DATA 440 Final Report: Two-Step Intrusion Detection System

Daniel Han
Professor Vasiliu
Capstone in Data Apps
December 16th, 2024

I. Abstract

The rapid growth of internet usage and device connectivity over the past decade has brought unprecedented opportunities but also significant cybersecurity challenges. Cyber attacks are increasingly frequent and sophisticated, costing the global economy trillions of dollars annually. Intrusion Detection Systems (IDS) are a crucial tool for identifying and mitigating these attacks. This project presents a two-phase intrusion detection pipeline leveraging machine learning techniques: a One-Class Support Vector Machine (OVM) for anomaly detection and a Convolutional Neural Network (CNN) for attack classification. The NSL-KDD dataset was used as the basis for training and testing the models. The OVM step efficiently detects attacks with optimized hyperparameters from a GridSearchCV, achieving an F1-score of 89% on training data and 88% on the test set. Subsequently, the CNN categorizes flagged attacks into generalized classes, with an overall test accuracy of 99.83%. These two models are then combined into a consolidated workflow that strikes a balance between accuracy and computational efficiency, making it suitable for real-world, high-throughput IDS environments.

II. Introduction

Technology and by extension internet usage has been on a meteoric rise since the 2010's. Just in the past ten years, the world has seen internet usage increasing from 2.77 billion in 2014 to 5.52 billion today in 2024 (DataReprotal, n.d.). Growing alongside that, is the increase in usage of devices. Devices encompass anything from mobile phones to laptops, desktops, anything that is connected to the internet and can be interacted with by the user. 70% of the world, around 5.75 billion people, utilize a mobile device, and spend an average of 6 and a half hours per day connected to the internet (DataReprotal, n.d.). This statistic will only increase as the vast majority of smartphone users come from younger generations, and as time moves on this will only exacerbate the amount of phone users (DataReprotal, n.d.).

One of the most pressing consequences of this rise in internet usage is concern for criminal activity involving these devices. To understand how to defend against attacks, we must also understand what the thought process behind committing this crime is in the first place. There are four main components when considering the intentions behind a cyber attack, many of which are similar to the typical factors to look for in crime. These components include: method, how did the attack work; vector, how did the attack launch and move over time; motive, why was the attack carried out; target, who was attacked. The focus of this project is on the method of attack. There are a plethora of methods for attackers to achieve their goals, to understand some of these methods, The OWASP Top 10 serves as a vital guide for developers to be aware of the most critical cybersecurity vulnerabilities in web applications. The most critical of these methods include failure to properly manage and limit user permissions, improper encryption methods for sensitive information, and vulnerabilities to code injection in web applications. This is how an attack is started, motivated mainly by financial incentives, stealing user information or gaining control over network systems to leverage their strength for money. Cybercrime is projected to cost $10.5 trillion dollars to the world's economies by 2025, a lucrative trade for attackers everywhere (Cybersecurity Ventures, 2021). The prominent industry when this is seen is in healthcare, where data breaches are all too common and millions of dollars are spent in paying ransoms for confidential information.

Cybersecurity considers the total networked system security, from the computer, to the network, to the information itself. The purpose of cybersecurity is to protect three core tenets: confidentiality, which prevents unauthorized access; integrity, which prevents corruption; and availability, which ensures that services are accessible. These tenets are integrated into the development techniques such as trust boundaries and intrusion detection systems.

The intrusion detection system is what this project is focusing on, a security tool that continuously monitors a network for any suspicious activity and alerts a separate system such as a system administrator that a possible intrusion is occurring. The intrusion detection system (IDS) in this project leverages multiple machine learning techniques to distinguish between malicious and benign network activity and categorize malicious activity into generalized attack surfaces while maintaining a real-world practicality and performance-minded outlook.

III. Datasets

The dataset used in this project is the NSL-KDD dataset taken from kaggle. This dataset contains its own separated training and testing sets consisting of raw internet traffic data. As shown in figure 1, there are 43 feature variables detailing a

```
@relation 'KDDTest'
@attribute 'duration' real
@attribute 'protocol_type' {'tcp','udp', 'icmp'}
@attribute 'service' {'aol', 'auth', 'bgp', 'courier', 'csnet_ns', 'ctf', 'daytime', 'discard', 'dor
@attribute 'flag' { 'OTH', 'REJ', 'RSTO', 'RSTOS0', 'RSTR', 'S0', 'S1', 'S2', 'S3', 'SF', 'SH' }
@attribute 'src_bytes' real
@attribute 'dst_bytes' real
@attribute 'land' {'0', '1'}
@attribute 'wrong_fragment' real
@attribute 'urgent' real
@attribute 'hot' real
@attribute 'num_failed_logins' real
@attribute 'logged_in' {'0', '1'}
@attribute 'num_compromised' real
@attribute 'root_shell' real
@attribute 'su_attempted' real
@attribute 'num_root' real
@attribute 'num_file_creations' real
@attribute 'num_shells' real
@attribute 'num_access_files' real
@attribute 'num_outbound_cmds' real
@attribute 'is_host_login' {'0', '1'}
@attribute 'is_guest_login' {'0', '1'}
```

