Explainability of Cybersecurity Threats Data Using SHAP

Rafa Alenezi
Department of Computer Science
North Dakota State University
Fargo, USA
rafa.alenezi@ndsu.edu

Simone A. Ludwig
Department of Computer Science
North Dakota State University
Fargo, USA
simone.ludwig@ndsu.edu

***Abstract*—Nowadays, cybersecurity threats have become a worrisome issue that need to be addressed in all of the world. Almost all people have smart devices that are connected worldwide and many are using social media platforms, and thus, most of their personal information is used and shared. An example of cybersecurity threats are malicious URLs and malware, which are very likely to impact general users. For the research community, detecting new types of attacks is a challenge. Most of the past research studies focused on surveying malicious attack detection. The classification models detect the kind of attacks by using machine learning approaches. Interpreting machine learning models is also an important issue. Tree, Deep, and Kernel of SHAP (Shapley Additive Explanations) are well-known techniques, which achieve efficient performance in interpreting the results. In this paper, two cyber data sets are investigated, both being five-class data sets, for which the Random Forest Classifier, XGBoost Classification, and the Keras Sequential algorithms are applied. The obtained results confirm that applying the classifiers to generate the models are good choices to detect cybersecurity threats. The efficacy of these models' performance was evaluated by measuring the precision, recall, F1-score, accuracy, and confusion matrix. In addition, three SHAP methods are used to explain the output of the resulting machine learning models for the five-class data sets.**

***Index Terms*—SHAP, XGBoost, RFC, Sequential, Explain, TreeShap, KernelShap, DeepShap**

I. Introduction

The web has recorded significant developments, with impressive communication features being provided. Despite the exponential growth, the online realm faces multiple challenges, including cyber threats. Already, numerous cases of financial fraud are reported constantly. Under normal circumstances, the hackers steal personal information, which is used to execute crime with the data being obtained from social media sites [1]. With technology being dynamic, complex penetration strategies are being developed, with the criminals always experimenting with newer schemes. Consequently, information technology professionals and users must be vigilant to thwart hackers' unrelenting efforts, which is time-consuming and resource-intensive. Hence, robust protective measures should be developed to protect online users against malicious activities on the internet [2]. Technology is fluid, which calls for constant defensive approaches to safeguard data within an organization's or individual's control.

Predicting the risk of attack occurrences is a significant challenge for researchers. The development of a reliable and secure analysis system is vital to protecting digital assets. Therefore, prevention approaches have proved useful in minimizing cyber crimes' threats, such as automated scans [3]. Mamun et al. [4] suggest methods to detect and classify malicious URLs based on their attack type through lexical analysis. For different kinds of malware threats, the researchers' efforts focus on surveying malware, for example, the Android malware variant [5]. Given the Android environment's uncontrolled development, a mandatory safety requirement aims to prevent infiltration by malicious codes. However, without a thorough understanding of malware infection approaches, it is challenging to implement an effective mitigation solution. Without proper structures, the prevention of online attacks is challenging, thus, compromising the security of information with online databases. This paper investigates two standard data sets of cyber-attacks: malicious URLs and Android malware. An explainable artificial intelligence (XAI) is used through the Shapley Additive Explanations (SHAP). Thus, this research contributes towards applying three methods of SHAP which are TreeShap, KernelShap, and DeepShap to explain three different models of Machine Learning (ML) classification. The ML algorithms used are Random Forest Classifier (RFC), XGBoost Classification, and the Keras Sequential algorithm. ML models are built to predict and estimate the expected performance on the test sets of both data sets used. The systematic analysis allows researchers to identify dominant patterns in system infiltration.

Additionally, forecast interpretation is imperative to distinguish model qualities and shortcomings. The analysis is achieved by analyzing the contribution of the input to decipher the expectations iteratively. Moreover, the approach allows the extrication of unused data from complex information intelligence captured by ML models. With the three methods of SHAP, sufficient explanation regarding the outputs of the ML models' decisions for the five classes for each data set is provided. The five classes express the different attacks, with the features describing the causes affecting the outcomes. However, more details are presented in the data description section.

The paper is organized into five sections following the

introduction. The related work is discussed in Section II. In Section III, the background is detailed, including the algorithms used for both classifications applying the SHAP methods. The experiments and results are illustrated in Section IV, where the results are obtained from various models of the ML algorithms. In addition, the SHAP explanation results are provided and discussed. Section V summarizes the paper and presents the paper's findings.

II. Related Work

Most cybersecurity research focuses on describing the risks, crime incidences, and their overall effects on individuals and companies. Nurse in [2] presents an approach to cyber crimes and community challenges, which addresses the inherent risks common on digital platforms. Cyber criminals' styles and methods are mentioned, including malware, account hacking, and malicious URLs, among many other techniques. In [6], the use of ML to detect cyber-attacks in real-time was investigated where the model was built using Random Forest and Decision Tree achieving the best performance.

In [7], the theory of XAI using an Ontology Graph (OG) and transfer learning was used to simulate the human brain to explain the outputs. The researchers established that the main problem was the explainable part of artificial intelligence, likely to be used in future development. Thus, a reliable analysis provides the critical role needed to interpret the outputs, which are essential in making correct decisions. Additionally, the approach provided validity and reliability to the results. In [8], Fernando et al. used two methods of XAI DeepSHAP and LIME to explain the output of Neural Retrieval Models (NRMs) for the text based adhoc search where DeepSHAP was more efficient since it uses more precise terms, whereas LIME focused only on eminent terms.

Research presented by Ibor et al. [9] examined the utilization of unsupervised measurable and administered learning strategies to identify threats by mapping hyper-alerts within class label risks. According to [10], Lakshmanaprabu et al. used RFC to classify the e-health data, and the precision of the proposed technique was 94.2%. In [11], RFC was used to built predictive models with the system producing very good results whereby the Receiver Operating Characteristics (ROC) curve was above 90%.

Furthermore, Chen et al. [12] describe XGBoost as a model that incrementally constructs new frameworks, focusing on the errors in the previous system's classification. In the method, the model can unravel real-world scale issues with a minimum number of resources.

Research presented by Ho [13] indicated that KernelSHAP was utilized to interpret expectations of RNN models and the average time of computation was less than ten minutes. Consequently, the explanations analyzed at different scales (people, cohorts, and all patients) were consistent with clinical expectations.

In [14], the authors used Keras Sequential to build their model for network intrusion detection by classifying the data set into two classes (normal and attack). Several explanation methods were utilized that were SHAP, LIME, Contrastive Explanations Method (CEM), ProtoDash and Boolean Decision Rules via Column Generation (BRCG). The accuracy of their model was 82.4%.

