Scientific Research Publishing

Journal of Information Security, 2024, 15, 545-556
https://www.scirp.org/journal/jis
ISSN Online: 2153-1242
ISSN Print: 2153-1234

Optimizing Network Security via Ensemble Learning: A Nexus with Intrusion Detection

Anu Baluguri[1], Vasudha Pasumarthy[1], Indranil Roy[2], Bidyut Gupta[3], Nick Rahimi[1]

[1]School of Computing Sciences and Computing Engineering, University of Southern Mississippi, Hattiesburg, USA
[2]Department of Computer Science, Southeast Missouri State University, Cape Girardeau, USA
[3]School of Computing, Southern Illinois University, Carbondale, USA
Email: Anu.Baluguri@usm.edu, Vasudha.Pasumarthy@usm.edu, iroy@semo.edu, bidyut@cs.siu.edu, Nick.Rahimi@usm.edu

How to cite this paper: Baluguri, A., Pasumarthy, V., Roy, I., Gupta, B. and Rahimi, N. (2024) Optimizing Network Security via Ensemble Learning: A Nexus with Intrusion Detection. *Journal of Information Security*, **15**, 545-556.
https://doi.org/10.4236/jis.2024.154030

Received: July 24, 2024
Accepted: October 19, 2024
Published: October 22, 2024

Abstract

Network intrusion detection systems need to be updated due to the rise in cyber threats. In order to improve detection accuracy, this research presents a strong strategy that makes use of a stacked ensemble method, which combines the advantages of several machine learning models. The ensemble is made up of various base models, such as Decision Trees, K-Nearest Neighbors (KNN), Multi-Layer Perceptrons (MLP), and Naive Bayes, each of which offers a distinct perspective on the properties of the data. The research adheres to a methodical workflow that begins with thorough data preprocessing to guarantee the accuracy and applicability of the data. In order to extract useful attributes from network traffic data—which are essential for efficient model training—feature engineering is used. The ensemble approach combines these models by training a Logistic Regression model meta-learner on base model predictions. In addition to increasing prediction accuracy, this tiered approach helps get around the drawbacks that come with using individual models. High accuracy, precision, and recall are shown in the model's evaluation of a network intrusion dataset, indicating the model's efficacy in identifying malicious activity. Cross-validation is used to make sure the models are reliable and well-generalized to new, untested data. In addition to advancing cybersecurity, the research establishes a foundation for the implementation of flexible and scalable intrusion detection systems. This hybrid, stacked ensemble model has a lot of potential for improving cyberattack prevention, lowering the likelihood of cyberattacks, and offering a scalable solution that can be adjusted to meet new threats and technological advancements.

Keywords

Machine Learning, Cyber-Security, Data Preprocessing, Model Training

1. Introduction

Network security is still a major concern for companies in a variety of industries in the digital age. The sophistication of cyber threats and network architectures are driving up the need for advanced intrusion detection systems (IDS), which are often inadequate. These systems need to change not just to keep up with new threats but also to handle the increasing volume and variety of network transactions. This study presents a sophisticated and robust framework for network intrusion detection that uses a stacked ensemble technique to improve detection robustness and accuracy by combining several machine learning models. In the digital age, network security continues to be a top concern for businesses across many industries. The need for sophisticated and robust intrusion detection systems (IDS), which are frequently insufficient, is increasing due to the complexity of cyber threats and network architectures. These systems must adapt to handle the growing volume and diversity of network transactions in addition to staying ahead of emerging threats. This research presents an advanced framework for network intrusion detection that combines multiple machine learning models with the stacked ensemble technique to improve detection robustness and accuracy. Multiple base models, such as a Multi-Layer Perceptron (MLP), K-Nearest Neighbors (KNN), Naive Bayes, and Decision Trees, make up the ensemble framework's core. Each model captures unique insights into the possible threats by processing the input data independently and contributing its predictions. The information is then effectively synthesized to produce a final determination of the nature of the traffic by using these predictions as inputs for a meta-model that is implemented using Logistic Regression. The methodology for implementing the stacked ensemble IDS is described in this paper, along with the steps involved in data preprocessing, feature engineering methods, and the reasoning behind the choice of base models [1]. It also covers the training procedure, which makes use of cross-validation to guard against overfitting and guarantee the generalizability of the model in various network scenarios. The effectiveness of the system in practical situations is reflected by the evaluation metrics used to evaluate the accuracy, precision, recall, and F1 score of the IDS. The stacked ensemble approach significantly enhances network security by integrating multiple models into an advanced infrastructure. This integration not only improves accuracy but also delivers a resilient solution capable of adapting to emerging threats. To safeguard against malicious intrusions, this research contributes to the continuous efforts in the field of cybersecurity by presenting a versatile and scalable solution that can be deployed across diverse IT environments.

