Title

Ludovico Toscano

# Abstract

In this thesis, we will see how explainable algorithms such as Shap can play a key role in improving the niche of network intrusion detection systems by helping to analyze HIKARI-2021 and NFS-2023-TE. Starting from a recent study that shows how most recent works about NIDS use complex models without a real need, we demonstrate how a decision tree could be better than a complex GA2Mmodel such as the explainable boosting machine. Then thanks to the usage of Shap we analyze different models and demonstrate why these datasets aren’t suitable for real word usage.

# Introduction

Section 1 - Datasets used

In February 2024 the three datasets with more citations in Google Scholars were CIC-IDS 2017 (Sharafaldin et al., 2018) with 3149 citations, NLS-KDD (Tavallaee et al., 2009) with 5131 citations, and UNSW-NB15 (Moustafa & Slay, 2015) with 2740 citations these two are improvements based on the raw data of KDD Cup ’99 (Lee et al., 1999). Our focus is on two datasets derived from CIC-IDS 2017, illustrating why it may no longer be the best option.

By making CIC-IDS 2017 the authors have achieved for the first time the goal of covering all the 11 criteria given by (Gharib et al., 2016), by using CICFlowMeter (Lashkari et al., 2017). However, the analysis of the pcap files by (Engelen et al., 2021) and (Liu et al., 2022) revealed over 5% corruption in dataset labeling and non-compliance of CICFlowMeter with TCP connection closure standards (Brownlee et al., 1999). This led to the creation of WTMC-2021. One year later WTMC-2021 was improved with the name of CRiSIS-2022 (Lanvin et al., 2023) by adding the attack port scan which wasn’t labeled, and by sorting and removing the duplicates in the pcap file. In 2023 (Pekar & Jozsa, 2024) have noticed that even after the patching CICFlowMeter wasn’t closing the connection after a FIN or RST flag and was also making some missing or negative values, so they decided to make NFS-2023-TE by using NFStream (Aouini & Pekar, 2022). NFS-2023-TE creates a new flow at first FIN or RST flag like most of the flow analyzers do (Hofstede et al., 2014) while NFS-2023-nTE creates a new flow after a timeout like WTMC-2021 and CRiSIS-2022 do, this second dataset has been made only for the sake of comparison with his predecessor.

Is possible that shortly things will change, since the authors of CICFlowMeter said in an email exchange that they are making a new tool written in Python with more than 50 new features for a total of 130 features.

Hikari 2021 (Ferriyan et al., 2021) is a dataset that is inspired by CIC-IDS 2017 but includes 7 new requirements, they provide as well the pcap files alongside a csv file and a pkl of the dataset. They provided 84 traffic features and 2 target features one multi-categorical and the other one binary, among the 84 traffic features most of them are inspired by the ones of CIC-IDS 2017. In this dataset, the authors have focused only on web attacks since 80% of the attacks nowadays are done on the application layer. So, they have performed 3 different attacks, but on the dataset, a fourth one is present since while they were analyzing the background traffic, they found a crypto miner attack. The raw data have been extracted using tcpdump, while the labeling and the analysis of the background traffic were made with Zeek which has also added some features that aren’t present in CIC-IDS 2017. These features are the IP address, source, and destination ports, and uid. Since the release of the paper, the dataset has received an update that improves the dataset by increasing the size of the minority classes.

We decided to make our models using NFS-2023-TE, HIKARI-2021, since as far as we know they are the best dataset available.

Section 2 - Explainable algorithm

Shap (Lundberg & Lee, 2017) is a Python library that makes use of a game theoretic approach by computing an approximation of the Shapley values to create post hoc explanations for any machine learning models. With this work among other explainers, they introduced Kernel Shap a model-agnostic explainer adapted from Lime (Ribeiro et al., 2016), and Deep Shap a model-specific explainer for neural networks that has adapted the algorithm of Deep Lift (Shrikumar et al., 2017).