Fig. 1. A snapshot of some of the feature variables in the NSL-KDD dataset and their data types, note the mixture of continuous and categorical data types in this dataset.

multitude of metadata for internet traffic. There is categorical data such as 'service', representing the type of network traffic connection, and numerical data such as 'src_bytes', the total number of bytes sent from the source to the destination.

In order for this dataset to be usable for our purposes, we need to convert the categorical data into numerical data that we can use. So, we first create dummy variables for all the categories. The resulting data frame contains over 120 feature variables, which, in the interest of computational cost, is far too many variables to train our models on.

To reduce the number of variables we are working with, we used PCA analysis on the dataset, a dimensionality reduction and feature extraction algorithm that reduces the dimensions of a dataset while maintaining as much of the information as possible. While we want to reduce the feature space of the data, we also want to hold on to as much information as possible, so as to not reduce any accuracy when training the model. The explained variance ratio indicates how much of the variance is still captured by the data with reduced features, so we use a 99% explained variance ratio which results in 58 features as seen in figure 2.

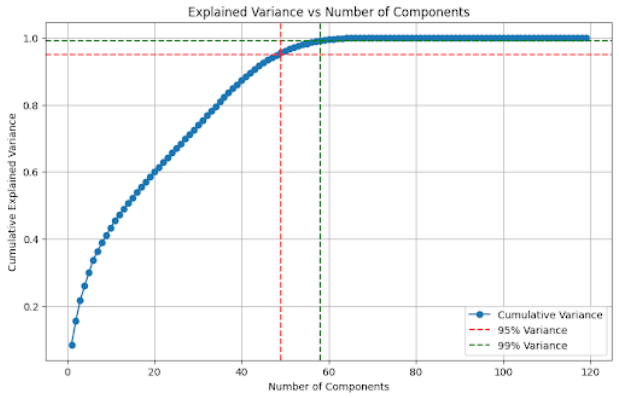


Fig. 2. Graph of the results of the PCA analysis. The optimal amount of feature variables to capture 99% of the variance was 58, and 49 for 95% of the variance.

IV. Methodology

A high-level overview of the organization of this project is seen in figure 3. The method can be generally split into two overarching phases, the training phase and the testing phase. Starting from where we left off with PCA analysis, we have a few more data modifications before we begin training the models.The variable we are trying to predict is called "class", which distinguishes between normal and anomalous traffic. The anomalous traffic is further split into specific attacks, such as buffer_overflow_, portsweep, and other well-known attacks. We reshape this data to just have a binary class, normal, and an umbrella term "attack" for any anomalous activity. Additionally, we scale the training data using StandardScaler, and we filter out only values that result in normal traffic. We

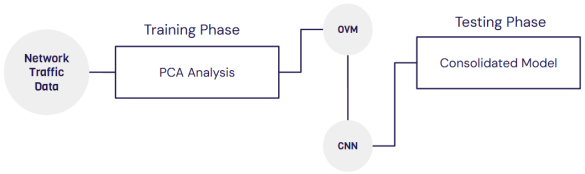


Fig. 3. A high level diagram of the roadmap of the project.

only use normal traffic because of the nature of the algorithm we are using, a One-Class Support Vector Machine (OVM). The OVM differs from a normal Support Vector Machine (SVM) in a few key respects. A One class svm's main purpose is to classify a norm and an anomaly, meaning, it treats every situation as a binary classification issue. As a result, it is only necessary for the OVM to have a single labeled class in the dataset. For our purposes, this is the "normal" class that we assigned to all traffic that is not an attack. This is especially useful for this purpose, because instead of creating the hyperplane to separate two classes, the OVM creates a hypersphere around the "norm", which finds the smallest enclosing perimeter around the specified points while allowing for errors. The mathematical basis behind this is,

$$\min_{\mathbf{w},\xi_i,\rho} \frac{1}{2}\|\mathbf{w}\|^2 + \nu \sum_{i=1}^{n} \xi_i - \rho \tag{1}$$

subject to:

$$(\mathbf{w} \cdot \phi(\mathbf{x}_i)) \geq \rho - \xi_i, \quad \xi_i \geq 0 \quad \forall i \tag{2}$$

