Impact of Feature Encoding on Malware Classification Explainability

Elyes Manai
Department of Computer Science & Software Engineering
Laval University, Quebec, Canada.
Elyes.Manai.1@ulaval.ca

Mohamed Mejri
Department of Computer Science & Software Engineering
Laval University, Quebec, Canada.
Mohamed.Mejri@ift.ulaval.ca

Jaouhar Fattahi
Department of Computer Science & Software Engineering
Laval University, Quebec, Canada.
Jaouhar.Fattahi.1@ulaval.ca

2023 15th International Conference on Electronics, Computers and Artificial Intelligence (ECAI) | 979-8-3503-2138-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/ECAI58194.2023.10193964

***Abstract*—This paper investigates the impact of feature encoding techniques on the explainability of XAI (Explainable Artificial Intelligence) algorithms. Using a malware classification dataset, we trained an XGBoost model and compared the performance of two feature encoding methods: Label Encoding (LE) and One Hot Encoding (OHE). Our findings reveal a marginal performance loss when using OHE instead of LE. However, the more detailed explanations provided by OHE compensated for this loss. We observed that OHE enables deeper exploration of details in both global and local contexts, facilitating more comprehensive answers. Additionally, we observed that using OHE resulted in smaller explanation files and reduced analysis time for human analysts. These findings emphasize the significance of considering feature encoding techniques in XAI research and suggest potential for further exploration by incorporating additional encoding methods and innovative visualization approaches.**

***Index Terms*—Explainability, XAI, feature encoding, malware classification, preprocessing, LE, OHE, XGBoost.**

I. Introduction

Machine learning has witnessed remarkable advancements in recent years, enabling the development of sophisticated models that achieve impressive performance on various tasks. As these tasks and the data they are trained on become more complex, so does the model complexity. This often causes the decision-making process to lack transparency, making it difficult to understand the reasons behind their predictions. In a society that uses AI for an ever-growing number of use cases, however, that lack of understanding can pose serious risks to the users. Averting these risks and allowing more control over what our AI is doing, thus allowing more responsible AIs, is the goal behind the Explainable Artificial Intelligence (XAI) subfield. This subdomain of AI focuses on making black-box models transparent by providing understandable explanations for their decisions. XAI also allows us to combine the powerful pattern-recognition learning capabilities of AI with human-readable explanations that humans can instinctively understand and explain. the algorithms used in XAI usually work by finding out what parts of the input and of the model weights most affect the model's predictions. The end result will be a summary of each feature's contribution to the model. How helpful are these summaries, however, which we can call the quality of the generated explanations, depends on several parameters such as the chosen algorithm, the model architecture, and the data preprocessing technique. This last parameter, however, is not as popular as the others. While most XAI research focuses on algorithms, use cases, and the quality of explanations generated, there is a lack of research on the impact of preprocessing on generated explanations. We think that the preprocessing technique has a sizable impact on the quality of generated explanations and should be more explored. More specifically, we are interested in the feature encoding step of the preprocessing pipeline. Since XAI methods summarize feature contribution, the way we encode our models will directly affect the understandability of the generated explanations. Since preprocessing directly affects model performance, considerations must be taken to not trade off too much performance for better explanations, as better explanations on an unprecise model are not useful. Nonetheless, we think that a minor performance loss for a major boost in explainability is worth it, as it also opens up the door for better model and data understanding, bias discovery, robustness tests, and overall higher quality assurance. This is especially important in critical industries such as Medicine, Finance, and Cyber Security. To showcase the added value of our idea in a real use case, we will apply Machine Learning and Explainability on a common problem in Cyber security: Malware Classification. It is one of the most common tasks that Machine Learning is applied to in modern antiviruses and Intrusion Detection Systems. We will train a model on a publicly available malware dataset, apply the XAI algorithm, switch the preprocessing technique and compare the generated explanations. We will show that new rules and pain points can be detected and further explored by just changing the preprocessing technique. To the best of our knowledge, no prior studies specifically addressed the subject of the direct impact of preprocessing on explanation quality in the field of XAI have been identified in the existing literature. Our comprehensive review of the literature revealed that research in XAI is more geared towards XAI algorithms [1]–[4], the generated explanations [5], [6], alternative ways to bake explainability into the input features [7], [8] and other related problems [9], [10]. Our focus in this paper can be summarized as follows : Given that XAI algorithms use the input features

979-8-3503-2138-8/23/$31.00 ©2023 IEEE

as the key components for the generated explanations, it is safe to assume that the type of feature encoding used will directly affect the clarity of the explanations. The more explicit the feature, the more detailed should be the explanation we get. With that in mind, we will study two main questions in this paper : Does feature encoding affect explainability? If yes, what encoding yields better explainability and why?

