Machine Learning (ML) and Data Mining (DM) Methods for Cybersecurity Intrusion Detection

Asrith Venkata Polapragada
Department of Computer Science
Georgia State University
Atlanta, Georgia, United States
apolapragada1@student.gsu.edu

Avyuktkrishna Ramasamy
Department of Computer Science
Georgia State University
Atlanta, Georgia, United States
aramasamy2@student.gsu.edu

Dr. Peng Wang
Department of Computer Science
Georgia State University
Atlanta, Georgia, United States
wpeng2@gsu.edu

***Abstract*—Growing sophistication of novel cyber attacks is rendering traditional signature-based methods of intrusion detection increasingly inadequate. The purpose of this paper is to shed some light on the importance of machine learning-driven intrusion detection in the everchanging landscape of cybersecurity. Utilizing the DrDoS_DNS dataset – a refined subset of the NSL-KDD dataset comprised of 1000 instances and 16 features – we investigate the performance of several supervised learning algorithms, specifically Decision Trees, Support Vector Machines (SVM), and Logistic Regression. Our experiment evaluates these methods in terms of metrics such as speed and accuracy.**

Keywords—IDS, DrDOS_DNS, Decision Trees, SVM, Logistic Regression, Cyber Attack Methods

I. INTRODUCTION

Today our computer systems, networks, computer infrastructures are constantly under new cyber threats that are trying to breach the security that has been put in place traditionally. The traditional defenses are firewalls, antivirus applications, intrusion detection systems (both hardware- and software-based), as well as network monitors to name a few. The main purpose for these defenses is to stop the attacks from breaching and protect the assets be it devices, network and data. In this paper our focus is going to be intrusion detection systems or IDSs and what purpose these systems serves essentially.

Intrusion Detection Systems (IDS) in particular are part of the essential line of defense against threats. It continuously monitors network traffic and logs looking for suspicious patterns and an alert is raised in real-time. However with the emergence of new attack methods like DrDoS (distributed reflective denial of service), particularly DNS amplification attacks, the traditional signature-based IDS have been struggling to keep pace since the signatures keep shifting.

Machine Learning (ML) and Data Mining (DM) offers promising alternative approach to this particular challenge. The ML driven IDS has the ability and capacity to learn and adapt based on the network behavior in near real-time verses the static signature-based models. The ML driven IDS can recognize attack patterns by developing models for normal network activity and then raising alerts for deviations through anomaly-based and hybrid detection techniques. Anomaly based methods may result in higher rates of positives than the hybrid implementations which combine the accuracy of signature-based along with the adaptability of anomaly detection. This has shown to provide a robust solution for modern cybersecurity.

In this paper our focus is going to be on the detection of DrDoS attacks that uses ML-enhanced IDS. We have approached it based on the DrDoS_DNS dataset that offers a streamlined, advanced version of the classic NSL-KDD dataset, that is free from duplicates and null/missing-value records. This avoids bias and allows the model to be trained robustly. The features of this dataset are the range of network flow characteristics – protocol type, flow duration, packet count metrics and throughput statistics. This serves as a rich basis for the preprocessing and feature engineering steps that are critical to the success of any machine learning model.

We have evaluated three prominent supervised learning algorithms, viz., decision trees, support vector machines (SVM), and logistic regression to verify their effectiveness in determining the type of traffic between benign vs attack. The cost comparison underscores the trade-offs that are inherent in model selection for IDS applications that require both speed and precision.

The other sections of this paper are organized as follows. Section II has reviews related to work in ML-based intrusion detection and goes into the evolution from datasets like KDD Cup 99 and NSL-KDD to the refined DrDoS_DNS dataset. Section III has the details of our methodology that includes data preprocessing, feature engineering, and selection of the model. Section IV lays out the experimental results and a comparative analysis of our chosen algorithms. Finally, Section V ends with what can be done in future research and the potential for integrating ML-based IDS in real-world applications on cybersecurity infrastructures.

