Indonesian Journal of Electrical Engineering and Computer Science
Vol. 32, No. 1, October 2023, pp. 517~529
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v32.i1.pp517-529

Enhanced website phishing detection based on the cyber kill chain and cloud computing

Saba Hussein Rashid, Wisam Dawood Abdullah
Department of Computer Science, Computer Science and Mathematics College, Tikrit University, Tikrit, Iraq

Article Info

Article history:

Received Apr 3, 2023
Revised Jul 12, 2023
Accepted Jul 16, 2023

Keywords:

AWS
Crimeware-as-a-service
Cyber kill chain
Decision tree
Random forest
Support vector machine
Website phishing

ABSTRACT

Crimeware-as-a-service (CaaS) presents a growing cybersecurity threat by facilitating the acquisition of tools for website phishing attacks. Detecting these attacks requires effective techniques to obtain accurate results in real time. Cloud machine learning (CML) emerged as a promising solution with the powerful tools of amazon web services (AWS). This study proposes a novel approach combining cyber kill chain concept with AWS technologies to enhance website phishing detection, using AWS SageMaker to preprocess an 11,430 uniform resource locators (URL) dataset and train 3 algorithms, which are: decision tree (DT), random forest (RF), and support vector machine (SVM), evaluate their performance through batch transform, and deploy them as separate endpoints. Prediction functions are then conducted on each endpoint and compared to batch transform results. Our findings demonstrate that the combination of the cyber kill chain concept and AWS CML significantly enhances website phishing detection by achieving results of 97% for RF in 0.48 seconds, 94% for SVM in 0.94 seconds, and 93% for DT in 0.52 seconds. By leveraging CML algorithms and breaking down attacks into stages, our approach identifies and disrupts attacks earlier, preventing damage. This research highlights the value of our approach in improving cybersecurity and protecting against website phishing attacks.

Corresponding Author:

Saba Hussein Rashid
Department of Computer Science, Computer Science and Mathematics College, Tikrit University
Street of Tikrit-Mosil, Al-Qadissiyah quarter, 009642, Tikrit, Salaheddin, Iraq
Email: sabahussein88@tu.edu.iq

1. INTRODUCTION

The increasing digitization of the world led to a rise in cybercrime, particularly through particularly through crimeware-as-a-service (CaaS), where criminal organizations provide illicit services and tools for online attacks like phishing, malware distribution, and data theft [1]. CaaS operates similarly to legitimate software-as-a-service (SaaS) providers but is illegal and used for criminal activities [2]. Phishing attacks, facilitated by CaaS, are social engineering attacks that target individuals and organizations for financial gain or system harm [3], [4]. These attacks involve tricking victims into providing confidential information or downloading malicious entities by copying or modifying legitimate websites. Phishing attacks are becoming more sophisticated with the use of advanced technologies, making them harder to detect [5].

Cyber kill chain, a concept that has been developed by Lockheed Martin, describes the stages of a cyberattack and is widely adopted in cybersecurity. Understanding and disrupting these stages is crucial for organizations to defend against cyber threats [6]. Cloud machine learning (CML) integrates machine learning with cloud computing, providing scalable platforms for detecting and analyzing phishing attacks [7]. Machine learning as a service (MLaaS) is an essential part of website phishing detection, offering tools for efficient execution of machine learning models without considering the underlying service structure. CML enables quick and accurate analysis of large data volumes to identify attack patterns and anomalies [8].

Amazon web services (AWS) is a cloud-based platform that offers a scalable environment for creating machine learning models. It combines virtual hardware and software resources, including S3, SageMaker, cloudwatch, and EC2, to process large amounts of data efficiently [9]. AWS provides developers with full control over their working environment and is designed for computationally intensive workloads [10]. AWS SageMaker and cloudwatch are integral components for developing and monitoring machine learning models [11], [12]. Previous studies have addressed website phishing detection using machine learning models, but faced challenges with real-time processing and limited incorporation of cloud computing resources or the cyber kill chain concept [13]–[22]. To overcome these limitations, new techniques have emerged that integrate cloud computing and the cyber kill chain concept, improving detection accuracy and speed.

This paper presents a method that utilizes CML and the cyber kill chain concept for enhanced website phishing detection. The approach involves preprocessing a dataset of uniform resource locators (URL) using AWS SageMaker and S3, training three machine learning algorithms, evaluating performance with SageMaker batch transform, deploying algorithms as separate endpoints with SageMaker endpoints, and conducting prediction functions. Results from endpoint prediction are compared to those from batch transform, and time measurements are obtained using AWS cloudwatch logs. AWS EC2 provides virtual resources for the process, aiming to improve accuracy and speed of website phishing detection by leveraging AWS machine learning services and the cyber kill chain concept.

2. METHOD

This study presents a cloud-based ML model that aims to enhance the accuracy and speed of website phishing detection by combining the steps of the cyber kill chain concept and AWS. Figure 1 represents the framework of the suggested CML model in the case of an attack. Figure 2 represents the steps of the proposed model which consists of seven stages.

In the first stage, the main focus is to align the reconnaissance step of the cyber kill chain concept with the data collection stage. This alignment ensures a systematic and methodical approach to gathering the required dataset. During this stage, the system first acquires the dataset from the designated source, whether it be from internal records or external data providers. Subsequently, a comprehensive analysis of the obtained data is conducted to understand its structure, quality, and relevance to the project's objectives. Once the dataset is thoroughly analyzed and validated, it is then uploaded to the AWS S3 bucket.

The second stage involves data preprocessing, which adapts the weaponization step of cyber kill chain model. This stage preprocesses the dataset by performing six steps, such as feature reduction, feature selection, standardization, label encoding, dataset splitting, and feature scaling. These preprocessing steps are performed using SageMaker studio notebook and the results are stored in the S3 bucket while the preprocessing resources are obtained from AWS EC2 instance (ml.m4.xlarge) which provides four virtual CPUs, 16GB of memory, and high network performance for 32/64-bit operating systems.

In the third stage, the delivery step of the cyber kill chain concept is adapted to the training stage of the proposed model, where three ML algorithms (decision tree (DT), random forest (RF), and support vector machine (SVM)) are trained using SageMaker studio python scripts. The results of the training are stored in the S3 bucket, the time of training is recorded in AWS cloudwatch, and the resources required for training are obtained from the EC2 instance (ml.m4.xlarge). In the fourth stage, the exploitation step of the cyber kill chain concept is adapted to the batch transform stage of the proposed model, which uses the training results to obtain evaluation results for each algorithm to measure model performance and accuracy, batch transform time for each algorithm is calculated in cloudwatch and the required resources are obtained from the EC2 instance (ml.m4.xlarge).

The fifth stage the fifth step involves adapting the installation process of the cyber kill chain concept. This adaptation enables the deployment of each algorithm using three separate single-model endpoints, which enhances the model's scalability and efficiency. Furthermore, during the deployment, the system records the deployment time in CloudWatch, allowing for monitoring and analysis of performance metrics. To support the deployment process, the system acquires deployment resources from an EC2 instance with the specifications (ml.m4.xlarge).