II. Concepts

A. Feature Encoding

Feature encoding, also known as feature transformation or feature representation, is a crucial step in data preprocessing where categorical or textual features are converted into numerical representations that can be effectively used by machine learning algorithms. This is a mandatory step as ML algorithms only deal with numerical features. The choice of encoding technique directly impacts the ML performance. Here are some common feature encoding techniques:

- **One-Hot Encoding:** Each category within a categorical feature is represented by a binary feature. If a feature has n categories, it is encoded into n binary features, where only one feature is active (1) for a particular category, and the rest are inactive (0). One-hot encoding is useful when there is no inherent order or relationship among the categories.
- **Label Encoding:** Label encoding assigns a unique numerical label to each category within a categorical feature. Each category is represented by a distinct integer value. Label encoding is suitable when the categories have an ordinal relationship or when using algorithms that can directly work with integer inputs.
- **Ordinal Encoding:** Similar to label encoding, ordinal encoding assigns numerical labels to categories. However, ordinal encoding takes into account the order or rank of the categories and assigns values accordingly. For example, ”low,” ”medium,” and ”high” could be encoded as 1, 2, and 3, respectively.
- **Binary Encoding:** Binary encoding represents categories as binary bit patterns. Each category is assigned a unique binary code, and each bit in the code represents the presence or absence of a category. Binary encoding can be efficient for high-cardinality categorical features and reduces the dimensionality compared to one-hot encoding.
- **Embedding:** Embedding techniques are commonly used for encoding textual or high-dimensional categorical features. Embeddings are dense, low-dimensional representations that capture semantic relationships between categories. Embeddings are learned using techniques like Word2Vec [11], [12] or categorical embedding layers in deep learning models [13].

B. Explainability

Explainability in the context of machine learning [14]–[16] refers to the ability to understand and interpret the decisions or predictions made by a machine learning model. It involves gaining insights into how and why a model arrives at a particular output, providing transparency and comprehensibility to the decision-making process. There are various approaches to achieving explainability:

- **Model-Agnostic Approaches:** These methods aim to explain any black-box machine learning model without relying on its internal structure. They involve techniques like feature importance analysis, partial dependence plots [17], and surrogate models, which provide insights into the relationship between input features and model predictions.
- **Rule-Based Approaches:** These approaches aim to generate human-readable rules that describe the decision-making process of the model. Rule-based models, such as decision trees or rule lists, can provide explicit if-then statements that explain how specific features influence predictions.
- **Interpretable Model Architectures:** Some machine learning models, such as linear regression, logistic regression, or decision trees, inherently provide interpretable explanations. Their simplicity and transparency allow users to understand the impact of each feature on the final prediction.
- **Local Explanations:** Local explanation methods focus on explaining individual predictions rather than the model as a whole. Techniques like LIME [2] (Local Interpretable Model-Agnostic Explanations) or SHAP [1] (SHapley Additive exPlanations) provide insights into which features contributed the most to a particular prediction.
- **Visualizations:** Visualizations play a significant role in explaining complex models and high-dimensional data. Techniques like heatmaps, bar plots, scatter plots, or saliency maps help in visualizing feature importance, decision boundaries, or highlighting influential regions in the data.