II. RELATED WORKS

IDS have evolved from traditional signature-based approaches to more dynamic methods over the years utilizing ML and data mining (DM) techniques. Earlier IDS implementations depended on fixed attack signatures and rule-based systems in order to recognize known malicious behavior; however the attackers began exploiting vulnerabilities and launching novel zero-day attacks. Gradually the limitations of static signature-based systems became increasingly apparent.

Recent academic research focused on applying ML models for the purpose of improving the IDS performance. The model will learn the nuances of regular network behavior and then flag any deviations as potential threats - this is the principle behind anomaly-based detection. Stewart, Kolajo, and Daramola (2024) conducted a systematic literature review of ML-driven approaches, and they highlighted algorithms such as SVM k-nearest neighbors (KNN) decision trees convolutional neural networks (CNN) random forests and

naïve Bayes as the most commonly used classifiers [1]. Their work shows that even though the ML-based system can dramatically improve detection rates, there are still some challenges in relation to training the model, selecting features, and maintaining low false positive rates.

A significant advancement is application of sophisticated feature engineering techniques to better extract some discriminative attributes from a given set of network data. In a recent study, Sapna Sadhwani et al. (2025) demonstrated that intelligent IDS for IoT network security built on resilient feature engineering, could achieve accuracy results exceeding 98% by process of carefully structuring the input features and applying only lightweight ML algorithms [2]. This work shows how dedicated feature extraction/selection can mitigate some of the shortcomings in traditional IDS approaches.

Real-time IDS procedures have emerged in parallel that utilize DM techniques to further boost the system's performance and adaptability. For example, the paper "Real-Time Data Mining-Based Intrusion Detection" proposes framework which not only improves the accuracy of detecting intrusion by using artificial anomaly generation, but also incorporates cost-sensitive modeling in order to handle the computational resources more effectively [4]. Artificial anomaly generation is used to simulate some attack patterns that may be rare in the real-world dataset, for the purpose of enriching the training data and helping the detection models to learn a broader variety of malicious behavior. This means instead of only relying on naturally occurring training data, the system is deliberately creating some synthetic anomalies that would mimic the potential attack scenarios. This can consequently improve the behavior generalization of both misuse-based methods and anomaly detection.

Cost-sensitive modeling is applied in the framework by analyzing computational cost associated with feature sets, then designing a multiple-model cost-based procedure. This approach would make use of various classification models each one optimized under particular cost constraints so that models who are less computationally intensive are chosen, while still maintaining the high level of accuracy in intrusion detection. From a practical standpoint, this layered strategy allows system to adjust/balance between speed and precision based on the current availability of system resources and network conditions. Complementing the efforts, "UNSW-NB15: A Comprehensive Dataset for Network Intrusion Detection Research" one of the papers, provides a modern and detailed dataset that enabled ML-based analysis of contemporary threats [5]. It adds further value by offering realistic basis for advanced DM techniques' testing, helping the researchers and practitioners alike to validate and also improve their detection algorithms under dynamic conditions.

Number of research experiments have employed supervised learning clustering and/or deep learning to enhance anomaly detection of ML-driven IDS. These studies effectively demonstrated that such techniques can improve identification of novel/subtle attack patterns while reducing the false positives. Series of experiments have confirmed effectiveness of these approaches in enhancing the cybersecurity defenses [6][7][8][9]. Additionally metaheuristic optimization methods have been applied to further refine the intrusion detection - Hulsure et al. (2025) proposes XGBoost-based IDS which integrates some advanced optimization techniques including genetic algorithms and improved Whale Optimization Algorithm to enhance the model's precision and reduce computational cost [3]. These hybrid methods combine the adaptability of an anomaly-based detection with the reliability of the misuse-based recognition and thus effectively addresses the evolving landscape of cyber threats.

Alongside the methodological advances marked by these various studies the benchmark datasets used to train the ML models for intrusion detection have also undergone significant changes and refinement in order to better align any experimental frameworks with real-world threats. The dataset developed in 1999 for Knowledge Discovery and Data Mining (KDD) competition provided an early benchmark; however it suffered from issues such as duplicate records and null/missing-value records. Its successor NSL-KDD addressed these problems and further improvements led to specialized datasets such as DrDoS_DNS - this dataset is comprised of 6785 instances and 16 features. By distilling NSL-KDD down into two major types of records - DrDoS_DNS or BENIGN it helps facilitate a focused evaluation of various ML algorithms to be tested.