Lately, other two papers (Lundberg et al., 2018, 2020) introduced Tree Shap a model-specific explainer for trees and ensembles of trees, which is so fast compared to Deep Shap and Kernel Shap that in most cases can explain the entire training set in a few seconds. However, the library Fast Tree Shap (Yang, 2021) introduced an optimized version of the Tree Shap algorithm which is faster in single-core but can overcome the single-core limitation of the original algorithm allowing even faster parallel execution. Since Fast Tree Shap keeps all the characteristics of Tree Shap when we talk about Tree Shap, we will refer to both Tree Shap and Fast Tree Shap algorithm.

Tree Shap explainers, as well as Linear Shap, can provide true-to-the-model explanations (H. Chen et al., 2020) as opposed to the other explainers that are true-to-the-data. By breaking the dependence between features following the rules of casual inference (Janzing, 2019) true-to-the-model explanations assign a value different from zero to a feature only if it’s used by the model. In the case of true-to-the-data explanation if two features correlate 100% the same importance will be assigned to them even if one is not used.

Even though Tree Shap is a fast post hoc explainer, a glass box model like the EBM (Explainable Boosting Machine (Lou et al., 2013)) which is a GA2M model doesn’t require any post hoc explanation at all promising to be accurate as a state-of-the-art black box model without trading off the explainability.

Section 3 – Related works

In the survey (Zhang et al., 2022), the authors highlight some limitations of explainable AI for cyber security, a big subject that includes intrusion detection. We will focus on how two future improvements highlighted in this survey can be solved by using boosted models such as the Random Forest (Breiman, 2001), LightGBT (Ke et al., 2017), XGBoost (T. Chen & Guestrin, 2016), CatBoost (Dorogush et al., 2017) with Fast Tree Shap, or EBM.

The first limitation highlighted is the trade-off between model performance and explainability, this is because most of the related works make use of complex back box deep models. Most of the time these back boxes can be explained only with model agnostic models like Kernel Shap, but this process is computationally expensive leading to a tradeoff between the accuracy of the explanation. Moreover, adversarial attacks (Slack et al., 2020) have proven to work against LIME (Ribeiro et al., 2016) and Kernel Shap. By using EBM or Tree Shap isn’t possible to realize this attack.

While the mentioned issues are related to the explainable cyber security word (Catillo et al., 2023) have written a paper about some of the most common issues encountered in machine learning models applied to network intrusion detection systems. They have listed two kinds of issues, the ones about public intrusion datasets, and the implications from the machine learning perspective. The following is the analysis of the issues of the dataset used.

Simplification of the data collection environment

As far as we know, no dataset has achieved this requirement.

Contemporaneity and effectiveness of the attack

CIC-IDS 2017 as discussed in (Catillo et al., 2021) contains some DDoS attacks that aren’t effective nowadays, and by being a 7-year-old dataset fails to meet this requirement.

Hikari 2021 focuses only on encrypted traffic at the application layer by saying that 80% of the attacks are done at this level. As far as we know this is the most updated dataset available and the attack provided are supposed to be effective.

Representativeness of the normal baselines

Both these datasets fail to meet this requirement.

Bugs of the feature extractor and incorrect ﬂow records

To avoid this problem NFS-2023-TE was made with NFStream which was chosen since it’s an open-source tool with a wide user base, and the labeling process has been documented and released alongside the Python code used.

Because Hikari 2021 doesn’t provide any source code, and they mention that the labeling was made with Zeek alongside an undefined Python tool we decided to analyze the dataset before using it, this part will be discussed lately. However, in an email exchange with the authors, they said that the source code would be released.

Data Labeling

Both datasets aren’t made solely by looking at the combination of timestamps, IP, and ports but by analyzing the payload. In the case of NFS-2023-TE, the labeling is based on the different analyses available performed over the pcap file, while in HIKARI 2021 an attack was found during the analysis of the background data. However, since HIKARI 2021 doesn’t provide the timestamp and part of the payload is encrypted to ensure privacy isn’t possible to access if there is any flaw in the labeling.

Class imbalance

Both the datasets aren’t balanced, the smallest class of NFS-2023-TE with 11 samples is Hearthbleed while in HIKARI-2021 is Bruteforce with 7988 samples. In the first case, over-sampling is not an option, while for the latter different strategies can be adopted. We decided to do stratified under-sampling for both datasets, for HIKARI-2021 all the classes have 7988 samples while for NFS-2023-TE all the classes will be 738 or less.

