IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 15, No. 1, February 2026, pp. 681~694
ISSN: 2252-8938, DOI: 10.11591/ijai.v15.i1.pp681-694

Comparison between ensemble and linear methods for website phishing detection

Saba Hussein Rashid, Saba Alaa Abdulwahhab, Farah Amer Abdulaziz
Department of Computer Science, College of Computer Science and Mathematics, Tikrit University, Tikrit, Iraq

Article Info

Article history:

Received Oct 11, 2024
Revised Jan 4, 2026
Accepted Jan 22, 2026

Keywords:

AutoGluon
Cloud ML
Ensemble model
Linear learner
Website phishing

ABSTRACT

In the current digitalized world, the notion of cybersecurity has become crucial in everyday life, and the issue of privacy takes the leading role in the technological agenda of the global community. One such social engineering attack that is currently prevalent is phishing, which is a common technique used by cybercriminals to intercept sensitive data. Despite the presence of certain limitations which can restrict its usefulness, machine learning (ML) has evolved into an interesting approach to identify phishing attacks. Cloud ML is an effective solution that uses cloud computing solutions to create, train, and deploy models that provide a faster and more accurate result as well as support large datasets. This paper compares the ensemble method of Amazon SageMaker's AutoML tool, AutoGluon, with the linear method of SageMaker's linear learner algorithm for website phishing detection. Key factors examined include training techniques, training time, batch transform time, endpoint prediction time, and model accuracy. The results demonstrate that while AutoGluon outperforms linear learner in terms of accuracy and prediction speed, linear learner is faster in training and batch transform processes.

Corresponding Author:

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

1. INTRODUCTION

With privacy as a major issue, cybersecurity is essential in today's environment [1]. Cybercriminals use phishing, a well-known social engineering attack, to obtain personal information [2], as attackers utilize e-commerce websites, banks, and credit card companies as their guises to trick people into sending sensitive data [3]. Phishing is defined as "a criminal mechanism using both technical subterfuge as well as social engineering for stealing personal identity data and financial account credentials of consumers" by Anti-Phishing Working Group (APWG), with 1,003,924 phishing attacks reported globally in the first quarter of 2025. Numerous techniques were put forth to identify and classify phishing attempts, such as blacklist method and heuristic analysis of the source code. Yet, the heuristic approach is difficult to use, classifier development takes time, and the blacklist has trouble identifying intermittent phishing sites [4]. Through examining website patterns and features, machine learning (ML) is good in identifying phishing attacks and helps distinguish between malicious and trustworthy websites. Compared to conventional rule-based algorithms, it could detect minor phishing indicators such as suspicious uniform resource locators (URLs) and domain age with more accuracy. These methods might be hampered by issues in handling unbalanced datasets, adjusting to novel attack strategies, and guaranteeing model generalization across many websites [5]. To develop, train, and deploy models for fast, accurate results and managing huge data, cloud ML makes

advantage of cloud computing services. This kind of service has numerous benefits including scalability, cost-effectiveness, and simplicity of use. It makes use of ensemble as well as linear ML techniques, which combine several models to create more reliable prediction model. The diversity of models is leveraged by ensemble techniques to improve error reduction as well as generalization over single models [6].

A cloud ML model called AutoML streamlines the process of choosing, creating, and improving ML models automatically, making it easier for non-experts to complete activities, such as feature engineering, data preprocessing, hyperparameter tuning, algorithm selection, and model evaluation, which in turn cuts down on the amount of time specialists need to construct high-quality models [7]. Amazon created the open-source AutoML toolkit AutoGluon, which is built on Python. Automatic feature data type analysis, low predictive attribute discarding, handling of missing values, raw data preprocessing, and data separation into training and validation sets are all done by it. With the use of repeated k-fold bagging for preventing overfitting, it trains a variety of models, including k-nearest neighbors (KNN), extreme gradient boosting (XGBoost), random forests (RF), categorical boosting (CatBoost), light gradient boosting machine (LightGBM), extremely randomized trees (ExtraTrees), and neural networks [8]. In cloud ML, linear techniques depict the relation between input parameters and predicted outputs using a linear function. Those methods can handle huge datasets as well as real-time applications with high-dimensional data since they presume the target parameter is a linear combination of input features [9]. With the use of a linear function for representing input parameters as well as target variables in datasets with linear relations, Amazon Web Services (AWS) SageMaker provides linear learner, a supervised ML technique appropriate for classification and regression applications [10].

AutoGluon and SageMaker's linear learner have been extensively studied in the past for a variety of ML applications, including time series forecasting, classification, and regression [11]–[21]. Yet, there is a clear lack of research on the application of such methods particularly in online website phishing detection. However, the algorithms incorporated within the AutoGluon architecture have been separately investigated for website phishing detection, establishing a robust baseline for justifying the current comparison [22]–[29]. In this study, AutoGluon is compared with linear learner, emphasizing fundamental elements such as training time, training methods, endpoint prediction time, and accuracy. Moreover, a large dataset of 11,430 preprocessed URL samples maintained on Amazon Simple Storage Service (S3) is used to make the trade-off between accuracy and the need for speedy, efficient processing, which allows for the selection of one of such models. Amazon CloudWatch is used to follow up the performance, while Amazon SageMaker is used to build, train, and deploy the model. The virtual resources required for model development are acquired through Amazon Elastic Compute Cloud (EC2) instances.

2. METHOD

This study exhibits a comparison of two AWS SageMaker frameworks designed to detect website phishing by employing two dissimilar yet complementary ML concepts within the SageMaker cloud platform. The process is comprised of the subsequent stages: in the first stage, the AWS SageMaker settings was configured and set according to the requirements of the proposed model, in the second stage, the dataset was downloaded from the source, manually configured, and uploaded to the S3 bucket for further processing, the third stage involved dataset preprocessing, which was conducted using a SageMaker Python Script specifically written for the dataset and integrated in the notebook, the results of this stage was saved into S3. In the fourth stage, the training of both AutoGluon and linear learner was conducted on the preprocessed training data where AutoGluon was trained using a SageMaker Python Script, called later in the notebook, while linear learner was directly integrated into the notebook, the training time was recorded using AWS CloudWatch and the results were saved in the S3 bucket, in the fifth stage, an offline batch transform prediction was conducted on the training results to measure the accuracy of the models and the time needed to obtain the results, the sixth stage included the deployment of the proposed models using two separate single-model endpoints to measure the time required for the prediction to be conducted in a real-time scenario. The concluding phase involves acquiring the endpoint prediction results and evaluating them against the training and batch transform outcomes.