In [15], researchers used the Deep Feed Forward (DFF) and Random Forest (RF) models to classify their data set. In addition, they utilized the SHAP technique to explain the importance and influence of the data set features.

Mahbooba et al. [16] addressed Explainable Artificial Intelligence (XAI) to enhance trust management in Intrusion Detection Systems by using the Decision Tree (DT) model. They claimed that DT achieves a marginal improvement compared to Logistic Regression and Support Vector Machines.

The study in [17] proposed a method to detect valuable alerts of cyber security threats using SHAP to identify the important features selected based on the SHAP plots. Random Forest (RF) and XGBoost were used to test the Malware Data set where RF's accuracy was 98%, and 97% for XGBoost.

In this paper, three methods of SHAP (TreeSHAP, KernelSHAP, and DeepSHAP) are employed to explain the predictions of classifiers applied that are RFC, XGBoost, and the Sequential model. The strategies are useful in mapping alerts of threats to web and social media. Additionally, the accuracy and performance of models are investigated in order to identify the best performing classifier and explainer tool for both data sets.

III. Background

In this section, the models of ML and explanations of the SHAP method applied to the models are described. First, RFC, XGBOOST, and SEQUENTIAL ML models are described. Then, the SHAP methods are explained, which are applied to the results of the ML models that are run on two cybersecurity data sets.

A. Random Forest Classifier

Random Forest Classifier (RFC) is used with TreeShap explanation, which is one of three ensemble methods that play a main role for the interpretation of predictions [18]. In RFC three main parameters are used: The first criterion, criterion='gini', measures the quality of a split or the Gini index that uses Equation (1) to decide how nodes on a decision tree divide. It uses the class and probability to determine the Gini value of each branch of a node, determining which of the branches is more likely to occur. Here, P_i represents the relative frequency of the class observed in the data set, and C represents the number of classes. The criterion='entropy' is another option to determine how nodes branch in a decision tree based on the probability as shown in Equation (2). For selecting the best features, we chose Gini because its range is between 0 to 0.5, while Entropy is between 0 to 1 [11]. Other parameters are "max_depth", which is for depth of the

tree, and "n_estimators" is the number of trees in the forest (integer).

$$Gini = \mathbf{1} - \sum_{i=\mathbf{1}}^{C} (P_i)^{\mathbf{2}} \tag{1}$$

$$Entropy = \sum_{i=\mathbf{1}}^{C} -P_{\mathbf{i}} * \mathbf{log_2}(P_{\mathbf{i}}) \tag{2}$$

RFC is the implementation that trains a model for a data set where random samples are selected from the data set. Then, a decision tree is created based on this selection. After that, a prediction result is obtained from each decision tree created and the most voted prediction result is the final prediction result [19].

B. XGBoost Classification

XGBoost classification is used with KernelShap explanation. XGBoost is an ensemble machine learning technique that uses the gradient boosting framework for machine learning prediction. XGBoost is well known for its fast execution and scalability [20]. XGBoost is derived from extreme gradient boosting, which is a mix of gradient boosting and XGBoost. It uses the second-order gradients and advanced regularization to obtain more accurate approximations. XGBoost is an example of boosting where the values of initial predictions and errors are calculated. Then, a model is trained with independent variables and residual errors to get the predictions [12]. Three parameters are used:

1) binary_logistic: logistic regression is used for binary classification;
2) output probability is max_depth =10 is the maximum depth of a tree;
3) n_estimators=800 is the number of boosting rounds [20].

The model is trained and evaluated resulting in the predictions.

C. Sequential Model

A sequential model is used as a classifier consisting of a linear stack of layers. Each layer has an input tensor and output tensor.

In the first layer, we use the ReLU (Rectified linear unit) activation function [21]. ReLU is defined by Equation (3). In the final layer, we use the softmax activation to generalize the first layer function [21]. Softmax is defined by Equation (4). After creating the network architecture, the loss function, optimizer, and metrics for prediction are defined.

$$F(x) = (0,x) \tag{3}$$

$$Softmax(x_j) = \frac{e^{(x_i)}}{\sum_i e^{(x_i)}} \tag{4}$$

D. Introduction to SHAP

Shapley values were leveraged to represent SHapley Additive exPlanation (SHAP) for the feature influence scoring. Shapley values consider all possible predictions for an instance using all possible combinations of inputs. Since this is an exhaustive approach, SHAP provides properties like consistency and local accuracy [22]. The Shapley value is defined via a value (Equation (5)) where $s \subseteq F$ represents all feature subsets, F is the set of all features, $f_{S\cup i}$ is trained with that feature present, f_S is trained with the feature withheld. Then, predictions from the two models are compared based on the current input $[f_{S\cup i}(x_{S\cup i}) - f_S(x_S)]$, where (x_S) represents the values of the input features in set S [23].

$$\emptyset_i = \sum_{S \subseteq \mathbf{F}\{\mathbf{i}\}} \frac{|\mathbf{S}|!(|\mathbf{F}| - |\mathbf{S}| - \mathbf{1})!}{|\mathbf{F}|!} [f_{S\cup i}(x_{S\cup i}) - f_S(x_S)] \tag{5}$$

SHAP takes three steps [24] which are: (1) computation of Shapley value explanations, (2) capture feature interactions by extending local explanations, and (3) interpreting global model structure based on many local explanations by defining desirable properties.

E. TreeExplainer of SHAP

The TreeExplainer is applied with the ML Random Forest Classifier (RFC) model to enable tractable computation of the optimal local explanations by defining desirable properties as defined by SHAP. The decomposition of the decision path into one component per feature explains the predictions of a decision tree. Thus, the decision is tracked by traversing the tree and explaining a prediction y by the additive contributions at each decision node as shown in Equation (6).

$$y = bias + \sum_{m=\mathbf{1}}^{M} feature\ contribution\{m, x\} \tag{6}$$

where $bias$ is the contribution of root node and feature contribution, $\{m, x\}$ is the contribution of feature m in predicting the outcome corresponding to an input x [25].

F. KernelExplainer of SHAP

The KernelExplainer is applied to the ML XGBOOST classifier model. It performs a local regression by taking the prediction method and the data to perform the SHAP values to compute the importance values of each feature as Shapley values. Two parameters are used in KernelExplainer: (1) $predict\_proba()$ is used to retrieve the probabilities of each target class; (2) $link = Logit$ is a function to make the feature importance values have log-odds units [26]. The approximation in Equation (7) is used to evaluate the conditional expectation [27].

$$f_{T,KernelSHAP}(\mathbf{x}) \approx \frac{1}{K} \sum_k f(x_T, x_T^k) \tag{7}$$

where x_T^k , k=1,...,K are samples from x_T.

