Jurnal Informatika Ekonomi Bisnis

http://www.infeb.org

2024 Vol. 6 Iss. 1 Hal: 38-46 e-ISSN: 2714-8491

Analysis and Comparison of Machine Learning Techniques for DDoS Attack Classification in Network Environments

Gregorius Airlangga[1]✉

[1]Atma Jaya Catholic University of Indonesia

gregorius.airlangga@atmajaya.ac.id

Abstract

This research presents a comparative analysis of machine learning techniques for classifying Distributed Denial of Service (DDoS) attacks within network traffic. We evaluated the performance of three algorithms: Logistic Regression, Decision Tree, and Random Forest, including their scaled-feature counterparts. The study utilized a robust methodology incorporating advanced data preprocessing, feature engineering, and Synthetic Minority Over-sampling Technique (SMOTE) to address class imbalance. The models were rigorously tested using a cross-validation framework, assessing their accuracy, precision, recall, and F1 score. Results indicated that the Random Forest algorithm outperformed the others, demonstrating superior predictive accuracy and consistency, albeit with higher computational costs. Logistic Regression, when feature-scaled, showed significant improvement in performance, highlighting the importance of data normalization in models sensitive to feature scaling. Decision Trees provided a quick and interpretable model, though slightly less accurate than the Random Forest. The research findings highlight the trade-offs between predictive performance and computational efficiency in selecting machine learning models for cybersecurity applications. The study contributes to the cybersecurity domain by elucidating the efficacy of ensemble techniques in DDoS attack classification and underscores the potential for model improvement through scaling and data balancing.

Keywords: DDoS Attack, Machine Learning, Logistic Regression, Decision Trees, Random Forest.

1. Introduction

As the digital landscape continues to evolve, cybersecurity threats like Distributed Denial of Service (DDoS) attacks are becoming increasingly sophisticated and damaging [1], [2], [3]. These attacks disrupt essential online services by flooding networks with excessive traffic, posing significant threats to the stability and security of digital infrastructures [4], [5], [6]. The complexity and dynamic nature of these attacks necessitate advanced detection and mitigation strategies [7], [8], [9]. This research is situated within this context, aiming to enhance the detection and classification of DDoS attacks using innovative machine learning approaches. The literature on DDoS attack detection is vast and diverse. Studies such as those who explored various machine learning techniques for network anomaly detection, providing valuable insights but often limited by static modeling approaches [10], [11], [12]. Another research emphasized the potential of deep learning methods, yet these require extensive computational resources and large datasets [13]. On the other hand, another work investigated classical machine learning models, highlighting their efficiency but pointing out their limitations in handling complex and high-dimensional data typical in network traffic [14]. Furthermore, the work from other source on the issue of imbalanced datasets prevalent in network security, proposing various sampling techniques to enhance model performance [15].

The current state of the art incorporates a range of techniques, from traditional machine learning models to more recent deep learning frameworks [16], [17]. Despite the progress, there remains a crucial gap in comprehensively comparing the effectiveness of different machine learning models, particularly in the context of DDoS attack classification. Many studies focus on a single model or a specific aspect of the classification problem, lacking a holistic approach that considers various models under uniform experimental conditions.

Addressing this gap, our research makes several key contributions, firstly is an extensive data preprocessing and feature engineering, in this process, we apply a series of advanced data preprocessing techniques to refine the network traffic data, ensuring high-quality inputs for model training. Additionally, our unique feature engineering strategy enhances the models' capacity to distinguish between normal traffic and DDoS attacks. Secondly, by implementing diverse machine learning models. This process is a central aspect of our study to do comparative analysis of three widely used machine learning models: Logistic Regression, Decision Trees, and Random Forest. This comparison provides a comprehensive view of their performance in the context of DDoS attack classification.

We also employ PCA for effective dimensionality reduction, allowing us to manage the complex nature of

Accepted: 05-01-2024 | Revision: 09-01-2024 | Publication: 31-03-2024 | doi: 10.37034/infeb.v6i1.795

network data. Simultaneously, our use of SMOTE addresses the challenge of class imbalance, a common issue in network security datasets [18], [19], [20]. To do In-depth comparative evaluation, the evaluation method involves a rigorous analysis of the models based on accuracy, precision, recall, and F1 score. This thorough comparative study helps in identifying the most effective model under various scenarios, contributing significantly to the field of network security. This study's implications extend beyond the immediate realm of DDoS attack classification. By providing a detailed comparison of different machine learning models, our research contributes to the broader understanding of their applicability in cybersecurity. The methodologies and insights gained can be applied to other areas of network security, potentially aiding in the development of more robust and adaptable defense mechanisms. Furthermore, our work sets a foundation for future research, encouraging further exploration into the comparative analysis of machine learning models in cybersecurity.

