Information, Computing and Intelligent Systems, 2025, No. 7, 72 – 82

UDC 004.8 : 004.94 https://doi.org/10.20535/2786-8729.7.2025.334076

DDOS ATTACK DETECTION WITH DATA IMPERFECTIONS USING MACHINE LEARNING ALGORITHMS

Artem Dremov
https://orcid.org/0009-0005-7214-9458

Artem Volokyta*
https://orcid.org/0000-0001-9069-5544

National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine

*Corresponding author: artem.volokita@kpi.ua

The issue of DDoS attacks remains a prevalent one even in recent years. Modern environment is highly dynamic and is characterized by a large amount of traffic flow. Existing research covers several models, techniques and approaches to detecting DDoS traffic, which aim to optimize the detection in controlled datasets. However, unintentional noise or data corruption may lower the efficacy of such methods. As such, determining most effective ways to detect DDoS traffic in conditions of data imperfections is necessary for reliable network performance.

Therefore, the object of this research Is the usage of machine learning algorithms for detection of incoming DDoS attacks. The purpose of this research is to determine the performance of ways to detect incoming DDoS attacks with machine learning algorithms based on detection accuracy, while simulating imperfect data conditions. The study also examines the impact of class rebalancing on modified data.

To achieve the aim of this research a variety of machine learning algorithms were implemented and tested on a `CIC-DDoS2019` dataset. The data is modified by removing values and introducing noise, tested, the classes are resampled and the dataset is tested again. The goal is to achieve over 90% accuracy in a classification task of the type of DDoS attack and to determine how much the changes affect the performance of the algorithms.

The results of the testing indicated that several solutions reach the target mark and changes to the dataset in realistic conditions do not significantly affect the final result. However, all models tested show a decrease in accuracy compared to unmodified data with more complex models showing higher resilience (smaller decrease in accuracy). In addition, resampling of the data shows comparable decrease in accuracy of the models with more complex models being affected less.

The results of this study may be used in development of an algorithm of repairing the corrupted data or development of models more resistant to such data changes. Additionally, the results of this study may be used when considering models for practical implementations of a DDoS traffic classification system.

Keywords: machine learning, DDoS, network security, network traffic analysis, data resampling.

1. Introduction

Distributed Denial of Service (DDoS) attacks remain a persistent threat to network infrastructure, disrupting service availability and causing significant economic and operational damage. Classifying incoming traffic as a possible DDoS attack quickly and accurately is necessary to mitigate the effect of the attack while minimizing the effect on normal operations. This challenge remains difficult in part due to the distributed nature of the attack, making it necessary to evaluate a large number of incoming traffic flows for each attack [1, 2].

Machine Learning (ML) algorithms have been widely explored for detecting such attacks due to their ability to learn complex traffic patterns and generalize to unseen data. However, these algorithms have been tested on curated and pre-processed or simulated datasets. In realistic conditions, some of the data may be corrupted and considered missing due to imperfections in the packet capture stage, or noise from large incoming traffic flow may be introduced. In addition, due to privacy, security, and operational constraints, publicly available datasets often feature class imbalance that may introduce bias to the available data, affecting many ML algorithms [3].

Therefore, ML for DDoS detection with imperfect data conditions and handling class imbalance is a promising area of research.

2. Literature review and problem statement

An article presented in [4] describes specific ways missing and noisy data may occur during data collection, and the impact it has on the performance of ML algorithms. It also describes the impact of class imbalance on algorithm performance. In addition, it gives a high-level overview of different data processing techniques, such as data cleansing, feature selection, etc., which also have an impact on performance of algorithms.

A number of studies were conducted over the years on the application of ML algorithms in the problem of DDoS detection. One study [5] gives an overview of datasets and ML algorithms used in the studies of DDoS detection problem. In addition, it describes the issues often overlooked in literature, such as class imbalance, categorical classification of DDoS attacks instead of binary, usage of offline datasets and other issues. A different study [6] shows the usage of different algorithms on different datasets, but mostly focuses on setting up an experiment in an experimental testbed to replicate traffic flows used in popular datasets, and evaluates the performance of models under realistic time constraints. However, both of these studies highlight the usage of highly curated datasets that do not feature data imperfections, or these imperfections are removed during data processing.

A study presented in [7] gives a problem overview in the field of IoT. However, of note is the review of datasets used and specific conclusions regarding class imbalance in several datasets, possibility of data manipulation that may affect the dataset, as well as lack of frequently updated state-of-the-art datasets. It also describes potential future trends, such as increasing frequency and difficulty of identification of DDoS attacks.

Yet another study [8] presents an overview of common datasets and methods used for the problem. In particular it describes the challenges of access to realistic datasets and the detection of high-rate and low-rate DDoS attacks.

A study presented in [9] provides tests of several algorithms not described previously, as well as providing additional insight on feature selection and data processing involved in handling this issue.