G. DeepExplainer of SHAP

The DeepExplainer is applied with the ML Sequential model. It is an enhanced method of the DeepLIFT algorithm (Deep SHAP), which approximates the conditional expectations of SHAP values using a selection of background samples [23]. This is done by integrating over many background samples. Deep estimates approximate SHAP values by calculating the difference between the expected model output on the passed background samples and the current model outputs as shown in Equation (8).

$$Difference = f(x) - Exp(f(x)) \quad (8)$$

where $f(x)$ is current outputs, and $Exp(f(x))$ is the expected output.

IV. Experiments and Results

In this section, the two data sets are used and described followed by the explanation of the outputs of the different SHAP explainer algorithm that are used based on the ML models. Furthermore, the accuracy of the resulting models are analyzed.

A. Data Set Description

Two data sets were taken from the Canadian institute for Cybersecurity, which is called UNB [28]. The first data set is called "URL data set (ISCX-URL2016)" and named malicious URLs. It contains 36,707 records and 80 columns (79 features and one outcome column (classes) for the five different attack types of URLs which are Benign URLs, Spam URLs, Phishing URLs, Malware URLs, and Defacement URLs. Features are obfuscation techniques used as common method for masking malicious URLs. More details regarding these features can be found on the UNB site [28]. The features that were analyzed and classified contributed to explain the five classes of Malicious URLs. The second data set is called "CICMalDroid 2020" and is named Android Malware data set. It contains 11,598 APK files samples (records) and 470 (469 extracted features (columns), and one column containing five categories (classes) which are: Adware, Banking malware, SMS malware, Riskware, and Benign). The features relate to system calls, binders, and composite behaviors.

B. Results Obtained

The results obtained using three classifiers of ML models were explained by using three versions of SHAP that are TreeSHAP, KernelSHAP, and DeepSHAP. The experiments focused on explaining and detecting the obfuscation techniques that influence each attack type of URLs and identify Android malware by explaining and detecting the features of system calls, binders, and composite behaviors that cause them. This allows to present the results to the end user in a clear, simple, and understandable way.

C. Results - TreeExplainer of SHAP

A bar graph is chosen for TreeExplainer of SHAP to explain the RFC model. It is used to explain the features that influence the classes in the data set. Feature importance is ranked in descending order based on the effect the prediction has on each class. Therefore, the main role of the Tree-Explainer is to explain and interpret the impact each feature has on the model output of the target class based on the test set for both data sets.

*1) **Malicious URLs Data Set**:* Figure 1 shows the most important features of the SHAP values that affect classes of the attack type URLs. As shown, the colors represent the features of the obfuscation techniques that are SHAP values. For example, obfuscation technique called "domain_token_count", which is located in high level of this plot, resulted in classes "Benign" and "Spam" URLs more than the remaining class types. Conversely, in the low level of the plot the "fileNameLen" affects class "Malware URLs" more than the other class types of URLs. Also, the "urlLen" feature influenced the class "Malware" more than the other classes. Thus, based on the influence of the features, the class "Spam" in blue color was identified most often.

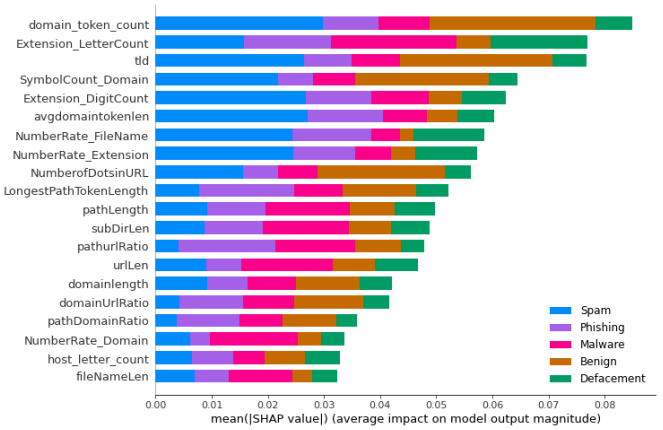

Figure 1. Five-Class Tree-SHAP of Malicious URLs

*2) **Android Malware Data Set**:* Figure 2 displays SHAP values that are used to identify the features that resulted in malware for each of the classes shown in different colors. For example, the "pread64" feature, which is located on top of this plot influenced the class "SMS malware" more often than the other classes, while the "getSubscriberId" feature, which is located in the low level of this plot influenced the class "Riskware" more than the other malware classes. This plot shows the class "SMS malware" in blue color resulting in the most often outcome.

D. Results - KernelExplainer of SHAP

For the following experiments, KernelExplainer of SHAP is used to explain one class using the Dot graph for the XGBoost classifier model. It shows the results as a graph showing the combination between a scatter plot and density estimation by