Six evaluation metrics are used in this study: accuracy score, precision, F1-score, recall, and receiver operating characteristic (ROC). It aims to compare the performance of the bagging ensemble strategy of AWS SageMaker AutoGluon with the simple linear strategy of AWS linear learner to give some idea on the trade-off between the automation-based ensemble scheme and the traditional linear model in phishing detection setting. The analysis focuses on four fundamental aspects, including detection accuracy, training time, offline prediction time, and online inference speed. The systematic approach adopted in the paper is presented in a systematic way and is broken down into the following steps, as shown in Figure 1.

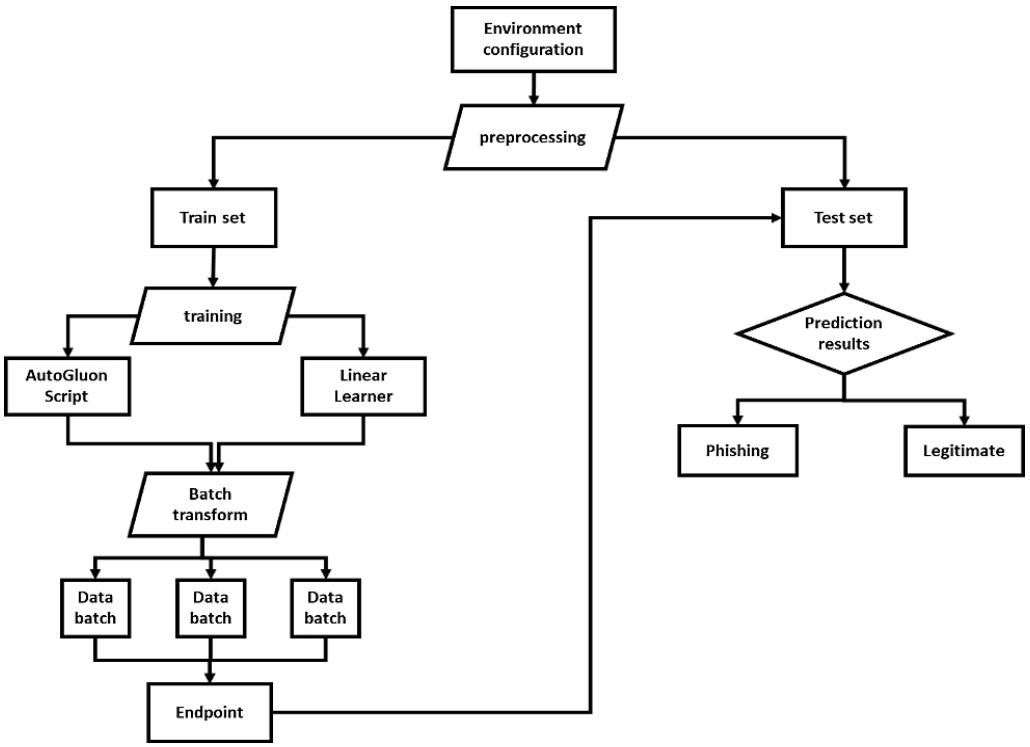

Figure 1. The general flowchart of the proposed methodology

2.1. Environment configuration

The experiment detailed in this paper was executed using a laptop with an Intel Core i5-10210U CPU at 1.60 GHz, 8 GB of RAM, and the Windows 11 Home Edition operating system. The AWS SageMaker Studio platform was employed to create and run Python codes in Jupyter Notebook environment. S3 facilitated the storage and retrieval of all experimental artifacts, while the "ml.m4.xlarge" virtual instance type from EC2 provided four virtual CPUs, 16 GB of virtual memory, and high-performance networking for a 64-bit operating system. AWS CloudWatch was utilized to monitor the performance of the SageMaker model and to log the processing, training, and prediction durations.

2.2. Dataset configuration

The "Web page phishing detection" dataset utilized in this work is available on Mendeley data [30]. It has 87 features and 11,430 URLs. To increase accuracy and gain a deeper understanding of the dataset's behavior, the 87 characteristics in the dataset are classified as seen in Figure 2: 56 features derived from URL syntax; 24 features resulting from the content of websites; and 7 features obtained from website services.

2.3. Dataset preprocessing

After analyzing the dataset for this work, it has been concluded that it is balanced, with 5,715 samples of legitimate URLs and 5,715 examples of phishing URLs, all of which had no missing values. There is no need for an earlier preprocessing step as AutoGluon could preprocess the data. However, the purpose of this step is to preprocess the data before training in linear learner. As a result, the preprocessing script, integrated as a SageMaker Studio Python Script, is reduced to the following five steps, which are shown in Figure 3. To reduce duplication and any detrimental effects on model performance, highly correlated features were first excluded from the feature selection process with the use of Pearson correlation. Three features were removed because their correlation rates were higher than 90%. Subsequently, to improve model performance through lowering noise, feature reduction was carried out by removing the nominal feature "URL," which represented the names of the dataset's URLs. Label encoding was used to transform categorical label values into numeric values, ensuring compatibility with ML techniques. The encoded "status" feature, which denotes dataset classifications, has "phishing" assigned to one (1) and "legitimate" set to zero (0). Additionally, feature scaling was carried out to reconfigure data within a given range, often 0 to 1, using Python's MinMaxScaler(). By taking this step, features with values between 2 and 100 are not allowed to dominate the other features. Following that, a 70:30 ratio was used to split the dataset

into training as well as testing sets, with 30% (3,425 samples) and 70% (8,001 samples) going to each set. Finally, S3 was used to upload both sets for additional processing.