C. Malware Detection

To demonstrate our work, we will take the common task of detecting malware. Malware are malicious pieces of software that are designed to infiltrate and damage information systems without the users' consent [18]–[22]. The term malware covers a lot of categories such as viruses, ransomware, worms, trojans, backdoors, spyware, keyloggers, adware, bots, and rootkits. Malware analysts have to discover exactly what happened to a system and make sure that the machines damaged by malicious software are isolated from the organization's network. The analysis done to single out the suspicious parts of the software can sometimes take a group of analysts and several hours or even days. Since undetected malware can have devastating consequences on any organization, malware detection has been deemed one of the most important tasks in cybersecurity. Several types of systems have been built to detect and capture malware such as Intrusion detection systems, antiviruses and firewalls, and these systems keep getting smarter thanks to the combined shared knowledge of the cyber security community and the rapid advancement of

technology. Current Malware detection systems use Machine Learning and Deep Learning to detect anomalies in files and network packets to protect the systems they're installed on. Since Machine learning has been known for its fantastic classification capabilities, more and more complex architectures and models are being tested and deployed to the current market.

III. Implementation and experimental results

A. Dataset

For this project, we found a Malware classification dataset from the 2015 Microsoft Malware Classification Challenge [23]. The public variant we managed to download contains 19611 rows and 78 features. Each row represents a single file. The dataset is imbalanced as there are 14599 malware files and 5012 non-malware files, so 3 times as much malware. The dataset has no missing data and all features are numerical aside from the "Name" one.

B. Preprocessing

The "Name" feature has been modified by the competition organizers to include "virus" if the file is malware and thus be removed since it does not represent real-life data. We do not apply any other preprocessing on the data aside from feature encoding. In this work, we apply two encoding techniques to all the features:

- **Label Encoding:** Each feature value is represented by a unique integer.
- **One Hot Encoding:** Each feature value becomes a separate binary column where 1 means the file's value of that feature is the column name, and 0 if not. This allows for more precise knowledge of what went wrong.

C. Machine Learning Modeling

For training, we choose XGBoost [24], [25] as our base model and train it using its default parameters, namely 100 estimators, a max depth of 5 and a learning rate of 0.1. We use the free Google Colab coding environment which offers a single sever with 12.7GB of RAM and a single NVIDIA T4 GPU with 15GB of GPU RAM. To evaluate our model, We use four popular metrics: Accuracy, Precision, Recall and F1. In a nutshell, accuracy measures the overall correctness of the model's predictions by calculating the proportion of correctly classified instances out of the total number of instances. Precision quantifies the proportion of true positive predictions out of all positive predictions made by the model, indicating the model's ability to correctly identify positive instances and minimize false positives. Recall measures the proportion of true positive predictions out of all actual positive instances in the dataset, representing the model's ability to capture positive instances and minimize false negatives. Finally, the F1 score combines precision and recall into a single metric by taking their harmonic mean, providing a balanced assessment of the model's accuracy and considering both false positives and false negatives. We showcase the performance results of XGBoost on the label encoded dataset in Table I.

TABLE I
XGB results on both encoding techniques

Encoding Technique	F1	Accuracy	Precision	Recall
Label Encoding	0.991	0.993	0.992	0.998
One Hot Encoding	0.988	0.989	0.987	0.998

Although we did not preprocess our data, aside from encoding them differently, we managed to get pretty good results. We can therefore directly go to the explainability part.

D. Explainability

For starters, we are going to take away non useful features because one hot encoding all 77 features created 85102 features, which kept crashing our environment due to insufficient RAM. To do that, we will use XGBoost's built in feature importance function to list each feature's impact on the model's decision making. In Table II, we extract the top 10 influential features and sort them from most to least important.

TABLE II
Feature Importance according to XGBoost

Rank	Feature	Importance score
1	MajorSubsystemVersion	0.6215
2	Subsystem	0.1362
3	MinorOperatingSystemVersion	0.0575
4	MajorLinkerVersion	0.0454
5	SizeOfStackReserve	0.0198
6	Characteristics	0.0100
7	SectionMaxChar	0.0084
8	ImageBase	0.0081
9	SizeOfHeapReserve	0.0081
10	TimeDateStamp	0.0059

According to Table II, the combined score of the 10 most important features are 0.9381 which means that they represent 93.81% of the model's decision making power. We, therefore, can just keep these 10 features and not use the rest. Doing so, we get the results shown in Table III.

Comparing the results shown in Table III to those in Table I show that although we did lose a bit of performance, the drop is marginal (less than 1%). This means that if the One Hot encoding does provide us with more explainability power, it would be recommended to use. For the next par, we will use a dedicated Explainability Algorithm called Shapley Additive Explanations (SHAP) to dig deeper into the model's inner reasoning.