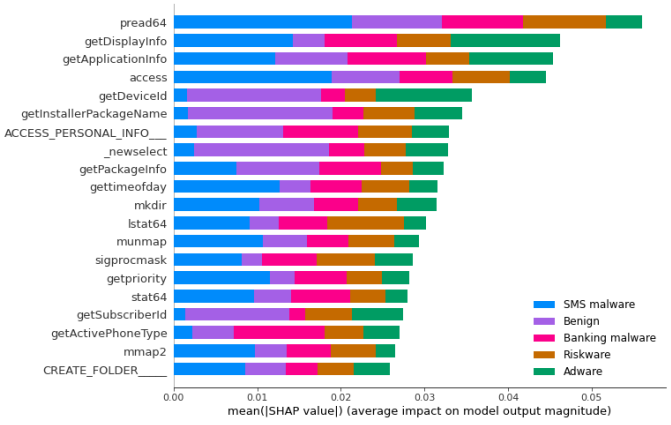

Figure 2. Five-Class Tree-SHAP of Android Malware

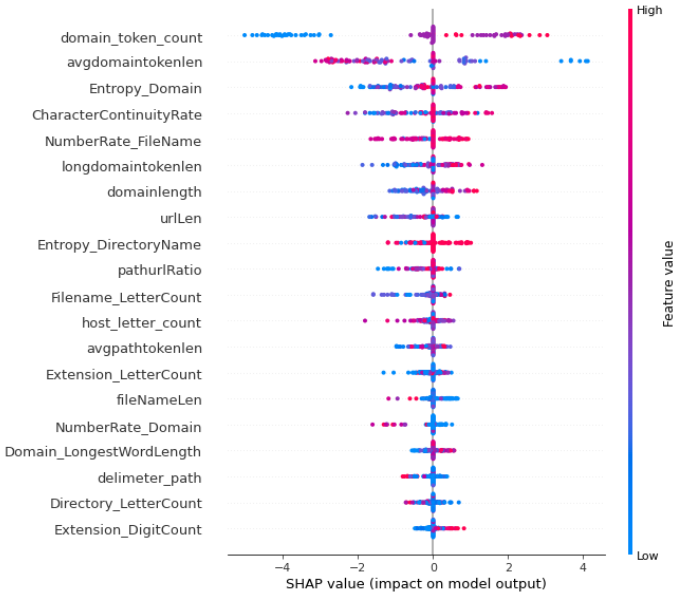

Figure 3. Five-Class Kernel SHAP of Malicious URLs

letting dots pile up when they do not fit. The average feature value at that position is represented with the colors where the red dots represent mostly high feature values while blue dots represent mostly low feature values in the data set based on their location either left or right of the vertical axis. The dot graph shows the positive and negative relationship of the predictors with the target class as dots based on the training data. Feature importance is ranked in descending order based of the effect each feature has on the prediction of the class.

*1) **Malicious URLs Data Set**:* Figure 3 shows that feature "domain_token_count" has a high and positive impact on the class "Spam" rating. The large values of this feature does increase the prediction of class "Spam". The "high" values are denoted in red color, and the "positive" impact is shown on the horizontal axis. The horizontal location shows that an impact of this value is associated with a higher or lower prediction according to the scale. Similarly, we can see

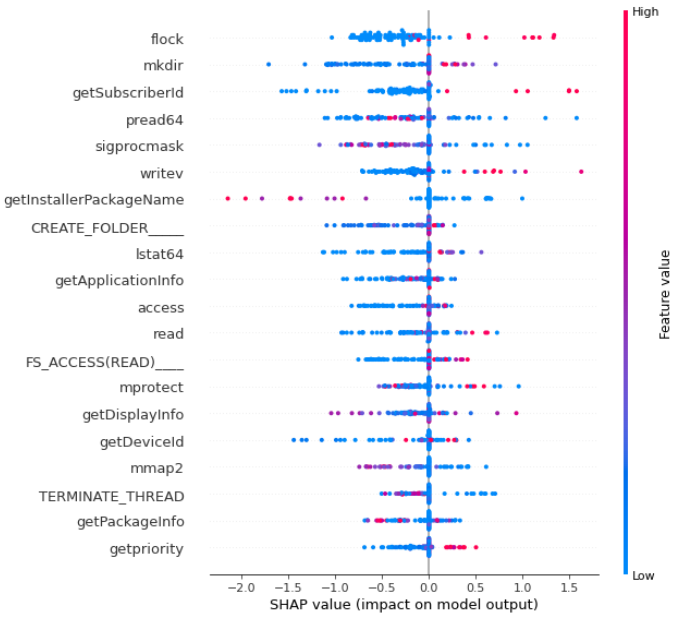

Figure 4. Five-Class Kernel-SHAP of Android Malware

that "avgdomaintokenlen" is negatively correlated with class "Spam". Also, the large values of "Entropy_DirectoryName" clearly increases the prediction of class "Spam", while it can be seen that large values of "fileNamemeLen" decrease the prediction of class "Spam".

*2) **Android Malware Data Set**:* Figure 4 shows the values of the "flock" feature having a high and positive effect on class "SMS malware" indicated by the red color on the horizontal axis, which increases the likelihood of class "SMS malware". On the other hand, large values of feature "pread64" decreases the likelihood of class "SMS malware". Also, the values of "getSubscriberid" and "writev" increase the prediction of class "SMS malware". Conversely, the values of "sigprocmask" and "getinstallerPackageName" decrease the prediction of class "SMS malware".

E. Results - DeepExplainer of SHAP

In the following section, DeepExplainer of SHAP is used to explain one class using the Force plot for the Sequential Classifier model. It shows how each feature contributes to influence the model output from the base value prediction (the average predicted outcome over the entire training set) to the corresponding model output of the target class. The plot is centered around the x axis by the explanation of an expected feature, which impacts the target class. Features influencing the prediction positively are shown in red, while those influencing the prediction negatively are shown in blue. Influencing the prediction to higher or lower values depends on the **output value (f(x))** compared to the **base value**. If the **output value** is greater than the **base value** for the prediction, then the features in the red color will push the prediction to the right (higher) range or vice versa. Also, at the bottom of the plot the largest effects of feature values are printed.

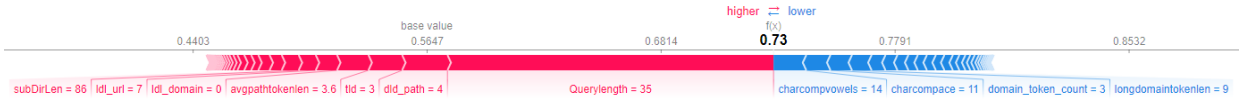

Figure 5. Five-Class Deep-Malicious URLs

Figure 6. Five-Class Deep-SHAP of Android Malware

*1) **Malicious URLs Data Set**:* Figure 5 shows the different features and how each is contributing to influence the model output class "Spam" from the base value. Since the **output value (f(x))** was 0.73, which is greater than the **base value** of 0.5647, the prediction shown in red color of features belonging to class "Spam" are pushed to the right (higher values). For example, "Querylength" has a positive impact on the class "Spam", which is seen in the red area and influences the prediction of class "Spam" to the right. Conversely, the feature "charcompvowels" shown in the blue area is negatively related to class "Spam" where it pushes the prediction of class "Spam" to the left (lower value). But the force to drive the prediction to the right (higher value) is the largest because the feature values that are in the red area are greater than the other feature values that are in the blue area.

*2) **Android Malware Data Set**:* Figure 6 shows how the each feature contributes to influence the model output for class "SMS malware" away from the base value due to the **output value (f(x))** with a value of 0.72, which is greater than the **base value** of 0.5881. The prediction in red color of features influencing class "SMS malware" are pushed to the right (higher value). For example, feature "FS_ACCESS(READ)___" in the red area is a force to drive the prediction of class "SMS malware" to the right (higher value). Conversely, the feature "FS_ACCESS(CAREATE__APPEND)__" in the blue area pushes the prediction of class "SMS malware" to the left (lower value). But the strong force to drive the prediction to the right (higher value) is located in the red area since the feature values in the red area are greater than the feature values that are in the blue area.

