**Detection of Denial of Service (DOS) Attacks Using Machine Learning Models**

Dr. Mohamed Turkey1, Artificial Intelligence, Artificial Intelligence, Egyptian Russian University Badr City, Cairo, Egypt.

Eng. Elsayed Mohamed1, Artificial Intelligence, Artificial Intelligence, Egyptian Russian University Badr City, Cairo, Egypt.

Eng. Moaz Mohamed1, Artificial Intelligence, Artificial Intelligence, Egyptian Russian University Badr City, Cairo, Egypt.

Eng. Badr Elsayed1, Artificial Intelligence, Artificial Intelligence, Egyptian Russian University Badr City, Cairo, Egypt.

**Abstract:** Denial of Service (DoS) attacks pose a significant threat to cybersecurity, disrupting the availability of services by overwhelming network resources. As these attacks grow in complexity, traditional detection methods struggle to keep pace. This paper explores the application of machine learning models, specifically Random Forest and Decision Tree classifiers, to detect DoS attacks[1], providing a robust and scalable solution. Utilizing the PyCaret framework, we evaluated multiple models, fine-tuned the best performers, and achieved high accuracy in attack detection. Our findings underscore the potential of machine learning in enhancing cyber defense mechanisms, particularly in the face of evolving attack strategies.

**Keywords:** Denial of Service, DOS Attacks, Machine Learning, Cyber Security, Random Forest, Decision Tree, PyCaret

1. **Introduction**

In the ever-evolving landscape of cyber threats, Denial of Service (DoS) attacks remain a persistent and significant challenge. These attacks aim to disrupt the normal operation of targeted systems or networks by flooding them with an overwhelming volume of traffic or requests[7]. The consequences of successful DoS attacks can be severe, ranging from temporary service outages to substantial financial losses and reputational damage.

Traditional DoS detection methods, often based on rule-based systems or signature matching, have become increasingly inadequate in the face of sophisticated and adaptive attack techniques[9]. These methods often struggle to distinguish between legitimate traffic spikes and malicious DoS traffic, leading to either missed detections or false alarms.

Machine learning, a subfield of artificial intelligence, offers a promising alternative for DoS attack detection[3]. Machine learning algorithms can learn patterns and anomalies in network traffic data, enabling them to identify DoS attacks with greater accuracy and adaptability than traditional methods[12]. This paper investigates the application of two machine learning models, Random Forest and Decision Tree, for DoS attack detection. We leverage the PyCaret framework, an open-source machine learning library, to streamline the model development and evaluation process.

In the modern digital era, where connectivity and online presence are ubiquitous, the threat landscape has become increasingly complex and sophisticated[5]. Among the myriad of cyber threats, Denial of Service (DoS) attacks stand out due to their capacity to cripple online services, causing significant disruptions and financial losses. DoS attacks aim to overwhelm a network, service, or application with a flood of illegitimate requests, rendering it inaccessible to legitimate users. As the dependency on internet-based services grows, the urgency to develop robust methods to detect and mitigate these attacks becomes paramount.

Traditional methods of detecting DoS attacks often rely on signature-based detection or rule-based systems[11]. While these approaches can be effective against known threats, they fall short when it comes to identifying new, evolving, or sophisticated attack vectors. This limitation underscores the need for more dynamic and adaptive detection mechanisms.

Machine learning (ML) offers a promising avenue to enhance DoS attack detection. By leveraging ML algorithms, it is possible to analyze vast amounts of network traffic data, identify patterns[1], and distinguish between normal and malicious activities. Unlike traditional methods, ML models can learn from data, adapt to new threats, and improve their detection capabilities over time.

The application of machine learning to DoS attack detection involves several key steps: data collection and preprocessing, feature extraction, model training, and evaluation. Various ML techniques, including supervised learning, unsupervised learning, and deep learning[3], have been explored to tackle this problem. Supervised learning models, such as decision trees, support vector machines, and neural networks, are trained on labeled datasets where attack and non-attack traffic are pre-identified. Unsupervised learning models, like clustering algorithms[2], can identify anomalies in the network traffic that may indicate a DoS attack without prior knowledge of attack patterns.