2. Research Method

2.1 Data Collection and Preprocessing

The dataset used in this study was sourced from Kaggle, specifically designed to contain network logs pertinent to DDoS attacks. The 'DDoS Attack Network Logs' dataset comprises various network attributes that are key to discerning traffic patterns indicative of attacks. The initial step involved loading the dataset using the ArffLoader function, tailored for handling the ARFF (Attribute-Relation File Format) commonly utilized in machine learning datasets.

Furthermore, we create dataframe creation in order to do post-loading, the data was structured into a Pandas DataFrame. This format is conducive for data manipulation in Python, offering a wide array of functionalities for data analysis. After that, we create feature-target separation, in this process, the dataset was bifurcated into feature columns ('df_X') and a target column ('df_target'). This separation is a cornerstone of supervised learning, where the model learns to predict the target variable from the features.

Furthermore, data encoding is implemented, the dataset contained byte sequences which were converted to strings for uniformity and ease of processing. This encoding step is crucial for handling categorical data in subsequent analysis. Then, specific columns were cast to integer and object types to maintain consistency with Python's data processing libraries. We also conducted a thorough check for missing values and duplicate entries to ensure data integrity. Handling missing values is vital to prevent inaccuracies during model training.

2.2 Feature Engineering

Feature engineering is a pivotal step where domain knowledge is leveraged to extract and optimize features from raw data. Firstly, categorical data handling, in this process, the categorical values in the 'FLAGS' column were replaced with numerical codes, as most machine learning algorithms necessitate numerical inputs. Secondly, one-hot encoding, this technique was employed to process categorical variables, converting them into a format amenable to machine learning algorithms, thus aiding in improving model accuracy.

Thirdly is logarithmic transformation, this process is important to address data skewness, logarithmic transformations were applied to various packet-related features. This approach is often effective in normalizing data distributions, enhancing the performance of learning algorithms.

2.3 Dimensionality Reduction and Class Imbalance Handling

PCA (Principal Component Analysis) was employed to mitigate the challenge of high-dimensional data. PCA reduces the dimensionality while retaining most of the data variance, aiding in simplifying the dataset. To address the class imbalance prevalent in network security datasets, SMOTE (Synthetic Minority Over-sampling Technique) was utilized. This technique synthesizes new samples from the minority class, balancing the dataset for training.

2.4 Model Implementation

In the model implementation phase of our research, we dedicated our efforts to the deployment of three distinct machine learning algorithms, each chosen for their relevance and potential in addressing the classification challenges posed by DDoS attack detection in network traffic data. Logistic Regression was the first of the three models we implemented. As a probabilistic linear classifier, Logistic Regression is traditionally prized for its simplicity and interpretability. In our implementation, we adapted the model to handle the binary classification task by modeling the log-odds of the probability of an attack as a linear combination of the input features. This required careful consideration of the feature space and the relationships between the features and the probability of an attack to ensure that the logistic function's output could be effectively threshold to distinguish between the two classes of interest.

Decision Trees were selected for their intuitive representation of decision-making processes, mirroring the if-then-else decision rules that can be readily understood. To implement the Decision Tree, we constructed a flowchart-like structure that recursively split the data into homogenous subsets. This was achieved by identifying the features that resulted in the most significant reduction in a given impurity measure (such as Gini impurity or entropy) at each node. Given the diverse nature of network traffic data, the Decision Tree was fine-tuned to prevent overfitting while

maintaining sufficient complexity to capture the underlying patterns indicative of DDoS attacks.

Random Forest was the final model we implemented, chosen for its robustness and accuracy resulting from its ensemble approach. By integrating multiple Decision Trees, each trained on a different subset of the data and features, the Random Forest model mitigates the overfitting tendencies of individual trees. In our study, the Random Forest model was composed of numerous trees whose predictions were aggregated through majority voting for classification. We adjusted the number of trees and the depth of each tree to optimize the trade-off between model bias and variance, ensuring that the model captured the essential characteristics of the data without being swayed by noise.

Each model was implemented with a particular emphasis on optimizing for the idiosyncrasies of our dataset, which included an imbalance between the classes and a wide range of feature scales. We employed techniques such as feature scaling and class weighting to tailor each model to the dataset's specific characteristics and requirements. Through rigorous hyperparameter tuning and validation, we ensured that each model achieved a high level of performance while avoiding the pitfalls of overfitting or underfitting. The outcome of this careful implementation was a set of models that were well-suited to our data and capable of providing insights into the nature of DDoS attacks within network traffic.

