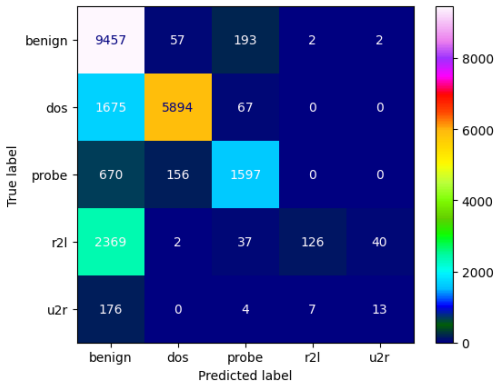
Figure 2. Confusion metrices and error rates before applying SMOTE

Decision tree



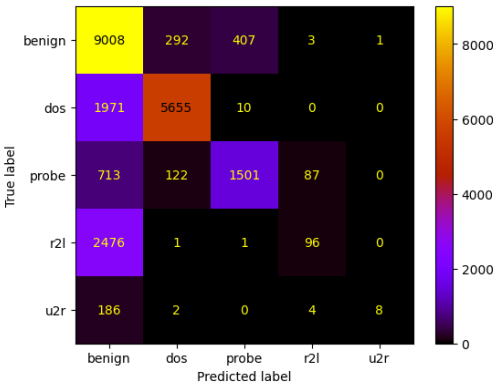
0.2378903477643719

KNN



0.24205997161107173

Linear SVC



0.27838892831795603

Table 2 provides information on the count and percentage of different types of activities detected by some system or tool. The types of activities are categorized as Benign, DOS, Probe, R2L, and U2R. The "Count" column indicates the number of activities that were classified as belonging to each type. For example, there were 67,343 activities classified as Benign, 45,927 activities classified as DOS, 11,656 activities classified as Probe, 995 activities classified as R2L, and 52 activities classified as U2R. The "Percentage" column shows the proportion of each type of activity out of the total number of activities observed. For example, 53.46% of the activities were classified as Benign, 36.46% as DOS, 9.25% as Probe, 0.79% as R2L, and 0.04% as U2R.

Table 2. Class counts

Type	**Count**	**Percentage**
Benign	67,343	53.46%
DOS	45,927	36.46%
Probe	11,656	9.25%
R2L	995	0.79%
U2R	52	0.04%

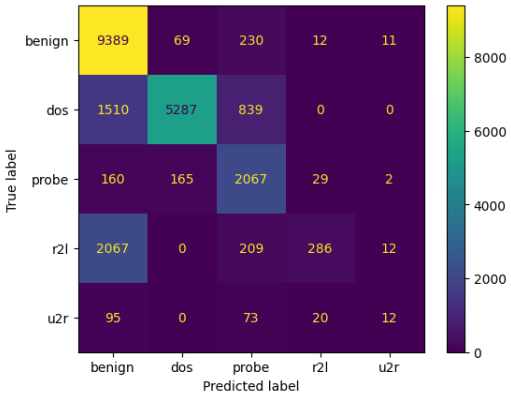
Table 3. SMOTE over and under sampler

	Benign	DOS	R2L	Probe	U2R
Over-sampler	67,343	67,343	67,343	67,343	67,343
Under-sampler	25,194	25,194	25,194	25,194	25,194

Table 3 provides information on the number of samples generated for each type of activity using two different sampling methods: Over-sampler and Under-sampler. The types of activities are categorized as Benign, DOS, R2L, Probe, and U2R. The "Over-sampler" column indicates the number of samples generated for each type of activity using the Over-sampler method. The number of samples generated is the same for all types of activities, with 67,343 samples generated for each activity type. This means that the Over-sampler method generates new samples for each activity type until the number of samples in each category is equal to the number of samples in the most frequent category, which in this case is Benign. The "Under-sampler" column shows the number of samples generated for each type of activity using the Under-sampler method. The number of samples generated is the same for all types of activities, with 25,194 samples generated for each activity type. This means that the Under-sampler method randomly selects samples from each activity type until the number of samples in each category is equal to the number of samples in the least frequent category, which in this case is U2R.

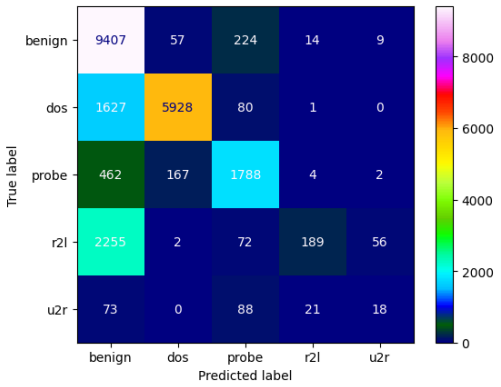
Figure 2. Confusion metrices and error rates after applying SMOTE