1) The SHAP algorithm: SHAP [1], [26] was introduced in 2017 and provides a unified way of explaining the contribution of each input feature to the final prediction of the model, based on calculated values called Shapley values. A Shapley value is a measure of the marginal contribution of a feature

TABLE III
XGBoost results on the top 10 features

Encoding Technique	F1	Accuracy	Precision	Recall
Label Encoding	0.992	0.992	0.991	0.998
One Hot Encoding	0.985	0.985	0.983	0.998

to the prediction, averaged over all possible combinations of features in the dataset. To calculate the Shapley values for a particular prediction, SHAP applies a game-theoretic approach based on the concept of cooperative games. It considers each feature value as a "player" in the game and computes the contribution of each player to the final prediction. It then calculates the average contribution of each player across all possible coalitions of players, weighting each coalition by its probability of occurrence. This approach results in a set of Shapley values, which represent the relative importance of each feature to the prediction for a specific instance. These Shapley values can be used to generate an explanation for the prediction, showing which features had the greatest impact and how they affected the final outcome. The mathematical formula used by SHAP to generate the Shapley Values is presented in Figure 1.

$$\phi_i(x) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} [f(x_S \cup \{x_i\}) - f(x_S)] \tag{1}$$

Once generated, SHAP uses these values to display plots for both global explanations and local explanations.

2) Global feature importance: We use the SHAP algorithm to generate global summary plots that highlight the importance of each feature in the model's decision-making similarly to what we have done in Table II. Figures 1 and 2 display the importance plots for the Label Encoded dataset and the One Hot Encoded dataset, respectively.

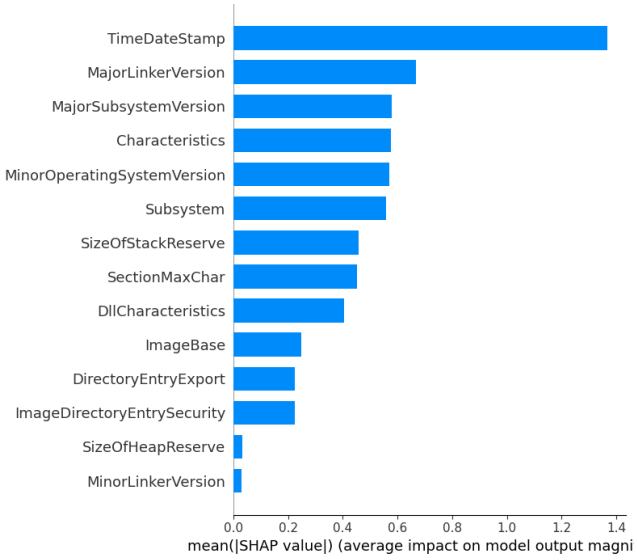

Figure 1. Label Encoding global importance plot

IV. Discussion

The main difference between these plots is that while we know what feature is more important with Label Encoding, we know what exact value of that feature is more important with One Hot Encoding. This means that we get more specificity