Moreover, the advent of deep learning has brought new capabilities to the detection of DoS attacks[7]. Deep learning models, with their ability to process large-scale data and capture intricate patterns, have shown significant promise in identifying complex attack behaviors that traditional ML models might miss.

In this research, we aim to explore and evaluate various machine learning models for detecting DoS attacks. By comparing their performance and analyzing their strengths and weaknesses[16], we seek to provide insights into the most effective strategies for deploying ML-based DoS attack detection in real-world scenarios. The ultimate goal is to enhance the resilience of online services against DoS attacks, ensuring continuity and reliability in an increasingly connected world.

1. **Methodology**
   1. **Data Acquisition**

The dataset used in this study was obtained from a combination of multiple sources, ensuring a diverse representation of network traffic patterns.[4] The dataset included both benign and malicious traffic, with the latter encompassing various types of DoS attacks.

* 1. **Data Splitting**

The dataset was **divided into training and testing** sets to evaluate the models' performance on unseen data. An 80/20 split was used, with **80% of** the data allocated for **training** and **20% for testing**.

* 1. **PyCaret Setup**

PyCaret, an open-source machine learning library, was employed to streamline the model selection and evaluation process. The **setup() function** initialized the **PyCaret environment**, automatically handling tasks like **data preprocessing** and **model selection**. The target variable, indicating whether a network flow was benign or malicious, was specified.

Before training the machine learning models, the dataset underwent preprocessing to enhance data quality and **model perfor**mance. This involved handling **missing values**, **encoding categorical features**, and **scaling numerical features**. Additionally, the Synthetic Minority **Over-sampling Technique (SMOTE)** was applied to address the **class imbalance issue** with parameter **‘fix\_imbalance=True’**, where the number of benign samples significantly outnumbered the malicious samples.

* 1. **Model Comparison**

The **compare\_models()** function in PyCaret was used to **evaluate the performance of various machine learning models**. This function trains and evaluates multiple models, providing a comprehensive comparison based on different metrics such as **accuracy**, **precision**, **recall**, **F1-score, Kappa, and MCC.**

* 1. **Model Creation and Tuning**

Based on the comparison results, the **Random Forest (RF)** and **Decision Tree (DT)** models were selected for further analysis. The **create\_model() function** was used to **create instances** of these models, and the **tune\_model() function** was employed to **optimize their hyperparameters**, enhancing their predictive capabilities.

* + 1. **Random Forest**

The Random Forest model is an ensemble learning method that constructs multiple decision trees during training and outputs the class that is the **mode of the classes (classification)** or **mean prediction (regression)** of the individual trees. Random forests correct for decision trees' habit of overfitting to their training set, such as Figure 1[3].

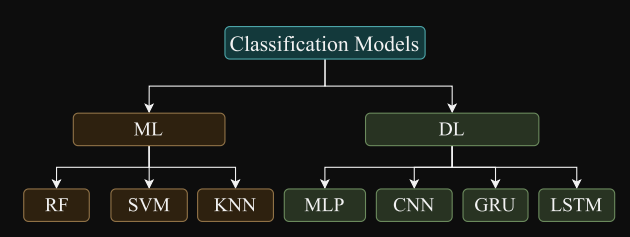
**Here's how it works**

**Bootstrapping**: The algorithm creates multiple subsets of the training data through random sampling with replacement. **Each subset** is used to train an **individual decision tree**.

**Random Feature Selection**: At **each node** of a decision tree, a random subset of features is considered for splitting. This randomness helps to decorrelate the trees and **reduce overfitting.**

**Tree Construction**: Each decision tree is **grown to its full depth** without pruning, using a greedy algorithm to **select the best split at each node** based on a criterion like **Gini impurity** or information gain.

**Aggregation**: For **classification** tasks, the final prediction is made by taking **a majority vote** among all the decision trees. For **regression** tasks, the **average prediction** of the trees is used.

****

**Figure 1**

**Mathematical Representation**

The Gini impurity is a common criterion used for splitting nodes in decision trees within a Random Forest. It measures the probability of misclassifying a randomly chosen element in a dataset if it were randomly labeled according to the class distribution in the subset.

