Assessment of Machine Learning Techniques for Building an Efficient IDS

Sotirios Panagiotis Chytas
Computer Science and Telecommunications
University of Thessaly
Lamia, Greece
schytas@uth.gr

Leandros Maglaras
School of Computer Science and Informatics
De Montfort University
Leicester, UK
leandros.maglaras@dmu.ac.uk

Abdelouahid Derhab
Center of Excellence in Information Assurance (CoEIA)
King Saud University
Riyadh, Saudi Arabia
abderhab@ksu.edu.sa

George Stamoulis
Computer Science and Telecommunications
University of Thessaly
Lamia, Greece
georges@uth.gr

***Abstract*—Intrusion Detection Systems (IDS) are the systems that detect and block any potential threats (e.g. DDoS attacks) in the network. In this project, we explore the performance of several machine learning techniques when used as parts of an IDS. We experiment with the CICIDS2017 dataset, one of the biggest and most complete IDS datasets in terms of having a realistic background traffic and incorporating a variety of cyber attacks. The techniques we present are applicable to any IDS dataset and can be used as a basis for deploying a real time IDS in complex environments.**

***Index Terms*—Security, IDS, Machine Learning**

I. Introduction

In recent years cyber attacks, especially those targeting Critical National Infrastructures that provide essential information or services are becoming more sophisticated and difficult to detect. The protection of their network and information systems becomes a significant issue [1]. The attacks on such systems vary from reconnaissance attacks, to attempts of penetrating to the internal network and installation and execution of malicious code that can be used in order to steal sensitive data or even change the behavior of specific physical equipment with devastating consequences. A category of attacks, entitles Advanced Persistent Threats (APTs) are stealthy attacks, with the ability to stay undetected, concealing themselves within targets network, and interacting just enough to achieve the defined objectives. For example, APT actors may use zero-day exploits to avoid signature-based detection, and encryption to obfuscate network traffic.

In order to tackle this growing trend academia and industry are joining forces in an attempt to develop novel systems and mechanisms that can defend their systems. Along with other preventive and reactive security tools and solutions that are proposed, such as movel access control and authentication mechanisms, intrusion detection systems (IDS) are deployed as a second line of defense. IDSs can distinguish between normal and malicious actions [2] using either specific rules or patterns of normal behavior of the system.

An Intrusion Detection System (IDS) is a device or software application that monitors a network or system for malicious activity or policy violations. Intrusion detection and prevention systems are primarily focused on identifying possible incidents in systems and networks, logging information, reporting attempts, and learning. IDS have become a necessary addition to the security infrastructure of nearly every organization [3]. Based on the current practical situations, machine learning (ML) technologies have been employed to improve the efficiency of intrusion detection systems, which is one of the most commonly used security infrastructures to protect networks from attacks.

Recently many novel algorithms have been proposed. In [4] authors proposed a hierarchical intrusion detection system based on the combination of three different classifiers, the REP Tree, the JRip algorithm and Forest PA, that can be used for IoT. In another work, authors in [5] proposed a novel IDS that incorporates physical characteristics in order to detect spoofing attacks in a networks of connected vehicles. In a recent survey work [2] the importance of deep learning techniques in detecting novel attacks was revealed.

In this work, we focus on network-level threats and we present the most prominent machine learning ideas that can solve that problem. We use the CICIDS2017 dataset, a huge dataset with more than 2.8 million rows of benign and malignant cases.

Usually IDS are based on [6], [7]:

1) Signature-based detection (Compare against saved signatures. Does not generalize to new threats.)
2) Anomaly-based detection (Detect any deviations from normal behaviours based on data. Does generalize.)
3) Stateful protocol analysis detection (Compare against pre-determined behaviours and patterns. Also does not generalize to new threats.)

978-1-7281-7407-5/20/$31.00 ©2020 IEEE
DOI 10.1109/SMART-TECH49988.2020.00048

Authorized licensed use limited to: UNIVERSITY TENAGA NASIONAL. Downloaded on May 13,2023 at 02:57:13 UTC from IEEE Xplore. Restrictions apply.

Machine learning techniques belong to the second category (Anomaly Detection), and this is the category we are going to analyze.