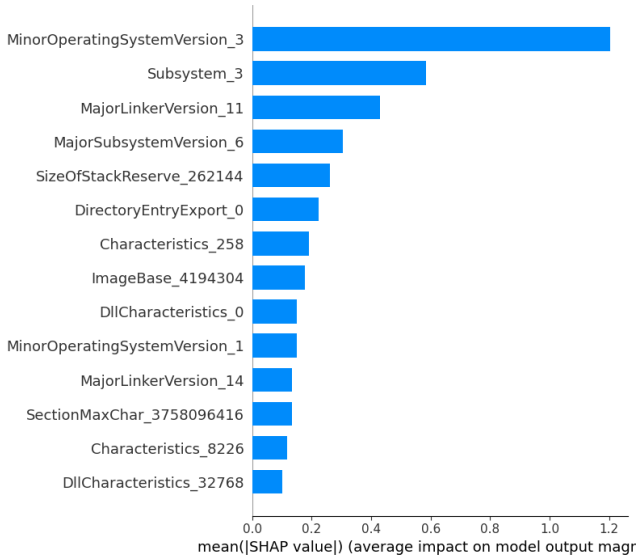

Figure 2. One Hot Encoding global importance plot

as a feature's importance is the sum of the importance of its unique values. A top ranking feature in the Label Encoding model could have therefore reached its rank because of the importance of some of its values, but not the others. Using One Hot Encoding, we can single out what values exactly are the most relevant to further analyze. For example, the "MinorOperatingSystemVersion" feature in 1 has a mean SHAP value of almost 0.6, ranking fifth. However, in 2, we can see that is actually Version 3 of this feature that is really impactful, ranking first with a mean SHAP value of more than 1.2. Yet, version 1 of this feature only has a score of almost 0.2, and the rest of the version are not in the top 10 features. So using One Hot Encoding, we can single out files with the Version 3 of "MinorOperatingSystemVersion" and further analyze them separately in hopes of creating an easy rule for them or see what more we can learn. One drawback of this plot is that it is not easy to read when we have hundreds or thousands of features. In this example, we have 16087 features. It will be unproductive to use this plot to study feature importance. Instead, we can extract the raw SHAP values of all one hot encoded features, group them by original feature, and plot them side by side in another plot. We propose the plots in Figures 3 and 4 where we plot the importance of the different values of the "MajorSubsystemVersion" feature side by side, horizontally and vertically respectively. We chose this feature instead of the number 1 ranking "MinorOperatingSystemVersion" feature because it has considerably fewer distinct values making it easier to plot, wasting less space and delivering the same message. These figures allow us to better visually grasp the relativity in importance between the different values of a feature. This way, we can add or remove values to and from a watchlist and also construct rules for particular values. We can now combine this with the confidence score of the model at inference to start a routine, a check or apply a rule when

the score doesn't hit the certainty thresshold. At that point, we would start investigating individual instances, thus needing different explanations called local explanations.

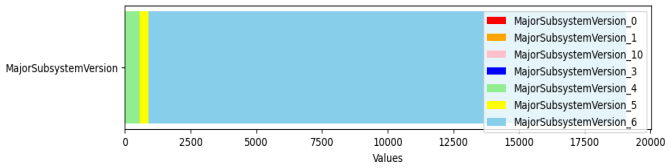

Figure 3. Horizontally stacked bar plot of the "MajorSubsystemVersion"'s distinct values importance

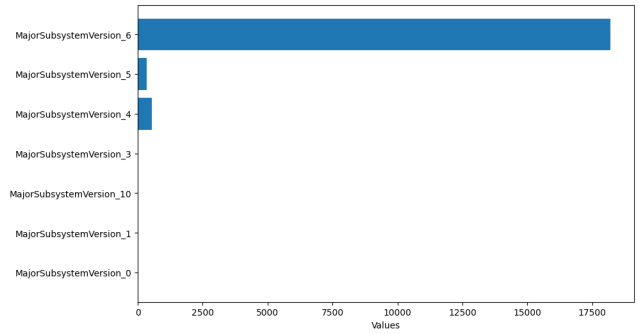

Figure 4. Vertically stacked bar plot of the "MajorSubsystemVersion"'s distinct values importance

1) local feature importance: Local explanations focus on individual instances, displaying to the user the step-by-step contribution of each feature on the model's decision. Using SHAP's local explanation plots, we get Figures 5 and 6 which display the local explanation of instances 2 and 3 respectively, first using Label Encoding first and then One Hot Ecoding.