In the sixth stage, the primary objective of the sixth stage is to adapt the command-and-control step of the cyber kill chain concept to the prediction stage of the proposed model. This adaptation is crucial for establishing a seamless and efficient prediction process. In this stage, the system executes a prediction function on each endpoint, enabling the model to generate accurate and reliable final prediction results.

The final stage of the proposed model adapts the action on objectives step of the cyber kill chain concept, which involves obtaining the prediction results and comparing them to the results of batch transform.

The time required for prediction is recorded in cloudwatch and compared to the time of the batch transform as well. By leveraging AWS services and the cyber kill chain concept, the proposed model aims to enhance the speed and accuracy of website phishing detection.

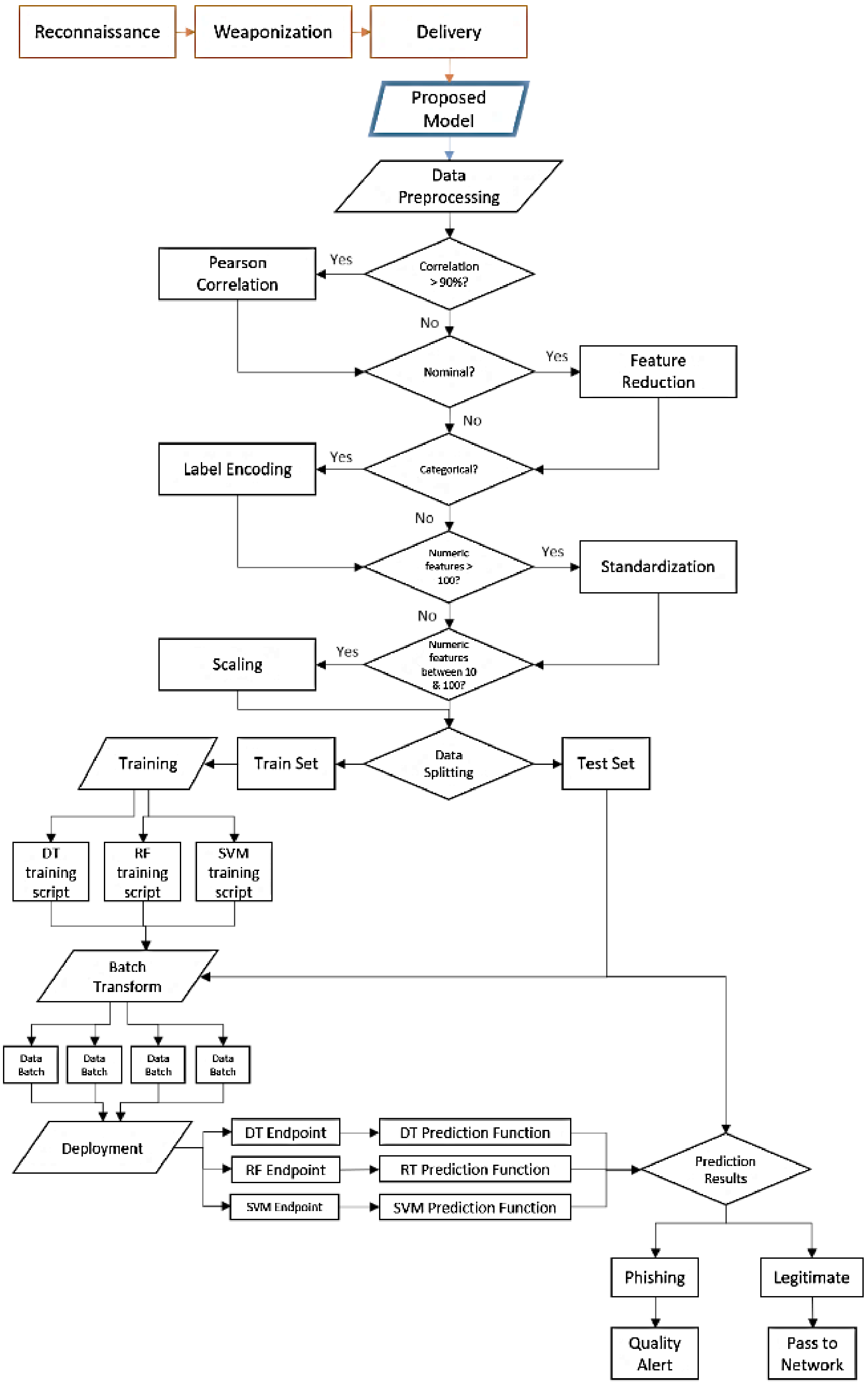


Figure 1. The framework of the proposed model in case of an attack

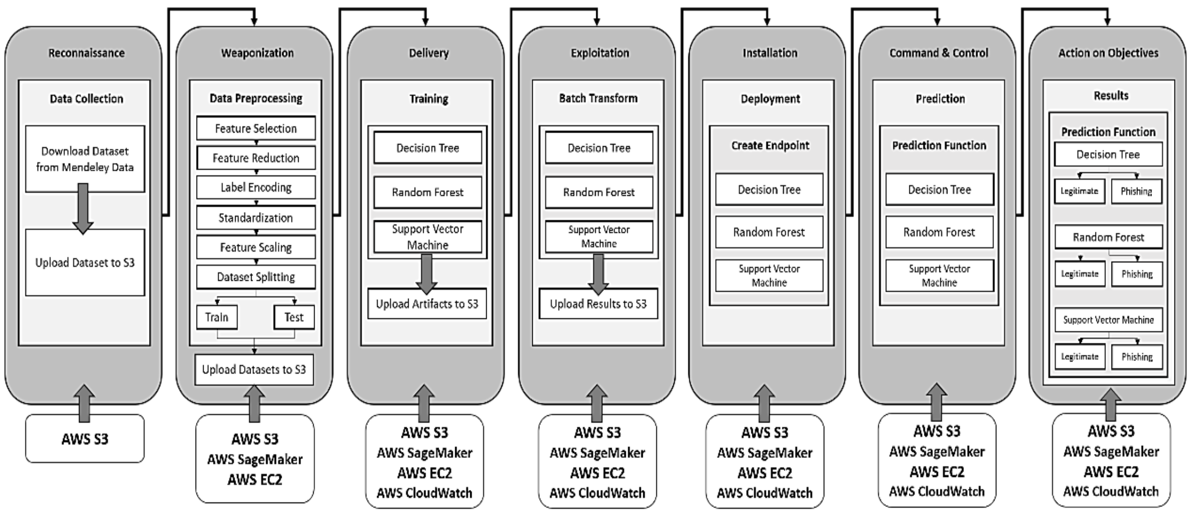


Figure 2. The steps of the suggested model

2.1. Dataset collection and description

In the dataset description stage, the first step of cyber kill chain, reconnaissance, is combined and adapted. This step is where the attacker gathers information about the target through passive reconnaissance or active reconnaissance. In our proposed model, the dataset that has been utilized in the present study, as described in Table 1, is named “web page phishing detection.” It is downloaded from Mendeley data, encompasses 11,430 URLs, and was created in May 2020. The dataset contains 89 features, with 87 being numeric, one being a nominal feature representing the dataset’s index in URLs, and one categorical feature representing the labels. The 87 retrieved features were divided by the study to 3 groups: 56 features generated from URL syntax, 24 features derived from the website content, and 7 features derived from website services.