F. ML Model Performance

The following section explains the confusion matrix for each model, and presents the tables of the classification results of the ML models as well as the ROC curve of each model followed by a summary of the results of all ML models applied.

*1) **Confusion Matrix**:* Figures 7 through 12 show the confusion matrices of the five classes for all ML models for both data sets.

- **Confusion Matrices of Malicious URLs Data Set:**

The confusion matrix in Figure 7 shows that 3,677 samples were correctly classified for all five classes of the **RFC model applied to the Malicious URLs data set**, with the Spam class having the highest number with 1,091 correctly classified samples. The Malware class classified 854 samples, the Phishing class classified 781 samples, the Benign class classified 508 samples, and the Defacement class classified 443 samples correctly. For each class there are also a few misclassifications as shown in the matrix.

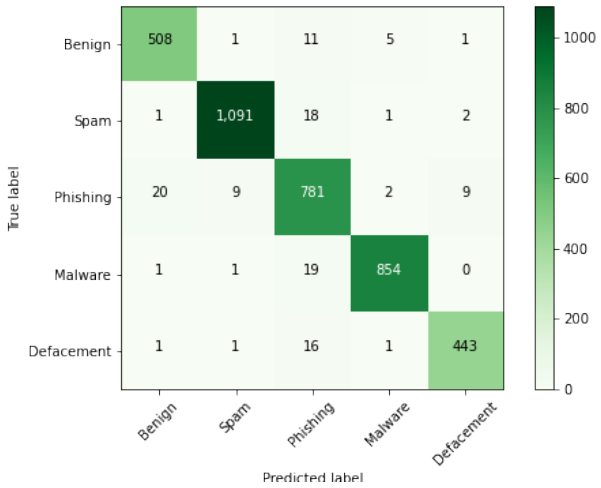

Figure 7. Confusion Matrix for RFC model of Malicious URLs

Figure 8 shows the Confusion Matrix of the **XGBOOST model applied to the Malicious URLs data set**. It correctly classified 3,722 samples in all five class categories of the Malicious URLs data set. For example, the Spam class achieved 1,108 correct classifications whereas the Defacement class is the one with the least number of correctly classified 449 samples correctly. Values in-between are the Malware class with 868 samples, the Phishing class with 787 samples, and the Benign class with 510 samples correctly classified.

The confusion matrix of the **Sequential model applied to the Malicious URLs data set** is shown in Figure 9. It shows the correct classification of 3,546 samples for all five classes of the Malicious URLs data set with the following correct classifications: Spam class 1,095 samples, the Malware class 814 samples, the Phishing class 722 samples, the Benign class 482 samples, and the Defacement class classifying 433 samples correctly.

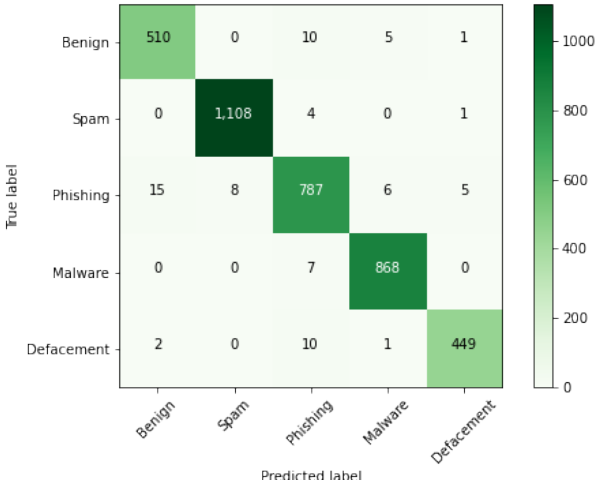

Figure 8. Confusion Matrix for XGBOOST model of Malicious URLs

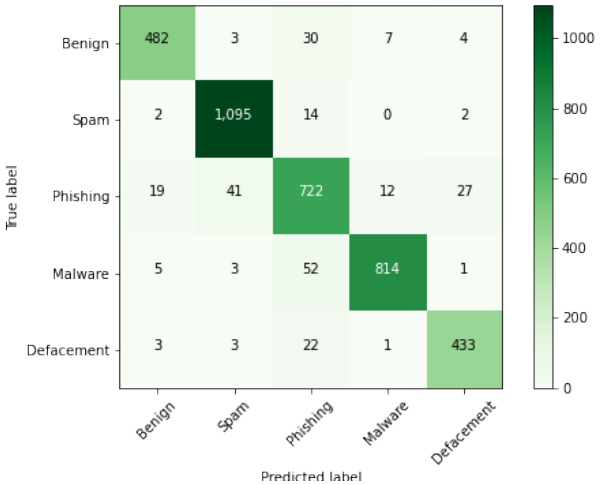

Figure 9. Confusion Matrix for Sequential model Malicious URLs

In summary, the XGBOOST model showed the best performance since the number of correctly classified samples was 3,722, which is the highest among all ML models. Also, it is observed that the Spam class achieved the highest numbers of samples that were correctly classified comparing all ML models applied to the Malicious URLs data set.

- **Confusion Matrices of Android Malware Data Set:**

In Figure 10, the confusion matrix shows the 2,183 samples that were correctly classified for all five classes of the **RFC model applied to the Android Malware data set** with the highest number of classifications for the SMS malware class (807 samples) followed by the Riskware class with 467 samples, the Bank malware class with 373 samples, and the Benign class with 307 samples. The lowest number of samples was for the Adware class with 229 samples.