Decision tree



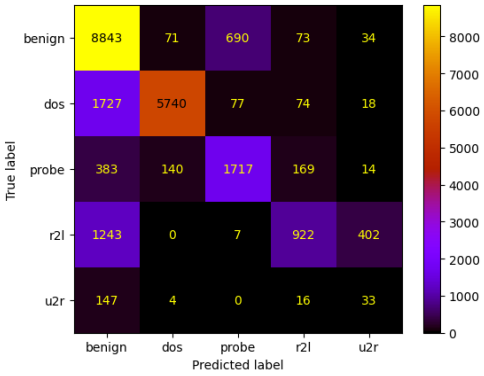
0.24410042583392477

KNN



0.23128105039034774

Linear SVC



0.2346078779276083

The table shows the error rates of three different machine learning algorithms: Decision tree, K-Nearest Neighbor (KNN), and Linear Support Vector Classification (Linear SVC). The error rates are measured before and after applying the SMOTE (Synthetic Minority Over-sampling Technique) algorithm. Before applying SMOTE, the Decision tree had an error rate of 0.237890, the KNN algorithm had an error rate of 0.242060, and the Linear SVC algorithm had an error rate of 0.278389. After applying SMOTE, the Decision tree algorithm had an increased error rate of 0.244100, the KNN algorithm had a decreased error rate of 0.231281, and the Linear SVC algorithm had a decreased error rate of 0.234608. The results suggest that the performance of the algorithms varied with the SMOTE algorithm, with some algorithms showing an improvement in performance, while others showed a decrease. It is worth noting

that the Decision tree algorithm had a slight increase in error rate after applying SMOTE, indicating that the SMOTE algorithm may not always improve performance for all machine learning models.

The error rates of three machine learning classifiers (Decision tree, KNN, and Linear SVC) were measured before and after applying the SMOTE algorithm. Before applying SMOTE, the Linear SVC algorithm had the highest error rate (0.278389), followed by the Decision tree algorithm (0.237890), and KNN algorithm (0.242060). This indicates that the Decision tree algorithm was the best performing model among the three classifiers before applying SMOTE. After applying SMOTE, the KNN algorithm had the lowest error rate (0.231281), followed by the Linear SVC algorithm (0.234608), and the Decision tree algorithm (0.244100). This indicates that the KNN algorithm was the best performing model among the three classifiers after applying SMOTE.

It appears that the KNN algorithm was the most effective in terms of reducing error rates after applying SMOTE, followed by the Linear SVC algorithm.

Conclusion

Machine learning has revolutionized the field of cybersecurity, especially in the area of network intrusion detection. Intrusion detection refers to the process of monitoring and analyzing network traffic to identify and respond to suspicious or malicious activities. Machine learning is a subfield of artificial intelligence that enables computers to learn from data and make predictions or decisions without being explicitly programmed. By combining machine learning algorithms with network traffic analysis, intrusion detection systems can achieve high accuracy, scalability, and efficiency, while reducing false positives and false negatives.

The traditional approach to intrusion detection involves using rule-based or signature-based methods, where predefined rules or signatures are used to detect known attack patterns. However, this approach has limitations, as it cannot detect new or unknown attacks, and it can generate a large number of false alarms if the rules or signatures are not precise or up-to-date. Moreover, attackers can evade signature-based detection by using polymorphic or obfuscated malware, or by exploiting zero-day vulnerabilities that have not been identified or patched.

In contrast, machine learning-based intrusion detection systems can learn from historical network traffic data and adapt to new or unknown attack patterns, without relying on predefined rules or signatures. Machine learning can identify subtle and complex patterns that may not be detectable by human analysts or traditional methods, and can distinguish between benign and malicious activities based on statistical and behavioral analysis. Machine learning can also reduce the time and effort required to update rules or signatures, as it can automatically learn from new data and adjust its model accordingly.

There are several types of machine learning algorithms that can be used for network intrusion detection, such as supervised learning, unsupervised learning, and reinforcement learning. Supervised learning involves training a model on labeled data, where each data point is associated with a class label that indicates whether it is benign or malicious. The model learns to classify new data based on its similarity to the training data, using techniques such as decision trees, support vector machines, logistic regression, and deep learning. Supervised learning can achieve high accuracy if the training data is representative and diverse, and if the features used to describe the data are relevant and informative.

Unsupervised learning involves training a model on unlabeled data, where the model learns to identify patterns and anomalies in the data without prior knowledge of the classes or labels. Unsupervised learning can be used for anomaly detection, where the model identifies data points that deviate from the normal behavior of the network. Unsupervised learning can use techniques such as clustering, principal component analysis, and autoencoders. Unsupervised learning can be useful for detecting new or unknown attacks, as it does not rely on predefined attack patterns, but it can generate false positives if the normal behavior of the network is not well-defined or if the anomalies are rare or subtle.