The most significant aspect here is the hyperparameter $\nu \in (0, 1]$. This hyperparameter controls how constrained the perimeter of the hypersphere is, thus how sensitive the model is towards outliers. Once the model is trained, the decision rule simply determines that if $f(\mathbf{x}) \geq 0$, the point $\mathbf{x}_i$ is considered normal, and if $f(\mathbf{x}) \leq 0$, it is considered an anomaly (Schölkopf et al., 2001). We then conduct a GridSearchCV process to find the best hyperparameters for the OVM model, including ν, gamma, and the best kernel.

After the OVM, we continue into the next step of our pipeline, the CNN model. Usually, a convolutional neural

Aspects	SVM	One Class SVM
Data Usage	Requires a large, labeled dataset	Only utilizes the most common label, others can be unknown
Purpose	Highest accuracy in distinguish different classes	Highest accuracy in distinguishing outliers, regardless of labels
Tuning	Encompasses margin and classification error	"Nu" determines the sensitivity towards outliers

Fig. 4. Simplified chart comparing the differences in use and structure between normal SVMs and the One Class SVM.

network (CNN) model is used for image classification tasks. There are three general steps to the CNN model, the convolutional layer, pooling layer, and fully connected layer. Using images as an example, it treats each pixel in the image as a feature variable and relates it to other pixels. In the convolutional layer, the model applies a series of filters (small matrices) onto the input data, these filters then apply themselves over the data to produce a feature map of the pixels, highlighting only the most important aspects of the data. The next layer, the pooling layer, reduces the size of the feature map, making the computation more efficient and less prone to overfitting. And finally, the fully connected layer flattens out the relevant feature matrices into single dimension arrays and combines them to make a final prediction. Looking at this process, it seems using images is essential to the CNNs inherent architecture and processes. However, if we look at images and data with a different perspective, we can create a novel way of utilizing this model. Images are treated just as a matrix of information, and since we have a large amount of feature variables in our dataset, 58, we can reshape this data to emulate the structure of an image. So, to emulate an image, we pad each data entry in the training set with 6 columns of zeros, so that the length of each row is 64. We then reshape this data into 8x8 "images" for the CNN to train on. The end-goal purpose of the CNN is to generalize flagged network activity into generalized attack surfaces so that a separate administrative system can take the proper action to mitigate the attack. These generalized attack surfaces are Denial of Service (DoS), Probe, User to Root (U2R), and Remote to Local (R2L) and a precautionary misclassified category that ensures any normal traffic that leaks through the OVM is captured by the CNN model.

In our final step, we aim to combine both of these processes, the OVM and the CNN into one continuous work flow to detect

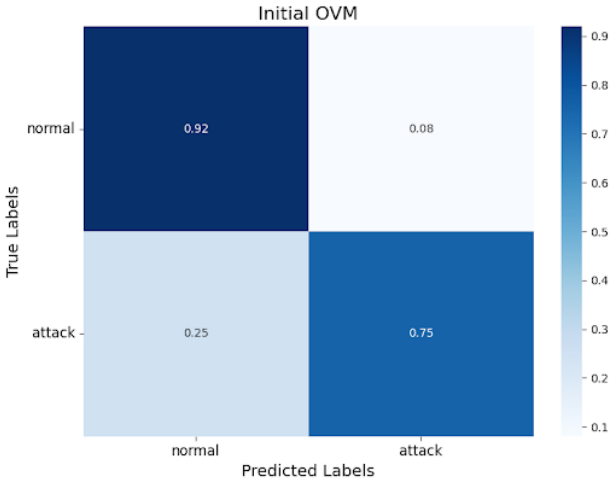
```
# Define the CNN model
cnn_model = Sequential([
    Conv2D(32, kernel_size=(3, 3), activation="relu", input_shape=(8, 8, 1)),
    MaxPooling2D(pool_size=(2, 2)),
    Dropout(0.25),
    Conv2D(64, kernel_size=(3, 3), activation="relu"),
    Flatten(),
    Dense(128, activation="relu"),
    Dropout(0.5),
    #5 categories
    Dense(5, activation="softmax")
])
```

Fig. 5. The specific CNN architecture used to categorize attacks into generalized attack surfaces, ends with 5 neurons at the end because there are 5 final categories.

and generalize network activity to the highest accuracy in our training set.