The confusion matrix in Figure 11 shows the correct classification of 2,197 samples in the diagonal of all five classes of the **Android Malware** data set using the **XGBOOST**

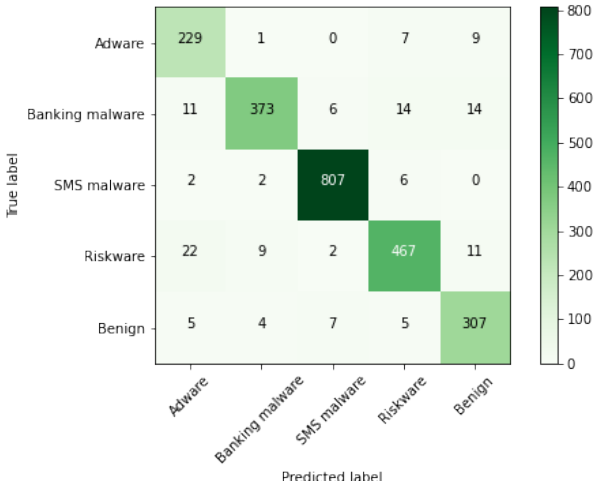

Figure 10. Confusion Matrix of RFC model of Android Malware

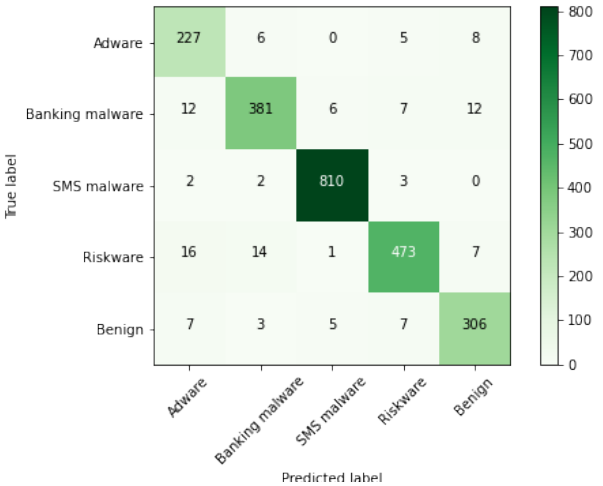

Figure 11. Confusion Matrix of XGBOOST model of Android Malware

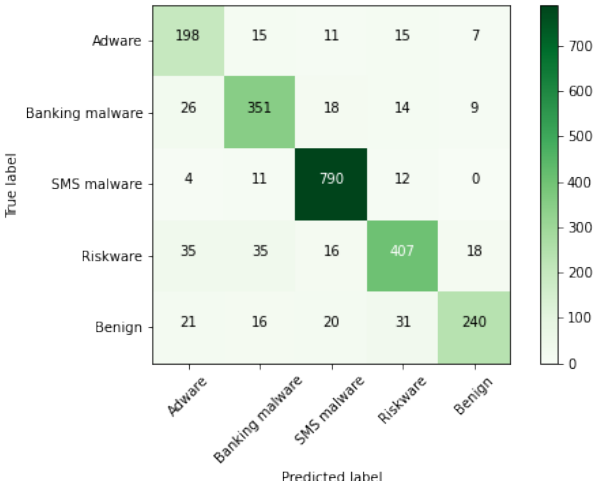

Figure 12. Confusion Matrix of Sequential model of Android Malware

model. The figure shows that the SMS malware class classified 810 samples, the Riskware class classified 473 samples, the Bank malware class classified 381 samples, the Benign class classified 306 samples, and the Adware class classified 227 samples correctly.

In Figure 12, the confusion matrix shows that 1,986 samples were correctly classified for all five classes of the **Sequential model applied to the Android Malware data set** with 790 for the SMS malware class, 407 for the Riskware class, 351 for the banking malware class, 240 for the Benign class, and 198 for the Adware class.

It can be concluded that the best performance was when using the XGBOOST model because it correctly classified 2,197 samples, which is the highest among all ML models applied. Finally, it is seen that all ML models correctly classified the highest numbers of Android Malware Data Set's samples to the SMS malware class.

*2) **ROC and Classification Tables of URLs Data Set**:* Firstly, the ROC (Receiver Operating Characteristic) curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR). It gives equal weight to the classification of each label. Therefore, the five curves represent the predictive quality of the five classes. Thus, in order to choose the best ML model we examined the ROC curve and accuracy rate of each class for each ML model.

Figure 13 shows the ROC curve for the XGBOOST ML model applied to the Malicious URLs data set showing the best performance among all models. Table I summarizes the ROC of all ML models applied to the test set of the Malicious URLs data set.

Secondly, Tables II through IV show the precision, recall, f1-score, and support for the 5-Class data based on all ML models that are applied to the Malicious URLs data set.

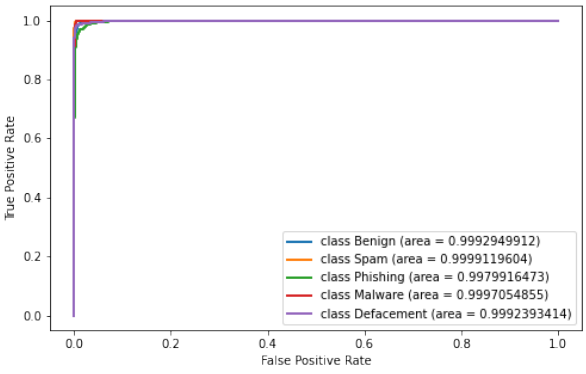

Figure 13. ROC Curve of 5-class to XGBOOST model of Malicious URLs

*3) **ROC and Classification Tables of Android Malware Data Set**:* Firstly, based on the examination of the ROC and accuracy rate of each class for each ML model, the XGBOOST achieved the best performance among other ML models. Figure 14 shows the ROC curve of the XGBOOST model and Table V the ROC of all ML models used to test five classes of Android Malware data set.

Table I
ROC OF ML MODELS FOR MALICIOUS URLS

	Sequential	XGBOOST	RFC
Benign	0.9873	0.9992	0.9988
Spam	0.9942	0.9999	0.9996
Phishing	0.9765	0.9979	0.9958
Malware	0.9928	0.9997	0.9996
Defacement	0.9896	0.9992	0.9988

Table II
RFC MODEL OF MALICIOUS URLS

	Precision	Recall	f1-Score	Support
Benign	0.96	0.97	0.96	526
Spam	0.99	0.98	0.98	1113
Phishing	0.92	0.95	0.94	821
Malware	0.99	0.98	0.98	875
Defacement	0.97	0.96	0.97	462
Accuracy			0.97	3797
Macro avg	0.97	0.97	0.97	3797
Weighted avg	0.97	0.97	0.97	3797

Table III
XGBOOST MODEL OF MALICIOUS URLS

	Precision	Recall	f1-Score	Support
Benign	0.97	0.97	0.97	526
Spam	0.99	1.00	0.99	1113
Phishing	0.96	0.96	0.96	821
Malware	0.99	0.99	0.99	875
Defacement	0.98	0.97	0.98	462
Accuracy			0.98	3797
Macro avg	0.98	0.98	0.98	3797
Weighted avg	0.98	0.98	0.98	3797

Table IV
SEQUENTIAL MODEL OF MALICIOUS URLS

	Precision	Recall	f1-Score	Support
Benign	0.94	0.92	0.93	526
Spam	0.96	0.98	0.97	1113
Phishing	0.86	0.88	0.87	821
Malware	0.98	0.93	0.95	875
Defacement	0.93	0.94	0.93	462
Accuracy			0.93	3797
Macro avg	0.93	0.93	0.93	3797
Weighted avg	0.93	0.93	0.93	3797

Table V summarizes the ROC of all ML models applied to the testing portion of the Android Malware data set.