Section 4 - Methodology

After having seen all the issues from the dataset side, is time to focus on the ML implications.

Attack-revealing features and ease of detection

In (D’hooge et al., 2023) they proved how some revealing features make it easy even for a OneR model to achieve high scores, while we could avoid ease of detection if we handled attack-revealing features by doing feature selection. In both datasets, all the features related to IP addresses, ports, Mac addresses, and the binary target label have been removed. NFS-2023-TE and NFS-2023-nTE also contain the timestamps of the connections that have been removed, moreover there is a feature called protocol which led to some biases since all the attacks are performed over TCP. HIKARI 2021 contains a label called uid and an index that has been removed.

Data partitioning

The random under-sampling, training split, and cross-validation were all made by stratifying using the attack classes preventing the loss of the smallest classes. We choose 80% of the data for training and the remaining for testing.

Unmotivated complexity

All the mentioned models have been compared to a decision tree and the fitting and classification time as well as the explanation time has been measured.

Use of the evaluation metrics

We used two metrics that sklearn calls F1 macro and F1 weighted, the first computes the F1 score of each class and then does the unweighted mean while the latter does the weighted mean.

Lack of transferability

For the moment being able to create a network intrusion detection system with a public dataset and being able to run in the real world by keeping the same results is still an open problem.

Section 5 – Results

Data set analysis

While in NFS-2023-TE we didn’t find any flaw in the dataset, in HIKARI 2021 we noticed that the flag count for FIN and RST is higher than how it’s supposed to be having a maximum of 140 FIN packages and 110 RST packages. Moreover, the longest flow lasted 17942 seconds which is about 4.9 hours. Since the capturing sessions of HIKARI 2021 lasted between 3 to 5 hours, this means that the tool used didn’t have a timeout nor wasn’t closing the connection at the first RST or FIN flag. To the best of our knowledge, none of the papers about HIKARI 2021 have noticed this issue, which if not noticed during the pre-processing of the dataset can be highlighted by the explanations of the algorithms.

Since there isn’t documentation about the labeling and the authors didn’t release the timestamps, this dataset is not suitable for real word usage, instead, we decided to show how explainability can be used to discover this kind of flaw.

Given that we found these flaws in the csv files, we decided to analyze also the pcap files by using the same tools of CRiSIS 2022 reordercap and editcap with a window of 500 microseconds. We found that 0,03% of the packages were duplicated and 0,23% were out of order, the authors have confirmed us that issue and they didn’t perform any preprocessing before making the dataset.

To achieve better results, we decided to opt for under-sampling for all the datasets, HIKARI-2021 all the classes have been sampled to have 7988 samples for each class, moreover, we merged Bening and background traffic. This number has been chosen since the smaller class is that big. While for NFS-2O23-TE under-sampling has been performed to have 738 sampled for each class except smaller classes. In this case, we decided to test different numbers of samples, and this was the one performing the best. This is a big limitation of this study, but the alternative of removing some attacks isn’t that better. With only 11 samples on the smallest class over-sampling was not considered to avoid overfitting on the sintetic data.

Model performance

The following training, classification, and explanation times have been measured on a Dell XPS 13 9315 with a 12th gen i7-1250u and 16gb of ddr5 ram running Fedora 39 with Linux 6.8. For the environment information, there is on the GitHub repository a YAML file with all the versions of the packages in the conda environment. Is important to notice that Sklearnex and daal4py have been used to improve inference time, and all the models have been set to use all the cores available.

The parameter tuning of Table 1 has been done following the suggestions of the documentation for each library, and by looking at what was working for the other models. We used the metrics f1 macro of sklearn to evaluate the results and when different parameters were giving similar results smaller training time was used to lead the decision.