2.5 Evaluation Metrics and Procedures

Model evaluation was conducted using cross-validation, a robust technique for assessing model performance on unseen data. We employed a range of metrics, including accuracy, precision, recall, and F1 score, to comprehensively evaluate each model. For categorical variables, one-hot encoding transforms a categorical variable with 'n' categories into 'n' binary features, each representing one category. For a category 'c', the feature corresponding to 'c' is 1, and all other features are 0.

For a variable x, the logarithmic transformation is given by, y = log(x), where 'log' is the natural logarithm. This is applied to skewed features to normalize their distribution. Furthermore, PCA involves the computation of the eigenvalue decomposition of a data covariance matrix or singular value decomposition of a data matrix, usually after mean centering the data for each attribute. The principal components are the eigenvectors of this covariance matrix. In order to measure effectiveness of the algorithm we used four metrics: accuracy, precision, recall and F1, these are described in the Equation (1-4).

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \quad (1)$$

$$Precision = \frac{TP}{(TP + FP)} \quad (2)$$

$$Recall = \frac{TP}{(TP + FN)} \quad (3)$$

$$F1 = \frac{2\,(Precision * Recall)}{(Precision + Recall)} \quad (4)$$

Where TP is a True Positives, TN is a True Negatives, FP is a False Positives and FN is a False Negatives.

2.6 Logistic Regression

Logistic Regression is commonly used for binary classification problems. It models the probability of a binary outcome using the logistic function as presented in the Equation (5).

$$P_{(y=1)} = \frac{1}{(1 + e^{-(\beta_0 + \beta_1 * x_1 + \cdots + \beta_n * x_n)})} \quad (5)$$

Where $P_{(y=1)}$ is the probability of the dependent variable equaling a '1', e is the base of natural logarithms, β_0, β_1, ..., β_n are the regression coefficients and x_1, x_2, ..., x_n are the feature variables.

2.7 Decision Tree

Decision Trees are a non-parametric supervised learning method used for classification and regression. The tree structure represents decisions based on feature values. A decision tree splits the data into subsets based on the value of input features. This splitting is repeated recursively, forming a tree structure. Commonly used metrics for determining splits are Gini Impurity and Information Gain (Entropy). These concepts is described in the Equation (6-7).

$$Gini = 1 - \sum_{k=1}^{N} (p_k)^2 \quad (6)$$

$$H(S) = -\sum_{k=1}^{N} p_k log_2(p_k) \quad (7)$$

Where p_k is the proportion of samples that belong to class k in the set S.

2.8 Random Forest

Random Forest is an ensemble learning method for classification and regression, which operates by constructing a multitude of decision trees at training time. The output of the Random Forest is determined by the aggregate of the predictions made by individual trees. Random Forest employs a technique known as Bootstrap Aggregating or Bagging. This method involves creating multiple subsets of the original dataset

with replacement, known as bootstrap samples. Each tree in the Random Forest is trained on one of these bootstrap samples. Mathematically, given a dataset D of size N, a bootstrap sample is a subset D_i (also of size N) sampled with replacement from D. This process is repeated to create as many datasets as there are trees in the forest.

Each decision tree in the Random Forest is constructed using a subset of features chosen at random at each split. If there are M features, a number m (where m << M) is specified such that at each split in the tree, m features are selected at random out of the M and the best split on these m is used to split the node. The value of m is constant during the forest growing.

For regression, the prediction of the Random Forest is given by averaging the predictions of all the individual trees. Mathematically, if $h(x, \Theta_i)$ is the prediction of the i-th tree, then the Random Forest prediction, H(x), for a given input x is described in the Equation (8).

$$H(x) = \frac{1}{N}\sum_{i=1}^{N} h(x_i, \theta_i) \qquad (8)$$

Where N is the number of trees, and Θ_i represents the parameters of the i_{-th} tree. For classification, the output is the class selected by most trees (majority voting). Each tree gives a 'vote' for a class, and the class with the most votes is chosen as the final prediction.

3. Result and Discussion

Figure 1-3 appear to be a set of boxplots comparing the performance metrics of three different machine learning models: Logistic Regression, Decision Tree, and Random Forest. Each boxplot shows the distribution of a specific metric (accuracy, precision, recall, F1 score) across multiple runs of the model.