Table 1. Description of dataset used in the proposed model

Characteristic	Value
Name	Webpage phishing detection
Source	Mendeley data
Type	Binary/Supervised
No. of samples	11430
No. of extracted features	87
Types of features	Numeric, nominal, and categorical
Missing data	No
Balanced	Yes

2.2. Dataset preprocessing

In this stage, the concept of the weaponization step of the cyber kill chain is adopted. At this stage, the attacker creates a weapon to exploit the vulnerabilities and weaknesses identified during the reconnaissance stage. In this study, based on the analysis of the dataset, we reached the conclusion that the dataset is balanced, containing 50% legitimate URLs (5,715 samples) and 50% phishing URLs (5,715 samples) with no missing values, in that way simplifying the preprocessing step into the following six procedures:

First, feature selection was conducted using Pearson correlation to eliminate highly correlated features, reducing redundancy and potential negative impacts on model performance. Three features with a correlation ratio above 90% were removed. Next, feature reduction was applied by removing the nominal feature “URL,” which represented the names of the URLs in the dataset, aiming to enhance model accuracy by reducing noise. Categorical label values were encoded into numeric values through label encoding, ensuring compatibility with machine learning algorithms. The “status” feature representing dataset classes was encoded, with “legitimate” converted to 0 and “phishing” converted to 1. Standardization using python’s standardScaler () was employed to transform features, giving them an average value of 0 and a standard deviation of 1. This step was crucial for algorithms like SVM that require standardized data. Standardization was performed on features with values higher than 100.

Additionally, feature scaling was performed using python's MinMaxScaler () to rescale data within a fixed range, typically between 0 and 1. This normalization prevented the dominance of certain features over others and was applied to features with values between 10 and 100. The data-set has been split to train and test datasets using 70:30 ratio, with 70% (8,001 samples) assigned to the train dataset and 30% (3,425 samples) to the test dataset. Finally, the datasets were uploaded to S3 for further processing, ensuring easy access and storage for the proposed model.

2.3. Training algorithms

This stage will be matched to the delivery step of the cyber kill chain concept. This step involves the delivery of the weapon to the target by the attacker through various methods. This study will use three algorithms to train the model and evaluate its performance. These algorithms are DT, RF, and SVM. These algorithms will be embedded into the SageMaker notebook using separate python scripts for each algorithm.

2.3.1. Decision tree

Machine learning algorithms like DT could be applied to regression and classification problems. It creates a model that resembles a tree, with internal nodes standing in for the data-set's features, branches for decision-making processes depending on the feature values, and leaves for the class labels. The feature that maximizes the information gain or minimizes the impurity measure is used by the method to recursively divide the data into smaller subgroups. DT models can manage both numerical and categorical data, are easily interpretable and visualized, and can handle high-dimensional datasets. These are just a few benefits of using DT models. They are also simple and easy to understand, validate, and interpret. Given a set of training samples $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ where x_i is the feature vector and y_i is the corresponding class label, a DT is constructed recursively using the following steps: choose the most suitable feature to divide data based on a criterion like Gini index or information gain. Based on the selected feature, divide the data into two subsets. Once a stopping requirement, like maximum depth or minimal number of samples per leaf, has been met, recursively repeat steps 1 and 2 on each subset. Assign the predicted class label to the sample that has the highest percentage of that class within each leaf node. Depending on a measurement of how well the feature divides the classes, the best feature to split the data is chosen. Information gain, which is a frequently used metric and is defined as:

$$IG(D, f) = H(D) - H(D|f) \tag{1}$$

where D is the set of training samples, f is a feature, $H(D)$ is the entropy of D, and $H(D|f)$ is the conditional entropy of D given f. Entropy is a measure of impurity or uncertainty, and is defined as:

$$H(D) = -\sum(p(i) * \log_2 p(i)) \tag{2}$$

where $p(i)$ is the proportion of samples in class i. The higher the entropy, the more uncertain the class distribution. The conditional entropy is defined as:

$$H(D|f) = \sum(\frac{|D(j)|}{|D|} * H(D(j))) \tag{3}$$

where $D(j)$ represent the subset of D for which feature f has value j. The lower the conditional entropy, the better the feature separates the classes [23].

2.3.2. Random forest

The RF is another popular algorithm which can be utilized for the tasks of classification and regression. It's a collection of a number of the trees that are trained independently on selected training data-sets. The information of the classification is determined then through voting amongst all trained trees. therefore, RF typically achieves a better accuracy of classification in comparison with a single tree. The algorithm is referred to as the random forest due to the way that the randomness is injected into tree-building process for ensuring that every tree differs. Considering a set of the training samples $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, where x_i is the feature vector and y_i is the corresponding class label, RF algorithm constructs a model which predicts binary class label of a new input vector *x* as follows: randomly choose a sub-set of the features from the original set of feature: Let *F* be the total number of the features and *k* be the number of the features to select at every split. Selecting *k* features may be randomly carried out with the use of a uniform distribution, and *k* value can be set as function of *F*, usually, $\sqrt[2]{F}$, construct a decision tree with the use of a sub-set of the training samples, which are chosen by either bootstrap sampling or sub-sampling with no replacement: Let *S* be the total number of the training samples and *m* be the number of the samples to select for every tree. The sampling may be carried out either with the replacement (i.e., bootstrap sampling) or with no replacements (i.e., sub-sampling). The value of m may be set as function of S, usually, *S/3*, repeat step 1 and step 2 for a fixed number of times or to the point where a stopping criterion is met: Let T be total number of the trees to construct. T value can be set through the cross-validation or grid search, and aggregate predictions of all of the DTs through either weighted voting or majority voting: Let $T_i(x)$ be the binary prediction of the tree i for input vector x. In the majority voting, the final prediction can be expressed by:

$$h(x) = argmax\left(1 \leq j \leq C\right) \sum T_i(x) = j \tag{4}$$

where C represent the number of the classes, and $argmax$ represent the function returning the index of maximum element in the vector. In weighted voting, each tree is assigned a weight based on its performance, and the final prediction is given by:

$$h(x) = argmax\left(1 \leq j \leq C\right) \sum (w_i T_i(x) = j) \tag{5}$$

where w_i is the weight of tree i, and the weights can be calculated based on the accuracy or the error of each tree [24].