**Gini(D) = 1 - Σ (pk)^2**

* D is the dataset
* pk is the proportion of class k instances in D

The lower the Gini impurity, the better the split.

* + 1. **Decision Tree**

A decision tree is a flowchart-like model used for classification and regression tasks. It recursively partitions the data based on feature values, creating a tree-like structure where each internal node represents a decision based on a feature, each branch represents an outcome of the decision, and each leaf node represents a class label or a numerical value[5].

**Here's how it works**

**Feature Selection**: The algorithm selects the most informative feature to split the data at each node. This is often done using criteria like Gini impurity or information gain.

**Splitting**: The data is divided into subsets based on the selected feature's value.

**Recursion**: Steps 1 and 2 are repeated for each subset until a stopping criterion is met (e.g., maximum depth, minimum samples per leaf).

**Prediction**: The leaf node reached by a new data point determines its predicted class or value.

**Mathematical Representation**

The Gini impurity is also used in decision trees for feature selection. The information gain is another criterion that measures the reduction in entropy (uncertainty) achieved by splitting the data based on a feature.

**Information Gain(D, a) = Entropy(D) - Σ [(|Dv| / |D|) \* Entropy(Dv)]**

* D is the dataset
* a is the feature
* Dv is the subset of D for which feature a has value v
  1. **Evaluation**

The performance of the tuned RF and DT models was assessed using **evaluate\_model().** This function provides a range of evaluation metrics, including **accuracy**, **precision**, **recall**, **F1-score**, and a **confusion matrix**, offering insights into the models' strengths and weaknesses.

* 1. **Model Finalization and Saving**

After evaluation, the finalized models were saved using save\_model(). This step ensures that the trained models can be easily loaded and used for future predictions without retraining.

* 1. **Visualization**

To gain a deeper understanding of the models' **decision-making processes**, **feature importance plots** were generated. These plots illustrate the relative importance of each feature in predicting DoS attacks. Additionally, **confusion matrices** were visualized using **plot\_model()**, providing a clear representation of the models' **classification performance**.

* 1. **Model Testing**

The saved models were loaded using **load\_model()** and then used to make **predictions on the testing set**. The predicted labels were compared with the **actual labels** to assess the models' **accuracy on unseen data**.

* 1. **Fernet algorithm for Encryption and Decryption**

The Fernet algorithm is a symmetric encryption technique available in Python's cryptography library. [2] It's designed for robust data encryption, ensuring that encrypted data remains confidential and unaltered without the correct key.

**How Fernet Works:**

**1.** **Key Generation:**

\* A unique, **secret key is generated**. This key is crucial for both encryption and decryption processes.

**2. Encryption:**

\* The data to be encrypted is combined with an **Initialization Vector** (IV). The IV is a **random value** that ensures that **encrypting the same data** multiple times produces different ciphertexts, enhancing security.

\* The **combined data** and **IV** are encrypted using the Advanced Encryption Standard (**AES) algorithm** in Cipher **Block Chaining** (CBC) mode. AES is a widely used and trusted symmetric encryption standard.

\* A Hash-based Message Authentication Code (**HMAC)** using **SHA256** is generated from the encrypted data. This HMAC serves as a digital signature, ensuring data integrity and **preventing unauthorized modifications.**

\* The **IV, ciphertext, and HMAC are combined and encoded using Base64 encoding**. This encoding makes the encrypted data **easier to store** and transmit across systems.

**3. Decryption:**

\* The Base64-encoded data is decoded.

\* The **HMAC is extracted** and verified **using the same key**. If the **HMAC doesn't match,** it indicates the data has been tampered with.

\* The **IV is extracted**, and the **ciphertext is decrypted** using the **AES algorithm in CBC mode** with the original key.

\* The decrypted data is returned, ensuring it's identical to the original data before encryption.

**Key Features**

\* Symmetric Encryption: The **same key is used for both encryption and decryption**, simplifying key management.

\* **AES-128** in **CBC Mode**: Employs a **strong encryption** standard with a 128-bit key and **CBC mode for added security**.