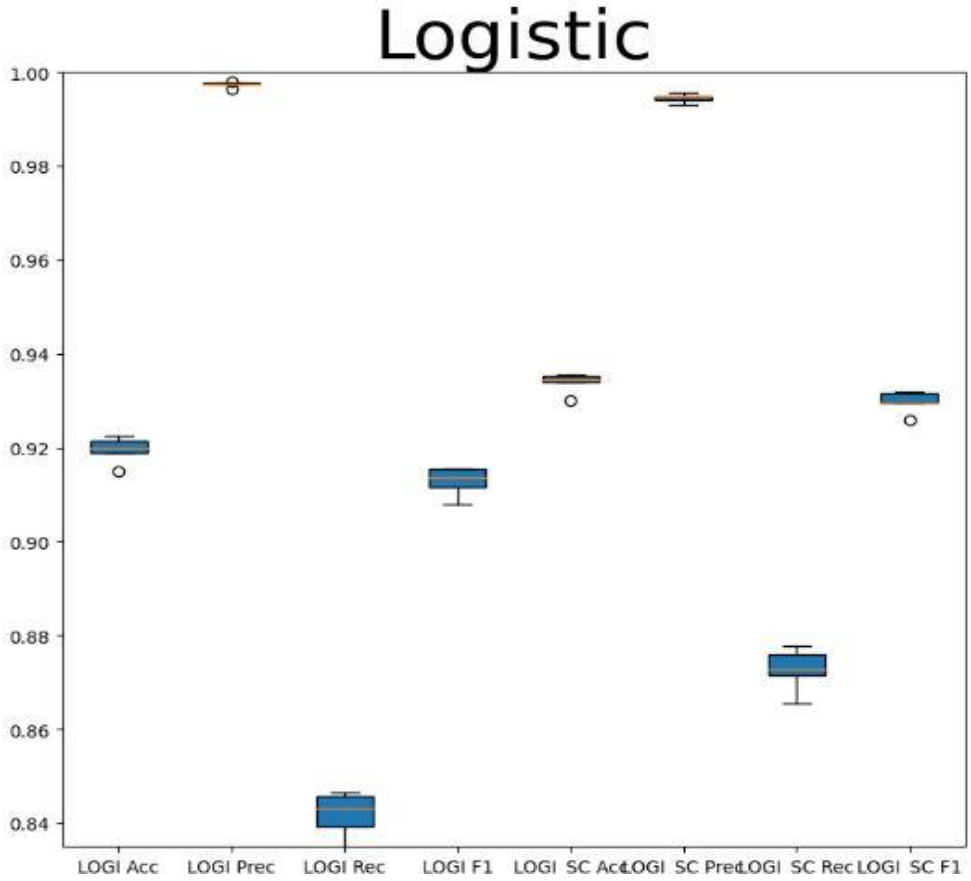

Figure 1. Logistic Regression Result

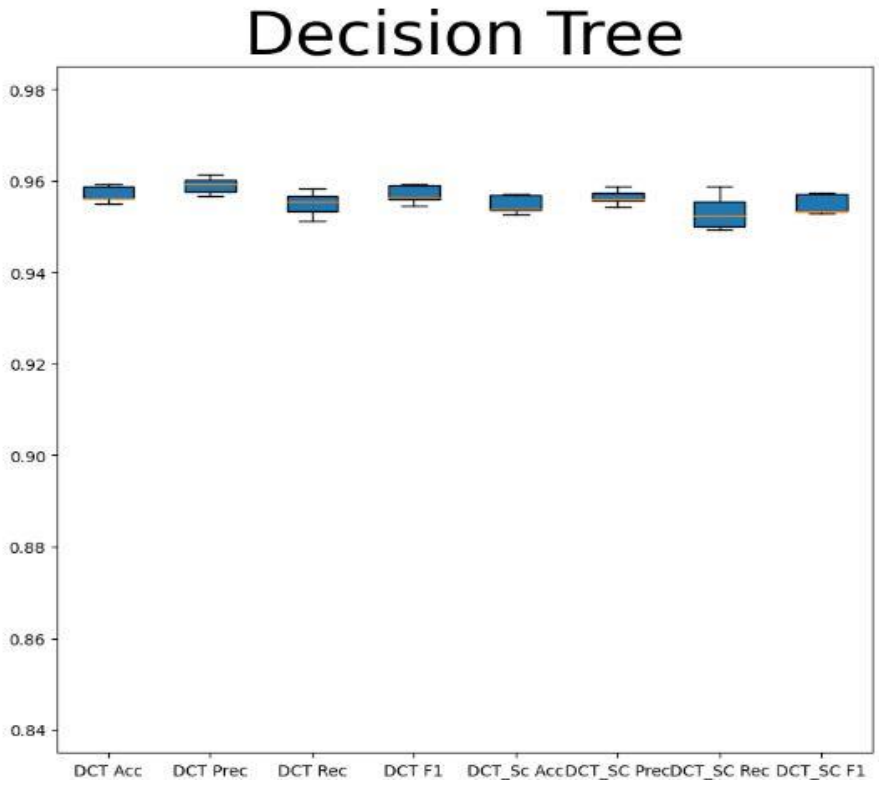

Figure 1. Decision Tree Result

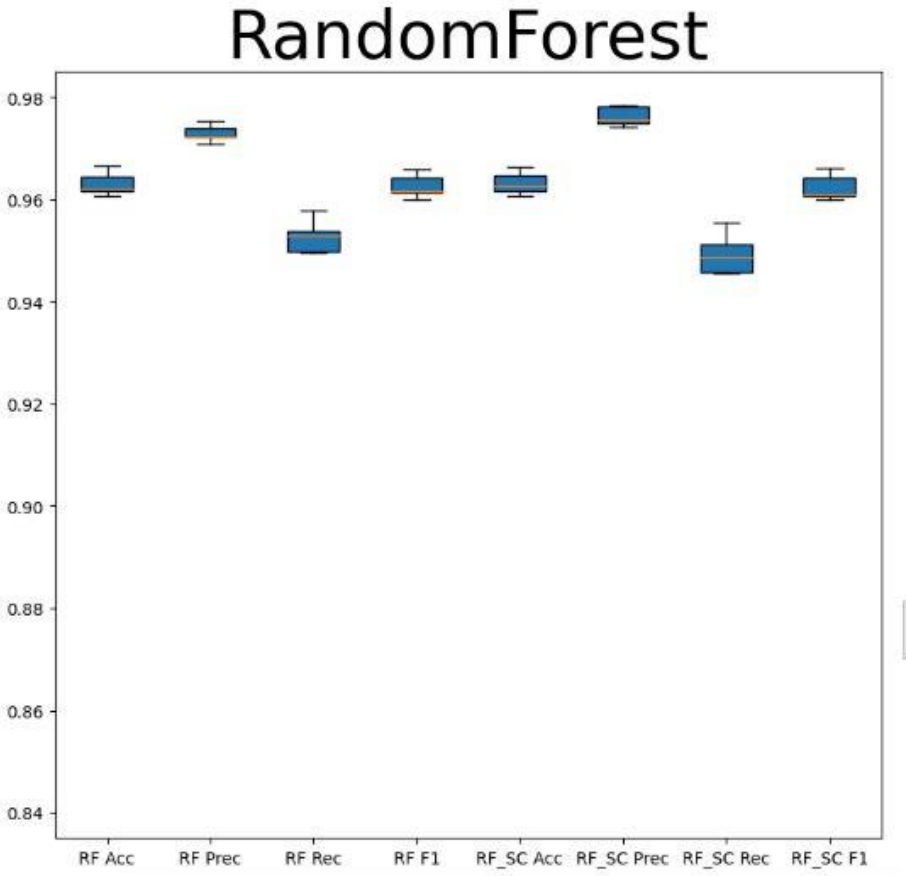

Figure 3. Random Forest Result

Logistic Regression appears as the first set of boxplots. Logistic Regression is a statistical model that, in this context, predicts the probability of a binary outcome. The boxplots indicate that accuracy has a median around 0.92, suggesting that, on average, the model correctly predicts the outcome 92% of the time. However, there is notable variability in the accuracy, as evidenced by outliers. These outliers could indicate cases where the model's performance deviated significantly from the median. Precision and recall both have medians close to 0.90. Precision measures the model's accuracy in predicting positive labels, while recall assesses how well the model captures actual positive instances. The similarity of these medians suggests a balanced trade-off between these two metrics. The F1 score, which harmonizes precision and recall into a single metric, reflects this balance with a similar median. When we consider the scaled variants of these metrics (indicated with an "_SC" suffix), the median values are slightly higher, and the interquartile ranges (IQRs) are tighter. This indicates an improvement in model performance when feature scaling is applied, which is common with Logistic Regression as it can be sensitive to the scale of input variables.

Moving to the Decision Tree model, the boxplots demonstrate a consistently high median performance across all metrics, with accuracy peaking around 0.95. This suggests that the Decision Tree model, which works by partitioning the data into subsets based on feature value thresholds, is highly effective at classification in this case. The high precision and recall indicate that the Decision Tree makes accurate predictions and is good at capturing the majority of the relevant cases. Notably, there is minimal variation in performance with scaling, as the boxplots for the scaled and unscaled metrics are closely aligned. This is indicative of the Decision Tree model's robustness to feature scaling, as it does not rely on distance calculations that can be affected by the scale of the data.