2. Literature Review

An innovative method for reducing cyber threats against Internet of Things (IoT) networks is presented in the paper "An Ensemble Intrusion Detection Technique based on proposed Statistical Flow Features for Protecting Network Traffic of Internet of Things" by Nour Moustafa *et al.* It focuses on the DNS, HTTP, and

MQTT protocols. The study aims to improve the detection of malicious events, especially botnet attacks, by extracting statistical flow features from these protocols and using an AdaBoost ensemble learning method that combines Decision Trees, Naive Bayes, and Artificial Neural Networks. When compared to individual classification techniques and other cutting-edge methods, the suggested technique shows promise in terms of higher detection rates and fewer false positives. The paper offers some insights into how this could be expanded in the conclusion [2] [3].

The paper titled "Network Intrusion Detection using Supervised Machine Learning Technique with Feature Selection" by Kazi Abu Taher *et al.* presents a novel approach to classifying network traffic as either malicious or benign using supervised machine learning methods. The study investigates the effectiveness of different machine learning algorithms and feature selection techniques in building an intrusion detection system. Specifically, the authors compare the performance of Support Vector Machine (SVM) and Artificial Neural Network (ANN) based machine learning models, along with different feature selection methods. Through experimentation on the NSL-KDD dataset, the study concludes that the model utilizing ANN with wrapper feature selection achieves the highest detection rate of 94.02 percent, outperforming other models considered in the study. The paper emphasizes the significance of these findings in advancing research in intrusion detection systems, particularly in the detection of novel or zero-day attacks, which remains a challenging research area due to the high false positive rates of existing systems [4] [5].

The classification of network packets for intrusion detection without decrypting their contents is addressed in the paper "Network Intrusion Detection using Machine Learning Techniques" by Sumaiya Thaseen I *et al.* The study focuses on creating and sending packets across a network, which Wireshark then records in order to analyze them for intrusion detection. Following the creation of a dataset and preprocessing of the collected data with the Weka tool, the authors apply a number of machine learning algorithms, such as K-nearest Neighbors, Random Forest, Support Vector Machine, and Naive Bayes. The classification accuracy of these algorithms is 83.63, 98.23, 99.81, and 95.13 percent. The most accurate classifier, according to the study, is Random Forest, which divides packets into two categories: malicious and benign. In order to improve performance and accuracy in packet recognition and classification, the conclusion emphasizes Random Forest's superiority over other algorithms in classifying various types of data packets and recommends more research using deep learning algorithms. By shedding light on the efficiency of machine learning algorithms in categorizing network packets and suggesting possible directions for further research to increase detection accuracy through the use of deep learning techniques, this study advances the field of network intrusion detection [6] [7].

3. Data Description

The dataset used in this study was sourced from Kaggle and comprises a total of

50,384 entries, each containing 42 features. These features encompass a range of network metrics, including protocol type, service, and traffic attributes such as the number of bytes transmitted from source to destination, login status, and error rates. The final column, denoted as "class," serves as the target variable, delineating whether the network traffic is classified as "normal" or signifies an "anomaly." Specifically, the dataset consists of 26,898 instances labeled as "normal" and 23,486 instances labeled as "anomaly." This dataset provides a comprehensive foundation for conducting analyses and developing models aimed at network intrusion detection, facilitating the exploration of various machine learning techniques to differentiate between normal and anomalous network behavior [8] [9].

The class distribution of the network traffic data is shown in Figure 1. The x-axis represents the traffic classification (normal or anomalous), and the y-axis indicates the frequency of each class. In this case, the dataset appears to be balanced with approximately 26,898 normal traffic instances and 23,486 anomalous traffic instances. This balanced distribution reduces the possibility of bias problems that can arise during training with imbalanced datasets.