Although there are many works that present various machine/deep learning algorithms that can be used for an IDS [8]–[13], it seems to be a shortage of papers that analyze possible ways to use these algorithms to build an IDS based on various cases of needs and resources. Our paper's goal is to provide all the well-known techniques that can be used for an IDS. This paper, in combination with papers that offer a survey on specific machine learning algorithms [2], is a valuable document for any scholar who wants to build or customize an IDS based on machine/deep learning techniques. Our contributions in this work are:

- We present 2 major classification categories that are used for an IDS, and analyze them further.
- We provide specific results for all the presented methods in the CICIDS2017 dataset.
- We give insight into many design choices that occur during the machine learning pipeline, e.g, feature importance, threshold tuning, etc.

In section II we briefly present the dataset that we use. In sections III-VI we present and analyze various machine learning methodologies and algorithms that can be used, with more or less successful results. Section VII includes the conclusion and any future steps.

II. Dataset

We used the IDS 2017 dataset [14]. It is one of the most famous and biggest IDS datasets, with more than 2.8 million rows (concatenation of 8 smaller datasets).

A. Labels

The dataset contains data for 14 different attacks (DDOS, XSS, SQL-injection, etc) and normal cases (Benign). The distribution of the labels is provided in Fig. 1.

For this project, we change the names of the labels to BENIGN (0) and MALIGNANT (1) as we care about building an Intrusion Detection System an not an Intrusion Classifier. About 80% of the dataset consists of Benign cases and the rest 20% is various Malignant cases.

B. Features

The dataset consists of 78 features with some of them being

- Destination Port
- FLow Duration
- Total Length of Bwd Packets
- Fwd Packet Length Std

C. Preprocessing

We apply the following preprocessing steps to the dataset.

1) Remove any columns (features) whose most frequent value is encountered in more than 99.9% of the rows.
2) Remove one of the two columns of each pair if their absolute correlation is more than 0.99.

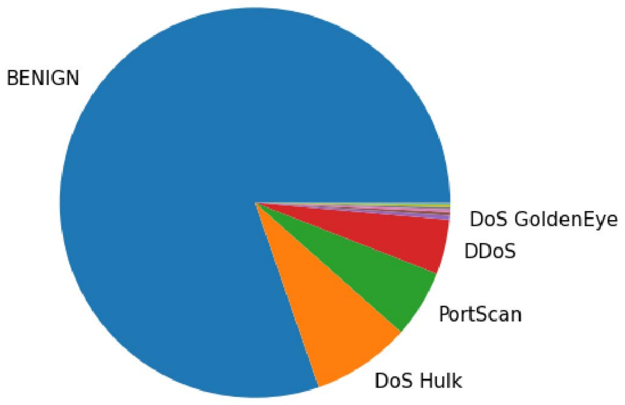

Fig. 1: There are 15 labels in total. However, as we can observe, the distribution is highly unbalanced. We display the names of the 5 most frequently encountered labels.

In that way, we end up with 52 features (originally 78). That makes our training time smaller and the whole training process easier. Obviously, we can proceed with the whole dataset which consists of 78 features. However, we will not observe any increase in our score, and, in fact, in some cases, it is likely to observe a decrease of the score, while the training time will be much greater.

III. Machine Learning techniques - overview

In this section, we provide a brief overview of the machine learning techniques that we analyze, in detail, in the following sections. We should note however that one of the most important aspects of any machine learning project is the dataset's size. It is well known in the machine learning community that more data beat a more sophisticated algorithm that is trained on fewer data.

As you will observe below, we achieve an almost perfect accuracy with a simple algorithm (Decision Tree, Random Forest). Our primary goal of this work is not to find the algorithm that achieves the best accuracy but to list various methodologies that can potentially be used for an IDS.

We separate the techniques that we present in two main categories:

1) Two-class or Multi-class classification
2) One-class classification

Usually, the first category yields better results (Sec. IV). However, we often do not possess a dataset that has benign and malignant cases, but only benign ones, which are much easier to find. In that case, we can only use one-class classification techniques. Although our dataset provides enough malignant cases, we present also the most successful and frequently used one-class classification techniques too (Sec. V).

Authorized licensed use limited to: UNIVERSITY TENAGA NASIONAL. Downloaded on May 13,2023 at 02:57:13 UTC from IEEE Xplore. Restrictions apply.

We used the Scikit-Learn [15] and the Keras [16] libraries for the development and the evaluation of the machine learning models.