2.3.3. Support vector machine

Popular algorithm SVM offers a quick and effective implementation. SVM creates a hyperplane, a 2D line dividing the classes, throughout the training phase. In order to identify whether a webpage is being utilized for phishing, SVM analyzes a variety of factors, like the presence or lack of specific characters. The most popular kernel for classification, especially in datasets with a lot of features, is the linear kernel. Because of its success in enhancing dataset performance and performance assessment, this kernel is extensively utilized. In machine learning, binary classification problems involve classifying data into two categories, which can be represented as $y_i \in \{1,-1\}\ \forall x_i$. The training data-set contains all x_i samples as well as their corresponding y_i values that indicate the class to which data belongs. The SVM algorithm attempts to find an optimal hyperplane that can divide data to two regions, with all positive class samples in one region and negative class samples in the other. Predictions for new data can be made through evaluating its position on the graph and returning a positive or negative value using a sign function. None-the-less, it is seldom possible to linearly solve binary classification problems, as a d-dimensional graph cannot be separated by a d -1-dimensional hyperplane. Therefore, kernel functions are utilized in order to map data to a higher dimension. In the event that the data remains inseparable even after using a kernel function, the SVM algorithm aims to minimize the error rate. Given a vector X of input vectors $x_i \in R_p$ for $i=1$ to n, where p is the number of input features, and a vector $Y \in \{-1,\ 1\}n$ of classes y_i, where n is the number of input vectors, SVM algorithm solves the following problem:

$$min_{v,q,z} \frac{1}{2} ||v||^2 + C \sum_{i=1}^{n} z_i \tag{6}$$

subject to,

$$h_i(v^T x_i + b) \geq 1 - z_i, i = 1, \dots, n \tag{7}$$

$$z_i \geq 0, i = 1, \dots, n \tag{8}$$

where the hyperplane is written as $h = v^T X + q$, v is the vector and q are the scalar quantity. The parameter C is utilized to regulate the tradeoff between bias and variance of the model by penalizing misclassifications. Lower values of C result in higher bias, while larger values increase variance. To avoid overfitting of the dataset, positive slack variables z_i are introduced for regularization. Through forming the Lagrangian equation of the problem and the substitution of values with the use of Karush-Kuhn-Tucker conditions, the dual problem can be obtained, which can be formulated as follows:

$$max_{\propto} W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j K(x_i, x_j) \tag{9}$$

subject to,

$$0 \leq \alpha_i \leq C, i = 1, \dots, n \tag{10}$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{11}$$

where α is a vector that consists of all Lagrangian multipliers α_i, $K(x_i, x_j) = \emptyset(x_i)^T \emptyset(x_j)$ is the kernel function of feature mapping $\emptyset$ that maps from input *x* to its features in higher dimension $\emptyset(x)$. Similarly, we can express the dual in vector form as:

$$min_{\alpha} \frac{1}{2} \alpha^T Q \alpha - e^T \alpha \tag{12}$$

subject to,

$$0 \leq \alpha_i \leq C, i = 1, \ldots, n \tag{13}$$

where $Q_{ij} = y_i y_j K(x_i x_j)$ and *e* is a vector of ones. Radial basis function (RBF) kernel has been utilized as kernel function for mapping data to higher dimensions. Because of its accuracy and dependable performance, RBF kernel function has gained popularity among many researchers in various domains. RBF kernel function can be described as follows:

$$K(x_i x_j) = \exp\left(-\gamma ||x_i - x_j||^2\right) \tag{14}$$

which is why, for constructing SVM model, it takes in 2 parameters, which are: *C* and *γ* that can be optimized with the use of the optimization algorithms [25].

2.4. SageMaker scripts

Through simply calling the algorithm inside SageMaker notebook, AWS SageMaker has been known to utilize the amazon machine learning (AML) algorithm library to give highly accurate built-in algorithms for the development and training of the scalable and effective cloud ML models. Not all ML techniques, nevertheless, are supported by the SageMaker environment. For building the models, the developers occasionally need to incorporate adjustable algorithms. In this situation, algorithm scripts offer a more adaptable way to execute algorithms with greater control over input/output format and hyperparameters, as well as the capability for containing data loading, training, pre-processing, and saving steps all in one script. This enables developers to improve the performance of their models and modify the algorithms to suit the particular requirements of their use cases, 3 scripts will be used in this work to activate three different ML algorithms for training and deploying models: RF, DT, and SVM. The training results will be put on S3 and later retrieved for the prediction stage.

3. RESULTS AND DISCUSSION

After training the model using the three algorithms mentioned previously, the training artifacts of each algorithm are uploaded to the S3 bucket and a batch transform procedure will be conducted on these artifacts to obtain the evaluation results of each algorithm. Next, endpoints will be created to deploy the model. Then, a prediction function will be conducted on each endpoint to obtain the final results, Finally, a comparison of results and times required to obtain these results with those of batch transform will be applied to determine the best accuracy and the highest speed of the acquired results.

3.1. Evaluation using batch transform

In this study, batch transform will be adapted to the exploitation step of the cyber kill chain concept. At this stage, the attacker exploits the vulnerability using the weapon that was delivered. Figure 3 represents the steps of batch transform of the proposed model. Through breaking the process down into records or mini-batches, the batch transform SageMaker feature may be used to create predictions from a trained model on large data-sets. As shown in the mentioned figure, it is appropriate for the one-time processing, provides insightful information about the status and effectiveness of the transform task, and stores the generated predictions in S3. Batch transform is more economical and flexible than real-time inference, but not being optimal for making predictions in real-time. It supports a broad variety of frameworks and models, could handle massive volumes of input data and scale to handle more data, and gives users the opportunity to customize input and output data formats. To analyze data gathered over a long period of time or to execute large-scale data analysis without having to worry about latency or online deployment, batch transform is often employed offline. Soltys [26] evaluation results are acquired with the use of 8 evaluation metrics: precision, accuracy, F-1, recall, receiver operating characteristic (ROC), negative prevalence, specificity, and mislabeling. Batch transform is then performed on the training outcomes of each method using the test set. As a result of achieving the greatest accuracy score, RF surpassed both DT and SVM, according to the data.

Figure 4 show the evaluation metrics used for the proposed model. The results show that RF had the highest accuracy score of 97% with high precision, F-1, recall, specificity, and negative prevalence, this indicates that the algorithm was able to predict the labels of most samples, correctly identify most positive samples with a low false alarm rate which leads to a high positive rate and low negative rate. RF also has the lowest mislabeling score of 0.03%, indicating that the dataset preprocessing is sufficient to process and the algorithm was able to learn from it effectively. RF classifier's excellent accuracy results could be ascribed to a number of variables. It starts by employing ensemble learning, which uses a group of DTs to lessen overfitting. Second, it uses randomness for reducing variance and noise when choosing features and data, which makes the model less sensitive to small changes in data. Also, it employs bagging methods to provide numerous data sets for each DT, reducing bias and enhancing generalization. Last but not least, the RF classifier lowers mislabeling by combining the output of various DT via a voting method. Those elements work together to give the RF classifier's excellent accuracy performance.