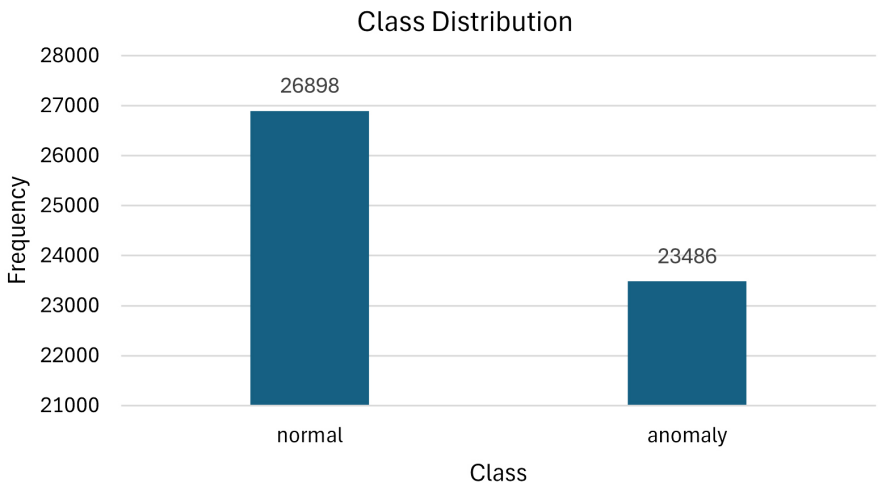
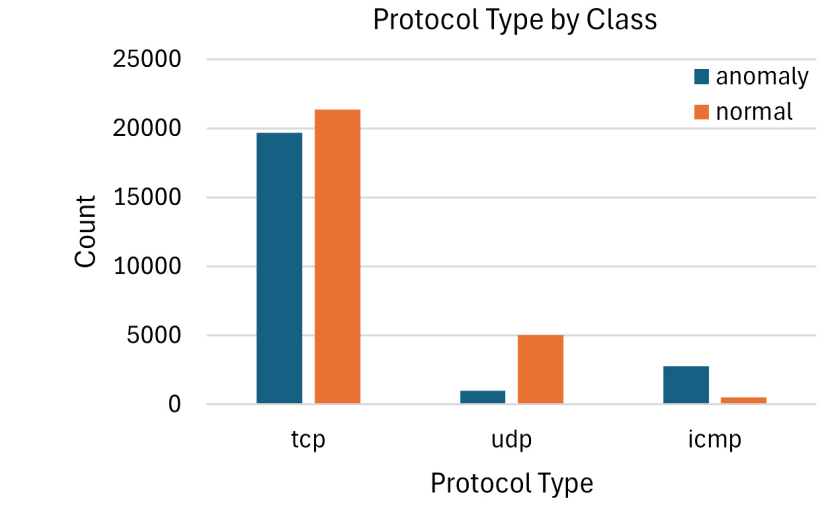
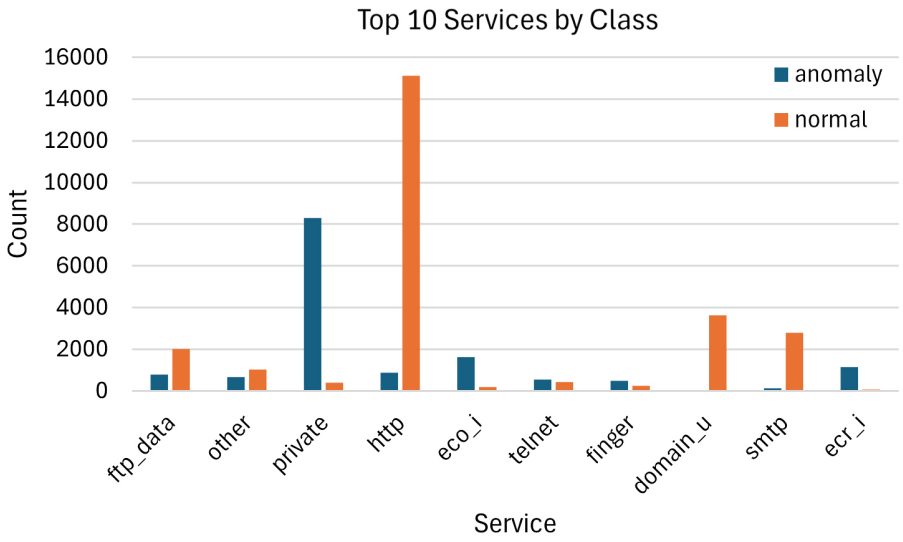


Figure 1. Data distribution by class.