Reinforcement learning involves training a model to make decisions and take actions in an environment, based on feedback and rewards. Reinforcement learning can be used for adaptive intrusion detection, where the model learns to optimize its actions based on the feedback it receives from the network or the system. Reinforcement learning can use techniques such as Q-learning, deep reinforcement learning, and evolutionary algorithms. Reinforcement learning can be useful for dealing with dynamic and complex environments, where the optimal actions may depend on multiple factors and goals.

The performance of machine learning-based intrusion detection systems depends on several factors, such as the quality and quantity of the training data, the choice of the machine learning algorithm, the selection of the features, the optimization of the hyperparameters, and the evaluation of the model. The training data should be diverse, representative, and balanced, to avoid bias and overfitting. The machine learning algorithm should be appropriate for the task, considering its complexity, scalability, and interpretability. The features should be relevant and informative, capturing the important aspects of the network traffic, such as packet headers, payload contents, timing, and sequence. The hyperparameters should be tuned to optimize the performance of the model, such as the learning rate, the regularization, the activation functions, and the number of layers or nodes in the neural network. The evaluation of the model should be done using appropriate metrics, such as accuracy, precision, recall, F1-score, ROC curve, and AUC.

One of the challenges of using machine learning for network intrusion detection is the lack of labeled data, especially for new or unknown attack types. Collecting and labeling data is time-consuming and expensive, and may not cover all possible scenarios or variations. Moreover, attackers can generate synthetic or adversarial samples to evade detection, by modifying or injecting data that resembles normal traffic. To overcome this challenge, researchers have proposed several techniques for data augmentation, such as generative adversarial networks, transfer learning, and active learning. Data augmentation can increase the diversity and quantity of the training data, and can improve the robustness and generalization of the model.

Another challenge of using machine learning for network intrusion detection is the trade-off between accuracy and latency. Machine learning models can achieve high accuracy, but may require significant computational resources and time to process large amounts of data in real-time. Moreover, machine learning models may introduce overhead and complexity to the network, by requiring additional hardware, software, or infrastructure. To address this challenge, researchers have proposed several techniques for optimizing the performance of machine learning models, such as model compression, model pruning, and hardware acceleration. Model compression can reduce the size and complexity of the model, by removing redundant or unimportant parameters or layers. Model pruning can reduce the number of parameters or connections in the model, by identifying and removing the least important ones.

Hardware acceleration can speed up the computations required by the model, by using specialized hardware or architectures, such as GPUs, TPUs, or FPGAs.

Machine learning-based intrusion detection systems have been applied to various types of networks, such as wired networks, wireless networks, cloud networks, and IoT networks. Each type of network has its own characteristics and challenges, such as bandwidth limitations, mobility, heterogeneity, and scalability. Machine learning-based intrusion detection systems can adapt to these challenges by using appropriate techniques and models that suit the specific requirements of each network.

Machine learning has emerged as a promising approach for network intrusion detection, by enabling computers to learn from data and detect complex and evolving attack patterns. Machine learning-based intrusion detection systems can achieve high accuracy, scalability, and efficiency, while reducing false positives and false negatives. Machine learning-based intrusion detection systems can adapt to new or unknown attacks, and can handle various types of networks. However, machine learning-based intrusion detection systems also face challenges, such as the lack of labeled data, the trade-off between accuracy and latency, and the complexity of the models. By addressing these challenges and improving the performance and robustness of machine learning-based intrusion detection systems, we can enhance the security and resilience of our networks and systems, and mitigate the risks and impacts of cyber attacks.

References

[1] H. Debar, M. Dacier, and A. Wespi, “Towards a taxonomy of intrusion-detection systems,” *Computer Networks*, vol. 31, no. 8, pp. 805–822, Apr. 1999.
[2] G. Vigna and R. A. Kemmerer, “NetSTAT: A network-based intrusion detection system,” *J. Comput. Secur.*, vol. 7, no. 1, pp. 37–71, Jan. 1999.
[3] J. Allen, A. Christie, W. Fithen, J. McHugh, and J. Pickel, “State of the practice of intrusion detection technologies,” Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst, 2000.
[4] S. Northcutt and J. Novak, “Network intrusion detection,” 2002.
[5] G. Vigna and R. A. Kemmerer, “NetSTAT: a network-based intrusion detection approach,” in *Proceedings 14th Annual Computer Security Applications Conference (Cat. No.98EX217)*, 1998, pp. 25–34.
[6] W. Lee and S. J. Stolfo, “Data mining approaches for intrusion detection,” 1998. [Online].
[7] T. F. Lunt, “A survey of intrusion detection techniques,” *Comput. Secur.*, vol. 12, no. 4, pp. 405–418, Jun. 1993.
[8] T. Verwoerd and R. Hunt, “Intrusion detection techniques and approaches,” *Comput. Commun.*, vol. 25, no. 15, pp. 1356–1365, Sep. 2002.
[9] A. Lazarevic, V. Kumar, and J. Srivastava, “Intrusion Detection: A Survey,” in *Managing Cyber Threats: Issues, Approaches, and Challenges*, V. Kumar, J. Srivastava, and A. Lazarevic, Eds. Boston, MA: Springer US, 2005, pp. 19–78.
[10] J. McHugh, “Intrusion and intrusion detection,” *Int. J. Inf. Secur.*, vol. 1, no. 1, pp. 14–35, Aug. 2001.
[11] R. A. Kemmerer and G. Vigna, “Intrusion detection: a brief history and overview,” *Computer* , vol. 35, no. 4, pp. supl27–supl30, Apr. 2002.
[12] S. E. Smaha and Others, “Haystack: An intrusion detection system,” in *Fourth Aerospace Computer Security Applications Conference*, 1988, vol. 44, p. 37.
[13] A. Sundaram, “An introduction to intrusion detection,” *Crossroads*, vol. 2, no. 4, pp. 3–7, Apr. 1996.

