Detecting Dangerous Network Requests with Machine Learning

Isaac Braun
School of Computing
Southern Adventist University
Collegedale, Tennessee 37315
Email: isaacbraun@southern.edu
Telephone: (419) 706-0480

Abstract—**Safety has always been a top priority, especially in network security. Firewalls play a crucial role in monitoring network requests to prevent harmful requests from causing damage to the systems or users on the network. However, the more advanced firewall systems can be expensive and inaccessible to individuals and small businesses. By applying machine learning, we can develop open-source and enhanced network monitoring systems that anyone can utilize. This paper serves as a starting point for such systems by testing various data engineering and classification methods. Due to imbalanced training data, the models generated may not be sufficient, but the findings and tests can be helpful in jumping-starting future development.**

I. Introduction

Network security is a critical and ever-evolving field. With attackers constantly changing their approaches and monitoring systems increasing in cost, individuals and small business owners may be unprotected. Here, machine learning can be employed to create adaptive and open-source systems that anyone can utilize. The experiments in this paper aim to be exploratory and serve as a basis for further development. With limited and imbalanced data, the goal was to find a successful way to balance the data and evaluate the performance of different classification models. These findings can inform future work on which model and data balancing methods to use and provide a starting point for deploying a model in a monitoring system.

II. Dataset

A. Source & Description

That data was sourced from Rudra Kumar via Kaggle.com [1], who captured over 7,000 requests using Zap Security Scanner [2], a network scanning tool. Each sample of data is a network request that falls into one of two classes, 'bad' or 'good,' which signifies whether the network request is malicious or safe to allow through. The features include the request method, path, and body; the rest are calculated based on their values. Ten features represent the counts of characters or strings included in the path or body. For example, characters such as semicolons, quotes, and percentages are counted, as well as the usage of keywords such as 'select,' 'order by,' 'drop,' and 'script.' Finally, two features contain the length of the path and body, respectively. These features combined can form patterns among legitimate and malicious network requests, making it a perfect use case for classification models.

B. Data Engineering

As the dataset was very clean, the initial cleaning only involved removing the path and body features, as they were no longer directly needed. Then, the data was split into two data frames, one for the features and one for the targets. The feature values were then normalized using a min-max scaler to create even significance across the board. A Chi-square test was done to find features that would not significantly impact the classification, returning the path and body length features. With these features removed, the results were worse across the board, so the results have not been featured. Lastly, the data was split into training and testing datasets, using an 80 to 20 percent split.

Even with a significant amount of data, there was still a large imbalance of the classes. Before splitting into training and testing datasets, the 'bad' class contained 5510 samples, while the 'good' class only had 287 samples. While some classification models may be able to produce respectable results with this imbalance, it was detrimental for the majority. Thus, this staggering imbalance called for evaluating three different resampling methods: random downsampling, random oversampling, and SMOTEENN.

Random down sampling is a simple process that takes the majority class, 'bad' in this case, and randomly removes samples. For example, in this implementation, the 'bad' samples were limited to 500, while the 'good' remained at the initial 230. Random oversampling follows a similar logic, but with the minority class, randomly duplicating samples until the classes are more balanced. The SMOTEENN approach, or Synthetic Minority Oversampling Technique plus Edited Nearest Neighbor, combines both downsampling and oversampling. In short, the SMOTE algorithm creates synthetic samples randomly based on a sample and its neighbors, moving through the data until the minority count proportion is fulfilled. Then, the ENN algorithm sorts through all the samples, deleting a sample and its neighbor if it doesn't match the majority class of the other nearest neighbors.

III. Results

Five different classification models were trained during these experiments, using the three differently balanced datasets for training. This allowed the evaluation of the models and data balancing methods in this use case. All the models were then tested on the training dataset that had been resampled using the SMOTEENN method.

A. *K-Nearest Neighbors*

The K-Nearest Neighbors algorithm was one of the few that performed decently without balancing the data, so unsurprisingly, it performed excellently and tied for the best accuracy. In Table I, it displays excellent performance across the board, with perfect and near-perfect recall in the 'bad' class. It also sports the most even performance when trained with downsampled data, scoring between 0.95 and 0.97 in every box.

B. *Decision Trees*

This algorithm also performed decently without balancing the data but did not perform as well as expected afterward, as seen in Table II. Nonetheless, it performed respectably with SMOTEENN, oversampled data, and very well with downsampled data. Even though the overall accuracy is not high for SMOTEENN and oversampled, the high recall for 'bad' and lower recall for 'good' means that the model would let very few malicious requests through but block more legitimate requests. If used in a very strict setting, this model may be better than others with higher accuracy scores.

C. *Logistic Regression*

The Logistic Regression algorithm was primarily scored consistently across the different data balancing methods, varying slightly with downsampled data, but not by much. While it boasts high recall scores for the 'good' class, this would make a poor model for stopping malicious requests as Table III shows low recall for 'bad' requests, meaning many would be let through.

D. *Support Vector Machines*

Table IV shows that this algorithm suffers from the same problems as Logistic Regression. It also produced decent accuracy and perfect 'good' class recall, but the low 'bad' class recall scores make it a poor model for this use case.