An application of DNN (Deep Neural Network) to the issue is given in a study [10]. In this study a combination of CNN (Convolutional Neural Network) and LSTM (Long Short-Term Memory) is used to create a hybrid DNN. Alongside this, the study describes feature selection process on popular datasets. The study achieved accuracy of 99% in testing.

A genetic algorithm for generation of optimal DNN is described in the study [11]. The GA (Genetic Algorithm) described is used to create populations of DNN with different layers, order of layers, parameters of layers – such as activation functions – used, and different loss functions used. The models were then tested on different datasets. The results described in the paper show that `MaxPool2D` and *Residual layers* seem to perform best in DNN for this task.

Research presented in [12] presents a method of using oversampling with `SMOTE` to improve class representation of minority class for DDoS detection task in IoT. Using a neural network, the study achieved close to 100% accuracy in the task.

Based on the analysis of the studies described above, the issues of using missing and noisy data to simulate realistic data collection conditions requires further research. In addition to this, combining different oversampling and undersampling techniques for handling class imbalance remains insufficiently explored. A further investigation into performance of different ML algorithms in described above conditions is necessary.

3. The aim and objectives of the study

The object of the study presented in this article is the detection and classification of DDoS traffic using ML algorithms in conditions where data is noisy or data corruption has occurred.

The purpose of this study is the implementation and testing of several ML algorithms in the task of classification of DDoS traffic. The algorithms are tested with inclusion of noise and missing values in the dataset. The dataset is resampled to balance the classes for the purpose of testing the performance of algorithms used in case of balanced class distribution.

To achieve this goal the following tasks are set:

– modify the DDoS traffic classification dataset by introducing noise values and missing values with the aim of introducing a realistic amount of data inconsistency;

– modify the DDoS traffic classification dataset by using oversampling and undersampling techniques in order to improve class distribution;

– implement DDoS traffic classification models using several ML algorithms. Test the performance of these models on an unmodified and modified datasets, compare the performance of the models. Determine the effect of data modifications on the problem of classifying DDoS traffic and determine which models are most resilient to specified data changes and how class rebalance affects the modified data. The performance of the models is primarily indicated by the accuracy of the attack or benign traffic detection.

4. The study materials and methods for ML algorithms for DDoS detection with simulated data imperfections

The research methods of the article are the search and analysis of theoretical material on possible solutions to problem of DDoS detection with ML algorithms. In addition to this, a solution to address class imbalance and introduction of data imperfections into the detection study is presented. Several testing scenarios were developed to ensure ML algorithms and models are tested under different conditions. Methods of modelling, experiment, observation, measurement, analogy and testing are also used to record, analyse and compare the performance of different models in outlined scenarios.

For the study `CICDDoS2019` dataset was used [13]. This is a popular dataset for evaluation DDoS detection models. It features 125000 training examples and is classified into 7 attack types and 1 class for benign traffic. The dataset contains a variety of data from simulation using `CICFlowMeter` traffic generation and analysis tool [14]. The dataset was created to reflect at the time latest cases of DDoS attacks.

4.1. ML techniques for DDoS traffic detection and experiment outline

An SGD (Stochastic Gradient Descent) based model was used for this problem [15–17]. In particular, an SGD based linear SVM (State Vector Machine) [18], linear regression [19] and OvR (One-vs-Rest) perceptron model [20]. The SGD itself represent an optimisation method for training common linear models described above.

$$f(x) = w^T x + b, \tag{1}$$

where w – weight vector, x – training vector and b – bias.

For multiclass classification we also have to introduce a *softmax* activation function in order to calculate class probabilities.

$$softmax(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}, \tag{2}$$

where z_i – input score (logit) for class i, K – total number of classes.

The result is a probability for each class. We can then determine the most probably class by applying an *argmax* over the *softmax* function to get the highest class probability.

The gradient descent process of SGD can be defined as:

$$w \leftarrow w - \eta \left[a\frac{\partial R(w)}{\partial w} + \frac{\partial L(w^T x_i + b, y_i)}{\partial w} \right], \tag{3}$$

where w – weights vector, η – learning rate, R – regularization term, L – loss function.

Additionally, a number of ensemble techniques was used in our testing. Ensemble techniques rely on construction series of decision trees that aim to correct the issues of previous trees, usually relying on gradient descent for adjusting weights, as described above.

The decision trees themselves are built using the following logic:

$$R_{left}(j, s) = \{x \in R : x_j \leq s\}, \tag{4}$$

$$R_{right}(j, s) = \{x \in R : x_j > s\}, \tag{5}$$

where j – feature, s – splitting point.

Random forest approach constructs a number of trees and uses a majority vote strategy to determine the output [21].

Gradient boosting techniques rely on constructing a strong model from a sequence of weaker estimators and decreasing the loss for the entire model [22].

$$\hat{y} = F_M(x_i) = \sum_{m=1}^{M} h_m(x_i), \tag{6}$$

where h_m – weak estimators (individual decision trees).