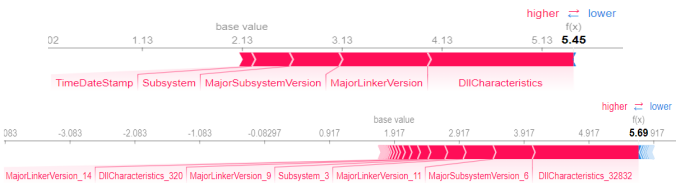

Figure 5. Local explanation for test observation number 2

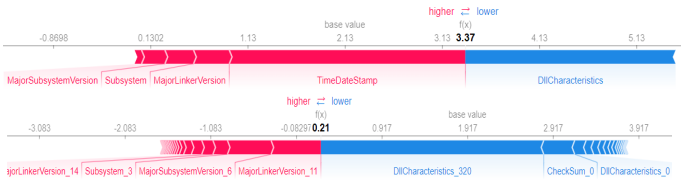

Figure 6. Local explanation for test observation number 3

Again, the added refinement of the exact feature value gives us a lot more insight into what pushed the model towards a certain classification. Although the one hot encoding in this case may seem useless since we already know what value of each feature the instance holds, it instead can be used as an assertion method to make sure there are no anomalies in the decision shifting. Finally, we can see that being trained on the individual values changes the base value and decision shift intensity of each feature, as it has been trained on more finegrained data and the model had the chance to learn combinations that go together. These combinations in a tree based model such as XGBoost can then be used extracted and used as normal conditional IF rules or analyzed to detect vulnerabilities that went under the radar. Even then, the feature encoding will have an impact on the generated rules.

2) IF-Rules: IF-Rules are logical statements that express conditional relationships between input variables and output decisions and follow a simple structure: IF a specific condition or set of conditions is satisfied, THEN a particular action or decision should be taken. The conditions and actions are typically expressed using logical operators, such as "AND," "OR," and "NOT." IF rules provide a transparent and interpretable way to encode domain knowledge and decision-making criteria into a system. Due to their nature, tree-based models can be seen as a collection of IF rules combined together to form a decision-making process. Each node in a decision tree represents an IF statement on a specific feature or attribute, and the tree structure guides the flow of decision-making based on these conditions. The splitting criteria at each node determine the conditions for branching into different paths, leading to subsequent nodes or leaves with specific outcomes or predictions. Since XGBoost is a tree based model, we can extract the IF-Rules it learned during the training phase and use them to build logical pipelines or to study them. An example of the IF-Rules learned by our XGBoost model can be seen in Figures 7 and 8 for Label Encoding and One Hot Encoding respectively.

```
booster[0]:
0:[MajorSubsystemVersion<6] yes=1,no=2,missing=2,gain=6773.84131,cover=3922
        1:[Subsystem<3] yes=3,no=4,missing=4,gain=1787.29395,cover=3138
                3:[TimeDateStamp<1445356928] yes=7,no=8,missing=8,gain=315.146484,cover=2828.5
                        7:[DllCharacteristics<34144] yes=13,no=14,missing=14,gain=4.33398438,cover=2770.25
                                13:leaf=0.193269417,cover=2762.5
                                14:[DllCharacteristics<34816] yes=25,no
```

Figure 7. Example IF-Rules for Label Encoding

```
booster[0]:
0:[MajorSubsystemVersion_6<1] yes=1,no=2,missing=2,gain=6620.83057,cover=3922
        1:[Subsystem_3<1] yes=3,no=4,missing=4,gain=1913.71191,cover=3149.75
                3:[DllCharacteristics_320<1] yes=7,no=8,missing=8,gain=167.949219,cover=2831.75
                        7:[DllCharacteristics_352<1] yes=15,no=16,missing=16,gain=77.3671875,cover=2765.75
                                15:[DllCharacteristics_448<1] yes=25,no=26,missing=26,gain=63.005859
```

Figure 8. Example IF-Rules for One Hot Encoding

While there is no apparent difference between the IF-Rules of the two encoding techniques, the difference lies in the metadata. In Table IV, we can see the difference in the rules' total text length in number of characters as well as the explanation file size in KB. We can see that One Hot Encoding resulted in less characters which means less file size. The indirect consequence of this is less analysis time, less system complexity and less ambiguity, all of which directly benefit analysts and systems.

TABLE IV
DIFFERENCE IN IF-RULES BETWEEN THE ENCODING TECHNIQUES

Encoding Technique	F1	Rules file size	Rules text length
Label Encoding	0.991	221 KB	226065 characters
One Hot Encoding	0.982	175 KB	180237 characters

V. CONCLUSION

In this paper, we studied the impact of feature encoding on the explainability of XAI algorithms. We took a malware classification dataset as an example on which we trained an XGBoost model. We tried two different types of feature encoding: Label Encoding and One Hot Encoding and found there is a marginal performance loss by the model by using OHE instead of LE. That loss was made up with thanks to the more detailed explanations we managed to make thanks to OHE. We found that OHE allows us to go deeper in the details when searching for answers, both globally and locally. We also found that using OHE yields smaller explanation files and results in less time spent analyzing by human analysts. We think this is an interesting aspect to be taken into consideration when working with XAI and could be expanded by including more feature encoding techniques and more creative plots.