Collectively the studies [1][2][3][4][5][6][7][8][9] represent the progression from static signature-based IDS to sophisticated ML-driven frameworks which utilize advanced feature engineering, anomaly detection, and specialized datasets to their advantage. These work provides a solid foundation for developing IDS that can not only enhance the detection rates, but also adapt effectively to the complexity of contemporary cyber threats.

III. METHODOLOGY

Our methodology consists of data acquisition, followed by clipping outliers, data normalization and feature engineering, model training, and finally, evaluation of performance based on specific metrics.

A. Data Acquisition and Exploration

DrDoS_DNS dataset contains 6,785 instances of network traffic data and 15 numerical features. The records have been categorized as either BENIGN or DrDoS_DNS attack traffic, this helps provide a structured dataset for IDS evaluation. As mentioned previously, compared with KDD 99 and NSL-KDD datasets, all the duplicate records as well as null/missing-value records have been removed. This eliminates any inherent bias the model may obtain due to training on repeatedly similar/same data. Some initial exploratory data analysis was first conducted in order to examine the dataset's structure, validate absence of null/missing-value records, and also to summarize distributions of features.

6568 instances were labelled as DrDoS_DNS (attack) whereas 217 are labelled as BENIGN (normal). According to our analysis for the research, the dataset is predominantly biased towards the positive label/class. The Pandas library was used to inspect the dataset - `df.info()` and `df.describe()` commands confirmed that the dataset was complete, and also provided insight into statistical distribution of each feature. Missing values have been checked using `df.isnull().sum()`.

This exploration of the data and understanding the structure and statistical properties it has, guided the subsequent preprocessing decisions. Histograms and boxplots were used in order to examine the feature distributions and detect any anomalies in data points. After studying the

distributions, it was ensured later on that data transformations (normalization and handling outliers) were appropriately applied to improve the suitability of the dataset for the ML model training.

B. Outlier Detection and Clipping

Presence of outliers in the dataset can (in the worst case scenario) significantly affect the capability of the ML model to make accurate predictions. This is particularly true for our use case since the extreme values may distort classification boundaries for intrusion detection tasks. In order to identify and mitigate the outliers, visual and statistical analysis methods were utilized. Boxplots were generated using `sns.boxplot()` and these boxplots helped to visually highlight any features with unusually high/low values. Thus with this graphical representation, it was possible to identify any extreme deviations that could interfere with the ML model training.

Using the IQR (interquartile range) method, statistical analysis was performed. IQR method is a reliable approach for finding outliers in a dataset. IQR is calculated as difference between 75th percentile (Q3) and 25th percentile (Q1). Data points lying outside 1.5(IQR) are classified as outliers. However, rather than remove these outliers like most traditional methods do, our research opted to perform clipping instead. Clipping is when the extreme values in these outliers are adjusted to match an acceptable value in IQR. This clipping was done in order to retain the same number of records and any valuable attack patterns, while also making sure that outliers do not interfere with the model training. Pandas library includes a `clip()` function which takes care of recalibrating all the values exceeding the upper and lower thresholds.

C. Data Normalization and Feature Engineering

The ML models usually perform optimally whenever the features have been scaled to uniform range, which prevents any single attribute from influencing the predictions disproportionately. As part of normalization process, Min-Max scaling process was used for converting values into a standardized range of 0 and 1. This normalization was later on particularly beneficial for SVM model, which relies on feature distances in order to construct classification boundaries. Python library `scikit-learn` provides a `MinMaxScaler()` to ensure that there was uniform feature distribution across all the instances.

Correlation heatmap was utilized to identify relationship between the features for the purpose of further refining the model inputs. The analysis has revealed strong correlation between the packet-based attributes, so this prompted us to consider dimensionality reduction techniques like PCA (principal component analysis). Finally, label encoding was applied to target variable `label` for converting BENIGN traffic to 0 and DrDoS_DNS attack traffic to 1 - this made it highly compatible later on with logistic regression algorithm since it has been specifically designed for modeling probability of binary outcome. Our research ensured that the data was structured optimally for enhancing classification accuracy and reducing noise in training by utilizing correlation-based selection and data normalization.