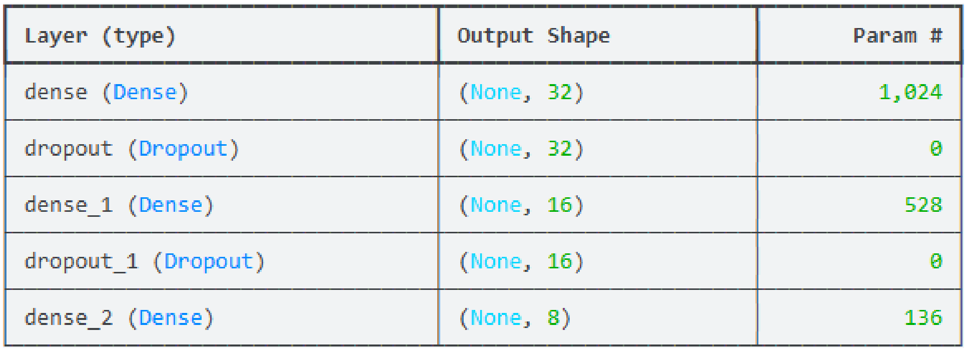
XGB (Extreme Gradient Boosting) follows the same principle, but uses second-order Taylor expansion of the loss and introduces regularization to avoid overfitting [22].

Catboost implements several improvements compared to traditional gradient boosting models.

It uses ordered target encoding, relying on data from previous rows to calculate the target. Additionally, it uses ordered boosting, relying on several separate models for fitting and calculating gradients, so that fitting data and gradient calculation data do not overlap.

DNN models are formed by combining several layers of, usually, densely connected neurons forming a layer artificial neural network. Such network tend to exhibit improved ability to learn complex data patterns compared to regular ANN [23].

For DNN model, a simple 3 layer model is used (excluding dropout layers). The first layer features 32 neurons and *ReLU* activation, second layer features 16 neurons and relu activation. Final layer serves as an output layer and uses *softmax* activation. Fig. 1 shows the structure of the described DNN.



Layer (type)	Output Shape	Param #
dense (Dense)	(None, 32)	1,024
dropout (Dropout)	(None, 32)	0
dense_1 (Dense)	(None, 16)	528
dropout_1 (Dropout)	(None, 16)	0
dense_2 (Dense)	(None, 8)	136

Fig. 1. DNN model summary

In order to best apply these algorithms, a following strategy is outlined:

1. Perform data cleaning and feature selection. For this we calculate highly correlating features and drop features that show correlation of more than 0.8.
2. Split the dataset into training and testing at 80/20 ratio.
3. Perform model testing on the result training samples for baseline model performance.
4. Introduce missing values and noise to the training data. Impute missing values using a mean of all values in the column. Train the models on noisy data and validate on clean test data.
5. Introduce oversampling with `ADASYN` and undersampling with Tomek links to the noisy data. Train and test models on resampled data and clean test data.

For model parameters optimisation `Optuna` library is used [24]. After calculating optimal parameters over 5 studies in baseline dataset, the parameters are saved for use in noisy and resampled datasets.

4.2. Noise generation and resampling strategy

For generating noise several data archetypes were determined.

1. Binary data indicating a specific traffic flag. For this data the value entry would be flipped to the opposite values with 2% probability (0 to 1, 1 to 0).
2. For integer data indicating data such as packet count, flow duration, etc, the strategy is to introduce a up to 10% deviation from the actual value for each data entry.
3. For float data the strategy is to introduce a up to 10% positive deviation from the actual value for each data entry.
4. After noise is introduced, 1% of values is randomly determined as missing. These values are imputed with mean of all values for this feature before feeding to the ML models since most models do not support missing values.

For oversampling `ADASYN` framework is used [25, 26]. The general oversampling algorithm used is described below:

$$x_{new} = x_i + \lambda \times (x_{zi} - x_i), \tag{7}$$

where λ – a random number [0, 1].

Compared to `SMOTE` – another oversampling method – `ADASYN` also uses a density distribution function to generate more minority classes samples in dense majority class areas.

$$r_i = \frac{\Delta_i}{K}, \tag{8}$$

where Δ_i – the number of examples of majority class in K neighbours of x_i.

For undersampling Tomek's links algorithm was used [27]. The principle is to calculate links between samples of 2 classes, based on the distance between them. If such links exist, they are removed as they are considered noisy and difficult to classify.

$$d(x, y) < d(x, z) andd(x, y) < d(y, z), \tag{9}$$

where x, y are samples of different classes and z is any other sample.