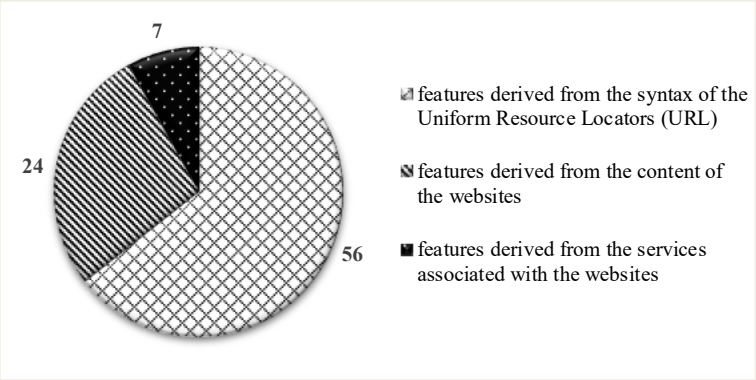

Figure 2. Analysis of the dataset used in the proposed methodology

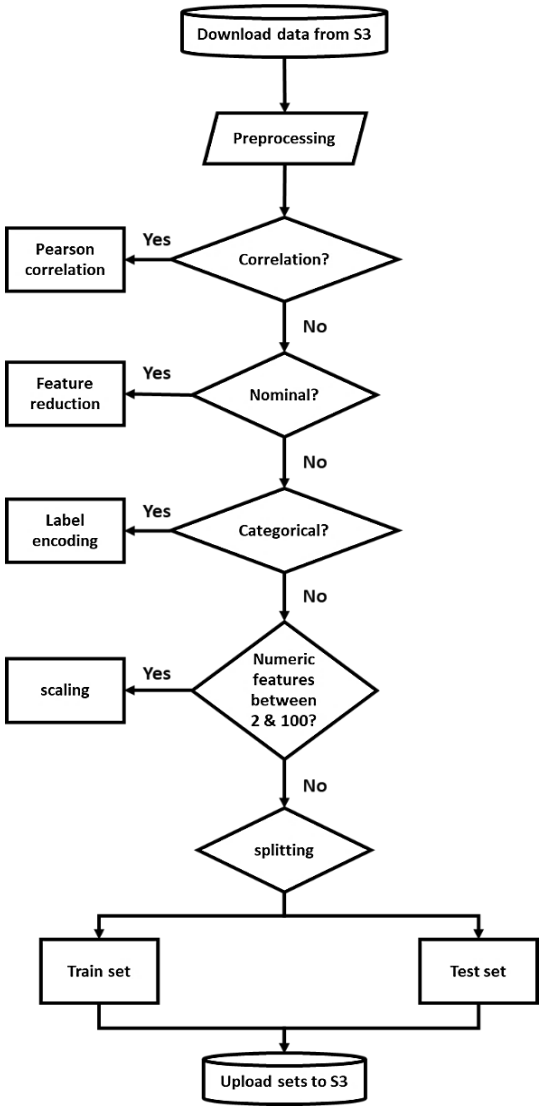

Figure 3. Steps of data preprocessing step of the proposed methodology

2.4. Training

To ascertain the optimal approach for managing large datasets in practical applications, the training stage contrasts SageMaker's ensemble method with the linear method. Two algorithms, AutoML AutoGluon and linear learner, are employed to train and assess the proposed model. Linear learner was invoked in the SageMaker notebook via specific code, as it is integrated within the SageMaker Studio Notebook architecture, while AutoGluon's code needed additional, separate Python Scripts to configure the training and inference hyperparameters.

2.4.1. AutoGluon

AutoGluon can be defined as an open-source AutoML framework developed by AWS which automates the construction of exact ML models with minimal manual effort [31]. The framework simplifies complicated procedures such as model selection, hyperparameter tuning, and feature engineering, making them available for experts and non-experts, improving accuracy by merging different models and methods and thus making it useful for tabular data applications such as classification [32]. AutoGluon operates within Python Scripts to generate reliable prediction models with minimal human input, as it uses bagging techniques to minimize variation and enhance stability, by training multiple models on random data, and make predictions based on majority votes [33]. AutoGluon also automates hyperparameter tuning by using random search algorithms, as shown in Table 1. After training, it analyzes models via cross-validation and aggregates the best performers for reliable predictions [34]. For this paper, AutoGluon was chosen due to its ability to automate major optimization processes by utilizing the combination distributed training among multiple diverse learners within its architecture enhances the robustness and generalization of the model, enabling rapid deployment, and making it preferable for real-world applications. AutoGluon utilizes the bagging approach to enhance the performance and strength of ML algorithms as depicted in following steps:

i) Base models (level 1)
   - AutoGluon trains a set of base models using algorithms like LightGBM, CatBoost, RF, ExtraTrees, XGBoost, KNN, and neural networks (FastAI).
   - Bagging is applied as a form of cross validation, where:
     a) The training data is divided into several bootstrapped samples (this research used five random subsets).
     b) Each base model is trained on a dissimilar set of data, enabling the model ensemble to learn using slightly dissimilar data distribution.
     c) The process generates several variations of the same base model, and each one of them reflects distinct feature associations and minimizes variance in the final prediction. Each trained model then generates predictions based on the patterns learned from its respective data subset, and their outputs are later combined to form a robust ensemble prediction.

ii) First-layer predictions
   - Once trained, the base models make predictions on a validation set of the training data.
   - These predictions are referred to as first-layer predictions and serve as input features for the second layer of models.

iii) Stacking (level 2)
   - Meta-models, or level 2 models, are created using the first-layer predictions as input.
   - In stacking, a meta-model is trained to combine the outputs from level 1 models.
   - This meta-model learns to weigh the predictions from different base models to minimize prediction errors.

iv) Weighted ensemble (level 2 model)
   - The "weighted ensemble" model represents the meta-model combining predictions from level 1 models with the use of a weighted average.
   - Weights are based upon performance of every one of the base models throughout the training—better-performing models have been assigned higher weight values, while poorer ones received lower weights.
   - This technique results in the improvement of the overall accuracy of prediction by giving a higher level of importance to the stronger models. Those steps have been illustrated in Figure 4.