Lastly, the Random Forest model, which is an ensemble of Decision Trees, shows the highest median scores and the least variability among the three models. The Random Forest boxplots show medians around 0.96 for accuracy, 0.95 for precision, and similarly high for recall and F1 score. The tight IQRs across these metrics suggest that the Random Forest model is not only accurate but also consistent in its predictions across different iterations. This is typical of Random Forests, which tend to perform well on a variety of datasets by reducing overfitting through averaging the predictions of multiple trees. Similar to the Decision Tree, the Random Forest model does not show significant changes in performance with feature scaling.

Random Forest appears to be the best performing model among the three, with the highest median values and tightest IQRs for all metrics, which indicates not only high performance but also consistency across runs. Decision Tree shows very high performance as well but with a slight decrease compared to Random Forest. Logistic Regression has the lowest median scores among the three models. However, the performance improves with feature scaling, which indicates that Logistic Regression is more sensitive to the scale of the data. Scaling does not significantly impact the performance of Decision Trees and Random Forests, which might be due to these models' intrinsic handling of feature scales. Outliers in the boxplots suggest that there are runs where the models either significantly overperform or underperform compared to the median, which could be due to the variability in the data or the randomness in the training process. Computation Time of Models can be seen on Figure 4.

	Model	Mean	Std	time_models
0	LOGI acc	0.91959	0.00260	8.64950
1	LOGI prec	0.99736	0.00047	7.91722
2	LOGI rec	0.84157	0.00483	8.30351
3	LOGI f1	0.91286	0.00277	8.13204
4	LOGI_SC acc	0.93387	0.00195	8.64610
5	LOGI_SC prec	0.99445	0.00091	8.44387
6	LOGI_SC rec	0.87272	0.00419	7.90912
7	LOGI_SC f1	0.92961	0.00216	8.52714
8	DCT acc	0.95708	0.00166	2.70445
9	DCT prec	0.95907	0.00175	2.71020
10	DCT rec	0.95498	0.00248	3.68586
11	DCT f1	0.95702	0.00187	3.56941
12	DCT_SC acc	0.95486	0.00184	2.77467
13	DCT_SC prec	0.95644	0.00154	2.76875
14	DCT_SC rec	0.95319	0.00351	2.81237
15	DCT_SC f1	0.95481	0.00201	3.71841
16	RFC acc	0.96309	0.00212	47.29631
17	RFC prec	0.97294	0.00157	48.90547
18	RFC rec	0.95279	0.00299	48.98979
19	RFC f1	0.96257	0.00211	48.16537
20	RFC_SC acc	0.96317	0.00209	46.46499
21	RFC_SC prec	0.97627	0.00172	44.97063
22	RFC_SC rec	0.94930	0.00371	46.47535
23	RFC_SC f1	0.96239	0.00239	44.98098

Figure 4. Computation Time of Models

In terms of computational efficiency, Logistic Regression is the fastest, with times ranging from approximately 6.85 to 8.53 seconds. The Decision Tree model is comparably fast, with times around 2.70 to 3.71 seconds. The Random Forest model, however, takes considerably longer, ranging from about 44.90 to 48.95 seconds, which is expected given that it builds multiple trees and combines their results. While the Random Forest model outperforms the other two in terms of

predictive metrics, it requires significantly more computational time, which might be a consideration in practical applications. Logistic Regression, after scaling, shows improved performance, and its quick computation makes it an attractive model for situations where speed is a critical factor. Decision Trees offer a good balance between speed and performance, with feature scaling not significantly affecting its results. The choice between these models would ultimately depend on the specific requirements of the application, including the acceptable trade-off between accuracy and computational resources. Test harness result can be seen on Figure 5.

```
1 db_results, results = test_harness(models, X_val_bal_sample, y_val_bal_sample, 5)
```

```
model : LOGI - time : 8.6495
model : LOGI - time : 7.91722
model : LOGI - time : 8.30351
model : LOGI - time : 8.13204
model : LOGI_SC - time : 8.6461
model : LOGI_SC - time : 8.44387
model : LOGI_SC - time : 7.90912
model : LOGI_SC - time : 8.52714
model : DCT - time : 2.70445
model : DCT - time : 2.7102
model : DCT - time : 3.68586
model : DCT - time : 3.56941
model : DCT_SC - time : 2.77467
model : DCT_SC - time : 2.76875
model : DCT_SC - time : 2.81237
model : DCT_SC - time : 3.71841
model : RFC - time : 47.29631
model : RFC - time : 48.90547
model : RFC - time : 48.98979
model : RFC - time : 48.16537
model : RFC_SC - time : 46.46499
model : RFC_SC - time : 44.97063
model : RFC_SC - time : 46.47535
model : RFC_SC - time : 44.98098
```