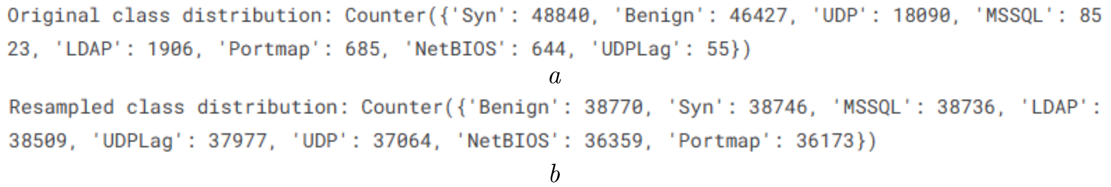
Depending on the handling strategy, either entire link or only majority class may be removed.

Combining oversampling and undersampling techniques allows us to balance out the classes' distribution between minority and majority classes as shown in the Fig. 2.

As can be seen; after resampling the proportion of lowest represented class `NetBIOS` has increased to match the proportion of other classes. Similar effect is seen on other minority classes. At the same time majority classes, such as `Syn` have their proportionality in the dataset reduced.

5. Results of the study of ML algorithms for DDoS detection with simulated data imperfections

As described before, three sets of experiments were conducted with the ML algorithms. First experiment was conducted after feature selection to establish a baseline of performance.



```
Original class distribution: Counter({'Syn': 48840, 'Benign': 46427, 'UDP': 18090, 'MSSQL': 85
23, 'LDAP': 1906, 'Portmap': 685, 'NetBIOS': 644, 'UDPLag': 55})
```

a

```
Resampled class distribution: Counter({'Benign': 38770, 'Syn': 38746, 'MSSQL': 38736, 'LDAP':
38509, 'UDPLag': 37977, 'UDP': 37064, 'NetBIOS': 36359, 'Portmap': 36173})
```

b

Fig. 2. Resampling results: *a* – class distribution before resampling; *b* – class distribution after resampling

Performance of each ML model was tested and recorded.

Second experiment was conducted after introducing noise and missing data entries into the dataset. Close to realistic rate of noise and data corruption was selected in order to test the models in close to realistic conditions, without further testing the robustness of models.

Third experiment was conducted after oversampling and undersampling the noisy data from the second experiment. This experiment was designed to test the effect of class balancing on the training results.

In each experiment 4 metrics were used to track model performance. *Accuracy* score is used to determine the overall accuracy of predictions.

$$A = \frac{1}{n}\sum_{i=1}^{n} 1(\hat{y}_i = y_i), \tag{10}$$

where $\hat{y}_i$ – predicted class label, y_i – actual class label.

Precision weighted by class – proportion of correct positive predictions of a class.

$$\mathrm{P} = \frac{\sum_{c=1}^{C} n_c \times \frac{TP_c}{TP_c+FP_c}}{\sum_{c=1}^{C} n_c}, \tag{11}$$

where n_c – number of true samples in class C, TP_c – true positives for class C, FP_c – false positives for class C.

Recall weighted by class – proportion of actual class samples correctly identified

$$R = \frac{\sum_{c=1}^{C} n_c \times \frac{TP_c}{TP_c+FN_c}}{\sum_{c=1}^{C} n_c}, \tag{12}$$

where FN_c – false negatives for class C.

F1 score weighted by class – mean of *precision* and *recall.*

$$\mathrm{P} = \frac{\sum_{c=1}^{C} n_c \times \frac{TP_c}{TP_c+FP_c}}{\sum_{c=1}^{C} n_c}. \tag{13}$$

The results of experiments are presented in Fig. 3, 4, 5, 6 and 7.

The results presented above show above 90% accuracy for most models in different scenarios.

6. Analysis of obtained results

In the process of conducting this study, several steps were made and experiments conducted. The results of these experiments are presented in this section:

1. Five ML models were tested and their performance measured on an unmodified dataset. A baseline of performance was established.

2. Data corruption and data noise were introduced. The performance of ML models mentioned before was recorded for the new dataset and the impact of data modifications was analysed.