Table 1. Hyperparameters of AutoGluon

Hyperparameter	Value	Purpose and effect
num_bag_folds	5	Specifies the number of folds for k-fold bagging to reduce overfitting.
num_bag_sets	1	indicates the number of complete bagging rounds.
num_stack_levels	0	Disables additional stacking layers to shorten training time.

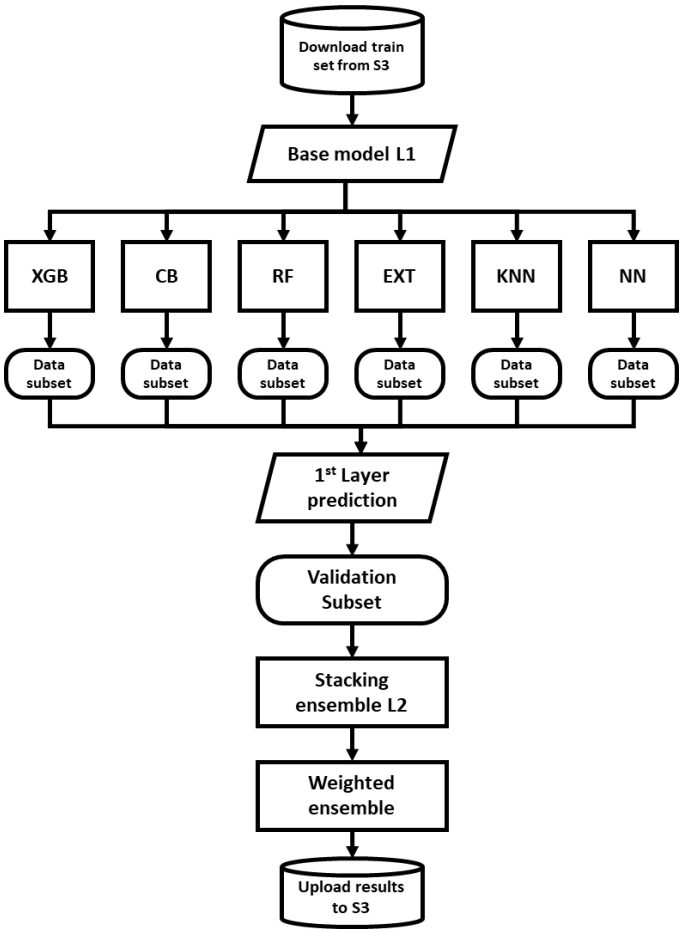

Figure 4. Steps of AutoGluon training

2.4.2. Linear learner

The technique of linear classification in Amazon SageMaker can be described as supervised learning approach that has been designed for the enhancement of binary classification tasks. It builds a linear decision boundary, which is exhibited as hyperplane, for the purpose of separating the data points of different classes by the assignment and optimization of the feature weights for class dissociation [35]. The model continuously adjusts those weight values throughout the training for minimizing errors and improving accuracy. This technique is appreciated since it is scalable, computationally efficient, and interpretable, which makes it suitable for large-scale applications [36]. According to Table 2, which presented the hyperparameters used for the algorithm, SageMaker's linear learner employs the stochastic gradient descent (SGD) for the purpose of handling large datasets. It utilizes automatic feature scaling for normalizing features of different scales and regularizing the techniques for overfitting prevention. SGD optimizes the model through the adjustment of weight values in error direction, guided by the gradient of loss function. The learning rate regulates the size of those adjustments, which leads to balancing the stability of the model and speed of convergence. The process of training continues to the point where minimal improvements are achieved, or a set number of iterations are completed [37]. The SageMaker integrated algorithm was chosen for comparison against AutoGluon in this paper because of its capability of handling large datasets effectively, offering a fast training and inference time, with easy interpretability, built-in regularization to reduce overfitting, and lower consumption of cloud resources, which gives it the benefit of fast deployment, making it suitable for time-sensitive phishing detection systems. A breakdown of the process that is involved in the training of a linear learner model in Amazon SageMaker has been depicted in Figure 5.

Objective function and regularization:

i) Objective function: linear learner uses an objective function to measure the error between predicted and actual labels. For binary classification, logistic regression is employed, estimating the probability that an input belongs to a specific class.

ii) Regularization:
- L1 regularization: encourages sparsity in the model by shrinking weights toward zero, which simplifies the model by ignoring less important features.
- L2 regularization: this is a term that is used to penalize big weights that give out smoother models. Both regularization methods are specified using the hyperparameter lambda that modulates the strength of the penalty in order to support the prevention of overfitting

iii) SGD optimization:
- Initialization: the model begins by initializing parameters, including weights for each feature and biases.
- Gradient calculation: the gradient of the objective function is calculated, representing the direction and magnitude of change in the error relative to the model's parameters.
- Mini-batch updates: instead of computing gradients on the entire dataset, the model updates its parameters using small subsets of the data (mini-batches), improving computational efficiency for large datasets.
- Weight and bias updates: weights and biases are iteratively adjusted based on the gradients to minimize the error. The learning rate controls how large each update step is. The process continues until either the weights converge, or the maximum number of iterations is reached.

Table 2. Hyperparameters of linear learner

Hyperparameter	Value	Purpose and effect
mini_batch_size	200	the number of samples processed before updating model parameters to lower computation time.
epochs	10	the number of complete passes through the training dataset to reduce overfitting.
regularization (L1, L2)	Auto	Applies both L1 and L2 regularization to reduce overfitting.