Table 1 - NFS-2023-TE and NFS-2023-nTe parameters

In Table 2 we have the training time, explanation time, and the sum of both in ms for each model tested with 6087 samples. For the explanation fast tree shap is set to automatically choose the best algorithm. These times are not accurate they have some variance, but they show the order of magnitude necessary to run each model. We can notice that even if EBM is the slowest during training if we also consider the explanation time it can be faster than lightgbm and xgboost. The random forest is close to the decision tree thanks to using only 10 trees which is considerably less than the GAM models. Another thing to consider is that even if Catboost uses fewer trees than the lightgbm and xgboost is slower to train.



Table 2 - NFS-2023-TE training times



Table 3- NFS-2023-TE – average F1 score of 5 fold

In Table 3 we have a comparison of the prediction time in μs for 1522 samples, in this case, we have used the magic function %timeit of ipython to compute the mean of different runs. Doing the mean is important not only because it is statistically more significant, but because with such smaller times a warmup of the function is necessary to reduce the computation time. Without the warmup run the classification time of the decision tree was 3 ms because at the first run, the computer loaded in memory the prediction function which takes more time than the prediction itself.

Looking at the EBM we can notice that is the best when it comes to F1 macro but the worst with F1 weighted. Comparing these results with the decision tree shows that the decision tree has a problem with Hearthbleed and Web Attack - SQL Injection it misclassifies 2 samples for the first and 4 with the latter, while the ebm misclassifies 111 samples in the DoS Slowhttptest. These results expose the need for a balanced dataset, we can argue that EBM is the best one because can handle well each attack, but on the other hand building a model over 11 samples will lead to something that isn’t going to work on the real word.



Table 4 - HIKARI 2021 parameters

Moreover, we need to also consider the prediction time by looking at both the F1 results lightgbm is probably the winner by being the second best in both macro and weighted score but is the worst in prediction time. Lightgbm is twice as slow as the random forest which scores nearly the same in the F1 metrics, making the random forest overall the best alternative. The reason why the classification time is so important is that in case of a DoS attack that opens and closes the connection at a fast enough speed, the model needs to keep up with each new flow generated otherwise the NIDS will run out of service. Another reason why this classification time is important is that can make a difference between enabling or not the use on the edge and lead to less expensive devices when the GPU is not required for running an NIDS.

Table 4 shows how the parameters for HIKARI-2021 lead to less complex models except for the ebm, one of the reasons that makes this possible is the fact that the dataset is balanced after the undersampling.



Table 5- HIKARI-2021 training times

Therefore, the training times for 31952 training samples except for the ebm are lower than the ones of NFS-2023-TE.



Table 6 - HIKARI-2021 F1 score and times

Since HIKARI-2021 is balanced there is no difference between F1 macro and F1 weighted, and since all the models score nearly the same the analysis comes to which one is the fastest which is the decision tree.

Model explanations

Figure 1 shows the feature importance of a decision tree for the class bot of NFS-2023-TE, the model has misclassified only 1 sample out of 737 by only using a small subset of the available features. The same pattern of using a few features applies to the other attacks, the only exception is the benign traffic which requires more features to be detected. The reason why benign assigns positive importance to more features is that most of the splits lead either to an attack or benign traffic.