Secondly, SVM scored 94% accuracy with a 95% precision score, and similar F-1 and ROC scores. SVM scored 94% recall, 96% negative prevalence, and 0.05 mislabeling score, the reason behind these results returns to the nature of SVM binary classification algorithm in maximizing margin between the classes instead of the use of probabilistic models which makes it robust to outliers and make the algorithm focusing on the classification of the positive samples correctly even if that means classifying a few negative samples incorrectly. Also, the use of SVM of the weighted cost function is one of the reasons the algorithm has a high negative prevalence value.

Additionally, DT achieved a 93% accuracy rating with a mislabeling score of 0.07% and comparable high precision, recall, F-1, negative prevalence, and specificity. Numerous variables contribute to DT classifier's excellent accuracy. First of all, DT are simple to interpret and comprehend. Second, DT are resistant to outliers since they do not make any assumptions regarding the distribution of data. In addition, DT employ feature selection methods to lessen data noise. To record complex decision boundaries, the feature space is recursively split into binary splits. Last but not least, DT use trimming strategies to prevent overfitting. Those elements work together to give the DT classifier's excellent accuracy performance.

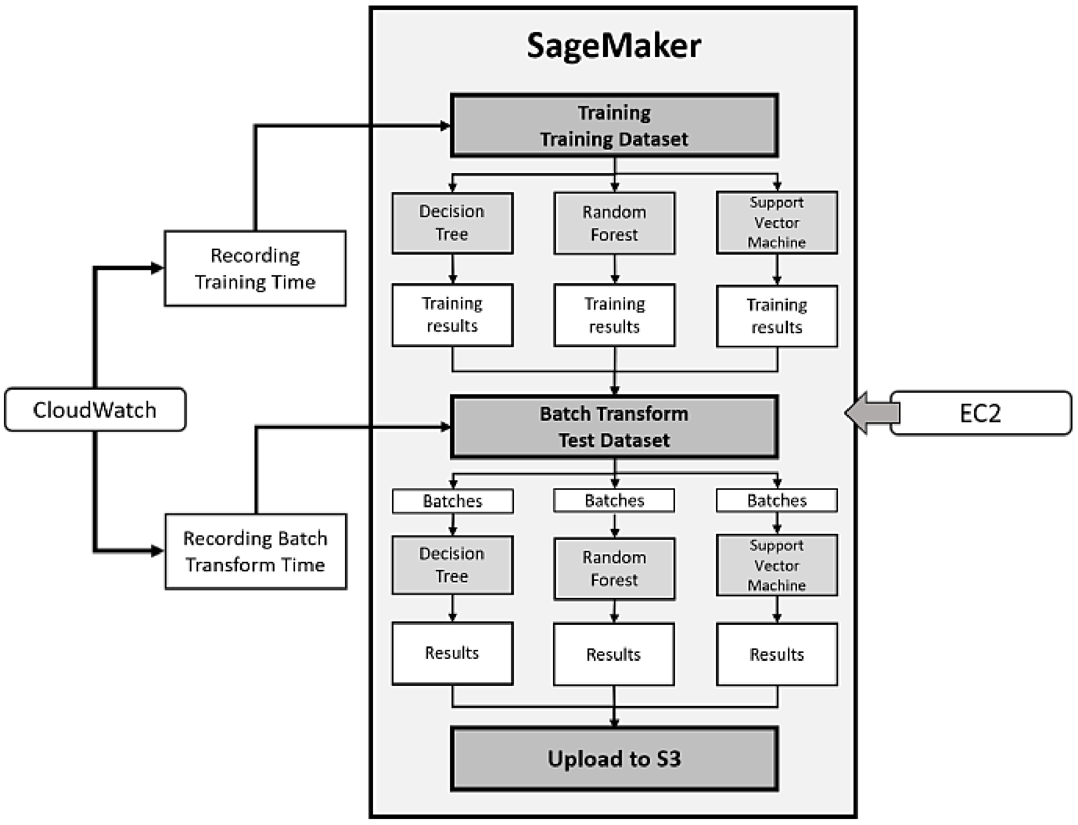
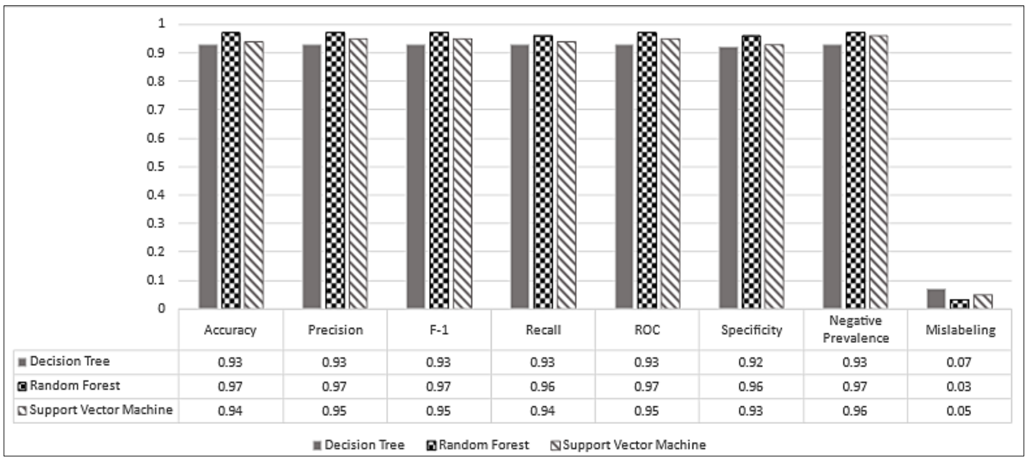


Figure 3. Steps of the batch transform procedure of the proposed model



	Accuracy	Precision	F-1	Recall	ROC	Specificity	Negative Prevalence	Mislabeling
■ Decision Tree	0.93	0.93	0.93	0.93	0.93	0.92	0.93	0.07
▣ Random Forest	0.97	0.97	0.97	0.96	0.97	0.96	0.97	0.03
▨ Support Vector Machine	0.94	0.95	0.95	0.94	0.95	0.93	0.96	0.05

Figure 4. Comparison of evaluation metrics of the proposed method

3.2. Deployment the model using endpoints

This study will adapt the Installation step of the cyber kill chain concept to the endpoint deployment stage, installation involves installing the malicious entity on the target's system by the attacker, while the action step involves. Endpoints are one of the key components of amazon SageMaker, which provides a way to deploy trained models and make predictions in real time. Endpoints are a critical aspect of building scalable models for use in real-world applications, such as phishing detection, fraud detection, and natural language processing. Figure 5 displays the steps of deploying the endpoint for the proposed model. After training the model using algorithms, a model artifact is created that contains the parameters and additional resources required by the model. Upon deployment, endpoints provide a URL that can be used to send new data to the model and return predictions based upon sent data. It is crucial to monitor and maintain endpoints to ensure the accuracy and effectiveness of the prediction. Endpoints have several advantages, including scalability to handle large datasets and high volumes of traffic, and the ability to make real-time predictions on new data that the model has not trained on. Endpoints can also avoid latency, even on large amounts of data. To create an endpoint, a model needs to be created which contains the container where the training results are stored on Amazon S3 and the model artifacts needed to deploy the model. Then, the endpoint configuration is created which contains the destination path of the captured data and the resources needed to deploy the endpoint. Finally, the endpoint is created, and the time to create the endpoint is dependent upon several factors like the size of the data and the number of models it contains. In this work, three distinct, single-model endpoints will be utilized to deploy each of the algorithms utilized for training the suggested model. On each endpoint, a prediction function will be run to determine the prediction outcomes and time for each method.