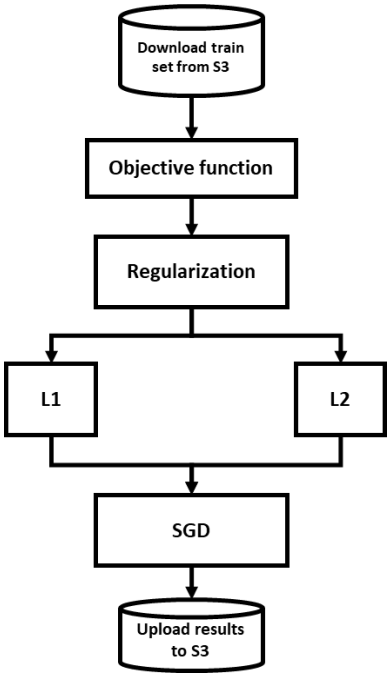

Figure 5. Steps of linear learner training

3. RESULTS AND DISCUSSION

Following training with both AutoGluon and linear learner, the training artifacts were uploaded to the S3 bucket and batch transform was then carried out on the results of both techniques for making offline predictions on the large dataset through dividing them into batches and uploading the results to S3 as well. Then, endpoint prediction was carried out by deploying endpoints as web service endpoints which could receive HTTP requests as well as responses with predictions in almost real time. The evaluation time and results for both methods were obtained using the test dataset.

3.1. Evaluation results

Following training with both AutoGluon and linear learner, the training artifacts were uploaded to the S3 bucket and batch transform was then carried out on the results of both techniques for making offline predictions on the large dataset through dividing them into batches and uploading the results to S3 as well. Then, endpoint prediction was carried out by deploying endpoints as web service endpoints which could receive HTTP requests as well as responses with predictions in almost real time. The evaluation time and results for both methods were obtained using the test dataset.

3.1.1. Evaluation results of AutoGluon

Since the ensemble architecture of AutoGluon generates predictions by aggregating the outcomes of many base learners, Therefore, Tables 3 and 4 present the leaderboard predictions and evaluation metrics, respectively, rather than a singular confusion matrix. The weighted ensemble model had the highest performance, with a test accuracy of 97%, as seen in Table 3, being the last level in the training order. This demonstrates that ensemble learning in the second prediction level, which combines the outputs from first prediction level models, operates by integrating many strategies. This final ensemble merges the predictions from many foundational models, which simultaneously reduces both bias and variance. In the first level, LightGBM, CatBoost, and ExtraTrees are instances of tree-based models that achieved second place with an accuracy of 96% after training in the third, fourth, and fifth order, respectively. Variety of URL, content, and service-based attributes. Their exceptional performance over the neural and distance-based models can be explained by the fact that they could capture complex relationships between features. The two next best models were RF and XGBoost with scores of 96% even though they had scored second and seventh in training order. This was because they used an ensemble-of-trees structure. The FastAI demonstrated 95% accuracy, placing it in the sixth order of training which is quite good but not superior compared to the other models due to the sensitivity of the features to scaling and the optimization of parameters. Finally, the first in the line of the training models was KNN, which did not fare well as the last, with an accuracy of 83%. Distance-based learning methodologies encounter difficulties with high-dimensional tabular data when several features exert minimal or correlated influence.

Table 3. Leaderboard prediction table of AutoGluon algorithms

No.	Algorithm	Accuracy score (%)	Stack level	Training order
1	Weighted ensemble	97	2	8
2	LightGBM	96	1	3
3	CatBoost	96	1	4
4	ExtraTrees	96	1	5
5	RF	96	1	2
6	XGBoost	96	1	7
7	FastAI	95	1	6
8	KNN	83	1	1

Table 4. The evaluation metrics of AutoGluon after batch transform

No.	Metric	Value (%)
1	Accuracy	97
2	Precision	97
3	Recall	96
4	F1-score	97
5	ROC	97

Table 4 summarizes the evaluation metrics obtained from the final AutoGluon ensemble on the test dataset. The weighted ensemble achieved the greatest performance for AutoGluon on the second prediction level, with 99% ROC, 97% F1-score, 97% accuracy, 97% precision, and 96% recall. This implies that there is a good ability to differentiate between phishing URLs and legitimate URLs. In addition, incorporating both false positives and false negatives into a single, balanced measurement. Furthermore, the excellent scores across all the measures indicate strong discrimination between phishing and legitimate labels and are indicative of the fact that the performance of AutoGluon's ensemble learning approach was exceptionally well on this dataset, as the usage of various algorithms is helpful in capturing diverse data parts, resulting in strong performance. AutoGluon's automated feature selection and hyperparameter tuning helped to improve the model's performance without overfitting, since diverse models in the ensemble may contribute to success. Tree-ensembled algorithms are well-suited to structured data, and their combination certainly improved overall performance, in addition to the stacking technique which minimized model variance and increased

generalization, allowing the model to perform well in phishing classes, alongside the balanced dataset and the effective regularization as they helped sustain high results in both accuracy and recall.

3.1.2. Evaluation results of linear learner

Based on the confusion matrix of linear learner displayed in Figure 6, it can be deduced that 1,616 samples were correctly classified as "legitimate," whereas 1,619 samples were appropriately classified as "phishing." 120 samples are wrongly classified as "phishing," whereas only 74 samples are incorrectly classified as "legitimate." Showing that linear learner has a low incidence of false positives, as it is able to accurately categorize a large proportion of instances that are positive. This indicates that linear learner is effective at managing large datasets and producing accurate predictions with a low amount of mislabeling. As shown in Table 5, linear learner obtained an accuracy of 94% with comparably high precision of 95%, F1-score of 94%, ROC of 94%, and a recall of 93%. The findings demonstrate that the model is effective at recognizing phishing websites while also avoiding false positives. The modest variation between accuracy and recall implies that the model slips up on the side of caution, focusing on lowering false positives while missing a few phishing websites. The high value of ROC is an indication of the fact that the model has great capacity for distinguishing between the phishing and the legal websites, which has been considered critical in this environment where the misclassifications might lead to severe consequences. Taking into consideration its high levels of accuracy, precision, and recall, the linear learner model would be highly dependable for the detection of real-world phishing, which makes it an invaluable tool for applications of cybersecurity.