D. Model Selection and Training

In order to evaluate the effectiveness of IDS, three classifiers - Decision Tree, SVM, and Logistic Regression - were trained with the preprocessed dataset. Using `train_test_split()` the dataset was split into two subsets - training set and testing set in a proportion of 80-20. This ensured that during testing, the models were evaluated using a portion of data unseen before. The approach helped facilitate a generally more robust performance assessment and helped to minimize any bias incurred from dataset-specific properties. The models themselves were selected because of their diverse approaches to classification of the data, ranging from tree-based decision making (in the case of Decision Tree) to linear separation techniques (in Logistic Regression and to some extent SVM as well).

Decision Tree classifier was implemented with `DecisionTreeClassifier(random_state=42, max_depth=4)`. Model was using Gini impurity criterion in order to determine the optimal splits. This provided a balance between accuracy of predictions and interpretability. The depth was limited to four since that would prevent overfitting while still allowing the model to capture any essential classification rules.

SVM classifier was trained using a linear kernel (`SVC(kernel='linear')`). The choice was based on the assumption that all the normalized features would amplify the linear separability, so that would help the SVM to construct a hyperplane which is efficient and can differentiate between BENIGN and DrDoS_DNS attack traffic. Even with high-dimensional network data, this approach ensures that the SVM should be able to generalize attack patterns effectively while being able to maintain low false positive rates.

Logistic Regression model was trained using `lbfgs` solver and maximum of 1000 iteration was set (`LogisticRegression(max_iter=1000)`). This served as a lightweight baseline classifier. To further analyze the separability, the model was retrained on PCA-reduced features which enabled visualization of classification boundaries within a two-dimensional space. the scroll down window on the left of the MS Word Formatting toolbar.

E. Performance Analysis and Evaluation Metrics

In order to measure the effectiveness of the classifiers, multiple different evaluation metrics were used. With `confusion_matrix(y_test, y_pred)`, confusion matrices were generated for each model. This is so that the classification performance could be quantified in terms of distinguishing true positives, false positives, true negatives, and false negatives from each other. The matrices provided some insights into each model's ability to correctly identify the instances which were DrDoS_DNS attacks while not misclassifying BENIGN traffic as malicious.

Other than confusion matrices, the F1-score, accuracy, precision, and recall were call computed. Accuracy (`accuracy_score(y_test, y_pred)`) assessed the overall correctness of the model, while precision indicated how reliable the model was for classification of attack instances. The recall value measures how the model is about to detect the actual attacks, and F1-score essentially provides a balanced metric combining precision and recall together.

For the purpose of examining computational efficiency, the training times for each model were recorded. Decision Tree classifier took 0.1472 seconds to be trained, SVM required 0.4372 seconds, and Logistic Regression was the fastest at 0.0587 seconds. The measured training times helped to inform which models could ideally be deployed in real-time IDS that required nearly instantaneous detection/identification of any potential attacks.

`clf.feature_importances_` contained the feature importance for Decision Tree model. For SVM and Logistic Regression, coefficients (`svm_model.coef_` & `log_model.coef_`) helped to analyze which specific features contributed most to the accuracy of the classification. Principle component analysis (PCA) enabled a decision boundary visualization which was used to confirm separability between the DrDoS_DNS attack traffic and BENIGN traffic.

IV. Results and Discussion

The evaluation results demonstrated that all three models, i.e, Decision Tree, Support Vector Machine (SVM), and Logistic Regression; achieved outstanding performance in detecting DrDoS DNS attacks.

A. *Decision Tree*

For attack traffic, the Decision Tree Classifier's precision and recall were 100%, and its overall accuracy was 99.97%. Likewise, the weighted F1-score was 100%. A hierarchy of feature-based decisions is revealed by the Decision Tree structure as shown in Fig1. The most important features in dividing the data into benign and attack classes were total_forward_packets, flow_bytes_per_seconds, and flow_packets_per_seconds as described in Fig4. Effective separation of the two classes is indicated by the tree structure's pure or nearly pure nodes (low Gini index) at the leaves.