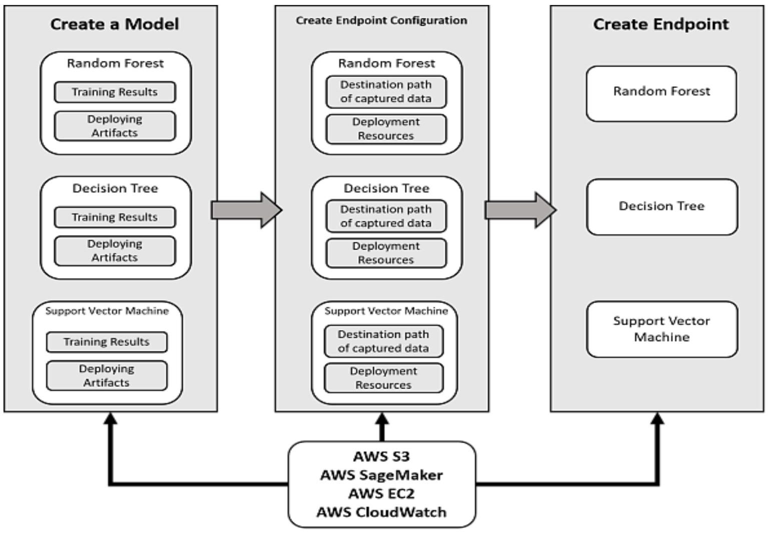


Figure 5. The steps of endpoint deployment of the proposed model

3.3. Comparison of time

By leveraging Amazon CloudWatch logs, the training, batch transform, and prediction times for each algorithm have successfully computed. This data-driven approach forms the foundation of a robust performance comparison between the algorithms. Through this comprehensive analysis, valuable insights into the efficiency of each algorithm are gained, facilitating an informed decision-making process and ensuring an agile, and high-performing model for website phishing detection.

3.3.1. Training time

Figure 6 displays the training times for each one of the algorithms following recording the times for each algorithm. As a result of not requiring the optimization of intricate mathematical operations, the results indicated that DT recorded 153 seconds of training time. Additionally, the way where the DT splits the data might result in a decrease in overfitting and an improvement in generalization performance. The reason for this is that RF has a parallelization feature that allows it to handle large datasets through using parallel computing model and virtual multi-core processors that are offered by Amazon EC-2.

Additionally, RF utilizes a subset sampling method for sub-sampling the training data, which decreases the amount of data that is processed at every node and, as a result, shortens the training time. The SVM achieved the quickest training time of 155.3 seconds due to its capacity to learn complex decision boundaries without explicitly computing the modified data. Additionally, SVM can optimize data effectively thanks to convex optimization when used with SVM. Additionally, compared to other algorithms, SVM's nature makes it easier to train and requires less time for training data because there are less parameters to adjust.

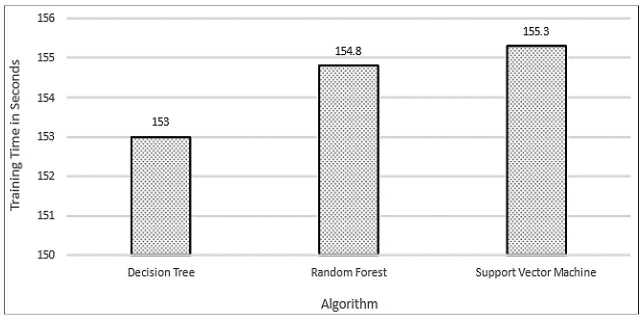


Figure 6. Training time of the algorithms used in the proposed model

3.3.2. Batch transform time

The results indicate that RF had the shortest batch transform time, with 300 seconds as shown in Figure 7, whereas SVM and DT had slightly slower processing times of 310 and 320 seconds, respectively. These batch transform times were obtained for the 3 algorithms that have been utilized in the suggested model from the cloudwatch logs. Those findings suggest that while batch transform could process extremely large datasets with high accuracy and is more cost-effective in comparison with the real-time inference because users just pay for the time spent processing the input data instead of maintaining a persistent endpoint, batch transform setup could be time-consuming because it requires the creation and configuration of numerous resources and batch transform jobs. Which is why, it cannot be used to make predictions in real time.

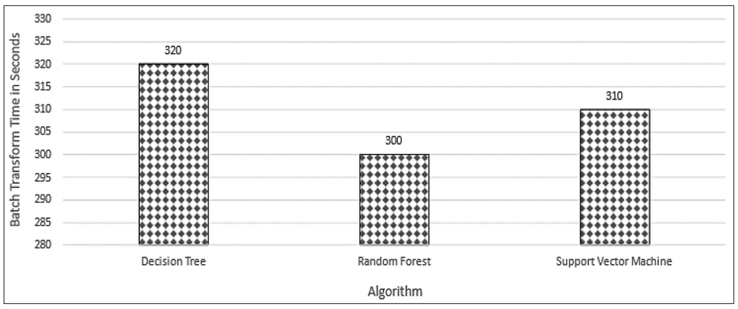


Figure 7. Batch transform time of the algorithms used in the proposed model

3.3.3. Endpoint prediction time

In this study, the last two steps of the cyber kill chain, namely command and control (C&C) and action on objectives, will be adapted to the last two stages of the proposed model. C&C involves establishing a command-and-control channel to communicate with the malicious entity and control the system, while in the action step the attacker achieves their objectives. After deploying the model using 3 separate single-model endpoints and conducting the prediction function on every endpoint, the AWS cloudwatch logs monitored the performance and recorded the time required for the prediction and evaluation. While the evaluation results remained unchanged after the endpoint deployment, the prediction time has decreased to a near real-time point as displayed in Figure 8.