\* **HMAC** with **SHA256**: Guarantees data integrity by verifying that **the encrypted data hasn't been modified**.

\* **Base64 Encoding:** Makes the **encrypted data safe for storage** and transmission across different systems.

**Mathematical Representation**

While the core of Fernet relies on established cryptographic standards like AES and HMAC, the overall process can be represented as:

**Encryption:**

**Ciphertext = Base64Encode(AES\_CBC(Data + IV, Key) + HMAC\_SHA256(EncryptedData, Key))**

**Decryption:**

**Data = AES\_CBC\_Decrypt(Ciphertext, Key, IV)**

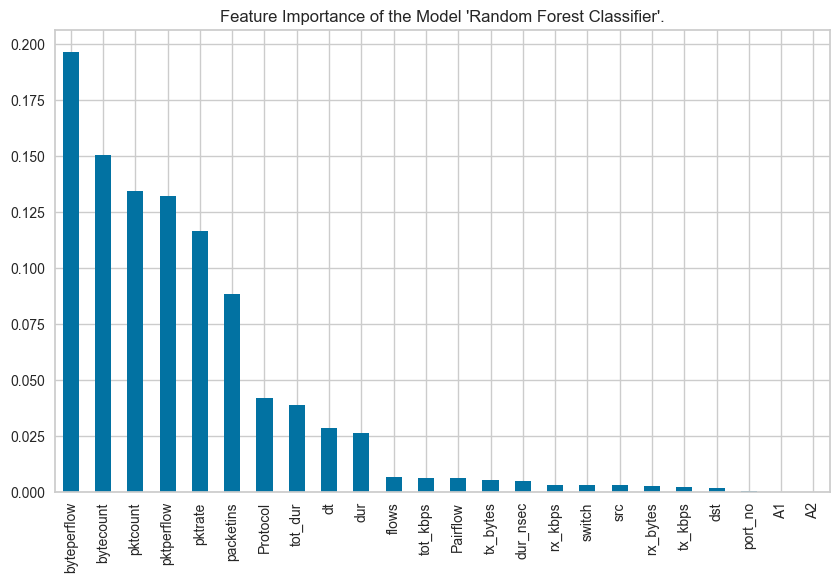
**Verify HMAC\_SHA256(EncryptedData, Key)**

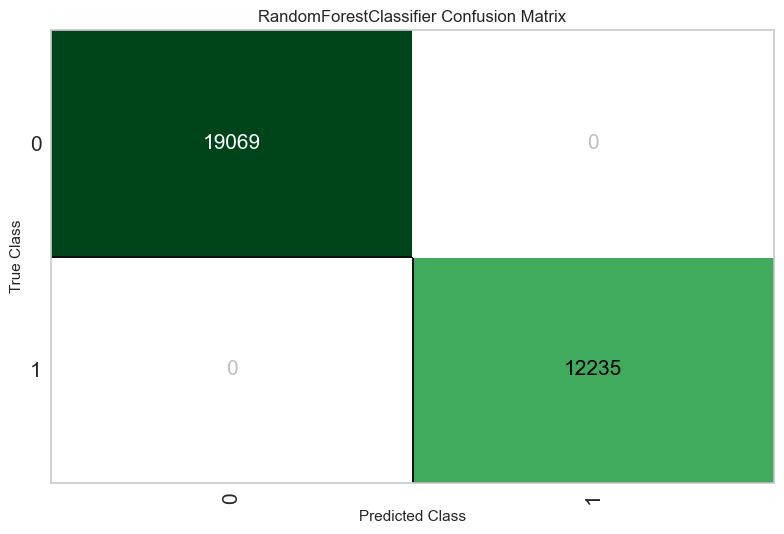
1. **Results**

The Random Forest model consistently outperformed the Decision Tree model across various evaluation metrics, such as figure 2, It achieved a higher accuracy, precision, recall, and F1-score, indicating its superior ability to detect DoS attacks. The confusion matrices further confirmed the Random Forest model's effectiveness, revealing fewer false positives and false negatives compared to the Decision Tree model, such as Table 1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Accuracy | Precision | Recall | F1-Score |
| Random Forest | 0.9999 | 0.9999 | 1 | 0.9999 |
| Decision Tree | 0.9998 | 0.9994 | 1 | 0.9997 |