Furthermore, Tables VI through VIII show the precision, recall, f1-score, and support for the 5-Class data based on all ML models that are applied to the Malicious URLs data set.

*4) **Summary of Results**:* Table IX shows the summary of the experiments. It summarizes and provides a comparison of both data sets. The data sets used are relatively large data sets. The URLs data set is distinguished by containing 36,707 samples and 79 features while the Android Malware data set is featured having 11,598 samples and 469 features. Both data sets' features were analyzed and classified to five classes based on the test data set portion of both. Moreover,

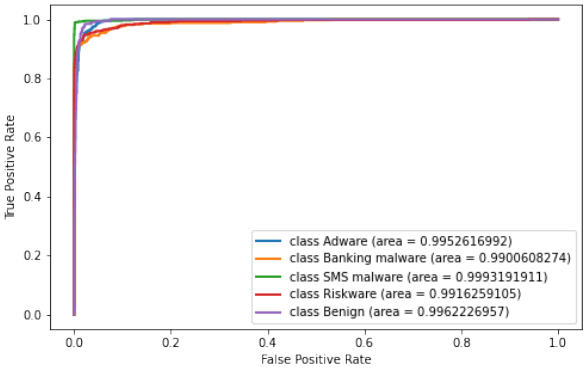

Figure 14. ROC Curve of 5-Class to XGBOOST model of Android Malware

Table V
ROC OF ML MODELS FOR ANDROID MALWARE

	Sequential	XGBOOST	RFC
Adware	0.8901	0.9952	0.9946
Banking malware	0.9068	0.9900	0.9890
SMS malware	0.9662	0.9993	0.9985
Riskware	0.8862	0.9916	0.9929
Benign	0.8661	0.9962	0.9943

Table VI
RFC MODEL OF ANDROID MALWARE

	Precision	Recall	f1-Score	Support
Adware	0.85	0.93	0.89	246
Banking malware	0.96	0.89	0.92	418
SMS malware	0.98	0.99	0.98	817
Riskware	0.94	0.91	0.92	511
Benign	0.90	0.94	0.92	328
Accuracy			0.94	2320
Macro avg	0.93	0.93	0.93	2320
Weighted avg	0.94	0.94	0.94	2320

Table VII
XGBOOST MODEL OF ANDROID MALWARE

	Precision	Recall	f1-Score	Support
Adware	0.86	0.92	0.89	246
Banking malware	0.94	0.91	0.92	418
SMS malware	0.99	0.99	0.99	817
Riskware	0.96	0.93	0.94	511
Benign	0.92	0.93	0.93	328
Accuracy			0.95	2320
Macro avg	0.93	0.94	0.93	2320
Weighted avg	0.95	0.95	0.95	2320

the accuracy of the resulting models was quite good for a 5-Class model outcome. For example, in the RFC model applied to the Malicious URLs data set, the accuracy was 97% while for the Android Malware data set the accuracy was 94%. For the XGBOOST model applied to the Malicious URLs data set, the accuracy was 98% while for the Android Malware the accuracy was 95%. For the Sequential model applied to the Malicious URLs data set the accuracy was 92% while for the Android Malware data set the accuracy was 86%. In summary,

Table VIII
SEQUENTIAL MODEL OF ANDROID MALWARE

	Precision	Recall	f1-Score	Support
Adware	0.70	0.80	0.75	246
Banking malware	0.82	0.84	0.83	418
SMS malware	0.92	0.97	0.94	817
Riskware	0.85	0.80	0.82	511
Benign	0.88	0.73	0.80	328
Accuracy			0.86	2320
Macro avg	0.83	0.83	0.83	2320
Weighted avg	0.86	0.86	0.86	2320

Table IX
SUMMARY OF COMPARISON LISTING BOTH DATA SETS

	URLs Data Set	Android Malware Data Set
Samples	36707	11598
Support	3797	2320
Features	79	469
Classes	5	5
RFC accuracy	97%	94%
XGBOOST accuracy	98%	95%
Sequential accuracy	92%	86%
Algorithm with most correct classifications	XGBOOST	XGBOOST

the best performance was achieved by the XGBOOST ML model among all ML models since it correctly classified the highest number of samples for both data sets.

V. CONCLUSION

The research investigated two big cybersecurity data sets (Malicious URLs, Android Malware), which were from the UNB site. Both data sets consisted of five classes. RFC, XGBOOST, and the Sequential algorithm were applied and three methods of SHAP were used to explain the feature contributions, which were TreeShap, KernelShap, and DeepShap. In addition, the evaluation used accuracy, ROC, classification tables, and confusion matrices.

The SHAP plots were analyzed in order to see which of the features have the most influence on the built models. The plots clearly show the different features and their effects or contributions they make to each model for each class. Thus, the Bar SHAP was successful to explain all classes and features that impact them for both data sets showns in different colors. The best result to determine the class “Spam” in the Malicious URLs data set, and class “SMS malware” in the Android Malware data set was based on the effects of the features. This result was confirmed by the classification tables and the confusion matrices of all ML models, while other SHAP methods help to explain the effects of features on one class.

In terms of accuracy, the RFC model achieved 94.0% for the Android Malware data set whereas the accuracy was 96.7% for the Malicious URLs data set. In addition, the accuracy was 98.0% for XGBOOST using the Malicious URLs data set while the accuracy was 94.7% using the same model on the Android Malware data set. Furthermore, the accuracy of the sequential algorithm for the Malicious URLs data set was 93.3%, and for the Android Malware data set the accuracy was

85.6%. The explanations were been made using the SHAP methods with high efficiency given the accuracy that was achieved using the ML algorithms. In addition, examining data points with low values turns out to be a good way of discovering noisy and/or mislabelled data points.

Finally as for future work, understanding which data has high value and which has a low value is important to improve the explanation capability of the cybersecurity threats data using SHAP methods. Thus, we suggest to use this information to guide future data collection activities and improve the data gathering in cybersecurity or any other domain.

REFERENCES

[1] R. P. Khandpur, T. Ji, S. Jan, G. Wang, C.-T. Lu, and N. Ramakrishnan, Crowdsourcing Cybersecurity: Cyber Attack Detection using Social Media. Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, 2017, doi: 10.1145/3132847.3132866.