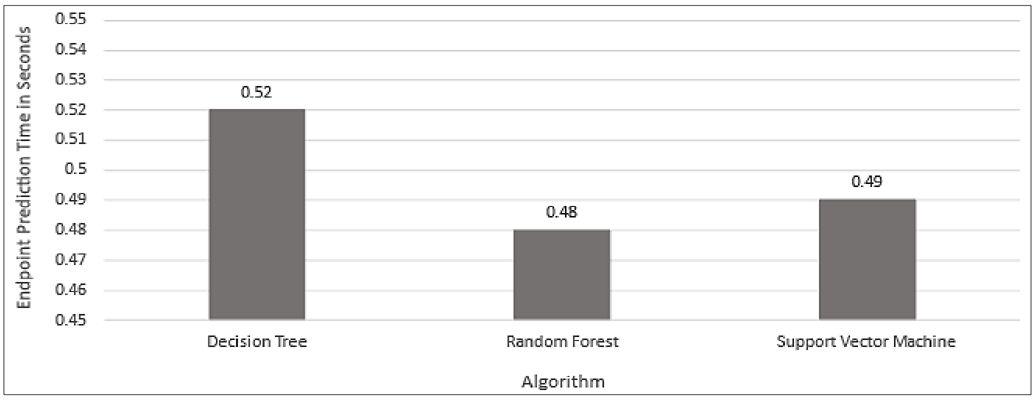


Figure 8. Endpoint prediction time for the algorithms used in the proposed model

The results showed that RF had the shortest prediction time of 0.48 seconds, DT recorded a prediction time of 0.25 seconds, while SVM recorded a prediction time of 0.49 seconds. The main reason why endpoint prediction time is faster than batch transform is that endpoint prediction requires much fewer computational resources and can be done quickly because it has all the necessary virtually unlimited resources and because the machine learning model is already trained, and the prediction can be generated quickly by passing the input data through the model's pre-trained weights. Furthermore, endpoint prediction is often performed on dedicated hardware, which is optimized for fast computation, further reducing the prediction time. Figure 9 show the comparison of training, batch transform, and endpoint prediction times of the proposed model.

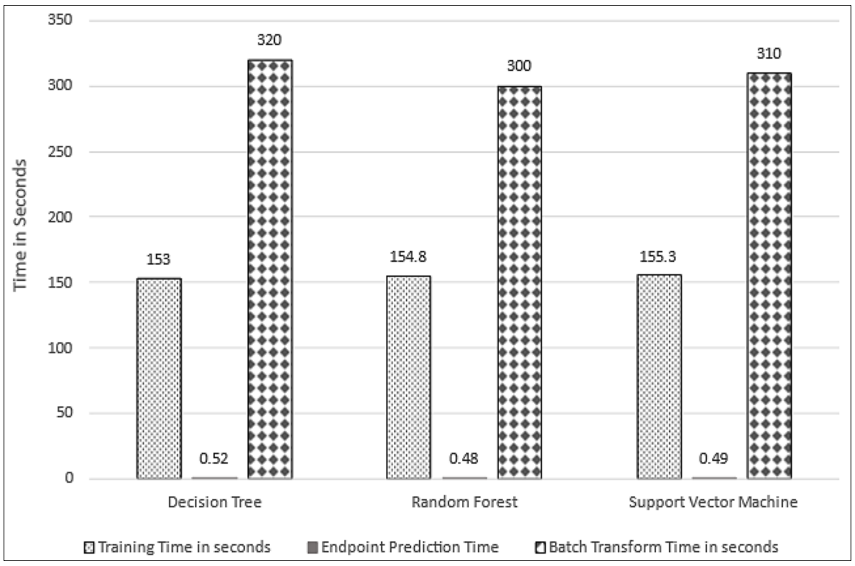


Figure 9. Comparison between training, batch transform, and endpoint prediction times of the proposed model

4. CONCLUSION

The cyber kill chain concept is combined with machine learning methods in this study to propose a novel approach for increasing the accuracy and speed of website phishing detection. To do this, the authors used SageMaker and S3 to preprocess a dataset of 11,430 URLs through AWS. With the use of SageMaker batch transform, three machine learning algorithms-DTs, RFs, and SVMs-were evaluated, trained, and deployed as individual endpoints. The times needed for such predictions have been derived with the use of AWS cloudwatch logs, and endpoint predictions have been compared to those produced with the use of batch transform. AWS EC2 delivered virtual resources in the form of the (ml.m4.xlarge) instance. The results specified that the RF algorithm had the shortest batch transform and endpoint prediction speeds, coming in at 300 and 0.48 seconds, respectively, and had the highest accuracy score of 97%. With a short batch transform and endpoint prediction time of 310 seconds and 0.49 seconds, respectively, SVM achieved an accuracy score of 94%. With an endpoint prediction time of 0.52 seconds, a batch transform time of 320 seconds, and training time of 153 seconds, DT had a 93% accuracy rating. The authors came to the conclusion that their method, which makes use of AWS machine learning capabilities and the cyber kill chain idea, significantly increases website phishing detection accuracy and speed when processing large datasets.

ACKNOWLEDGEMENTS

Author thanks AWS Academy in Tikrit University for their help in developing this paper.

REFERENCES

[1] J. M. Vidal, M. A. S. Monge, S. M. M. Monterrubio, L. I. B. Lopez, and A. L. V. Caraguay, "Profits at the dawn of cybercrime-as-a-service," in *Proceedings - 2019 International Conference on Information Systems and Software Technologies, ICI2ST 2019*, Nov. 2019, pp. 71–78, doi: 10.1109/ICI2ST.2019.00017.

[2] J. An and H. W. Kim, "A data analytics approach to the cybercrime underground economy," *IEEE Access*, vol. 6, pp. 26636–26652, 2018, doi: 10.1109/ACCESS.2018.2831667.

[3] H. Aldawood and G. Skinner, "An advanced taxonomy for social engineering attacks," *International Journal of Computer Applications*, vol. 177, no. 30. pp. 1–11, 2020, doi: 10.5120/ijca2020919744.

[4] J. V. Jawade and S. N. Ghosh, "Phishing website detection using fast.ai library," in *Proceedings - International Conference on Communication, Information and Computing Technology, ICCICT 2021*, Jun. 2021, pp. 1–5, doi: 10.1109/ICCICT50803.2021.9510059.

[5] K. R. Ferreira *et al.*, "Using remote sensing images and cloud services on Aws to improve land use and cover monitoring," in *2020 IEEE Latin American GRSS and ISPRS Remote Sensing Conference, LAGIRS 2020 - Proceedings*, Mar. 2020, pp. 558–562, doi: 10.1109/LAGIRS48042.2020.9165649.

[6] T. Dargahi, A. Dehghantanha, P. N. Bahrami, M. Conti, G. Bianchi, and L. Benedetto, "A cyber-kill-chain based taxonomy of crypto-ransomware features," *Journal of Computer Virology and Hacking Techniques*, vol. 15, no. 4, pp. 277–305, Dec. 2019, doi: 10.1007/s11416-019-00338-7.

[7] P. Mccaffrey, "Cloud technologies," *An Introduction to Healthcare Informatics*, Academic Press, pp. 307–316, 2020, doi: 10.1016/B978-0-12-814915-7.00021-1.