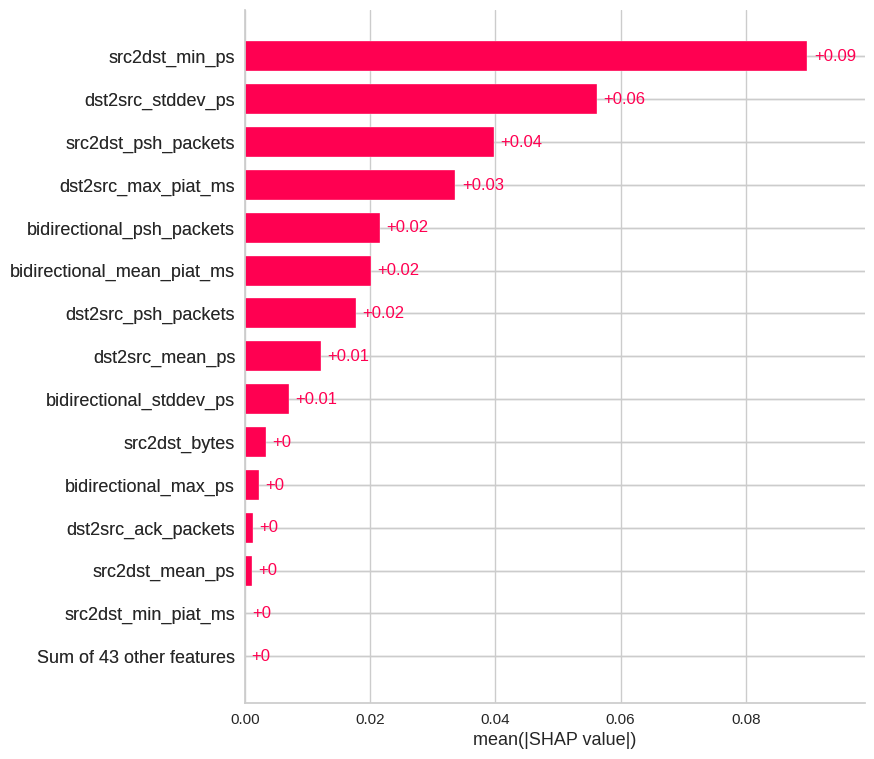


Figure 1 - NFS-2023-TE - decision tree feature importance with shap of bot class

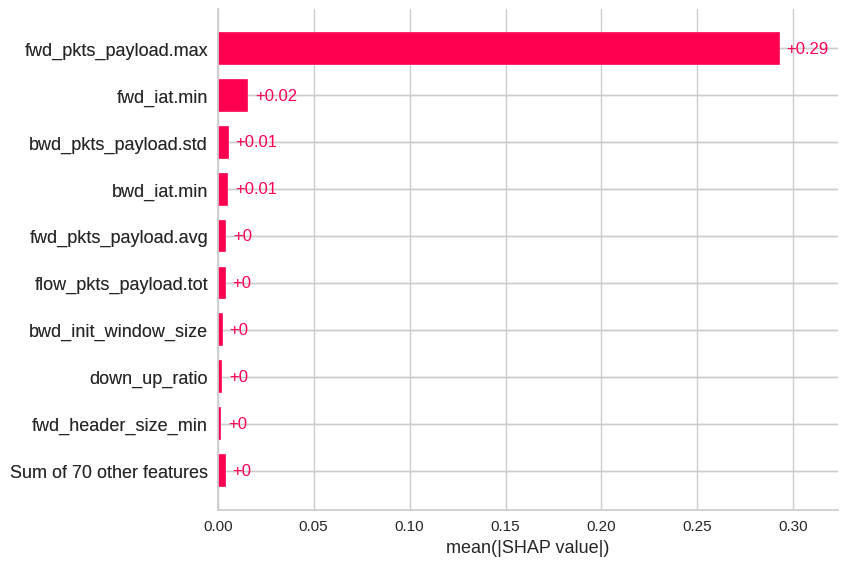


Figure 2 - HIKARI-2021 - decision tree feature importance with shap of bruteforce-XML class

In Figure 2 we can see that the decision tree has given almost all the importance to one feature, which has been used as a root node and in a node at the first branch classifying the node as bruteforce-XML with a gini coefficient of 0.041. This explanation has led to an analysis of the feature by grouping all the samples by class and what has been found is that the value of this feature at the 25th percentile is 746 which is higher than the 75th percentile of all the other classes. While in the class probing all the samples have 517 as the value for this feature which is the second most important.