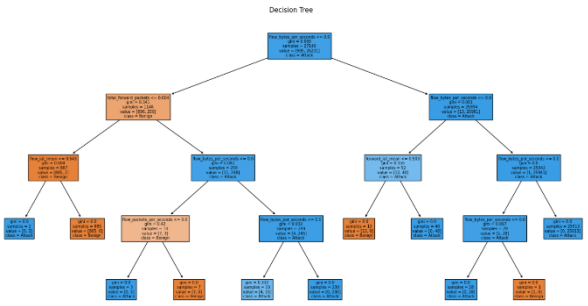

Fig1: Decision Tree Classifier classifying the labels (DrDos_DNS as positive class and BENIGN as negative class). Feature tests are performed using Gini index in order to classify the nodes. Blue nodes represent DrDoS_DNS (attack) labels and the orange nodes represent the BENIGN (normal) labels.

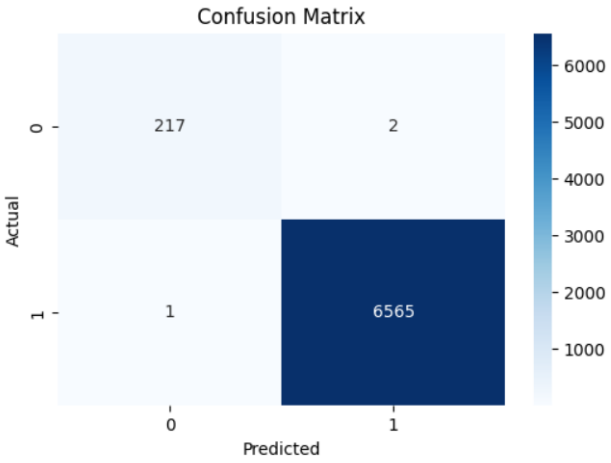

Fig2: Confusion Matrix of the Decision Tree Classifier highlighting True Positives and Negatives along with False Positives and Negatives

```
Accuracy: 0.9995578481945468
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.99      0.99       219
           1       1.00      1.00      1.00      6566

    accuracy                           1.00      6785
   macro avg       1.00      1.00      1.00      6785
weighted avg       1.00      1.00      1.00      6785
```

Fig3: Classification Report of Decison Tree Classifier showing Accuracy,Precision,Recall and F1 score

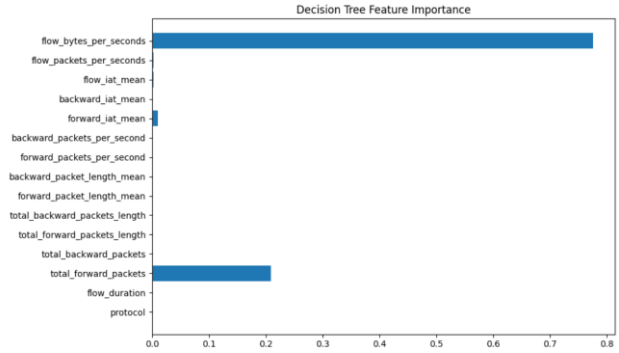

Fig4: Feature Importance Bar graph for Decision Tree Classifier

B. *Support Vector Machine*

The Support Vector Machine (SVM) model, which was trained with a linear kernel, performed similarly to the Decision Tree classifier with an accuracy rate of 99.97%. Attack detection accuracy and recall were 100%. The decision boundary visualization created after the feature space was reduced to two dimensions using Principal Component Analysis (PCA) is shown in Fig4. Attack traffic (red) and benign traffic (blue) are separated by a nearly linear border in this plot. The clear separation and minimal overlap between the two classes illustrate the model's ability to generalize well, even after significant dimensionality reduction. As shown in Fig. 8, feature importance was ascertained by examining the learned coefficients of the model. The backward_iat_mean and forward_iat_mean features had the greatest impact on attack detection. Features linked to DrDos_DNS traffic were indicated by positive coefficients, whereas BENIGN classification was preferred by negative coefficients. Each

feature's contribution to the SVM's decision boundary was represented by the coefficients' magnitudes.