Figure 5. Test Harness

As presented in the Figure 5, The logs indicate multiple runs of a Logistic Regression model, with and without feature scaling (denoted as LOGI_SC). The times recorded for these runs show that the model takes, on average, about 8 seconds to complete the test harness function. Logistic Regression is a foundational machine learning algorithm that models the probability of a binary outcome. It is generally favored for its simplicity and efficiency, especially in cases where the relationship between the independent variables and the binary outcome is approximately linear. The scaled version (LOGI_SC) implies that the data has been standardized or normalized to improve the model's performance, which can be particularly beneficial for Logistic Regression as it relies on the optimization of a loss function that can converge faster when features are on the same scale. Decision Trees are recorded next, also both in scaled (DCT_SC) and unscaled forms. These models are significantly faster, completing the test harness in roughly 2.7 seconds. This efficiency stems from the Decision Tree's flowchart-like structure, where binary decisions are made at each node, leading to a final classification at the leaves. Decision Trees are highly interpretable and do not require feature scaling, which is consistent with the similar execution times observed for DCT and DCT_SC. Finally, the Random Forest models, which are ensembles of Decision Trees, show a much higher execution time, averaging around 48 seconds. This substantial increase in time is expected due to the complexity of Random Forest, which builds multiple Decision Trees on various sub-samples of the dataset and averages their predictions. The nature of this algorithm makes it robust to overfitting and generally more accurate than a single Decision Tree, at the cost of increased computational complexity. Similar to Decision Trees, Random Forests do not inherently benefit from feature scaling (RFC_SC), as evidenced by the consistent execution times regardless of scaling.

In terms of the test harness function itself, the usage of cross-validation (CV) with five folds suggests a robust evaluation methodology. In CV, the dataset is split into five parts, and the model is trained and tested five times, with each part being used as the test set once. This method provides a thorough assessment of the model's performance and generalizability to new data. From these execution times, one can infer that while Logistic Regression and Decision Trees are quicker to train and evaluate, Random Forests take a considerably longer time. This trade-off between time and predictive

performance is a common consideration in machine learning. Practitioners must decide whether the improvement in prediction accuracy with Random Forest is worth the additional computation time, which may be a critical factor in real-time applications or when working with very large datasets. The information from the test_harness function is valuable for understanding not only the performance of the models but also the computational demands they place on the system. Such insights are crucial when it comes to selecting the right model for deployment in production environments, where both accuracy and efficiency need to be balanced according to the application's requirements.

4. Conclusion

Our findings reveal a nuanced landscape of model efficacy and computational efficiency. The Logistic Regression model demonstrated admirable predictive performance, particularly when feature scaling was applied, suggesting its utility in scenarios where model interpretability and operational speed are paramount. The Decision Tree model offered a compelling balance between speed and performance, reinforcing its reputation as a versatile and interpretable classifier. However, it was the Random Forest model that emerged as the superior performer in terms of accuracy, precision, recall, and F1 score, albeit with significantly higher computational demands. The scaled versions of these models (denoted with "_SC"), particularly for Logistic Regression, hinted at the importance of feature normalization in enhancing model predictions. Notably, such scaling did not markedly affect the tree-based models, underscoring their inherent robustness to feature magnitude variations. Reflecting on our methodological approach, the use of cross-validation provided a comprehensive understanding of model generalizability, while the Synthetic Minority Over-sampling Technique (SMOTE) addressed the common issue of class imbalance in cybersecurity datasets. Despite the strengths of our research, we acknowledge certain limitations. The scope of computational resources and the potential for model tuning were not exhaustively explored, which could yield further improvements in model performance. Moreover, the dynamic and evolving nature of DDoS attack patterns necessitates ongoing model adaptation and validation. For future work, we recommend the exploration of hybrid models and deep learning architectures, which may uncover new dimensions of predictive accuracy.

References

[1] Abbasi, M., Shahraki, A., & Taherkordi, A. (2021). Deep learning for network traffic monitoring and analysis (NTMA): A survey. Computer Communications, 170, 19–41. https://doi.org/10.1016/j.comcom.2021.01.021

[2] Ali, T. E., Chong, Y.-W., & Manickam, S. (2023). Machine Learning Techniques to Detect a DDoS Attack in SDN: A Systematic Review. Applied Sciences, 13(5), 3183. https://doi.org/10.3390/app13053183

[3] Bhatia, S., Behal, S., & Ahmed, I. (2018). Distributed denial of service attacks and defense mechanisms: current landscape and future directions. Versatile Cybersecurity, 55–97. https://doi.org/10.1007/978-3-319-97643-3_3

[4] Bhattacharyya, D. K., & Kalita, J. K. (2013). Network anomaly detection: A machine learning perspective. Crc Press. https://doi.org/10.1201/b15088