IV. Two-class classification

Two-class classification, just like the name states, deals with two types (classes) of data (in our case benign and malignant). Below we present two distinctions of this method. We also should note that the same principles can be also applied to the multi-class classification which deals with more than two classes (e.g. benign, DDoS, SQL-injection, ...).

A. Binary classification

The simplest way is called *binary classification* and assumes we have a dataset with two different labels that are equally important. In table I you can observe the results we achieve with various well-known algorithms. We depict both accuracy and F-score (F-scores gives a better measure of the incorrectly classified cases).

We trained the algorithms with 80% of the dataset (2.25 Million rows) and evaluated their performance at the rest 20% (0.56 Million rows). As we can see, tree-based algorithms achieve amazing results. In table II we have listed the False Positives and the False Negatives of each algorithm in order to get more insight into the performance of each one.

TABLE I: Accuracy and F-score results for the most well-known binary classification algorithms. (neural network's capacity against performance is depicted in more detail, in Fig. 2).

Algorithm	Accuracy	F-score
Logistic Regression	93.5%	83.4%
Decision Tree	99.87%	99.7%
Random Forest	**99.89%**	**99.74%**
Gradient Boosting	99.67%	99.2%
Neural Network	99.5%	98.7%

TABLE II: False positives & false negatives. False positives measure refers to benign cases that were wrongly classified as malignant and false negatives to incorrectly classified true malignant cases.

Algorithm	False Pos	False Neg
Logistic Regression	16696	19827
Decision Tree	345	350
Random Forest	**297**	**280**
Gradient Boosting	913	942
Neural Network	1824	1100

B. Weighted binary classification

Usually, in Intrusion Detection Systems we value more the malignant cases. This means that we care more to correctly classify the malignant cases even if this means we will classify a few benign cases as malignant. Visually, we can think of weighted binary classification as a stretch of the ”valuable” class' region. For example, in Fig. 3, if we value more the red class, we stretch the red region and make it bigger. The bigger

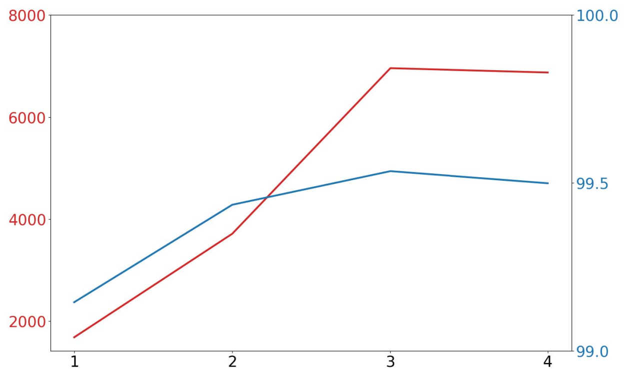

Fig. 2: The change on Neural Network's accuracy (right y-axis) as we increase the network's hidden layers (x-axis). We observe that the improvement is negligible. However, the training time (left y-axis) explodes. In the last case (4 hidden layers) the training time is approximately the same as in the previous case because the network overfits quickly and we used the early stopping technique to prevent that. (We used 100 neurons in each hidden layer. Different sizes lead to similar results).

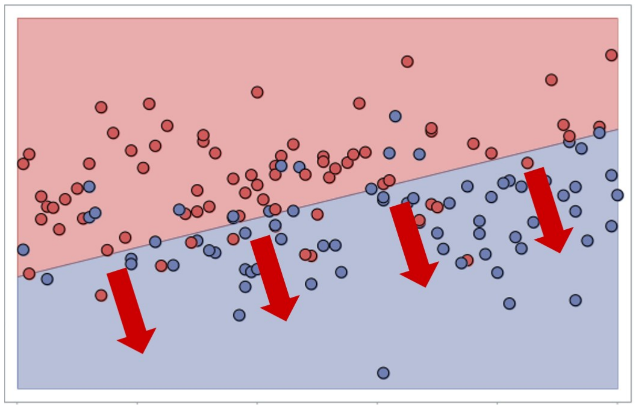

Fig. 3: The effect of the weighted binary classification to the decision boundary of the unweighted bimary classification. The arrows represent the move of the boundary in favor of the red class. The bigger the red class' weight, the bigger the arrows. [17]

the weight, the bigger (theoretically) that stretch is. We can clearly see that there is a trade-off between correctly classified red points, and misclassified blue points.