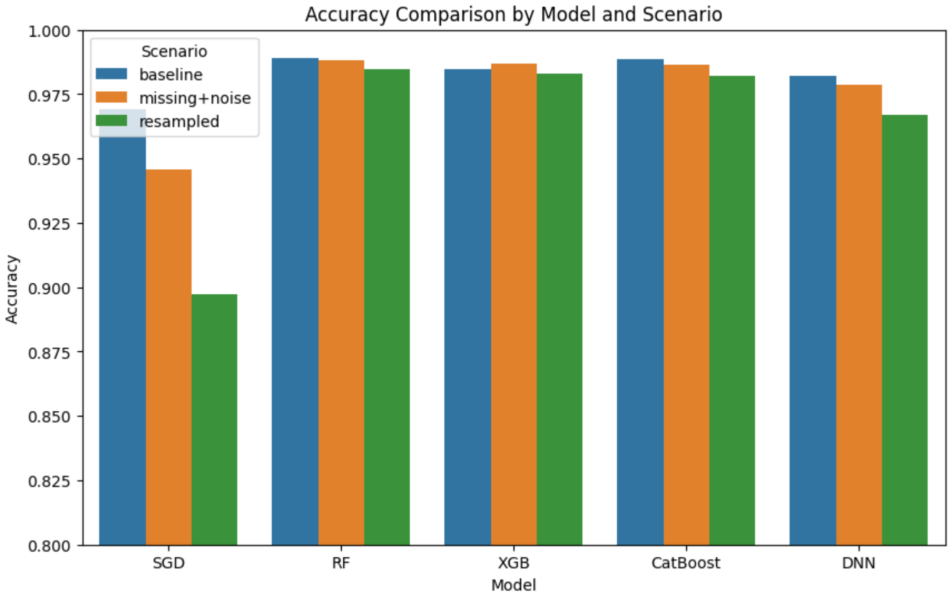


Fig. 3. Accuracy comparison of different scenarios

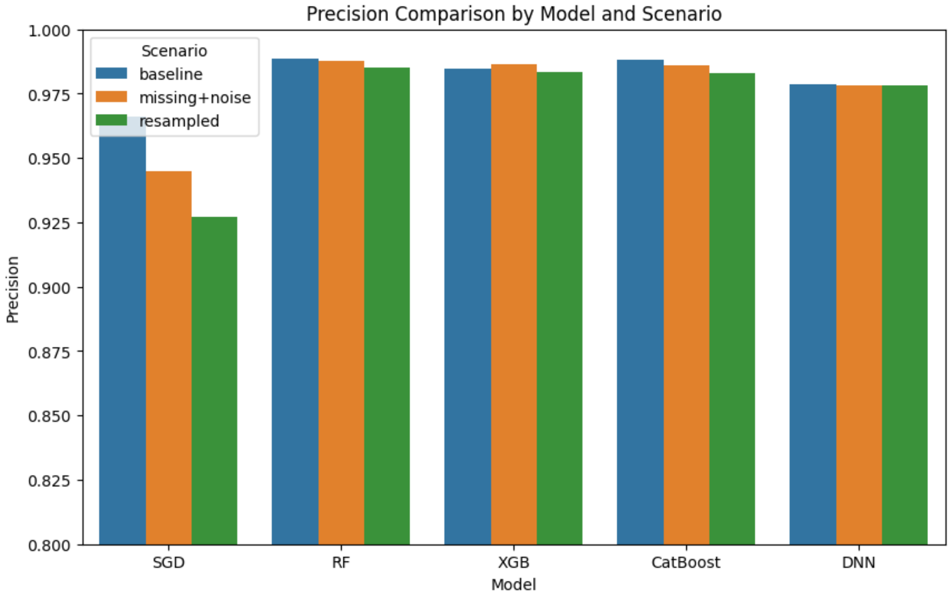


Fig. 4. Precision comparison of different scenarios

3. Data resampling utilizing ADASYN for oversampling and Tomek link for undersampling was performed and the impact of resampling on ML models was analyzed.

4. A comparative analysis of the ML models was done and the most well performing models were identified. An analysis into model performance was made and possible causes for underperforming models were presented.

Obtained results indicate, that by using a high volume of data collected from the traffic flow simulation, it is possible in perfect condition to solve this problem by applying a simple regression function, however when introducing noise and data imperfections, the linear model underperforms, as can be seen in Fig. 3.

All variants of ensemble techniques used in this study have shown great resilience to data modifications and perform consistently well in all scenarios. The DNN model, however, underperform in comparison in most scenarios, as can be seen in Fig. 3. This may be due to simple