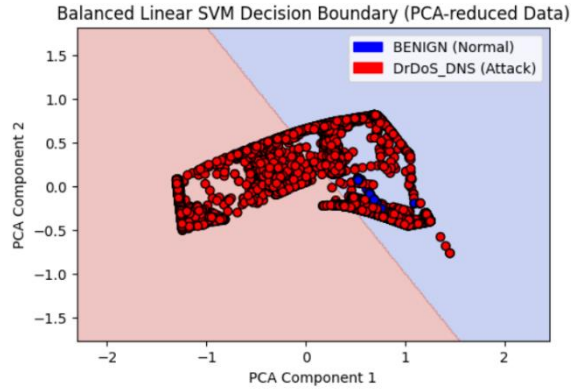

Fig5: SVM decision boundary after PCA dimensionality reduction. BENIGN traffic (blue) and DrDoS DNS attack traffic (red) are separated by a nearly linear boundary.

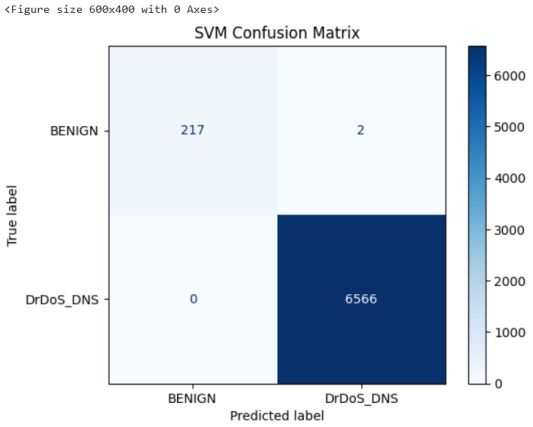

Fig6: Confusion Matrix of SVM

SVM Accuracy: 0.9997052321296979
SVM Classification Report:

	precision	recall	f1-score	support
0	1.00	0.99	1.00	219
1	1.00	1.00	1.00	6566
accuracy			1.00	6785
macro avg	1.00	1.00	1.00	6785
weighted avg	1.00	1.00	1.00	6785

Fig7: Classification Report of SVM

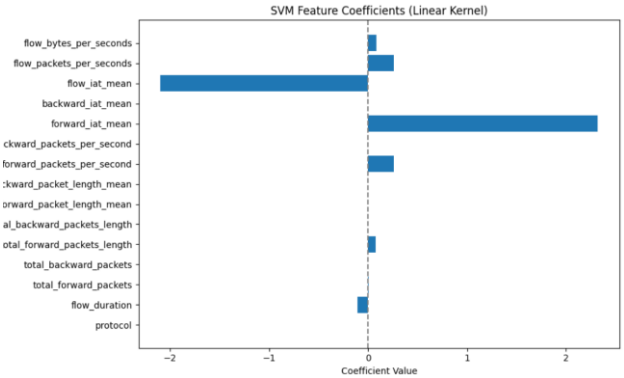

Fig8: Support Vector Machine (linear kernel) feature coefficients. Features with positive coefficients are associated with predicting attack traffic, while negative coefficients are associated with predicting benign traffic.

C. Logistic Regression

The Logistic Regression model achieved 99.97% accuracy with 100% precision and recall for attack traffic. Among the three classifiers, The decision boundary in the PCA-reduced space is linear but slightly skewed, in contrast to the SVM boundary. The majority of attack instances (red) are correctly classified, but a tiny portion of benign instances (blue) are misclassified as attacks, leading to a few false positives. The overall separation is still quite successful, though, showing that the model can faithfully capture the underlying structure of the dataset. As illustrated in Fig. 12, the learned coefficients were examined in order to assess the feature importance. Attack traffic classification was most significantly impacted by features like forward_iat_mean and backward_iat_mean. Negative coefficients were linked to BENIGN traffic, whereas positive coefficients helped predict attacks(DrDoS_DNS). The interpretability of the model was supported by the magnitude of the coefficients, which showed how strongly each feature affected the decision boundary.