Some of the machine learning algorithms offer the ability of weighted classification. Thankfully, tree-based techniques (which have the best results in binary classification) have this capability and, in this section, we experiment with that option. In table III you can observe the effect of the weighted classification on the False Positives and False Negatives. We

Authorized licensed use limited to: UNIVERSITY TENAGA NASIONAL. Downloaded on May 13,2023 at 02:57:13 UTC from IEEE Xplore. Restrictions apply.

used a weight of 1 for the Benign cases and a weight of 2 for the Malignant ones. Bigger weight differences do not offer an improvement, and, in fact, decrease the performance (Fig. 4).

TABLE III: False positives and false negatives for the weighted cases. Inside the parenthesis we present the results of the previous section i.e. same weights. The decrease of the false negatives is obvious as well as the increase of the false positives.

Algorithm	False Pos	False Neg
Logistic Regression	27355 (16696)	8785 (19827)
Decision Tree	355 (345)	**242** (350)
Random Forest	**345** (297)	259 (280)

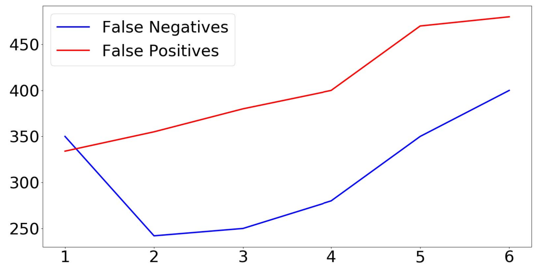

Fig. 4: The change on FN and FP (y-axis) based on the weight (x-axis) we apply to the malignact class (benign has always weight=1). We can see that too big values lead to worse results for both classes. We used Weighted Decision Trees for the graph.

C. *Threshold tuning*

To further improve the results of the algorithms, we can tune the probability threshold. By default, when a classifier outputs a probability of benign higher of 0.5 then we predict that this case is a benign one and malignant otherwise. However, we can alter this threshold. For example, we may set it to 0.7 if we value more the malignant cases, or to a value lower than 0.5 if we care more for the benign cases.

In order to properly tune the threshold without overfitting to the testing dataset, we need a new, unseen dataset. The training pipeline is depicted in Fig. 5. In Fig. 6 you can observe the effect of the threshold on False Positives and False Negatives for the unweighted Random Forest.

V. One-class classification

A usual case in anomaly detection problems is the one-class classification. Many datasets provide only benign cases (not the case here) and we are required to build an algorithm that can extract knowledge from these cases in order to accurately detect any malignant cases.

The one-class classifiers try to ”understand” the concept of the given class, with various techniques. This is a much harder

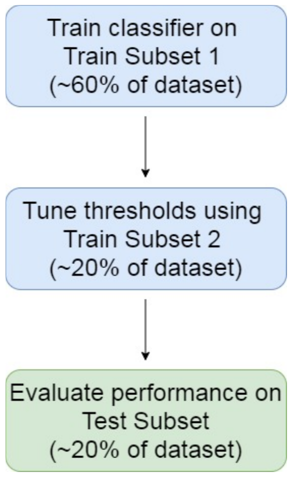

Fig. 5: Threshold tuning pipeline

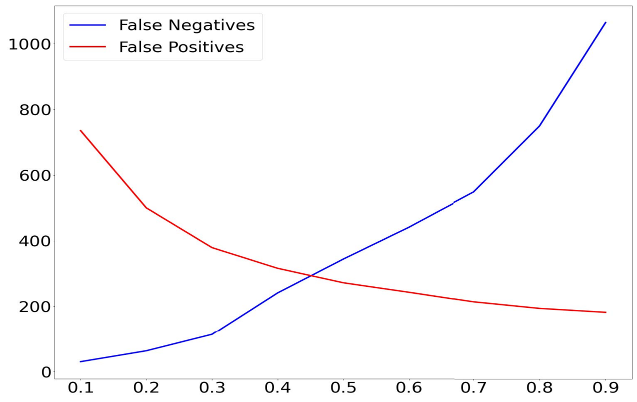

Fig. 6: The change on FN and FP (y-axis) based on the threshold (x-axis) we apply to the predicted probabilities.

task than binary classification, because the model’s ”world” is just this given class. In simple terms, the most common technique is to understand the true values of each feature as well as for combinations of these features, so that they will be able to detect any big deviations from these ranges, and mark these cases as outliers (malignant cases). Consequently, such algorithms perform much worse than the binary classifiers presented above.