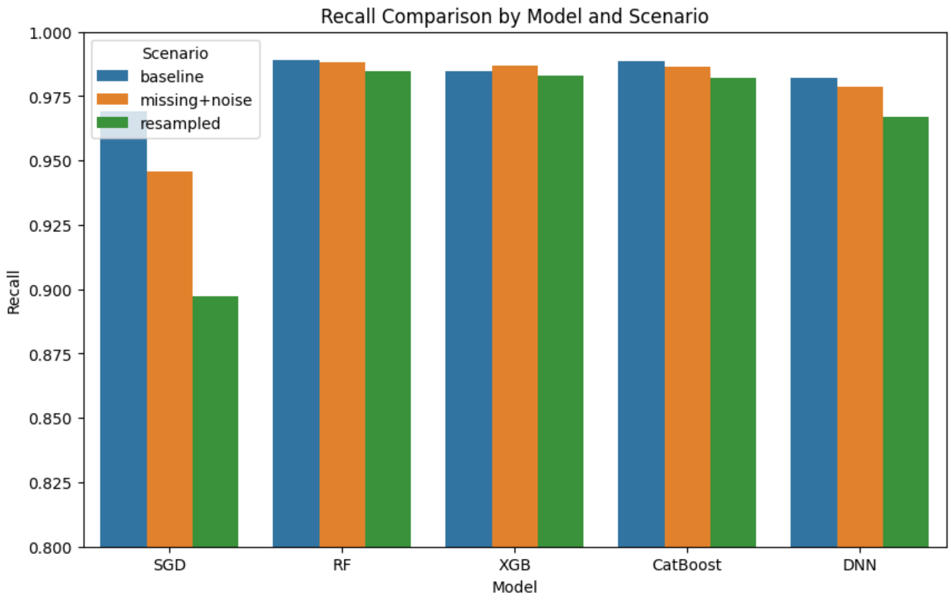


Fig. 5. Recall comparison of different scenarios

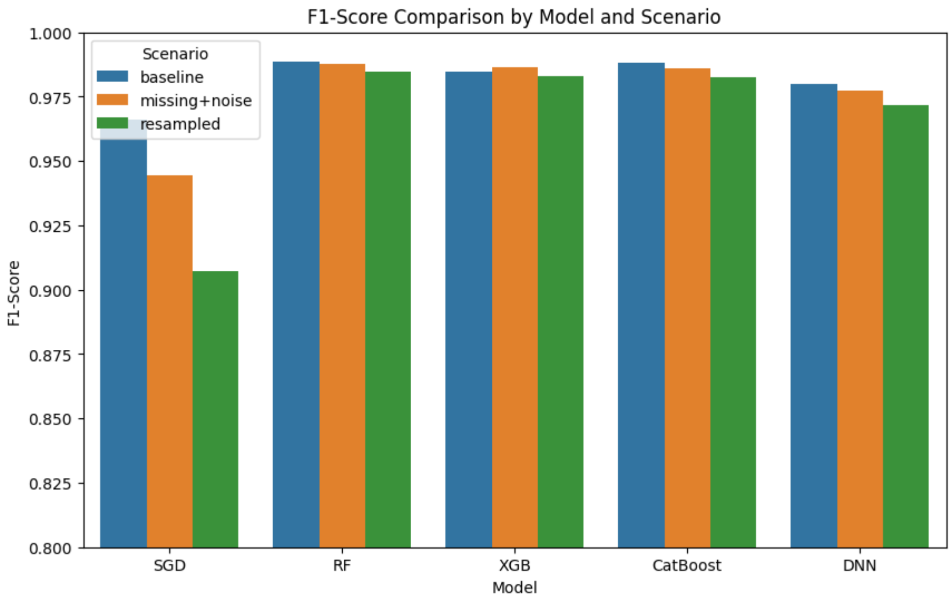


Fig. 6. F1 score comparison of different scenarios

and unsuitable architecture of the DNN.

With all algorithms a trend is seen as shown in Fig. 7, where introducing data resampling leads to lower accuracy scores. The nature of this phenomenon may be due to overfitting, as oversampling techniques in particular tend to create strong clusters of data, which may not always appear in realistic conditions.

DNN model also appears to struggle with *precision*, compared to *recall* and *F1* score, as seen in Fig. 4, 5 and 6. However such behavior is not changed after class resampling, which generally improves this metric. A possible cause of this may be that the features selected for classification have low separability or such issue is introduced at resampling stage.

Another peculiar result comes from DNN performance on resampled data. In training the DNN reached only 80% *accuracy*, however in testing, it reached 97% *accuracy*. This indicated possible underfitting on training data. Perhaps due to complexity introduced by highly dense clusters of

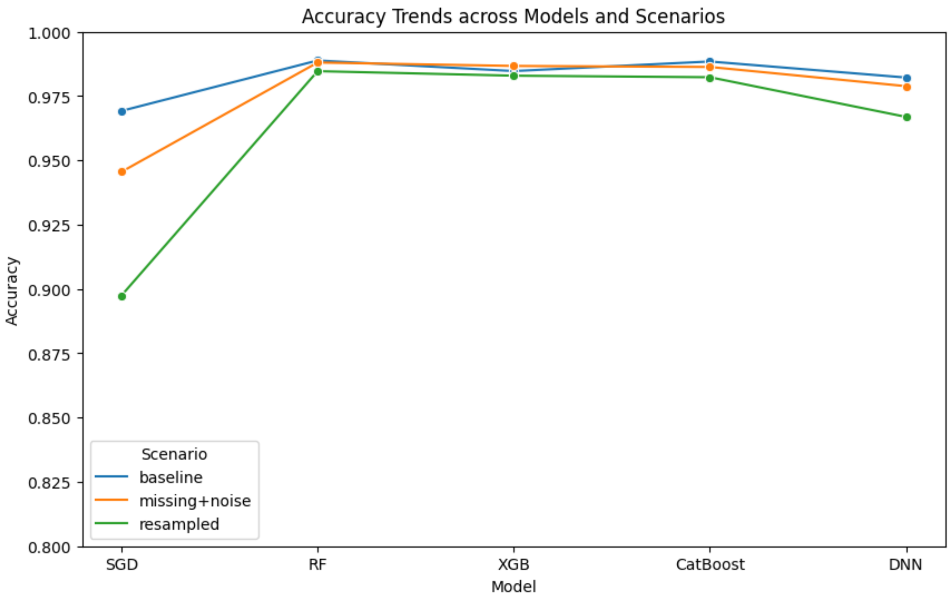


Fig. 7. Accuracy trend

multi-class data.