V. Results

In training our OVM, we decided to first compare results between pre and post PCA in order to observe and account for any information lost in the PCA process. In figure 6, we have the confusion matrix and classification report for the OVM using the dataset before PCA and using the default hyper-parameter settings for the OVM. The model is exceptional at capturing normal traffic, 92% of normal traffic identified correctly, however it is lacking in identification of attack traffic, correctly identifying only 75% of the data. Overall, the f1-score for the first trained model is an acceptable 83%.



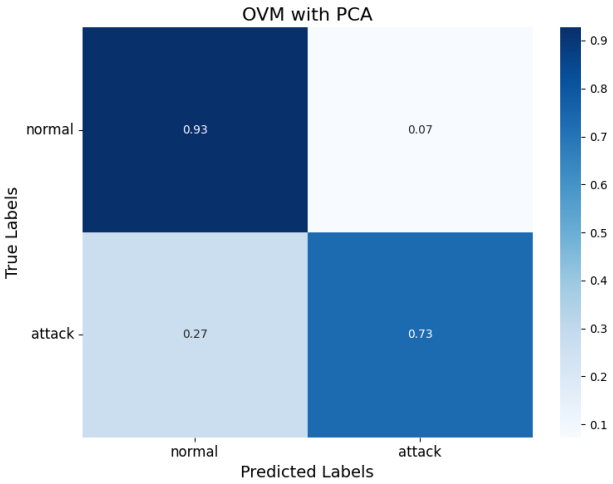
Classification Report:

	precision	recall	f1-score	support
Normal	0.93	0.75	0.83	12833
Attack	0.74	0.92	0.82	9711
accuracy			0.83	22544
macro avg	0.83	0.84	0.83	22544
weighted avg	0.85	0.83	0.83	22544

Fig. 6. Confusion matrix and classification report for the training phase of the One Class SVM before conducting PCA analysis on the dataset

In our second interaction of the OVM model, we trained the model on the PCA reduced dataset, going down from 119

variables to 58. Overall, accuracy scores are slightly lower than the model trained on the pre-PCA dataset. This however, is to be expected since reducing the feature space is going to lead to some information loss no matter what. The normal accuracy is slightly higher at 93%, while the attack accuracy decreases to 73%. The f1-score for this model is 82%, one point lower than the previous model.

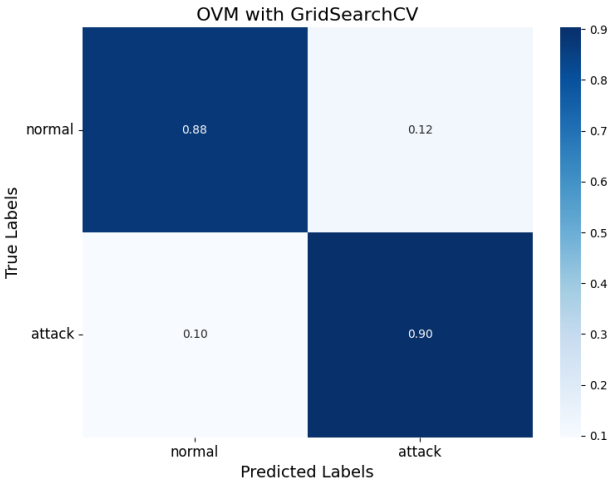


	precision	recall	f1-score	support
Normal	0.93	0.73	0.82	12833
Attack	0.72	0.93	0.81	9711
accuracy			0.82	22544
macro avg	0.83	0.83	0.82	22544
weighted avg	0.84	0.82	0.82	22544

Fig. 7. Confusion matrix and classification report for the training phase of the One Class SVM after conducting PCA analysis on the dataset, reducing the feature space from 119 variables to 58.

After conducting a GridSearchCV to find the optimal hyperparameters, we created a final model based on the PCA reduced dataset. The optimal hyperparameters identified were as follows: RBF for the kernel, auto for gamma, and 0.1 for nu. Here, the overall f1-score out performs the previous models, with a value of 89%. This model is also much more attuned to detecting attack surfaces, capturing 90% of the attack traffic. This however also comes at a cost, with a much lower accuracy for normal traffic at 88%, which means that the model predicts much more false positives than before.