[2] J. R. C. Nurse, Cyber crime and You: How Criminals Attack and the Human Factors That They Seek to Exploit. The Oxford Handbook of Cyber psychology, pp. 662-690, Nov. 2018, doi: 10.1093/oxfordhb/9780198812746.013.35.

[3] B. Seemma, S. Nandhini, and M. Sowmiya. Overview of Cyber Security. Nov. 11, 2018. https://www.researchgate.net/publication/329678338 (accessed: Dec. 24, 2020).

[4] M. S. I. Mamun, M. A. Rathore, A. H. Lashkari, N. Stakhanova, and A. A. Ghorbani, Detecting Malicious URLs Using Lexical Analysis. Network and System Security, pp. 467-482, 2016, doi: 10.1007/978-3-319-46298-1_30.

[5] F. Alswaina and K. Elleithy, Android Malware Family Classification and Analysis: Current Status and Future Directions. Electronics, vol. 9, no. 6, p. 942, 2020, doi: 10.3390/electronics9060942.

[6] M. Teixeira, T. Salman, M. Zolanvari, R. Jain, N. Meskin, and M. Samaka, SCADA System Testbed for Cybersecurity Research Using Machine Learning Approach. Future Internet, vol. 10, no. 8, p. 76, 2018, doi: 10.3390/fi10080076.

[7] N. SHI, Q. Zeng, and R. Lee, The Design and Implementation of Language Learning Chatbot with XAI using Ontology and Transfer Learning. Computer Science & Information Technology (CS & IT), 2020, doi: 10.5121/csit.2020.101124.

[8] Z. T. Fernando, J. Singh, and A. Anand, A study on the Interpretability of Neural Retrieval Models using DeepSHAP. Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2019, doi: 10.1145/3331184.3331312.

[9] A. E. Ibor, F. A. Oladeji, O. B. Okunoye, and C. O. Uwadia, Deep learning model for preicting multistage cyber attacks. Journal of Computer Science and Its Application, vol. 26, no. 1, 2020, doi: 10.4314/jcsia.v26i1.6.

[10] S. K. Lakshmanaprabu, K. Shankar, M. Ilayaraja, A. W. Nasir, V. Vijayakumar, and N. Chilamkurti, Random forest for big data classification in the internet of things using optimal features. International Journal of Machine Learning and Cybernetics, vol. 10, no. 10, pp. 2609-2618, 2019, doi: 10.1007/s13042-018-00916-z.

[11] L. Khaidem, S. Saha, and S. Roy Dey. Predicting the direction of stock market prices using random forest. arXiv.org. Apr. 29, 2016. https://arxiv.org/abs/1605.00003 (accessed: Dec. 24, 2020).

[12] T. Chen and C. Guestrin. XGBoost: A Scalable Tree Boosting System. arXiv.org. 2016. https://arxiv.org/abs/1603.02754 (accessed: Dec. 24, 2020).

[13] L. V. Ho, M. D. Aczon, D. Ledbetter, and R. Wetzel. Interpreting a Recurrent Neural Network. arXiv.org. Apr. 06, 2020. https://arxiv.org/pdf/1905.09865 (accessed: Dec. 24, 2020).

[14] S. Mane and D. Rao, Explaining Network Intrusion Detection System Using Explainable AI Framework, Mar. 12, 2021. https://arxiv.org/abs/2103.07110 (accessed Oct. 07, 2021).

[15] M. Sarhan, S. Layeghy, and M. Portmann. Evaluating Standard Feature Sets Towards Increased Generalisability and Explainability of ML-based Network Intrusion Detection. Aug. 29, 2021. https://arxiv.org/abs/2104.07183 (accessed: Oct. 08, 2021).

[16] B. Mahbooba, M. Timilsina, R. Sahal, and M. Serrano, Explainable Artificial Intelligence (XAI) to Enhance Trust Management in Intrusion Detection Systems Using Decision Tree Model. Complexity, vol. 2021, pp. 1-11, 2021, doi: 10.1155/2021/6634811.

[17] H. Kim, Y. Lee, E. Lee, and T. Lee, Cost-Effective Valuable Data Detection Based on the Reliability of Artificial Intelligence. IEEE Access, vol. 9, pp. 108959-108974, 2021, doi: 10.1109/access.2021.3101257.

[18] S. M. Lundberg, G. G. Erion, and S.-I. Lee. Consistent Individualized Feature Attribution for Tree Ensembles. Feb. 12, 2018.

[19] S. K. Lakshmanaprabu, K. Shankar, M. Ilayaraja, A. W. Nasir, V. Vijayakumar, and N. Chilamkurti, Random forest for big data classification in the internet of things using optimal features. International Journal of Machine Learning and Cybernetics, vol. 10, no. 10, pp. 2609-2618, 2019, doi: 10.1007/s13042-018-00916-z.

[20] R. Kumar and G. S, Malware classification using XGboost-Gradient Boosted Decision Tree. Advances in Science, Technology and Engineering Systems Journal, vol. 5, no. 5, pp. 536-549, 2020, doi: 10.25046/aj050566.

[21] A. Gulli, A. Kapoor, and S. Pal, Deep Learning with TensorFlow 2 and Keras. Packt Publishing Ltd, 2019.

[22] M. Zaeri-Amirani, F. Afghah, and S. Mousavi, A feature selection method based on Shapley value to false alarm reduction in ICUs A genetic-algorithm approach, Annu Int Conf IEEE Eng Med Biol Soc, vol. 2018, pp. 319–323, 2018.

[23] S. Lundberg and S.-I. Lee, A unified approach to interpreting model predictions, arXiv [cs.AI], 2017.

[24] S. M. Lundberg et al., Explainable AI for trees: From local explanations to global understanding, arXiv [cs.LG], 2019.

[25] P. Sharma, Evaluating Tree Explanation Methods for Anomaly Reasoning: A Case Study of SHAP TreeExplainer and TreeInterpreter. Lecture Notes in Computer Science, pp. 35-45, 2020, doi: 10.1007/978-3-030-65847-2_4.

[26] Welcome to the SHAP documentation — SHAP latest documentation. https://shap.readthedocs.io/en/latest/ (accessed: Dec. 24, 2020).

[27] D. Janzing, L. Minorics, and P. Blöbaum, Feature relevance quantification in explainable AI: A causal problem, arXiv [stat.ML], 2019.

[28] Canadian Institute for Cybersecurity — UNB. https://www.unb.ca/cic/ (accessed: Dec. 24, 2020).