REFERENCES

[1] S. M. Lundberg and S. Lee, "A Unified Approach to Interpreting Model Predictions," *CoRR*, vol. abs/1705.07874, 2017. [Online]. Available: http://arxiv.org/abs/1705.07874

[2] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why Should I Trust You?: Explaining the Predictions of Any Classifier," *CoRR*, vol. abs/1602.04938, 2016. [Online]. Available: http://arxiv.org/abs/1602.04938

[3] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic Attribution for Deep Networks," in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 2017, pp. 3319–3328. [Online]. Available: http://proceedings.mlr.press/v70/sundararajan17a.html

[4] M. T. Ribeiro, S. Singh, and C. Guestrin, "Anchors: High-Precision Model-Agnostic Explanations," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/11491

[5] K. Sokol and P. A. Flach, "One Explanation Does Not Fit All," *Künstliche Intell.*, vol. 34, no. 2, pp. 235–250, 2020. [Online]. Available: https://doi.org/10.1007/s13218-020-00637-y

[6] F. Cabitza, A. Campagner, G. Malgieri, C. Natali, D. Schneeberger, K. Stoeger, and A. Holzinger, "Quod erat demonstrandum? - Towards a typology of the concept of explanation for the design of explainable AI," *Expert Systems with Applications*, vol. 213, p. 118888, 09 2022.

[7] S. R. Islam, W. Eberle, S. K. Ghafoor, A. Siraj, and M. Rogers, "Domain Knowledge Aided Explainable Artificial Intelligence for Intrusion Detection and Response," in *AAAI-MAKE 2020, Palo Alto, CA, USA, March 23-25, 2020, Volume I*, ser. CEUR Workshop Proceedings, vol. 2600. CEUR-WS.org, 2020. [Online]. Available: https://ceur-ws.org/Vol-2600/paper1.pdf

[8] M. Hind, D. Wei, M. Campbell, N. C. F. Codella, A. Dhurandhar, A. Mojsilovic, K. N. Ramamurthy, and K. R. Varshney, "TED: Teaching AI to Explain its Decisions," in *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society, AIES 2019, Honolulu, HI, USA, January 27-28, 2019*, V. Conitzer, G. K. Hadfield, and S. Vallor, Eds. ACM, 2019, pp. 123–129. [Online]. Available: https://doi.org/10.1145/3306618.3314273

[9] A. Arias-Duart, F. Parés, D. Garcia-Gasulla, and V. Gimenez-Abalos, "Focus! Rating XAI Methods and Finding Biases," in *IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2022, Padua, Italy, July 18-23, 2022*. IEEE, 2022, pp. 1–8. [Online]. Available: https://doi.org/10.1109/FUZZ-IEEE55066.2022.9882821

[10] M. Setzu, R. Guidotti, A. Monreale, F. Turini, D. Pedreschi, and F. Giannotti, "GLocalX - From Local to Global Explanations of Black Box AI Models," *Artificial Intelligence*, vol. 294, p. 103457, may 2021.

[11] J. Fattahi, M. Ziadia, and M. Mejri, "Cyber Racism Detection Using Bidirectional Gated Recurrent Units and Word Embeddings," in *New Trends in Intelligent Software Methodologies, Tools and Techniques - SoMeT 2021, Cancun, Mexico, 21-23 September, 2021*, ser. Frontiers in Artificial Intelligence and Applications, H. Fujita and H. Pérez-Meana, Eds., vol. 337. IOS Press, 2021, pp. 155–165. [Online]. Available: https://doi.org/10.3233/FAIA210016

[12] J. Fattahi, M. Mejri, M. Ziadia, and R. Ghayoula, "SpamDL: A High Performance Deep Learning Spam Detector Using Stanford Global Vectors and Bidirectional Long Short-Term Memory Neural Networks," in *SoMeT 2022, Kitakyushu, Japan, 20-22 September, 2022*, ser. Frontiers in Artificial Intelligence and Applications, H. Fujita, Y. Watanobe, and T. Azumi, Eds., vol. 355. IOS Press, 2022, pp. 143–162. [Online]. Available: https://doi.org/10.3233/FAIA220246

[13] M. K. Dahouda and I. Joe, "A Deep-Learned Embedding Technique for Categorical Features Encoding," *IEEE Access*, vol. 9, pp. 114 381–114 391, 2021. [Online]. Available: https://doi.org/10.1109/ACCESS.2021.3104357

[14] M. Klimo, J. Kopčan, and L. Králik, "Explainability as a Method for Learning From Computers," *IEEE Access*, vol. 11, pp. 35 853–35 865, 2023.