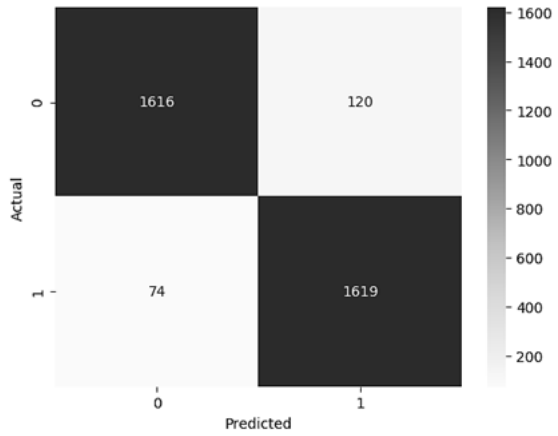

Figure 6. The confusion matrix of linear learner prediction after batch transform

Table 5. The evaluation metrics of linear learner after batch transform

No.	Metric	Value (%)
1	Accuracy	94
2	Precision	95
3	Recall	93
4	F1-score	94
5	ROC	94

3.1.3. Comparison of evaluation results

Figure 7 illustrates the comparison of the results of the evaluation of both AutoGluon and linear learner. The graph allows visualizing AutoGluon and linear learner in terms of the main assessment parameters. It was found that AutoGluon in all five measures, accuracy, precision, recall, F1-score, and ROC, has always reported higher values when compared to linear learner who has reported slightly lower but equally reliable results, which shows that the ensemble model has a strong balance between the detection of phishing websites and false alarms. This can be explained by the fact that AutoGluon has a mechanism of capturing non-linear, and complicated patterns in the data. The stacking technique further enhances model generalization and reduces variance, resulting in robust predictive behavior across both phishing and legitimate classes. Linear learner, by comparison, confirms that the model is competent at distinguishing between legitimate and malicious websites but remains limited by its linear decision boundary, which cannot fully represent the non-linear feature relationships typical of phishing data. The slightly higher precision relative to recall suggests that linear learner is more conservative in classification, producing fewer false

positives while missing a small portion of phishing samples. Despite these differences, both algorithms demonstrate strong capability, with AutoGluon favored for accuracy-critical applications and linear learner more suitable for time-sensitive or resource-limited environments.

Due to the lack of studies utilizing SageMaker's AutoGluon and linear learner for website phishing detection as mentioned before, Table 6 presents a comparison between the evaluation results of AutoGluon in this study and the results of the individual algorithms integrated within its architecture. It also includes the results of linear learner and a similar algorithm in construction and performance, all obtained from the related work mentioned earlier in the section 1. The results demonstrated that the proposed model outperformed in terms of accuracy, confirming the trend that model diversity improves phishing-detection robustness, and validating its efficiency for large website phishing use cases.

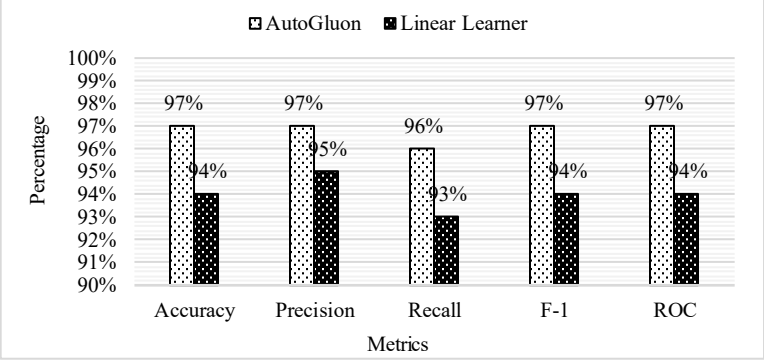

Figure 7. Comparison of evaluation results for both methods after batch transform

Table 6. Comparison of evaluation results with related works

Model	Dataset	Accuracy (%)	Environment	Reference
CatBoost	CIC-Bell-DNS	90	Google Colab	[22]
FastAI	ISCX-URL	96	Google Colab	[23]
LightGBM	UCI phishing domains dataset	95	Google Colab	[24]
RF	PhishTank	93	Google Colab	[25]
XGBoost	PhishOFE	94	Jupyter Notebook	[26]
ExtraTrees	UCI phishing domains dataset	96	Google Colab	[27]
Logistic regression	Combined of multiple Kaggle datasets	93	AWS SageMaker	[28]
KNN	Web page phishing detection	83	AWS SageMaker	[29]
AutoGluon	Web page phishing detection	97	AWS SageMaker	This study
Linear learner	Web page phishing detection	94	AWS SageMaker	This study

3.2. Time assessment

The paper estimated training, batch transform, and prediction times of each algorithm on an Amazon cloudwatch log, which provides valuable information on the effectiveness of particular algorithms. Figure 8 gives results of the time evaluation. AutoGluon used more memory and processing and memory resources than linear learner because it used the SageMaker ml.m4.xlarge instance, which used about 1.3 times more memory and required more time to train because it is a multi-model ensemble architecture and optimizes the resulting architecture. This extra computation cost, though, led to improved predictive accuracy and increased generalization, which shows that AutoGluon is suitable when large scale or accuracy-sensitive environments are needed. On the other hand, linear learner trained and inferred faster using less resources which validated. its effectiveness in time-critical or resource-limited applications. These results emphasize that while AutoGluon offers superior accuracy, linear learner remains more efficient and practical where computational constraints are a priority.