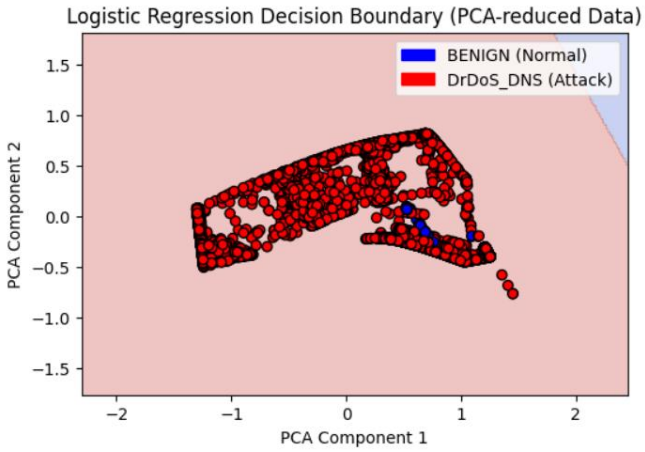

Fig9: Logistic Regression PCA plot

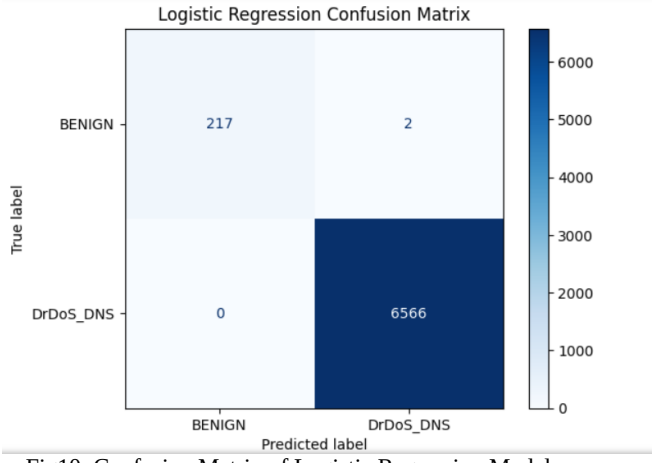

Fig10: Confusion Matrix of Logistic Regression Model

Logistic Regression Accuracy: 0.9997052321296979
Logistic Regression Classification Report:

	precision	recall	f1-score	support
0	1.00	0.99	1.00	219
1	1.00	1.00	1.00	6566
accuracy			1.00	6785
macro avg	1.00	1.00	1.00	6785
weighted avg	1.00	1.00	1.00	6785

Fig11: Classification Report for Logistics Regression Model

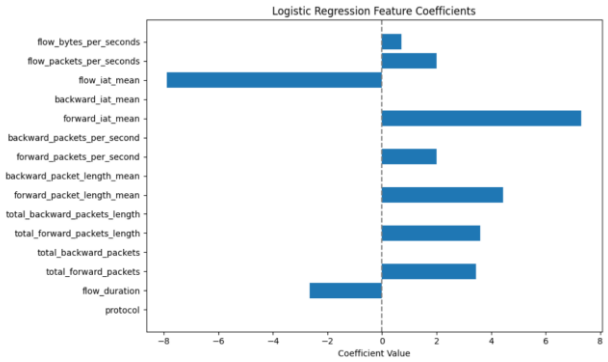

Fig. 12: Logistic Regression model feature coefficients. Features with positive coefficients contribute toward predicting DrDoS_DNS(attack) traffic, while features with negative coefficients contribute toward predicting BENIGN traffic.

V. CONCLUSION

The conducted research has explored the benefits of utilizing ML-based methods for intrusion detection, by building on legacy datasets (KDD 99, NSL-KDD) through the refined DrDoS_DNS dataset. Our methodology – encompassing data preprocessing, feature engineering, and model selection – paved the way for a comprehensive analysis of algorithms' performance in terms of speed, accuracy, and adaptability to the modern cyberthreat landscape.