A. *Isolation Forest*

Although our dataset provides both benign and malignant cases, we use only the benign cases as training dataset and we train an Isolation Forest, one of the most well-known one-class classifiers. The results we obtain are far worse than the ones we achieve when we train a binary classifier. Specifically, we achieve accuracy of 83.5% and F-score of 50.9% (30K False Positives and 63K False Negatives)

Authorized licensed use limited to: UNIVERSITY TENAGA NASIONAL. Downloaded on May 13,2023 at 02:57:13 UTC from IEEE Xplore. Restrictions apply.

Another well-known algorithm is the one-class SVM. However, its complexity is too high for such a huge dataset (in terms of both rows and columns) [18]. We mention it though because it is one of the few one-class classifiers and it can be very useful with even better results than Isolation Forest for smaller datasets, or for someone who possesses greater computational power.

B. Autoencoder

Many recent works [19], [20] have explored the idea of using an autoencoder to detect any anomalies. An autoencoder architecture trained on normal data (benign) is very likely to not be able to properly reconstruct a ”weird” example (malignant). The big advantage of this idea is the use of only benign cases (which are abundant in contrast to malignant ones) during the training phase.

We train an autoencoder using the 80% of the benign cases, at the architecture of the Table IV. After having trained the autoencoder, we have to determine the reconstruction error threshold in a similar way as in the previous section (Fig. 5). In Fig. 7 you can observe how the Accuracy and the F-score change as we change the threshold value. The results we get are much worse than these of the previous methods but similar to the other one-class classification algorithms. Careful tuning of the architecture and techniques such as Dropout [21] may improve the results but they will never be as good as the results we get with the tree-based methods. However, if the dataset consists only of benign cases, then one-class classification methods are the only available.

TABLE IV: Autoencoder architecture. We used the ReLU activation for all the hidden layers. The input is real-valued so the activation of the final layer should also be linear (no activation).

Encoder layer 1 (Input)	input size - Linear
Encoder layer 2	40 - ReLU
Encoder layer 3	30 - ReLU
Latent representation	20 - ReLU
Decoder layer 1	30 - ReLU
Decoder layer 2	40 - ReLU
Decoder layer 3 (Output)	input size - Linear

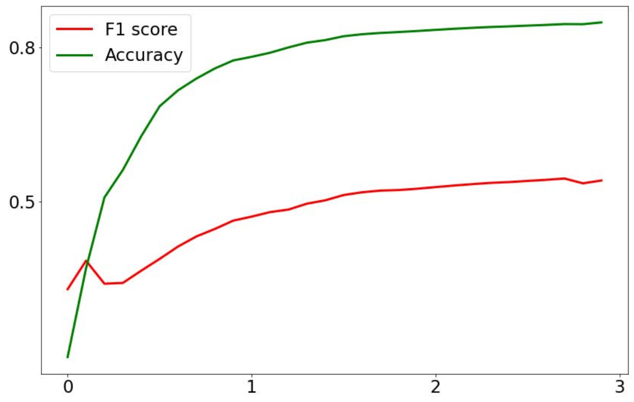

Fig. 7: The x-axis corresponds to different thresholds for the reconstruction error. Accuracy reaches as high as 80% while F-score reaches the value of 50%.

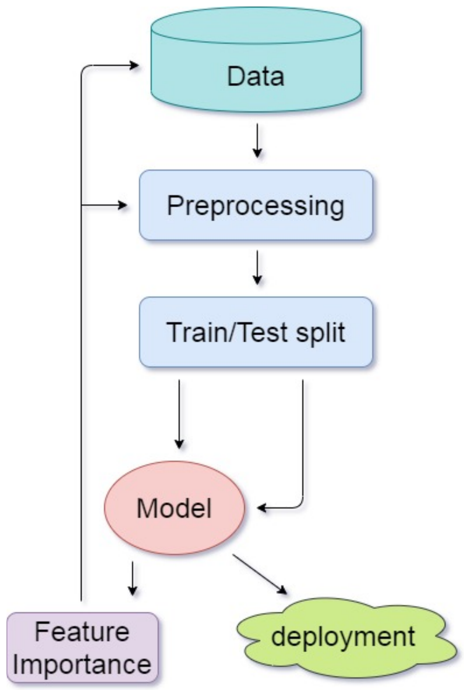

Fig. 8: The whole training pipeline. After preprocessing our data (Sec. II), we train the algorithms (Sec. IV, V), and use feature importance to improve the results by altering the dataset (Sec. VI)

VI. Feature importance

Lastly, we can use well-known machine learning algorithms in order to get more information about our features. In Fig. 9, we can observe how three algorithms (Logistic Regression, Decision Tree, Random Forest) ”rank” the features. For instance, we can observe that the feature ”Bwd Packet Length Std” is one of the most important features in all three algorithms.