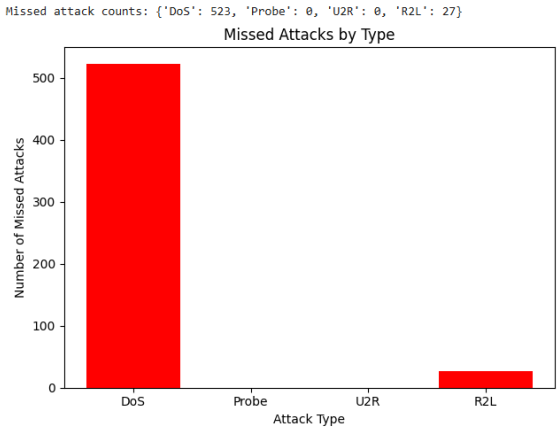
The CNN model was able to fully capture the complex relationships between variables, with an accuracy of 99.83% after 20 epochs on the training set. The CNN is much more accurate than the OVM at not only detecting normal and anomalous activity, but also generalizing them into specific attack surfaces. Once we consolidated both models into a continuous workflow, we evaluated the model's accuracy on the testing set. First we took a look at the specifics of the accuracy of the OVM and its binary classification. The OVM had a test f1-score of about 88%, slightly lower than on the training data. Looking at what types of attacks the OVM



	precision	recall	f1-score	support
Normal	0.91	0.90	0.91	12833
Attack	0.87	0.88	0.88	9711
accuracy			0.89	22544
macro avg	0.89	0.89	0.89	22544
weighted avg	0.89	0.89	0.89	22544

Fig. 8. Final confusion matrix and classification report for the training phase of the One Class SVM after conducting a GridSearchCV for the optimal hyperparameters.

missed, we can see that Denial of Service attacks (DoS) were the most frequently missed attacks by the OVM with 523 missed. Next are the R2L attacks at 27, and an extremely accurate 0 attacks missed for the probe and U2R attacks.

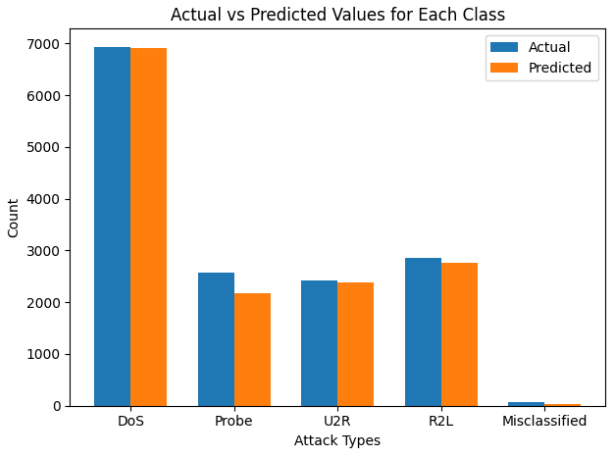


Accuracy: 0.8615
Precision: 0.8269
Recall: 0.9571
F1-score: 0.8872

Fig. 9. Bar chart and accuracy report for the OVM, the first step of the consolidated model, on the test dataset.

Evaluating the overall accuracy of the model, the CNN scores an f1-score of 88% on the test dataset. The raw

counts for actual and predicted attacks can also be seen, this visualization compares the raw count of attacks contained in the test set compared against the attacks that made it through the OVM and into the CNN and how they were classified. Note that the CNN did catch some of the misclassified datasets from the OVM, this represents data points labeled as attacks by the OVM but were further evaluated by the CNN to be normal traffic.




```
CNN Accuracy on attack samples: 0.9589
CNN Precision on attack samples: 0.9067
CNN Recall on attack samples: 0.8616
CNN F1-score on attack samples: 0.8795
```

Fig. 10. Bar chart and accuracy report for the consolidated model at the end of its evaluation on the testing set.

VI. Discussion

In the OVM step of our process, we see that the vast majority of missed attacks are Denial of Service attacks. This is because of the nature of the attack, DoS attacks are when an attacker uses a huge amount of bots to try and use an internet service simultaneously. Usually, when this happens, the service provider can't account for such a large surge in user activity. This overloads their servers and causes their application to have to shut down. The issue here lies in the fact that this sort of consequence can result from normal user activity as well. For example, limited time products can have a huge amount of users logging into the same application all at the same time. Concert tickets, clothing restocks, these are all perfectly normal events that happen that draw a huge amount of users. The bots used in denial of service attacks only have to log in to the website to achieve their goal, so there is not the same amount of traffic clues and patterns to identify these attacks. Instead, applications should use other measures such as throttling the amount of users logging in at once and other traffic jam methods to solve this particular problem.