The first graph shown in Figure 2. "Protocol Type by Class", shows the distribution of network traffic events across three common protocols (TCP, UDP, and ICMP) and classifies them as normal or anomalous. TCP is the most frequently used protocol, and most of its traffic is benign. UDP traffic is less frequent, and also has a lower anomaly rate. ICMP traffic is the least common, and also has a low anomaly rate. The second graph in Figure 2. "Top 10 Services by Class," dives deeper into the service types observed in the network traffic data. The most unusual traffic in this case comes from "private" services, indicating possible safety concerns. Conversely, HTTP traffic exhibits a low rate of anomalies and a significant volume of normal activity. Other services such as "telnet" and "ftp-data" display a mixture of normal and unusual traffic, which could indicate vulnerabilities or targeted attacks.



(a)



(b)

Figure 2. (a) Protocol type by class. (b) Top 10 services by class.

Figure 3 consists of series of histograms that show the distribution of various network traffic features. These features offer insights into typical network behavior patterns, aiding in anomaly detection. Most connections are brief (duration), involve small data transfers (source and destination bytes), and rarely connect repeatedly to the same host or service within a short timeframe (count and service count). Interestingly, a significant portion of connections have no errors (error rates), but there's also a noticeable presence of connections with 100 percent error rates. In terms of service usage, many connections consistently connect to the same service, with very few connections switching between different services (service rates). Overall, these histograms suggest that regular and predictable behavior characterizes most network traffic, while outliers, particularly those with high error rates or frequent service switching, could indicate potential anomalies or security threats.