Conclusions

From this study, the following goals were achieved:

1. A data modification process was introduced and tested on `CICDDoS19` dataset. This data modification process introduces missing values and data noise to the dataset which achieves a degree of data corruption that may be considered realistic for practical conditions of DDoS attack detection.

2. An undersampling method with Tomek links and oversampling method with `ADASYN` were used to solve class imbalance in the original dataset. The results of this resampling show equal distribution of classes on the before mentioned dataset. However, in practice it is determined to decrease DDoS attack classification accuracy in experiments conducted.

3. *Stochastic Gradient Descent*, *Random Forest*, *Extreme Gradient Boosting*, *CatBoost* and *Deep Neural Network* models were implemented to measure classification accuracy on modified and unmodified dataset. The results of the comparative analysis show indicate, that complex models, such as *RF*, *XGB*, *CatBoost*, sufficiently complex DNN, are not affected in a major way by modifications done. These results can be used in further research and design of intrusion detection systems for DDoS attacks, that are data corruption resistant and can operate with major class imbalance.

References

[1] C. Douligeris and A. Mitrokotsa, "DDoS attacks and defense mechanisms: Classification and state-of-the-art," *Computer Networks*, vol. 44, no. 5, pp. 643–666, 2004. https://doi.org/10.1016/j.comnet.2003.10.003

[2] G. Somani, M. S. Gaur, D. Sanghi, M. Conti, and R. Buyya, "DDoS attacks in cloud computing: Issues, taxonomy, and Future Directions," *Computer Communications*, vol. 107, pp. 30–48, 2017. https://doi.org/10.1016/j.comcom.2017.03.010

[3] M. Tahir, A. Abdullah, N. I. Udzir, and K. A. Kasmiran, "A novel approach for handling missing data to enhance network intrusion detection system," *Cyber Security and Applications*, vol. 3, p. 100063, 2025. https://doi.org/10.1016/j.csa.2024.100063

[4] "The impact of data quality on machine learning," Wipro, accessed Aug. 13, 2025. [Online]. Available: https://www.wipro.com/engineering/the-impact-of-data-quality-on-machine-learning/

[5] T. E. Ali, Y.-W. Chong, and S. Manickam, "Machine learning techniques to detect a DDOS attack in SDN: A systematic review," *Applied Sciences*, vol. 13, no. 5, p. 3183, 2023. https://doi.org/10.3390/app13053183

[6] F. S. Lima Filho, F. A. Silveira, A. de Medeiros Brito Junior, G. Vargas-Solar, and L. F. Silveira, "Smart detection: An online approach for DOS/ddos attack detection using machine learning," *Security and Communication Networks*, vol. 2019, pp. 1–15, 2019. https://doi.org/10.1155/2019/1574749

[7] A. A. Alahmadi et al., "DDoS attack detection in IOT-based networks using Machine Learning Models: A survey and research directions," *Electronics*, vol. 12, no. 14, p. 3103, 2023. https://doi.org/10.3390/electronics12143103

[8] A. A. Bahashwan, M. Anbar, S. Manickam, T. A. Al-Amiedy, M. A. Aladaileh, and I. H. Hasbullah, "A systematic literature review on machine learning and deep learning approaches for detecting ddos attacks in software-defined networking," *Sensors*, vol. 23, no. 9, p. 4441, 2023. https://doi.org/10.3390/s23094441

[9] F. L. Becerra-Suarez, I. Fernández-Roman, and M. G. Forero, "Improvement of distributed denial of service attack detection through machine learning and Data Processing," *Mathematics*, vol. 12, no. 9, p. 1294, 2024. https://doi.org/10.3390/math12091294

[10] A. Zainudin, L. A. Ahakonye, R. Akter, D.-S. Kim, and J.-M. Lee, "An efficient hybrid-DNN for ddos detection and classification in software-defined IIOT Networks," *IEEE Internet of Things Journal*, vol. 10, no. 10, pp. 8491–8504, 2023. https://doi.org/10.1109/JIOT.2022.3196942

[11] J. Zhao, M. Xu, Y. Chen, and G. Xu, "A DNN architecture generation method for ddos detection via genetic alogrithm," *Future Internet*, vol. 15, no. 4, p. 122, 2023. https://doi.org/10.3390/fi15040122

[12] Y. N. Soe, P. I. Santosa, and R. Hartanto, "DDoS attack detection based on simple ANN with smote for IOT environment," *2019 Fourth International Conference on Informatics and Computing (ICIC)*, pp. 1–5, 2019. https://doi.org/10.1109/ICIC47613.2019.8985853