3.2.1 Training time

As Figure 8 shows, the training time of AutoGluon is 412.8 seconds, which is not surprising given the complexity of the AutoGluon framework. Several steps are included in this framework, including model selection, hyperparameter optimization, and the development of ensembles, which can explain the time cost. Model ensembling is one of the significant variables that contribute to the more time spent on training. AutoGluon usually trains a number of models and aggregates their predictions by the bagging method. This algorithm requires more computational resources and time than training a single model, which is not the case

of linear learner. Also, the diversity of the models in AutoGluon, where a single architecture and method of optimization used in a model are not similar, implies that the framework must repeat the process of using different models and fuse their results into a single ensemble. This is a multi-model and heterogeneous approach that results in increased training time but possibly provides a sturdier prediction outcome. The training time of linear learner was 312.6 seconds which is significantly less than that of AutoGluon. The causes of this outcome can be partly attributed to the ease of the algorithm since linear learner be based on a simple linear model to perform classification and therefore process is inherently faster than the more complex multi-model-based methods employed by AutoGluon. Also, linear learner applies (SGD) as the optimization method, which also benefits to a large extent to the decrease in training time. In addition, ensembling is not applied in linear learner. In comparison with AutoGluon, thereby lowering the amount of time and resources needed to train.

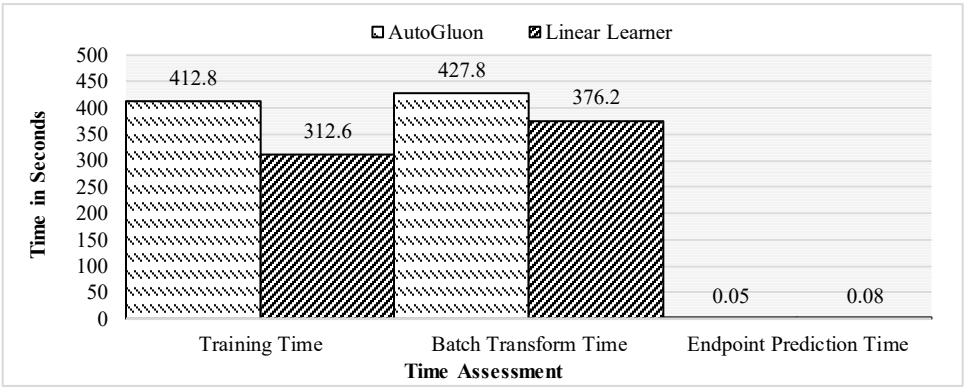

Figure 8. Results of time assessment for both methods

3.2.2. Batch transform time

To obtain the results of the abovementioned predictions, the batch transform time of AutoGluon was 427.8 seconds and that of linear learner was 367.2 seconds as shown in Figure 8. The much longer transform time taken by AutoGluon checks the complexity of all the models in the framework and the cost of computer calculation caused by it is ensemble strategy. Predictions that are made by various models of AutoGluon have to be pooled during batch transform. This process of aggregation involves voting on classification work as well as use of various computer resources and this overhead contributes to increasing processing time. Quite on the contrary, such complexity has enhanced predictive performance since it uses an ensemble approach. Moreover, linear learner has the advantage of a simpler and more efficient model structure, which has a reduced batch transform time. This simple method lowers the amount of computing load that leads to decreased processing time and linear learner is better applied in areas that demand speed. However, this advantage could also be achieved at the cost of forecasted accuracy as simple linear models may fail to identify as many patterns in the data as the complex ensemble methods.

3.2.3. Endpoint prediction time

Although endpoint prediction predictions are similar to the batch transform predictions, the amount of time taken to acquire the results differs. The prediction time of the auto-gluon on an endpoint was found 0.05 seconds compared to 0.08 seconds of linear learner as shown in Figure 8. It is also notable that the endpoint prediction time, which is used by linear learner is slower than the endpoint prediction time of AutoGluon since the former operates on a collection of models. The short prediction time proves the efficiency of the prediction process of AutoGluon that can develop quick results even with a number of models in its ensemble. One of the reasons behind this efficiency is the use of better assembly procedures. AutoGluon uses model distillation methods, that is, a smaller and faster model is trained to recreate the behavior of the complete ensemble, thus reducing the overhead of the prediction. In addition to this, AutoGluon also exploits pre-computation phases like the early preprocessing stage that was employed in this work that enables the system to reduce the time required in real-time inference. Moreover, AutoGluon has good aggregation algorithms to decrease the time involved in combining separate model predictions. It ensures that, although the ensemble methodology is employed, AutoGluon will still be significant in regard to performance, making it fit for real-time applications where prediction speed is critical. linear learner has a slightly slower endpoint prediction time than AutoGluon. Although still relatively efficient, this result validates that the linear learner's prediction process requires slightly longer time. Given the simplicity of the

linear model, this delay might be a reason for overhead from the endpoint architecture, where computing instances play an important role in affecting the steps of procedures. When comparing the two, AutoGluon's quicker endpoint prediction time demonstrates its effectiveness in real-time applications. This is especially useful for cases when instant responses are required. On the other hand, while linear learner's prediction time is slightly longer, it is still adequate for most real-time applications. The little difference in prediction time proposes that linear learner is still a practicable alternative, particularly given its faster training time and simpler model structure.

4. CONCLUSION

This paper included a comparison of AutoGluon and linear learner's advantages and trade-offs for the detection of web-site phishing. On the second prediction level, AutoGluon's weighted ensemble method had given a remarkable performance with 97% accuracy, 99% ROC, 97% F1-score, 97% precision, and 96% recall. In comparison, the performance of the linear learner has been lesser, with 94% ROC, 94% F1-score, 94% accuracy, 95% precision, and 93% recall. The higher prediction accuracy of AutoGluon comes at the expense of longer training and batch transform times, taking 412.8 seconds and 427.8 seconds respectively, unlike the faster training time of linear learner that equals 312.6 seconds, and batch transform time of 367.2 seconds. However, AutoGluon compensates for this with an endpoint prediction time of 0.05 seconds, where it outperformed linear learner's 0.08 seconds. In practice, the selection between these two models is dependent upon the operational circumstance. In large-scale or high-risk applications, such as enterprise-level phishing detection systems where reducing false negatives is essential, the superior accuracy of AutoGluon compensates the increased computing expense. Conversely, in real-time or resource-limited contexts—such as browser plug-ins or lightweight email filters—linear learner offers a more rapid and economical solution with satisfactory accuracy. This study illustrates that accuracy and speed have adverse impacts in the evaluated models, and the choice of method must correspond with the unique performance demands and computing limitations of the planned deployment context. Future work will concentrate on expanding this study by utilizing new and more varied phishing datasets to better assess the generalization proficiency of both AutoGluon and linear learner. Evaluating the models against novel and unobserved phishing techniques will yield a more definitive assessment of their resilience and flexibility in practical cybersecurity contexts. Furthermore, investigating the influence of sophisticated feature engineering methods on classification performance may enhance model precision and diminish false positives. Future research may integrate more advanced ML and deep learning (DL) frameworks into SageMaker to evaluate AutoGluon's ensemble methodology versus current architecture. Ultimately, implementing the models on advanced and high-performance AWS SageMaker and EC2 instances will facilitate a more profound analysis of computational scalability, memory usage, and processing duration, providing a thorough comprehension of the trade-off between accuracy and efficiency in extensive production environments.