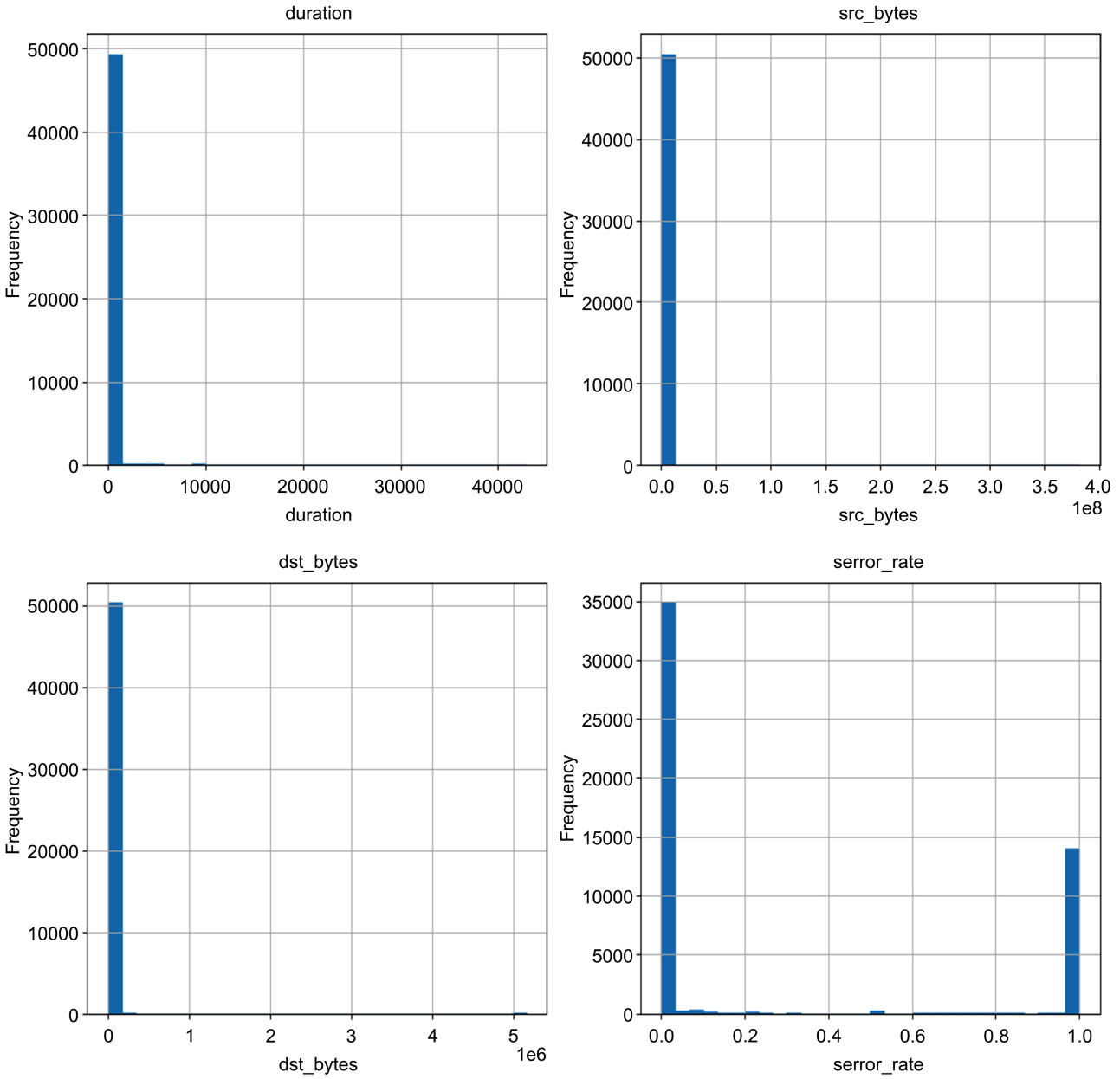


Figure 3. Numerical feature analysis.

The correlation matrix in **Figure 4**. provides insightful information about the connections between specific numerical features in the dataset. The matrix illustrates cases of no correlation between variables as well as strong and weak correlations, with correlation coefficients ranging from −1 to +1. While significant negative correlations, like that between "*serror_rate*" and "*diff_srv_rate*" is (−0.76), point to an inverse relationship between SYN error rate and service diversity, notable positive correlations, like that between "count" and "*srv_count*" is (0.47), suggest synchronized behavior in connection counts. These results highlight the significance of comprehending correlation ideas and realizing how feature selection and anomaly detection procedures are affected by them