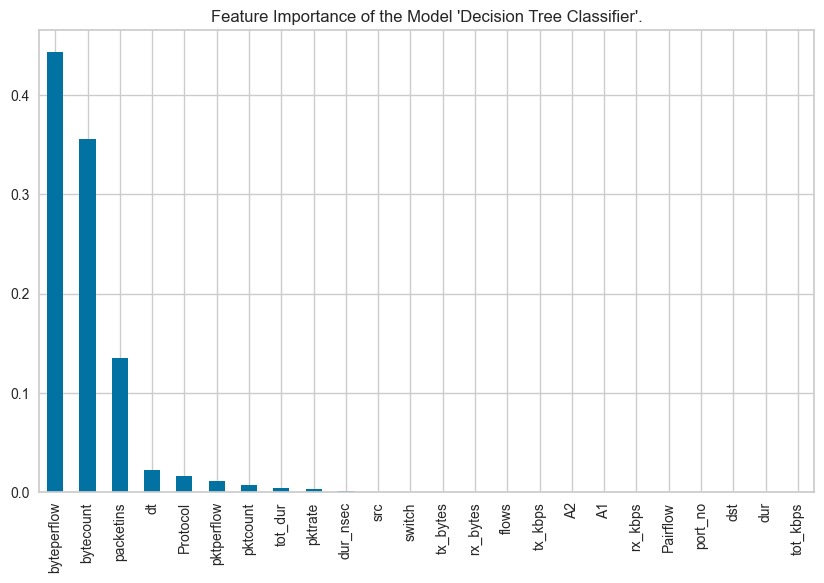
Table 1

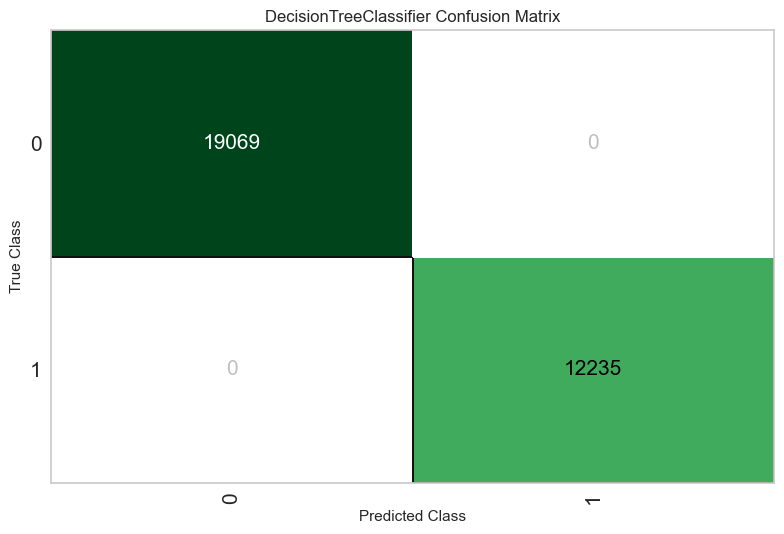
* 1. **Random Forest**

****

Graph 1

Figure 2

* 1. **Decision Tree**

****

Graph 2

Figure 3

1. **Discussion and feature work**

The results of this study highlight the effectiveness of machine learning, particularly the Random Forest model, in detecting DoS attacks. The PyCaret framework significantly streamlined the model development process, enabling rapid experimentation and evaluation. Future work will focus on exploring additional machine learning algorithms, incorporating real-time data streams, and developing adaptive models that can respond to evolving attack patterns. The integration of the Fernet encryption algorithm ensures data security, making this approach a promising avenue for future research and development in the field of cybersecurity.

1. **Conclusion**

This research demonstrates the potential of machine learning in bolstering cybersecurity defenses against DoS attacks. By leveraging the PyCaret framework and employing the Random Forest model, we achieved high accuracy in attack detection. The integration of the Fernet encryption algorithm further ensures data security, making this approach a promising avenue for future research and development in the field of cybersecurity.