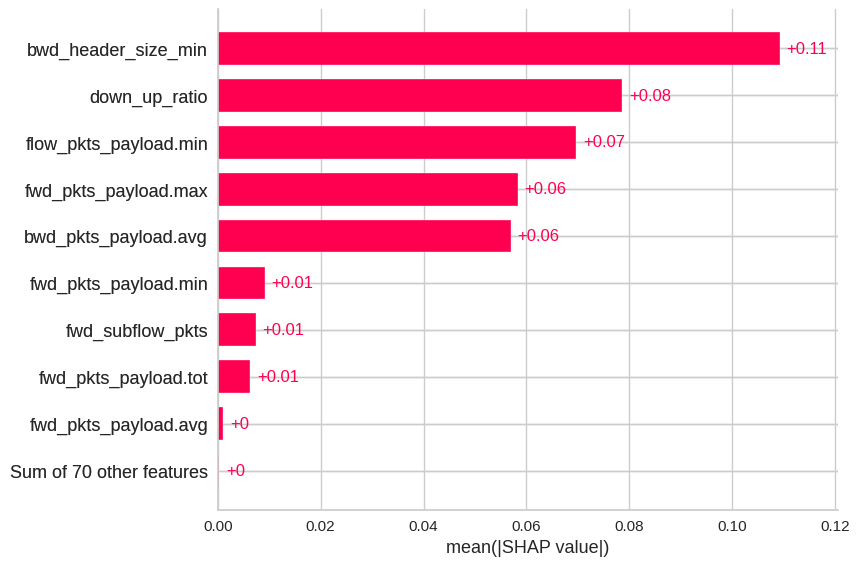


Figure 3- HIKARI-2021 - decision tree feature importance with shap of XMRIGCC CryptoMiner

Figure 3 is the feature importance of XMRIGCC CryptoMiner with the decision tree, again only a few selected features have been used the most important is bwd\_header\_size\_min which for this attack is always zero then we have down\_up\_ratio. Down\_up\_ratio for this class is always zero and is the most important feature for probing and the second most important for background, the reason is that background has an average of 0.8, probing is 1.3 while the remaining attacks have more than 8. Also, benign which has been merged with background shows an average of 2.

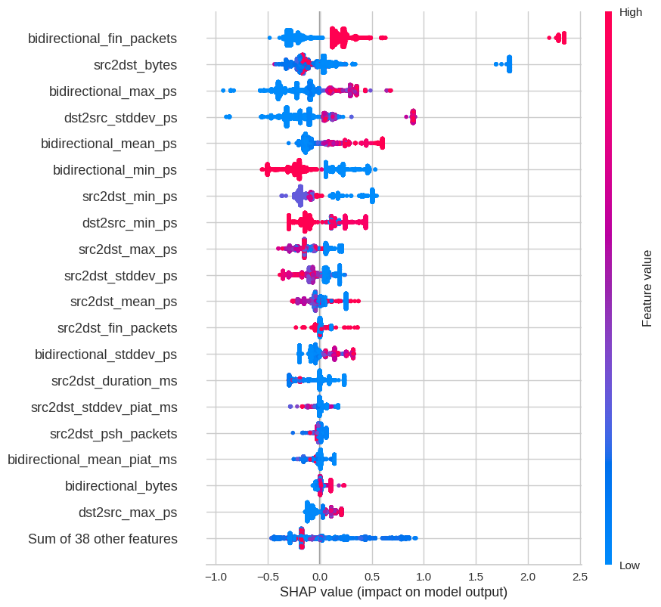


Figure 4 - NFS-2023-TE - catboost impact on the model output of DDoS with shap sorted by highest magnitude of impact

Figure 4 is the explanation of the catboost algorithm where we can see that the feature with the highest magnitude of all is bidirectional\_fin\_packets, which shows up for Bot, DDoS, DoS Slowhttptest, DoS Slowloris, Heartbleed, Infiltration, and Web Attack – XSS as one of the two most important feature in average or the one with the highest magnitude of impact. By analyzing this feature, we can notice that Bot and Web Attack -XSS always have 1 FIN packet, while DDoS averages 99% of the samples with 1 FIN packet. This characteristic makes the difference in the other attacks cited at the 85th percentile of the distribution showing 0 FIN packets. To understand why we have a lot of 0 we analyzed the duration and for Infiltration and Heartbleed, most of the connection gets closed because of the timeout, while for DoS Slowhttptest and DoS Slowloris, it’s hard to tell since they didn't reach the timeout either they send an RST packet.

All these analyses lead to the conclusion that using a model more complex than a decision tree or a random forest is difficult to justify for these datasets, but this doesn’t mean that the decision tree is ready for deployment in the real world. The reason behind the success of this algorithm is driven by the over-simplified test bed.