The experimental results presented in Section IV clearly indicate that by integrating tailored ML models into existing IDS, the detection capabilities of the same can be significantly enhanced. Among the three chosen algorithms, Logistic Regression adeptly balanced detection accuracy and computational speed. SVM takes a close second, maintaining the accuracy while falling short in terms of speed. Decision Tree algorithm was both slower and less accurate. Thus, it can be seen that the algorithm selected plays a critical role in the computational efficiency and detection precision of IDS.

For future work, further research should be conducted to explore utilizing deep learning, ensemble methods, and other ML models to improve the detection accuracy and lower the rates of false positives. Real-world deployment and live evaluations will be key in order to reliably test the scalability and adaptability of the model, uncover any operational challenges, and guide newer improvements for adaptive IDS.

In conclusion, our work illustrates that integration of ML and DM methods into existing IDS framework will represent a transformative leap towards a more accurate and resilient security defense for networked devices. As the digital infrastructure continues to evolve and sophisticated threats develop to match this, we hope that this work will help to bridge the gap between theoretical advances and practical solutions. We encourage other researchers to build on this foundation and explore newer avenues of approaches to protect and safeguard the technological ecosystems against cyber threats.

```
Decision Tree Time: 0.1472 seconds
SVM Time: 0.4372 seconds
Logistic Regression Time: 0.0587 seconds
```

Fig13.Time taken by each model

VI. REFERENCES

[1] D. Stewart, T. Kolajo, and O. Daramola, "Machine Learning for Intrusion Detection Systems: A Systematic Literature Review," *Lecture Notes in Networks and Systems*, pp. 623–638, 2024, doi: https://doi.org/10.1007/978-3-031-73110-5_42.

[2] S. Sadhwani, A. Sriwastawa, R. Muthalagu, and P. M. Pawar, "Feature engineering based intelligent intrusion detection system for IoT network security," *International Journal of Machine Learning and Cybernetics*, Apr. 2025, doi: https://doi.org/10.1007/s13042-025-02631-y.

[3] A. Hulsure, A. Yergude, A. Sawal, O. Aware, and A. Raut, "ADVANCED INTRUSION DETECTION SYSTEM USING MACHINE LEARNING," *International Research Journal of Modernization in Engineering Technology and Science*, pp. 2582–5208, doi: https://doi.org/10.56726/IRJMETS72234.

[4] W. Lee *et al.*, "Real Time Data Mining-based Intrusion Detection." Accessed: Apr. 27, 2025. [Online]. Available: http://ids.cs.columbia.edu/sites/default/files/dmids-discex01.pdf

[5] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," *IEEE Xplore*, Nov. 01, 2015. https://ieeexplore.ieee.org/document/7348942

[6] J. Zhu, C. Ding, Y. Tian, and G. Pang, "Anomaly Heterogeneity Learning for Open-set Supervised Anomaly Detection." Accessed: Apr. 27, 2025. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2024/papers/Zhu_Anomaly_Heterogeneity_Learning_for_Open-set_Supervised_Anomaly_Detection_CVPR_2024_paper.pdf

[7] P. Scherer, M. Vicher, P. Dráždilová, J. Martinovič, J. Dvorský, and V. Snášel, "Using SVM and Clustering Algorithms in IDS Systems Using SVM and Clustering Algorithms in IDS Systems." Accessed: Apr. 27, 2025. [Online]. Available: https://ceur-ws.org/Vol-706/paper25.pdf

[8] R. Kimanzi, P. Kimanga, D. Cherori, and P. K. Gikunda, "Deep Learning Algorithms Used in Intrusion Detection Systems -- A Review," *arXiv (Cornell University)*, Feb. 2024, doi: https://doi.org/10.48550/arxiv.2402.17020.

[9] V. Chakravarthy, D. Bell, and S. Bhaskaran, "Hybrid AI Learning Approaches for Intrusion Detection: A Review," *Studies in systems, decision and control*, pp. 665–681, Jan. 2024, doi: https://doi.org/10.1007/978-3-031-62102-4_56.