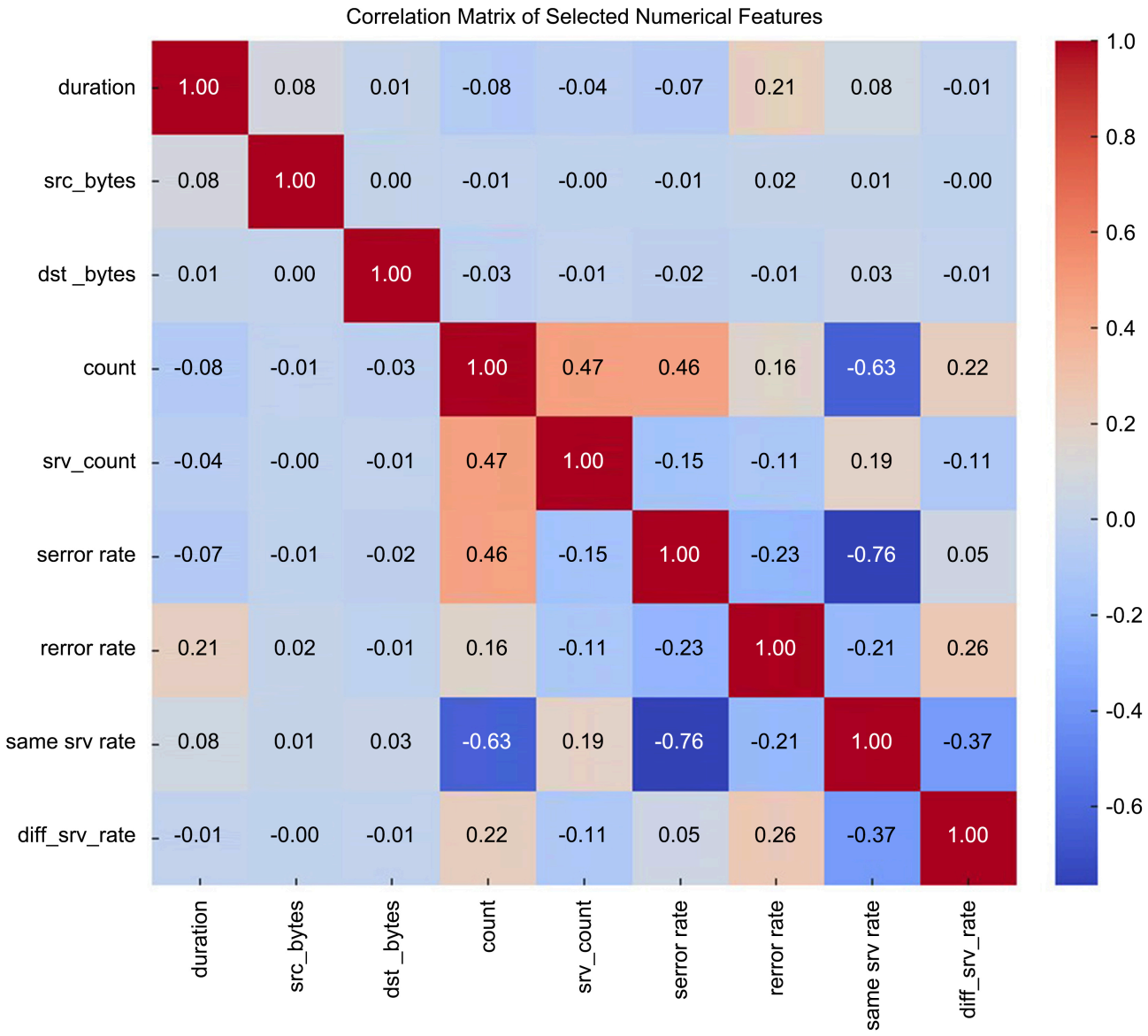


Figure 4. Numerical feature analysis.

4. Data Preprocessing

In the data preprocessing phase, the following steps were undertaken:

1) Loading Data: The dataset was imported into the Python environment using the pd.read_csv() function. This step involves reading the data from the file NID_data.csv and loading it into a pandas DataFrame for additional processing.

2) Removing Constant Features: Constant features—features that have the same value in every instance—were found and eliminated. This procedure is necessary because constant features don't increase the model's capacity for prediction. The quantity of unique values for every feature was determined by utilizing the apply() function in conjunction with pd.Series.nunique. The dataset's dimensionality and possible noise were decreased by discarding features that had just one unique value.

3) Separating Features and Target Variable: The features (X) and the target variable (Y) made up the two parts of the dataset. The "class" target variable indicates if the network traffic is typical or unusual. For supervised learning tasks, where the objective is to predict the target variable based on the input features, this separation is essential.

4) Encoding Categorical Columns and Standardization: By using pd.get_dummies(), the categorical variables in the feature set were converted into numerical representations through one-hot encoding. To transform categorical variables into a format that machine learning algorithms can use to make better predictions, one-hot encoding is required. After that, the standardized features were scaled using the StandardScaler() from sklearn.preprocessing. to have a mean of 0 and a standard deviation of 1. For algorithms that measure the distances between data points or assume normally distributed data, standardization is essential.

5) Principal Component Analysis (PCA): As part of dimensionality reduction, PCA was applied to the standardized feature set. PCA is a technique that transforms the original features into a new set of uncorrelated components, ordered by the amount of variance they capture. In this study, PCA was configured to retain 95% of the variance in the dataset, ensuring that the most critical information is preserved while reducing the number of features. This step helps in reducing computational complexity and mitigating the curse of dimensionality [10].

6) Label encoding: Following PCA, LabelEncoder from sklearn was used to transform the target variable. In this step, the numerical labels 0 and 1 were used to encode the categorical target variable, which indicated whether network traffic was "normal" or "anomaly". In order for machine learning algorithms to process the target variable in a binary classification task, this transformation was necessary [11].

7) Splitting Training and Testing Set: Then, using the train_test_split function from sklearn.model_selection., the dataset was divided into training and testing sets. The model was trained using the training set, which comprised 80% of the data, with the remaining 20% set aside for validating and testing the model's performance on untested data. To ensure reproducibility, or consistency in the results across multiple runs, a random state of 42 was chosen [12].