1. **Reference**

[1] N. M. Yungaicela-Naula, C. Vargas-Rosales and J. A. Perez-Diaz, "SDN-Based Architecture for Transport and Application Layer DDoS Attack Detection by Using Machine and Deep Learning," in IEEE Access, vol. 9, pp. 108495-108512, 2021, doi: 10.1109/ACCESS.2021.3101650.

keywords: {Denial-of-service attack;Computer crime;Proposals;Support vector machines;Real-time systems;Internet of Things;Entropy;Software defined networking;deep learning;machine learning;DDoS attack;transport layer;application layer;slow-rate attacks},

[2] Information Security and Machine Learning: Encryption Allocation Based on Recognizing of Text Patterns (Santos P H J1, Costa Junior C A D F2, Rodrigues J3, Carvalho L M4, Souza F H B5).

[3] A Survey of Machine Learning and Cryptography Algorithms(Innovative Machine Learning Applications for Cryptography (pp.105-118) M. Indira, K. S. Mohanasundaram, M. Saranya)

[4] Detecting Cryptography Misuses With Machine Learning: Graph Embeddings, Transfer Learning and Data Augmentation in Source Code Related Tasks(IEEE Transactions on Reliability PP(99), Gustavo Eloi de Paula Rodrigues, Alexandre Melo Braga, Ricardo Dahab)

[5] Machine Learning Techniques to Predict the Inputs in Symmetric Encryption Algorithm(innovative Machine Learning Applications for Cryptography (pp.163-172), M. Sivasakthi, A. Meenakshi)

[6]Frias-Martinez, E., Sanchez, A. & Velez, J., 2006. Support vector machines versus multi-layer

perceptrons for efficient off-line signature recognition. Engineering Applications of Artificial Intelligence, Volume 19, pp. 693-704.

[7]Gan, M., Wang, C. & Zhu, C., 2016. Construction of hierarchical diagnosis network based on deep

learning and its application in the fault pattern recognition of rolling element bearings. Mechanical Systems and Signal Processing, Volume 72-73, pp. 92-104.

[8]Jiang, D. et al., 2020. A probability and integrated learning based classification algorithm for high-level human emotion recognition problems. Measurement.

Khalifa, O., Islam, M., Khan, S. & Shebani, M., 2004. Communications cryptography. IEEE, pp. 220

223.

[9]Li, D., Chen, X. & Huang, K., 2015. Multi-attribute learning for pedestrian attribute recognition in

surveillance scenarios. 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR), pp. 111-115.

[10]Lima-Junior, F. R. & Carpinetti, L. C. R., 2019. Predicting supply chain performance based on SCOR® metrics and multilayer perceptron neural networks. International Journal of Production Economics, pp. 1938.

[11]Mazurowski, M. A. & Tourassi, G. D., 2009. Evaluating classifiers: Relation between area under the receiver operator characteristic curve and overall accuracy. 2009 International Joint Conference on Neural Networks, pp. 2045-2049.

[12]Oliveira, C., Xexéo, J. A. M. & Carvalho, C. A. B., 2006. Clustering and categorization applied to cryptanalysis. Cryptologia, Volume 30, pp. 266-280.

[13]Pejic-Bach, M., Bertoncel, T., Meško, M. & Krstić, Ž., 2020. Text mining of industry 4.0 job

advertisements. International Journal of Information Management, pp. 416-431.

[14]Souza, F. H. B. et al., 2019. Risk Prediction For Surgical Site Infection In Craniotomy Patients.

Antimicrobial Resistance & Infection Control, p. 34.

[15]Souza, W. A. R., De Carvalho, L. A. V. & XEXÉO, J. A. M., 2009. Ciphertexts Clustering is Equivalent to Plaintexts Clustering. Seventh Brazilian Symposium in Information and Human Language

Technology.IEEE., pp. 44-52.

[16]Thome, A. C. G., 2012. SVM Classifiers – Concepts and Applications to Character Recognition. In: Advances in Character Recognition. London: InTech, pp. 25-50.