This part provides information to the data analytics part and can lead to the decision to add some features that may help achieve better results. Additionally, we may choose to remove features that seem to be ”useless” and they just increase the training time without adding any value to the classification process. We depict the whole training pipeline in Fig. 8.

VII. Conclusion - Next steps

In this work, we presented various machine learning techniques that can satisfactorily solve the IDS problem. We tested every algorithm with the CICIDS2017 dataset, one of the biggest and most complete IDS datasets. We obtained the best results with Random Forest and weighted Decision Tree.

Authorized licensed use limited to: UNIVERSITY TENAGA NASIONAL. Downloaded on May 13,2023 at 02:57:13 UTC from IEEE Xplore. Restrictions apply.

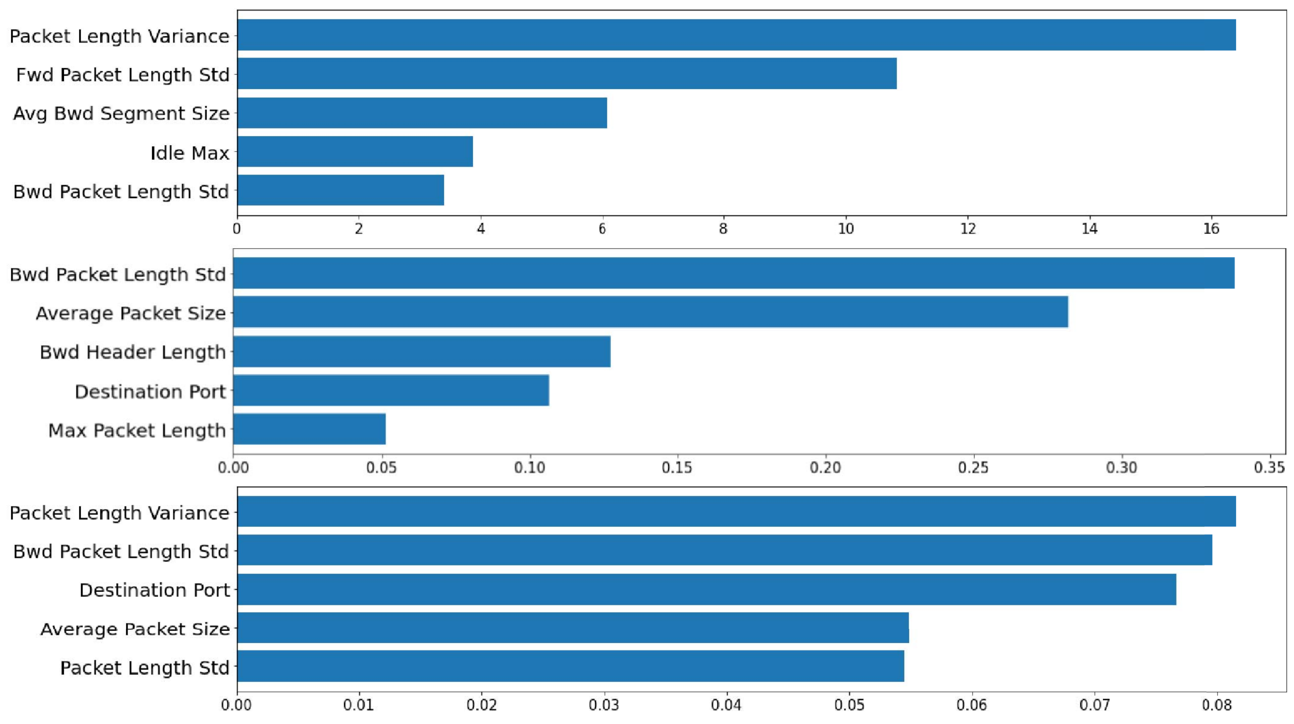

Fig. 9: From top to bottom. (1) Top-5 absolute coefficients of the Logistic Regression method. (2) Top-5 Weighted Decision Tree importances. (3) Top-5 Weighted Random Forest importances.