[13] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (ddos) attack dataset and taxonomy," *2019 International Carnahan Conference on Security Technology (ICCST)*, pp. 1–8, 2019. https://doi.org/10.1109/CCST.2019.8888419

[14] A. H. Lashkari, "Ahlashkari/CICFlowMeter: CICFlowmeter-v4.0" GitHub. https://doi.org/10.13140/RG.2.2.13827.20003

[15] F. Pedregosa et al., "Scikit-Learn: Machine learning in Python," Journal of Machine Learning Research, accessed Aug. 13, 2025. [Online]. Available: https://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html

[16] L. Bottou, "Stochastic gradient descent tricks," *Lecture Notes in Computer Science*, pp. 421–436, 2012. https://doi.org/10.1007/978-3-642-35289-8_25

[17] C. M. Bishop, Pattern Recognition and Machine Learning. New York: *Springer*, 2016.

[18] C. Cortes and V. Vapnik, "Support-Vector Networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995. https://doi.org/10.1007/BF00994018

[19] D. R. Cox, "The regression analysis of binary sequences," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 20, no. 2, pp. 215–232, 1958. https://doi.org/10.1111/j.2517-6161.1958.tb00292.x

[20] F. Rosenblatt, "The Perceptron: A probabilistic model for information storage and organization in the brain.," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958. https://doi.org/10.1037/h0042519

[21] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001. https://doi.org/10.1023/A:1010933404324

[22] J. H. Friedman, "Stochastic gradient boosting," *Computational Statistics & Data Analysis*, vol. 38, no. 4, pp. 367–378, 2002. https://doi.org/10.1016/S0167-9473(01)00065-2

[23] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015. https://doi.org/10.1038/nature14539

[24] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," *arXiv.org*, accessed Aug. 13, 2025. [Online]. Available: https://arxiv.org/abs/1907.10902

[25] H. He and E. A. Garcia, "Learning from Imbalanced Data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009. https://doi.org/10.1109/TKDE.2008.239

[26] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive Synthetic Sampling Approach for imbalanced learning," *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pp. 1322–1328, 2008. https://doi.org/10.1109/IJCNN.2008.4633969

[27] I. Tomek, "Two modifications of CNN," *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-6, no. 11, pp. 769–772, 1976. https://doi.org/10.1109/TSMC.1976.4309452

УДК 004.8 : 004.94

ВИЯВЛЕННЯ DDOS АТАК ПРИ НЕДОСКОНАЛОСТІ ДАНИХ ЗА ДОПОМОГОЮ АЛГОРИТМІВ МАШИННОГО НАВЧАННЯ

Артем Дремов
https://orcid.org/0009-0005-7214-9458

Артем Волокита
https://orcid.org/0000-0001-9069-5544

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна

Проблема DDoS-атак залишається актуальною навіть в останні роки. Сучасне середовище є дуже динамічним і характеризується великим обсягом трафіку. Існуючі дослідження охоплюють кілька моделей, технік і підходів до виявлення DDoS-трафіку, які спрямовані на оптимізацію виявлення в контрольованих наборах даних. Однак ненавмисний шум або пошкодження даних можуть знизити ефективність таких методів. Таким чином, для надійної роботи мережі необхідно визначити найефективніші способи виявлення DDoS-трафіку в умовах недосконалості даних.

Тому предметом цього дослідження є використання алгоритмів машинного навчання для виявлення вхідних DDoS-атак. Мета цього дослідження — визначити ефективність способів виявлення вхідних DDoS-атак за допомогою алгоритмів машинного навчання спираючись на точність виявлення, моделюючи умови недосконалості даних. У дослідженні також розглядається вплив перебалансування класів на модифіковані дані.

Для досягнення мети цього дослідження було впроваджено та протестовано різноманітні алгоритми машинного навчання на наборі даних `CIC-DDoS2019`. Дані модифікуються шляхом видалення значень та введення шуму, тестуються, класи передискретизуються, а набір даних тестується знову. Мета полягає в досягненні точності понад 90% у завданні класифікації типу DDoS-атаки та визначенні, наскільки зміни впливають на продуктивність алгоритмів.

Результати тестування показали, що кілька рішень досягають цільового показника, а зміни в наборі даних в реалістичних умовах суттєво не впливають на кінцевий результат. Однак усі протестовані моделі демонструють зниження точності порівняно з немодифікованими даними, причому більш складні моделі виявляють вищу стійкість (менше зниження точності). Крім того, передискретизація даних показує порівнянне зниження точності моделей, причому більш складні моделі зазнають меншого впливу.

Результати цього дослідження можуть бути використані для розробки алгоритму відновлення пошкоджених даних або розробки моделей, більш стійких до таких змін даних. Крім того, результати цього дослідження можуть бути використані при розгляді моделей для практичного впровадження системи класифікації DDoS-трафіку.

Ключові слова: машинне навчання, DDoS, мережева безпека, аналіз мережевого трафіку, передискретизація даних.