[5] Chalapathy, R., & Chawla, S. (2019). Deep learning for anomaly detection: A survey. ArXiv Preprint ArXiv:1901.03407. https://doi.org/10.48550/arXiv.1901.03407

[6] Elsayed, M. S., Le-Khac, N.-A., Dev, S., & Jurcut, A. D. (2020). Ddosnet: A deep-learning model for detecting network attacks. 2020 IEEE 21st International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM), 391–396. https://doi.org/10.1109/WoWMoM49955.2020.00072

[7] Iftikhar, A., Qureshi, K. N., Shiraz, M., & Albahli, S. (2023). Security, trust and privacy risks, responses, and solutions for high-speed smart cities networks: A systematic literature review. Journal of King Saud University-Computer and Information Sciences, 101788. https://doi.org/10.1016/j.jksuci.2023.101788

[8] Karatas, G., Demir, O., & Sahingoz, O. K. (2020). Increasing the performance of machine learning-based IDSs on an imbalanced and up-to-date dataset. IEEE Access, 8, 32150–32162. https://doi.org/10.1109/ACCESS.2020.2973219

[9] Kwon, D., Kim, H., Kim, J., Suh, S. C., Kim, I., & Kim, K. J. (2019). A survey of deep learning-based network anomaly detection. Cluster Computing, 22, 949–961. https://doi.org/10.1007/s10586-017-1117-8

[10] Lohachab, A., & Karambir, B. (2018). Critical analysis of DDoS—An emerging security threat over IoT networks. Journal of Communications and Information Networks, 3, 57–78. https://doi.org/10.1007/s41650-018-0022-5

[11] McIntosh, T., Liu, T., Susnjak, T., Alavizadeh, H., Ng, A., Nowrozy, R., & Watters, P. (2023). Harnessing GPT-4 for generation of cybersecurity GRC policies: A focus on ransomware attack mitigation. Computers & Security, 134, 103424. https://doi.org/10.1016/j.cose.2023.103424

[12] Mittal, M., Kumar, K., & Behal, S. (2023). Deep learning approaches for detecting DDoS attacks: A systematic review. Soft Computing, 27(18), 13039–13075. https://doi.org/10.1007/s00500-021-06608-1

[13] Osei-Kyei, R., Tam, V., Ma, M., & Mashiri, F. (2021). Critical review of the threats affecting the building of critical infrastructure resilience. International Journal of Disaster Risk Reduction, 60, 102316. https://doi.org/10.1016/j.ijdrr.2021.102316

[14] Popoola, S. I., Adebisi, B., Ande, R., Hammoudeh, M., Anoh, K., & Atayero, A. A. (2021). smote-drnn: A deep learning algorithm for botnet detection in the internet-of-things networks. Sensors, 21(9), 2985. https://doi.org/10.3390/s21092985

[15] Qazi, N., & Raza, K. (2012). Effect of feature selection, SMOTE and under sampling on class imbalance classification. 2012 UKSim 14th International Conference on Computer Modelling and Simulation, 145–150. https://doi.org/10.1109/UKSim.2012.116

[16] Rudd, E. M., Rozsa, A., Günther, M., & Boult, T. E. (2016). A survey of stealth malware attacks, mitigation measures, and steps toward autonomous open world solutions. IEEE Communications Surveys & Tutorials, 19(2), 1145–1172. https://doi.org/10.1109/COMST.2016.2636078

[17] Salim, M. M., Rathore, S., & Park, J. H. (2020). Distributed denial of service attacks and its defenses in IoT: a survey. The Journal of Supercomputing, 76, 5320–5363. https://doi.org/10.1007/s11227-019-02945-z

[18] Sengupta, S., Basak, S., Saikia, P., Paul, S., Tsalavoutis, V., Atiah, F., Peters, A. (2020). A review of deep learning with special emphasis on architectures, applications and recent trends. Knowledge-Based Systems, 194, 105596. https://doi.org/10.1016/j.knosys.2020.105596

[19] Shafin, S. S., Prottoy, S. A., Abbas, S., Hakim, S. Bin, Chowdhury, A., & Rashid, M. M. (2021). Distributed denial of service attack detection using machine learning and class oversampling. Applied Intelligence and Informatics: First International Conference, AII 2021, Nottingham, UK, July 30--31, 2021, Proceedings 1, 247–259. https://doi.org/10.1007/978-3-030-82269-9_19

[20] Srivastava, A., Parmar, V., Patel, S., & Chaturvedi, A. (2023). Adaptive Cyber Defense: Leveraging Neuromorphic Computing for Advanced Threat Detection and Response. 2023 International Conference on Sustainable Computing and Smart Systems (ICSCSS), 1557–1562. https://doi.org/10.1109/ICSCSS57650.2023.10169393