[14] R. Bace and P. Mell, “Intrusion Detection Systems,” 2001. [Online].
[15] B. Mukherjee, L. T. Heberlein, and K. N. Levitt, “Network intrusion detection,” *IEEE Netw.*, vol. 8, no. 3, pp. 26–41, May 1994.
[16] S. Axelsson, “Intrusion detection systems: A survey and taxonomy,” 2000.
[17] C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel, and M. Rajarajan, “A survey of intrusion detection techniques in Cloud,” *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 42–57, Jan. 2013.
[18] H.-J. Liao, C.-H. Richard Lin, Y.-C. Lin, and K.-Y. Tung, “Intrusion detection system: A comprehensive review,” *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, Jan. 2013.
[19] R. G. Bace, *Intrusion Detection*. Indianapolis, IN: Sams Publishing, 1999.
[20] D. E. Denning, “An Intrusion-Detection Model,” *IEEE Trans. Software Eng.*, vol. SE-13, no. 2, pp. 222–232, Feb. 1987.
[21] B. Ingre, A. Yadav, and A. K. Soni, “Decision Tree Based Intrusion Detection System for NSL-KDD Dataset,” in *Information and Communication Technology for Intelligent Systems (ICTIS 2017) - Volume 2*, 2018, pp. 207–218.
[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] 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.
[24] L. Dhanabal and S. P. Shantharajah, “A study on NSL-KDD dataset for intrusion detection system based on classification algorithms,” *International journal of advanced*. faratarjome.ir, 2015.
[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. Nakamura, Y. Kajiwara, A. Otsuka, and H. Kimura, “LVQ-SMOTE - Learning Vector Quantization based Synthetic Minority Over-sampling Technique for biomedical data,” *BioData Min.*, vol. 6, no. 1, p. 16, Oct. 2013.
[27] L. Torgo, R. P. Ribeiro, B. Pfahringer, and P. Branco, “SMOTE for Regression,” in *Progress in Artificial Intelligence*, 2013, pp. 378–389.
[28] J. Sun, J. Lang, H. Fujita, and H. Li, “Imbalanced enterprise credit evaluation with DTE-SBD: Decision tree ensemble based on SMOTE and bagging with differentiated sampling rates,” *Inf. Sci.* , vol. 425, pp. 76–91, Jan. 2018.
[29] S. Mishra, “Handling imbalanced data: SMOTE vs. random undersampling,” *Int. Res. J. Eng. Technol*, vol. 4, no. 8, pp. 317–320, 2017.
[30] J. A. Sáez, J. Luengo, J. Stefanowski, and F. Herrera, “SMOTE–IPF: Addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering,” *Inf. Sci.* , vol. 291, pp. 184–203, Jan. 2015.
[31] Y. Dong and X. Wang, “A New Over-Sampling Approach: Random-SMOTE for Learning from Imbalanced Data Sets,” in *Knowledge Science, Engineering and Management*, 2011, pp. 343–352.
[32] M. R. Prusty, T. Jayanthi, and K. Velusamy, “Weighted-SMOTE: A modification to SMOTE for event classification in sodium cooled fast reactors,” *Prog. Nuclear Energy*, vol. 100, pp. 355–364, Sep. 2017.
[33] H. Han, W.-Y. Wang, and B.-H. Mao, “Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning,” in *Advances in Intelligent Computing*, 2005, pp. 878–887.

[34] C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap, "Safe-Level-SMOTE: Safe-Level-Synthetic Minority Over-Sampling TEchnique for Handling the Class Imbalanced Problem," in *Advances in Knowledge Discovery and Data Mining*, 2009, pp. 475–482.
[35] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *jair*, vol. 16, pp. 321–357, Jun. 2002.