[8] B. Qolomany, I. Mohammed, A. Al-Fuqaha, M. Guizani, and J. Qadir, "Trust-based cloud machine learning model selection for industrial iot and smart city services," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2943–2958, Feb. 2021, doi: 10.1109/JIOT.2020.3022323.

[9] A. Choudhary, "A walkthrough of amazon elastic compute cloud (Amazon EC2): a review," *International Journal for Research in Applied Science and Engineering Technology*, vol. 9, no. 11, pp. 93–97, Nov. 2021, doi: 10.22214/ijraset.2021.38764.

[10] M. Aleisa, A. A. Hussein, F. Alsubaei, and F. T. Sheldon, "Performance analysis of two cloud-based iot implementations: empirical study," in *Proceedings - 2020 7th IEEE International Conference on Cyber Security and Cloud Computing and 2020 6th IEEE International Conference on Edge Computing and Scalable Cloud, CSCloud-EdgeCom 2020*, Aug. 2020, pp. 276–280, doi: 10.1109/CSCloud-EdgeCom49738.2020.00055.

[11] P. Khot, "Cloud migration with google cloud storage and AWS (S3 Service)," *International Journal for Research in Applied Science and Engineering Technology*, vol. 8, no. 10, pp. 555–558, Oct. 2020, doi: 10.22214/ijraset.2020.31953.

[12] E. Liberty *et al.*, "Elastic machine learning algorithms in amazon sagemaker," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Jun. 2020, pp. 731–737, doi: 10.1145/3318464.3386126.

[13] B. Espinoza, J. Simba, W. Fuertes, E. Benavides, R. Andrade, and T. Toulkeridis, "Phishing attack detection: a solution based on the typical machine learning modeling cycle," in *Proceedings - 6th Annual Conference on Computational Science and Computational Intelligence, CSCI 2019*, Dec. 2019, pp. 202–207, doi: 10.1109/CSCI49370.2019.00041.

[14] C. Y. Wu, C. C. Kuo, and C. S. Yang, "A phishing detection system based on machine learning," in *Proceedings - 2019 International Conference on Intelligent Computing and Its Emerging Applications, ICEA 2019*, Aug. 2019, pp. 28–32, doi: 10.1109/ICEA.2019.8858325.

[15] N. N. Gana and S. M. Abdulhamid, "Machine learning classification algorithms for phishing detection: a comparative appraisal and analysis," in *2019 2nd International Conference of the IEEE Nigeria Computer Chapter, NigeriaComputConf 2019*, Oct. 2019, pp. 1–8, doi: 10.1109/NigeriaComputConf45974.2019.8949632.

[16] M. G. Hr, A. Mv, S. Gunesh Prasad, and S. Vinay, "Development of anti-phishing browser based on random forest and rule of extraction framework," *Cybersecurity*, vol. 3, no. 1, p. 20, Dec. 2020, doi: 10.1186/s42400-020-00059-1.

[17] W. Bai, "Phishing website detection based on machine learning algorithm," in *Proceedings - 2020 International Conference on Computing and Data Science, CDS 2020*, Aug. 2020, pp. 293–298, doi: 10.1109/CDS49703.2020.00064.

[18] M. N. Alam, D. Sarma, F. F. Lima, I. Saha, R. E. Ulfath, and S. Hossain, “Phishing attacks detection using machine learning approach,” in *Proceedings of the 3rd International Conference on Smart Systems and Inventive Technology, ICSSIT 2020*, Aug. 2020, pp. 1173–1179, doi: 10.1109/ICSSIT48917.2020.9214225.
[19] B. Geyik, K. Erensoy, and E. Kocyigit, “Detection of phishing websites from URLs by using classification techniques on WEKA,” in *Proceedings of the 6th International Conference on Inventive Computation Technologies, ICICT 2021*, Jan. 2021, pp. 120–125, doi: 10.1109/ICICT50816.2021.9358642.
[20] A. A. Orunsolu, A. S. Sodiya, and A. T. Akinwale, “A predictive model for phishing detection,” *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 2, pp. 232–247, Feb. 2022, doi: 10.1016/j.jksuci.2019.12.005.
[21] Y. Wei and Y. Sekiya, “Sufficiency of ensemble machine learning methods for phishing websites detection,” *IEEE Access*, vol. 10, pp. 124103–124113, 2022, doi: 10.1109/ACCESS.2022.3224781.
[22] S. Das Guptta, K. T. Shahriar, H. Alqahtani, D. Alsalman, and I. H. Sarker, “Modeling hybrid feature-based phishing websites detection using machine learning techniques,” *Annals of Data Science*, Mar. 2022, doi: 10.1007/s40745-022-00379-8.
[23] R. Manne, “Machine learning techniques in drug discovery and development,” *International Journal of Applied Research*, vol. 7, no. 4, pp. 21–28, Apr. 2021, doi: 10.22271/allresearch.2021.v7.i4a.8455.
[24] T. O. Ojewumi, G. O. Ogunleye, B. O. Oguntunde, O. Folorunsho, S. G. Fashoto, and N. Ogbu, “Performance evaluation of machine learning tools for detection of phishing attacks on web pages,” *Scientific African*, vol. 16, p. e01165, Jul. 2022, doi: 10.1016/j.sciaf.2022.e01165.
[25] U. A. Butt, R. Amin, H. Aldabbas, S. Mohan, B. Alouffi, and A. Ahmadian, “Cloud-based email phishing attack using machine and deep learning algorithm,” *Complex and Intelligent Systems*, vol. 9, no. 3, pp. 3043–3070, Jun. 2023, doi: 10.1007/s40747-022-00760-3.
[26] M. Soltys, “Cloudifying the Curriculum with AWS,” in *Proceedings - Frontiers in Education Conference, FIE*, Oct. 2021, pp. 1–7, doi: 10.1109/FIE49875.2021.9637242.

BIOGRAPHIES OF AUTHORS

Saba Hussein Rashid is a candidate in the program of Master in computer science, Tikrit University, Iraq. She received her B.Sc. degree in computer science from Tikrit University, Iraq in 2010. She is currently working as a website administrator at Tikrit University, Iraq. She can be contacted at email: sabahussein88@tu.edu.iq.

Wisam Dawood Abdullah is an Associate Professor and a faculty member at Tikrit University. He received his B.Sc. Degree in Computer Science from Tikrit University, Iraq, and his M.S. degree in Information Technology (with a concentration in telecommunications and networks) from the University Utara Malaysia (UUM). He received an expert certification from Cisco Networking Academy CCNP, CCNA, CCNA security, IoT, entrepreneurship, grid, voice, wireless cloud, Linux, CCNA cybersecurity, and IT. In addition, he is a NetAcad administrator at Cisco Networking Academy, Iraq. Recently, he is selected as AWS Community Builder at Amazon. His research interest includes protocol engineering, network analysis, cybersecurity, cloud computing, network traffic analysis, data mining, future internet, internet of things, AI, and ML. He can be contacted at email: wisamdawood@tu.edu.iq.