The GridSearchCV results for the OVM contradict the previous results in that the accuracy for normal drops a significant amount. However, when considering the threat and danger of cyberattacks, it is preferable to have a higher number of false positives than false negatives. This approach prioritizes accidentally flagging normal users in the pursuit of attackers over missing the detection of actual attackers, even if it slightly disrupts normal traffic.

This is also the massive advantage of the OVM over normal SVMs, the nu hyperparameter can be tuned and changed for the specific circumstance of an intrusion detection system. For example, for a video streaming service, there is going to be wildly different types of traffic from all over the world and displaying different characteristics. So a looser nu parameter might be more advisable so that the system does not disrupt the experience of the streaming site for the user. There also might not be as much incentive to capture attackers due to the low amount of risk and potential loss compared to something like a healthcare system or government institution. These closed networks would tune their nu hyperparameter to be much more sensitive to outliers, prioritizing the detection of outliers over the preservation of the normal traffic. Here, it is necessary and preferable to take the extra precaution and disturbance of normal traffic for safety, due to the sensitive nature of their networks and the information they contain. So, this model can be applied to many different circumstances and be modified for specific use.

Looking at the raw results here, it is reasonable to conclude that this entire process is over-engineered, and that instead of using an ensemble of models for this problem, just a CNN model would have been sufficient. However, we aim to create a model that can practically be used in an intrusion detection system environment. When working with thousands or maybe tens of thousands of unique internet traffic data, performance and computational time is of the essence. Although the OVM to CNN model pipeline sacrifices some accuracy, the computational performance gained is much more significant. Instead of running each and every traffic point through a CNN model, a computationally costly endeavour, the model presented here can easily and quickly calculate whether a point lies outside of the hypersphere. Thus, immediate action can be taken, whether it be to lock the user out of access to the web application or to simply prompt a captcha, the specific type of attack does not need to be immediately identified. Further resources can be utilized after stemming the flagged activity, where then a system administrator can apply the appropriate response to the intrusion.

This project is open to many future steps, including a deeper analysis of the performance gain to accuracy loss we obtain from using the OVM and CNN combined model as opposed to just a CNN model. There could also be more research into how this would fit into a larger network protection system, including administrative systems that would have the authority to neutralize these attacks. And finally, feeding live data into the system and seeing how it holds up in a real-world context.

VII. Conclusion

This project demonstrates a practical, machine-learning-based approach to intrusion detection, combining a One-Class Support Vector Machine (OVM) with a Convolutional Neural

Network (CNN). The OVM efficiently identifies anomalous traffic, serving as a computationally lightweight pre-filter to flag potential attacks. The CNN then classifies these flagged instances into specific attack surfaces (DoS, Probe, U2R, R2L) with high accuracy. The combined OVM-CNN workflow is a realistic solution for high-traffic environments where efficiency is critical.

The results highlight that Denial of Service (DoS) attacks remain particularly challenging to detect due to their similarity to surges in legitimate user traffic. Addressing this limitation requires complementary mitigation strategies, such as rate-limiting or throttling mechanisms. Additionally, the tunability of the OVM's nu hyperparameter allows this model to be adapted for varying network environments—prioritizing either detection sensitivity or user experience depending on the application.

This work sets the foundation for future enhancements, including testing with live network traffic, deeper analysis of the accuracy-to-performance trade-off, and integrating this model into a comprehensive network defense system. By balancing computational efficiency and accuracy, this model contributes to building more effective and scalable intrusion detection systems capable of mitigating evolving cyber threats in real-world applications.

REFERENCES

[1] DataReportal. (n.d.). Global Digital Overview: Essential insights into how people around the world use the internet, mobile devices, social media, and e-commerce. Retrieved from https://datareportal.com/global-digital-overview

[2] Pew Research Center. (2019, March 7). Use of smartphones and social media is common across most emerging economies. Retrieved from https://www.pewresearch.org/internet/2019/03/07/use-of-smartphones-and-social-media-is-common-across-most-emerging-economies/

[3] Cybersecurity Ventures. (2021). Cybercrime to cost the world $6 trillion annually by 2021. Retrieved from https://cybersecurityventures.com/cybercrime-damages-6-trillion-by-2021/

[4] Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., & Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. Neural Computation, 13(7), 1443–1471. https://doi.org/10.1162/089976601750264965

[5] Kaggle. (n.d.). NSL-KDD Dataset for Network Intrusion Detection Systems (NIDS). Retrieved from https://www.kaggle.com/datasets/hassan06/nslkdd/data