Section 6 – Conclusion

We first saw that CICIDS-2017 is nowadays one of the most popular datasets cited in Google Scholar, and then we saw how new proposals such as NFS-2023-TE and HIKARI-2021 aim to improve the work done with CICIDS-2017. In particular, NFS-2023-TE uses the same raw data as CICIDS-2017 while HIKARI-2021 proposed a new updated test bed.

Then we analyzed the flaws in the current literature and the dataset in use, which has led to the propose of a new methodology seen in other work with the addition of the explainability part.

HIKARI-2021 has been analyzed using the criteria proposed by NFS-2023-TE leading to the discovery of some flaws in the raw data and the dataset generation.

Finally, the analysis of the results has shown how the comparison with simpler models such as the decision tree is necessary to have a benchmark for more complex models. Moreover, we have shown how explainable AI algorithms such as Shap need to be used to help build better models and to increase the accountability of the proposed models. Not only thanks to the explanations, we have discovered something about the dataset but during the entire work has helped to avoid mistakes. It happened more than once that a feature that wasn’t supposed to be in the training set was there and was identified thanks to Shap.

Controllare se è possibile ingegnerizzare down up ratio in nfs.

Aouini, Z., & Pekar, A. (2022). NFStream: A flexible network data analysis framework. *Computer Networks*, *204*. https://doi.org/10.1016/j.comnet.2021.108719

Breiman, L. (2001). Random forests. In *Random Forests* (Vol. 45). https://doi.org/10.4324/9781003109396-5

Brownlee, N., Mills, C., & Ruth, G. (1999). *Traffic Flow Measurement: Architecture*.

Catillo, M., Pecchia, A., Rak, M., & Villano, U. (2021). Demystifying the role of public intrusion datasets: A replication study of DoS network traffic data. *Computers & Security*, *108*, 102341. https://doi.org/10.1016/j.cose.2021.102341

Catillo, M., Pecchia, A., & Villano, U. (2023). Machine Learning on Public Intrusion Datasets: Academic Hype or Concrete Advances in NIDS? *Proceedings - 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks - Supplemental Volume, DSN-S 2023*, 132–136. https://doi.org/10.1109/DSN-S58398.2023.00038

Chen, H., Janizek, J. D., Lundberg, S., & Lee, S.-I. (2020). *True to the Model or True to the Data?* https://github.com/slundberg/shap/blob/

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *13-17-August-2016*, 785–794. https://doi.org/10.1145/2939672.2939785

D’hooge, L., Verkerken, M., Wauters, T., De Turck, F., & Volckaert, B. (2023). Castles Built on Sand: Observations from Classifying Academic Cybersecurity Datasets with Minimalist Methods. *International Conference on Internet of Things, Big Data and Security, IoTBDS - Proceedings*, *2023-April*, 61–72. https://doi.org/10.5220/0011853300003482

Dorogush, A. V., Ershov, V., & Yandex, A. G. (2017). *CatBoost: gradient boosting with categorical features support*. https://github.com/Microsoft/LightGBM

Engelen, G., Rimmer, V., & Joosen, W. (2021). Troubleshooting an Intrusion Detection Dataset: The CICIDS2017 Case Study. *Proceedings - 2021 IEEE Symposium on Security and Privacy Workshops, SPW 2021*, 7–12. https://doi.org/10.1109/SPW53761.2021.00009

Ferriyan, A., Thamrin, A. H., Takeda, K., & Murai, J. (2021). Generating network intrusion detection dataset based on real and encrypted synthetic attack traffic. *Applied Sciences (Switzerland)*, *11*(17). https://doi.org/10.3390/app11177868

Gharib, A., Sharafaldin, I., Habibi, A., & Ghorbani, A. (2016). *An Evaluation Framework for Intrusion Detection Dataset*.

Hofstede, R., Čeleda, P., Trammell, B., Drago, I., Sadre, R., Sperotto, A., & Pras, A. (2014). Flow monitoring explained: From packet capture to data analysis with NetFlow and IPFIX. *IEEE Communications Surveys and Tutorials*, *16*(4), 2037–2064. https://doi.org/10.1109/COMST.2014.2321898

Janzing, D. (2019). *Causal Regularization*.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. *31st Conference on Neural Information Processing Systems*. https://github.com/Microsoft/LightGBM.