5. Defining and Training the Stacking Ensemble

1) Defining the Stacking Ensemble:

- The ensemble classifier combines many base models with a final meta-model by using the StackingClassifier from sklearn.ensemble.
- Base models that have been defined as estimators consist of:
  - MLPClassifier: ReLU activation and two hidden layers (128 and 64 neurons) comprise this multi-layer perceptron neural network. Effective training is ensured by parameters like max_iter = 1000 and the "adam" solver.
  - KNeighborsClassifier: a distance-weighted voting algorithm that uses the K-nearest neighbors classifier to examine the five closest neighbors.

- GaussianNB: a big dataset-suitable Gaussian Naive Bayes model that assumes feature independence.
- DecisionTreeClassifier: Minimum samples per split and leaf, and a maximum depth of 10 to avoid overfitting in the decision tree.

- The final estimator, LogisticRegression, gains the ability to efficiently combine base model predictions.
- Setting cv = 5 ensures robustness through 5-fold cross-validation.

2) Training the Ensemble:

- The stacking ensemble is trained on the training data by invoking fit(X_train, y_train).
- The final estimator is trained using the predictions made by the base models, which are originally trained as a new feature set.
- Through this procedure, the final estimator might potentially improve overall performance by refining predictions based on information from several base models.

6. Training Process of Stacking Ensemble

1) Training Process:

- **Training Base Models on Original Data:** Within the stacked ensemble, the original training dataset (X_train) is used to individually train each base model. Since this dataset includes the original input features, every model can identify trends and connections in the information.
- **Generating Predictions for a New Dataset:** After the base models are trained, they generate predictions based on X_train. These predictions are not just for evaluation but are used as new features to create a secondary dataset. This new dataset serves as input for the final estimator, effectively transforming the problem into a higher-level learning task.
- **Ensuring Robustness with Cross-Validation:** To enhance the model's robustness and prevent overfitting, a cross-validation approach (with cv = 5) is employed. During this process, each segment of X_train is used once as a validation set while the remaining segments serve as the training set. The predictions made during each fold are combined to create a new dataset, which ensures that the final estimator does not receive overly optimistic information that could lead to overfitting.
- **Creating the Level-1 Dataset:** The predictions from all base models are compiled to form a new feature set, known as the Level-1 dataset. This dataset retains the original order of instances from X_train but replaces the original features with the predictions generated by the base models. This new representation captures the learned patterns from multiple models, allowing the final estimator to make more informed decisions.

2) Predicting Process:

- **Training the Final Estimator:** The Level-1 dataset, created from the base model's predictions, is used to train the final estimator, which in this case is a

LogisticRegression model. The final estimator learns how to best combine the predictions from the base models to improve the overall performance of the ensemble.

- **Making Final Predictions:** Once the final estimator is trained, it can be used to make predictions on new, unseen data (X_test). The process involves first passing the test data through the base models to generate predictions, which are then used by the final estimator to produce the final productions. This layered approach allows the stacked ensemble to leverage the strengths of multiple models and make more accurate and robust predictions.

7. Performance Evaluation of the Stacked Ensemble Model

After applying the stacked ensemble model, the results shown in Table 1, indicate high performance across multiple evaluation metrics:

Table 1. Performance metrics of the model.

Metric	Value	
Accuracy	0.9993	
Precision	0.9989	
Recall	0.9998	
Confusion Matrix	4707	6
	1	5363
F1 Score	0.9993	

- **Accuracy**: Achieved an accuracy of 99.93%, indicating the proportion of correctly classified instances.
- **Precision**: Demonstrated a precision of 99.89%, reflecting the proportion of correctly identified positive **predictions** out of all positive predictions made.
- **Recall**: Attained a recall rate of 99.98%, representing the proportion of correctly identified positive **instances** out of all actual positives in the dataset.
- **Confusion Matrix**: The confusion matrix reveals the model's classification performance:
  - True negatives (TN): 4707
  - False positives (FP): 6
  - False negatives (FN): 1
  - True positives (TP): 5363
- **F1 Score**: Yielded an F1 score of 99.93%, which is the harmonic mean of precision and recall, providing a balanced measure of model performance.

These results as shown in Table 1. collectively demonstrate the effectiveness of the stacked ensemble model in accurately classifying instances, with minimal misclassifications and high overall performance [13] [14].