The next steps include incorporating these models into a server and test them against our own, new data in a continuous, real-time fashion. After being sure of the effectiveness of the model, we can use them for real-world applications (Deployment step as depicted in Fig, 8), run stress tests and conduct thorough efficiency and communication overhead analysis.

REFERENCES

[1] L. A. Maglaras, K.-H. Kim, H. Janicke, M. A. Ferrag, S. Rallis, P. Fragkou, A. Maglaras, and T. J. Cruz, "Cyber security of critical infrastructures," *Ict Express*, vol. 4, no. 1, pp. 42–45, 2018.

[2] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *Journal of Information Security and Applications*, vol. 50, p. 102419, 2020.

[3] M. Martellini and A. Malizia, *Cyber and Chemical, Biological, Radiological, Nuclear, Explosives Challenges: Threats and Counter Efforts*. 2017.

[4] M. A. Ferrag, L. Maglaras, A. Ahmim, M. Derdour, and H. Janicke, "Rdtids: Rules and decision tree-based intrusion detection system for internet-of-things networks," *Future Internet*, vol. 12, no. 3, p. 44, 2020.

[5] D. Kosmanos, A. Pappas, L. Maglaras, S. Moschoyiannis, F. J. Aparicio-Navarro, A. Argyriou, and H. Janicke, "A novel intrusion detection system against spoofing attacks in connected electric vehicles," *Array*, vol. 5, p. 100013, 2020.

[6] M. Whitman and H. Mattord, *Principles of Information Security*. Thomson Course Technology, 2009.

[7] E. Kirda, S. Jha, and D. Balzarotti, "Recent advances in intrusion detection," *Proc. 12th International Symposium, RAID 2009, Saint-Malo, France.*, 9 2009.

[8] S. S. S. Sindhu, S. Geetha, and A. Kannan, "Decision tree based light weight intrusion detection using a wrapper approach," *Expert Systems with applications*, vol. 39, no. 1, pp. 129–141, 2012.

[9] S. A. Mulay, P. Devale, and G. Garje, "Intrusion detection system using support vector machine and decision tree," *International Journal of Computer Applications*, vol. 3, no. 3, pp. 40–43, 2010.

[10] P. B. Hasan M., Nasser M. and A. S., "Support vector machine and random forest modeling for intrusion detection system (ids)," *Journal of Intelligent Learning Systems and Applications*, vol. 6, no. 6, pp. 45–52, 2014.

[11] Y. Kim, M. S. Johnson, S. H. Knox, T. A. Black, H. J. Dalmagro, M. Kang, J. Kim, and D. Baldocchi, "Gap-filling approaches for eddy covariance methane fluxes: A comparison of three machine learning algorithms and a traditional method with principal component analysis," *Global Change Biology*, vol. 26, no. 3, pp. 1499–1518, 2020.

[12] S. A. R. Shah and B. Issac, "Performance comparison of intrusion detection systems and application of machine learning to snort system," *Future Generation Computer Systems*, vol. 80, pp. 157–170, 2018.

[13] S. K. Biswas, "Intrusion detection using machine learning: A comparison study," *International Journal of Pure and Applied Mathematics*, vol. 118, no. 19, pp. 101–114, 2018.

[14] https://www.unb.ca/cic/datasets/ids-2017.html.

[15] https://scikit-learn.org/stable/.

[16] https://keras.io/.

[17] https://blogs.sas.com/content/iml/2017/07/17/prediction-regions-classification.html.

[18] O. Chapelle, "Training a support vector machine in the primal," *Neural computation*, vol. 19, pp. 1155–78, 06 2007.

[19] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *Special Lecture on IE*, vol. 2, no. 1, 2015.

[20] M. Sakurada and T. Yairi, "Anomaly detection using autoencoders with nonlinear dimensionality reduction," in *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, pp. 4–11, 2014.

[21] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

Authorized licensed use limited to: UNIVERSITY TENAGA NASIONAL. Downloaded on May 13,2023 at 02:57:13 UTC from IEEE Xplore. Restrictions apply.