E. *Artificial Neural Networks*

The Artificial Neural Network algorithm, with scores displayed in Table V, looks incredible at first glance, with very even f1-scores across the different datasets, but upon further investigation, it disappoints. Once again, the model has perfect recall for the 'good' class but less than acceptable recall for the 'bad' class, making it a poor choice to reject malicious requests.

TABLE I
K-Nearest Neighbors

Data	Type	Precision	Recall	F1-Score	Support
SMOT-EENN	bad	0.89	0.97	0.93	1062
	good	0.97	0.88	0.93	1058
	accuracy			0.93	2120
	macro avg	0.93	0.93	0.93	2120
Random Over Sampled	bad	0.80	1.00	0.89	1062
	good	1.00	0.75	0.86	1058
	accuracy			0.88	2120
	macro avg	0.90	0.88	0.87	2120
Down Sampled	bad	0.97	0.96	0.96	1060
	good	0.95	0.97	0.96	1057
	accuracy			0.96	2117
	macro avg	0.96	0.96	0.96	2117

TABLE II
Decision Trees

Data	Type	Precision	Recall	F1-Score	Support
SMOT-EENN	bad	0.67	0.97	0.79	1062
	good	0.95	0.52	0.67	1058
	accuracy			0.74	2120
	macro avg	0.81	0.74	0.73	2120
Random Over Sampled	bad	0.82	1.00	0.90	1062
	good	1.00	0.77	0.87	1058
	accuracy			0.88	2120
	macro avg	0.91	0.88	0.88	2120
Down Sampled	bad	0.94	0.95	0.94	1060
	good	0.94	0.94	0.94	1057
	accuracy			0.94	2117
	macro avg	0.94	0.94	0.94	2117

TABLE III
Logistic Regression

Data	Type	Precision	Recall	F1-Score	Support
SMOT-EENN	bad	1.00	0.89	0.94	1062
	good	0.90	1.00	0.95	1058
	accuracy			0.94	2120
	macro avg	0.95	0.94	0.94	2120
Random Over Sampled	bad	1.00	0.85	0.92	1062
	good	0.87	1.00	0.93	1058
	accuracy			0.93	2120
	macro avg	0.94	0.93	0.93	2120
Down Sampled	bad	1.00	0.78	0.88	1060
	good	0.82	1.00	0.90	1057
	accuracy			0.89	2117
	macro avg	0.91	0.89	0.89	2117

IV. Conclusion

After evaluation, only two models would be well-suited for usage in a network firewall: K-Nearest Neighbors and Decision Trees. While the other three models produced eye-catching accuracy and f1-scores, they would all suffer the same fate of letting too many malicious requests through. K-Nearest Neighbors and Decision Trees tie for the highest

TABLE IV
SUPPORT VECTOR MACHINES

Data	Type	Precision	Recall	F1-Score	Support
SMOT-EENN	bad	1.00	0.87	0.93	1062
	good	0.89	1.00	0.94	1058
	accuracy			0.94	2120
	macro avg	0.94	0.94	0.94	2120
Random Over Sampled	bad	1.00	0.85	0.92	1062
	good	0.87	1.00	0.93	1058
	accuracy			0.93	2120
	macro avg	0.93	0.93	0.92	2120
Down Sampled	bad	1.00	0.76	0.87	1060
	good	0.81	1.00	0.89	1057
	accuracy			0.88	2117
	macro avg	0.90	0.88	0.88	2117

TABLE V
ARTIFICIAL NEURAL NETWORKS

Data	Type	Precision	Recall	F1-Score	Support
SMOT-EENN	bad	1.00	0.93	0.96	1062
	good	0.93	1.00	0.96	1058
	accuracy			0.96	2120
	macro avg	0.96	0.96	0.96	2120
Random Over Sampled	bad	1.00	0.88	0.94	1062
	good	0.89	1.00	0.94	1058
	accuracy			0.94	2120
	macro avg	0.95	0.94	0.94	2120
Down Sampled	bad	1.00	0.88	0.93	1060
	good	0.89	1.00	0.94	1057
	accuracy			0.94	2117
	macro avg	0.95	0.94	0.94	2117

'bad' recall scores with oversampled data, followed closely by SMOTEENN data. These models would be the best at rejecting most malicious requests, but with lower 'good' recall scores, they may frustrate many legitimate network users. Both models offer a very balanced prediction model using downsampled data, with K-Nearest Neighbors slightly taking the edge.

These results are exciting, and they show the value supervised machine learning can have in the field of network protection. For future work, these results can jump-start improved model development and provide guidance on which data balancing method to use for a specific algorithm. More real-world data should be procured and evaluated using the K-Nearest Neighbors and Decision Trees algorithms, as they showed the most promise in this particular use case.

REFERENCES

[1] R. Kumar, "Web Network Traffic," Sep. 2024. [Online]. Available: https://www.kaggle.com/datasets/rudrakumar96/web-firewall-good-and-bad-request/data

[2] Z. D. Team, "Zed Attack Proxy (ZAP)," Sep. 2024. [Online]. Available: https://www.zaproxy.org/