8. Conclusion

In conclusion, the implementation of the stacked ensemble model for network intrusion detection yielded exceptional results, showcasing high accuracy, precision, recall, and F1 score. Despite the possibility of overfitting due to the model's high accuracy, the focus remains on enhancing the model's ability to generalize and adapt to diverse intrusion scenarios. Moving forward, integrating Generative Adversarial Networks (GANs) into the model architecture could significantly enhance its detection capabilities. By training the model on synthetic data generated by GANs, it can adapt to a broader spectrum of network intrusion scenarios, thereby improving its robustness and detection accuracy. Additionally, the hybrid approach employed in this study not only demonstrated improved detection accuracy but also provided valuable insights, paving the way for developing more resilient and interpretable intrusion detection systems in the future, capable of addressing evolving cyber threats with greater efficacy and precision.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Moustafa, N., Turnbull, B. and Choo, K.R. (2019) An Ensemble Intrusion Detection Technique Based on Proposed Statistical Flow Features for Protecting Network Traffic of Internet of Things. *IEEE Internet of Things Journal*, **6**, 4815-4830. https://doi.org/10.1109/jiot.2018.2871719

[2] Kambe, T. (2007) Elementary Fluid Mechanics. World Scientific Publishing Co. Pte. Ltd. https://doi.org/10.1142/9789812706676

[3] Kambe, T. (2010) A New Formulation of Equations of Compressible Fluids by Analogy with Maxwell's Equations. *Fluid Dynamics Research*, **42**, Article ID: 055502. https://doi.org/10.1088/0169-5983/42/5/055502

[4] Penchala, S., Murad, S.A., Roy, I., Gupta, B. and Rahimi, N. (2024) Unveiling Text Mining Potential: A Comparative Analysis of Document Classification Algorithms. *Proceedings of* 39*th International Conference on Computers and Their Applications*, **98**, 103-115. https://doi.org/10.29007/lsgw

[5] Kambe, T. (1984) Axisymmetric Vortex Solution of Navier-Stokes Equation. *Journal of the Physical Society of Japan*, **53**, 13-15. https://doi.org/10.1143/jpsj.53.13

[6] Murad, S.A., Rahimi, N. and Md Muzahid, A.J. (2023) PhishGuard: Machine Learning-Powered Phishing URL Detection. 2023 *Congress in Computer Science*, *Computer Engineering*, *& Applied Computing* (*CSCE*), Las Vegas, 24-27 July 2023, 2279-2284. https://doi.org/10.1109/csce60160.2023.00371

[7] Ladyzhenskaya, O. (1969) The Mathematical Theory of Viscous Incompressible Flow, Translated from the Russian by Richard A. Silverman, Mathematics and Its Applications. 2nd Edition, Gordon and Breach.

[8] Galdi, G.P. (2011) An Introduction to the Mathematical Theory of the Navier-Stokes Equations. Springer.

[9] Gautam, A. and Rahimi, N. (2023) Viability of Machine Learning in Android Scareware Detection. *Proceedings of* 38*th International Conference on Computers and Their Applications*, **91**, 19-26. https://doi.org/10.29007/n5ft

[10] Majda, A.J. and Bertozzi, A.L. (2001) Vorticity and Incompressible Flow. Cambridge University Press. https://doi.org/10.1017/cbo9780511613203

[11] Broome, H., Shrestha, Y., Harrison, N. and Rahimi, N. (2022) SMS Malware Detection: A Machine Learning Approach. 2022 *International Conference on Computational Science and Computational Intelligence* (*CSCI*), Las Vegas, 14-16 December 2022, 936-941. https://doi.org/10.1109/csci58124.2022.00167

[12] Tao, T. (2015) Finite Time Blowup for an Averaged Three-Dimensional Navier-Stokes Equation. *Journal of the American Mathematical Society*, **29**, 601-674. https://doi.org/10.1090/jams/838

[13] Constantin, P. and Fefferman, C. (1993) Direction of Vorticity and the Problem of Global Regularity for the Navier-Stokes Equations. *Indiana University Mathematics Journal*, **42**, 775-789. https://doi.org/10.1512/iumj.1993.42.42034

[14] Constantin, P. (1990) Navier-Stokes Equations and Area of Interfaces. *Communications in Mathematical Physics*, **129**, 241-266. https://doi.org/10.1007/bf02096982