Lanvin, M., Gimenez, P. F., Han, Y., Majorczyk, F., Mé, L., & Totel, É. (2023). Errors in the CICIDS2017 Dataset and the Significant Differences in Detection Performances It Makes. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *13857 LNCS*, 18–33. https://doi.org/10.1007/978-3-031-31108-6\_2

Lashkari, A. H., Gil, G. D., Mamun, M. S. I., & Ghorbani, A. A. (2017). Characterization of tor traffic using time based features. *ICISSP 2017 - Proceedings of the 3rd International Conference on Information Systems Security and Privacy*, *2017-Janua*, 253–262. https://doi.org/10.5220/0006105602530262

Lee, W., Stolfo, S. J., & Mok, K. W. (1999). *Mining in a Data-flow Environment: Experience in Network Intrusion Detection*.

Liu, L., Engelen, G., Lynar, T., Essam, D., & Joosen, W. (2022). Error Prevalence in NIDS datasets: A Case Study on CIC-IDS-2017 and CSE-CIC-IDS-2018. *2022 IEEE Conference on Communications and Network Security, CNS 2022*, 254–263. https://doi.org/10.1109/CNS56114.2022.9947235

Lou, Y., Caruana, R., Gehrke, J., & Hooker, G. (2013). Accurate intelligible models with pairwise interactions. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *Part F1288*, 623–631. https://doi.org/10.1145/2487575.2487579

Lundberg, S. M., Erion, G., Chen, H., DeGrave, A., Prutkin, J. M., Nair, B., Katz, R., Himmelfarb, J., Bansal, N., & Lee, S. I. (2020). From local explanations to global understanding with explainable AI for trees. *Nature Machine Intelligence*, *2*(1), 56–67. https://doi.org/10.1038/s42256-019-0138-9

Lundberg, S. M., Erion, G. G., & Lee, S.-I. (2018). *Consistent Individualized Feature Attribution for Tree Ensembles*. http://github.com/slundberg/shap

Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, *2017-Decem*, 4766–4775. https://github.com/slundberg/shap

Moustafa, N., & Slay, J. (2015). UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). *2015 Military Communications and Information Systems Conference, MilCIS 2015 - Proceedings*, 1–6. https://doi.org/10.1109/MilCIS.2015.7348942

Pekar, A., & Jozsa, R. (2024). *Evaluating ML-Based Anomaly Detection Across Datasets of Varied Integrity: A Case Study*. https://intrusion-detection.distrinet-research.be/WTMC2021/tools

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). *“Why Should I Trust You?” Explaining the Predictions of Any Classifier*. https://doi.org/10.1145/2939672.2939778

Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSP 2018 - Proceedings of the 4th International Conference on Information Systems Security and Privacy*, *2018-Janua*(Cic), 108–116. https://doi.org/10.5220/0006639801080116

Shrikumar, A., Greenside, P., Shcherbina, A. Y., & Kundaje, A. (2017). *Not Just A Black Box: Learning Important Features Through Propagating Activation Differences*. https://arxiv.org/

Slack, D., Hilgard, S., Jia, E., Singh, S., & Lakkaraju, H. (2020). Fooling LIME and SHAP: Adversarial attacks on post hoc explanation methods. *AIES 2020 - Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, 180–186. https://doi.org/10.1145/3375627.3375830

Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. *IEEE Symposium on Computational Intelligence for Security and Defense Applications, CISDA 2009*, *Cisda*, 1–6. https://doi.org/10.1109/CISDA.2009.5356528

Yang, J. (2021). *Fast TreeSHAP: Accelerating SHAP Value Computation for Trees*. http://arxiv.org/abs/2109.09847

Zhang, Z., Hamadi, H. Al, Damiani, E., Yeun, C. Y., & Taher, F. (2022). Explainable Artificial Intelligence Applications in Cyber Security: State-of-the-Art in Research. *IEEE Access*, *10*(July), 93104–93139. https://doi.org/10.1109/ACCESS.2022.3204051