[15] G. Joshi, R. Walambe, and K. Kotecha, "A Review on Explainability in Multimodal Deep Neural Nets," *IEEE Access*, vol. 9, pp. 59 800–59 821, 2021.

[16] J. E. Zini and M. Awad, "On the Explainability of Natural Language Processing Deep Models," *ACM Comput. Surv.*, vol. 55, no. 5, pp. 103:1–103:31, 2023. [Online]. Available: https://doi.org/10.1145/3529755

[17] J. H. Friedman, "Greedy Function Approximation: a Gradient Boosting Machine," *Annals of statistics*, pp. 1189–1232, 2001.

[18] Ö. Aslan and R. Samet, "A Comprehensive Review on Malware Detection Approaches," *IEEE Access*, vol. 8, pp. 6249–6271, 2020. [Online]. Available: https://doi.org/10.1109/ACCESS.2019.2963724

[19] E. Doynikova, E. Novikova, and I. Kotenko, *Attacker Behaviour Forecasting Using Methods of Intelligent Data Analysis: A Comparative Review and Prospects*. In Advanced Topics in Systems Safety and Security Information. Edited by Emil Pricop, Grigore Stamatescu and Jaouhar Fattahi. Printed Edition of the Special Issue Published in Information, 2021, pp. 41–57. [Online]. Available: https://www.mdpi.com/books/book/3965-advanced-topics-in-systems-safety-and-security

[20] D. Ameyed, F. Jaafar, and J. Fattahi, "A Slow Read Attack Using Cloud," in *2015 7th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, 2015, pp. SSS–33–SSS–38.

[21] M. Ziadia, J. Fattahi, M. Mejri, and E. Pricop, "Smali+: An Operational Semantics for Low-Level Code Generated from Reverse Engineering Android Applications," *Inf.*, vol. 11, no. 3, p. 130, 2020. [Online]. Available: https://doi.org/10.3390/info11030130

[22] X. Lou and A. Tellabi, *Cybersecurity Threats, Vulnerability and Analysis in Safety Critical Industrial Control System (ICS)*. Cham: Recent Developments on Industrial Control Systems Resilience. Springer International Publishing, 2020, pp. 75–97. [Online]. Available: https://doi.org/10.1007/978-3-030-31328-9_4

[23] R. Ronen, M. Radu, C. Feuerstein, E. Yom-Tov, and M. Ahmadi, "Microsoft Malware Classification Challenge," *CoRR*, vol. abs/1802.10135, 2018. [Online]. Available: http://arxiv.org/abs/1802.10135

[24] J. Fattahi, M. Mejri, and M. Ziadia, "Extreme Gradient Boosting for Cyberpropaganda Detection," in *SoMeT 2021, Cancun, Mexico, 21-23 September, 2021*, ser. Frontiers in Artificial Intelligence and Applications, H. Fujita and H. Pérez-Meana, Eds., vol. 337. IOS Press, 2021, pp. 99–112. [Online]. Available: https://doi.org/10.3233/FAIA210012

[25] M. Liu, C. Guo, and S. Guo, "An Explainable Knowledge Distillation Method with XGBoost for ICU mortality prediction," *Comput. Biol. Medicine*, vol. 152, p. 106466, 2023. [Online]. Available: https://doi.org/10.1016/j.compbiomed.2022.106466

[26] R. Alenezi and S. A. Ludwig, "Explainability of Cybersecurity Threats Data Using SHAP," in *IEEE Symposium Series on Computational Intelligence, SSCI 2021, Orlando, FL, USA, December 5-7, 2021*. IEEE, 2021, pp. 1–10. [Online]. Available: https://doi.org/10.1109/SSCI50451.2021.9659888