ACKNOWLEDGMENTS

Authors thank AWS Academy in Tikrit University, Iraq for their help in developing this paper.

FUNDING INFORMATION

The authors state there is no funding involved.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Saba Hussein Rashid	✓	✓	✓			✓	✓		✓				✓	✓
Saba Alaa Abdulwahhab	✓			✓	✓			✓		✓	✓	✓		✓
Farah Amer Abdulaziz		✓		✓	✓					✓	✓	✓		✓

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

The authors state there is no conflict of interest.

DATA AVAILABILITY

The data that supports the findings of this study are openly available in Mendeley Data at http://doi.org/ 10.17632/c2gw7fy2j4.3, reference number [30].

REFERENCES

[1] G. A.-Aliyeva, J. Aliyev, and U. Sadigov, "Application of classification algorithms of ML in cybersecurity," *Procedia Computer Science*, vol. 215, pp. 909–919, 2022, doi: 10.1016/j.procs.2022.12.093.

[2] D. P. F. Möller, "Cyberattacker profiles, cyberattack models and scenarios, and cybersecurity ontology," in *Guide to cybersecurity in digital transformation: Trends, methods, technologies, applications and best practices*, 2023, pp. 181–229, doi: 10.1007/978-3-031-26845-8_4.

[3] A. Safi and S. Singh, "A systematic literature review on phishing website detection techniques," *Journal of King Saud University - Computer and Information Sciences*, vol. 35, no. 2, pp. 590–611, Feb. 2023, doi: 10.1016/j.jksuci.2023.01.004.

[4] M. Patil, N. Shivsharan, Y. Naik, H. Yeram, and A. Gawade, "Enhancing cybersecurity: a comprehensive analysis of ML techniques in detecting and preventing phishing attacks with a focus on xgboost algorithm," in *2024 International Conference on Intelligent Systems for Cybersecurity (ISCS)*, May 2024, pp. 1–6, doi: 10.1109/ISCS61804.2024.10581237.

[5] M. H. Alkawaz, S. J. Steven, and A. I. Hajamydeen, "Detecting phishing website using ML," in *2020 16th IEEE International Colloquium on Signal Processing and Its Applications (CSPA)*, Feb. 2020, pp. 111–114, doi: 10.1109/CSPA48992.2020.9068728.

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

[7] S. de Oliveira, O. Topsakal, and O. Toker, "Benchmarking automated ML (AutoML) frameworks for object detection," *Information*, vol. 15, no. 1, Jan. 2024, doi: 10.3390/info15010063.

[8] A. F. Glavan and V. Croitoru, "Autoencoders and automl for intrusion detection," in *2023 15th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, Jun. 2023, pp. 1–4, doi: 10.1109/ECAI58194.2023.10194229.

[9] AWS, "How linear learner works," Amazon Web Services. Accessed: Feb. 16, 2023. [Online]. Available: https://docs.aws.amazon.com/sagemaker/latest/dg/ll_how-it-works.html

[10] D. S. Watson, "The statistics of interpretable ML," in *2021 Yearbook of the Digital Ethics Lab*, 2022, pp. 133–155, doi: 10.1007/978-3-031-09846-8_10.

[11] F. V. S. Jothiraj and A. Mashhadi, "Personalized emotion detection using IoT and ML," *arXiv:2209.06464,* Sep. 2022.

[12] S. B. Ahamed, A. Anoop, R. A. Nazeema, and M. A. Khan, "A hybrid algorithm for detection of cloud-based email phishing attack," in *International Workshop on Cultural Perspectives of Human-Centered and Technological Innovations*, 2025, pp. 177–187, doi: 10.1007/978-3-031-77012-8_13.

[13] L. Kangethe, H. Wimmer, and C. Rebman, "Network intrusion detection system with ML intrusion detection system with ML as a service," *Journal of Information Systems Applied Research*, vol. 17, no. 3, pp. 4–15, Dec. 2024, doi: 10.62273/EWQL5023.

[14] S. Mishra and A. Tiwari, "Medical insurance cost prediction using aws sage-maker," in *2024 International Conference on Augmented Reality, Intelligent Systems, and Industrial Automation (ARIIA)*, Dec. 2024, pp. 1–6, doi: 10.1109/ARIIA63345.2024.11051598.

[15] H. A. Shoaib *et al.*, "An enhanced deep learning approach to potential purchaser prediction: autogluon ensembles for cross-industry profit maximization," *IEEE Open Journal of the Computer Society*, vol. 6, pp. 468–479, 2025, doi: 10.1109/OJCS.2025.3552376.

[16] C.-M. Rosca, A. Stancu, and C. Popescu, "ML models for sql injection detection," *Electronics*, vol. 14, no. 17, Aug. 2025, doi: 10.3390/electronics14173420.

[17] M. Yang, F. Li, and W. Qiu, "Integrating autogluon for real-time monitoring and classification of dental equipment performance," *IEEE Access*, vol. 13, pp. 2844–2854, 2025, doi: 10.1109/ACCESS.2024.3523519.

[18] A. V. Özcan *et al.*, "Forecasting day-ahead electricity prices for the electricity market with dynamic time period," *Energy*, vol. 338, Nov. 2025, doi: 10.1016/j.energy.2025.138766.

[19] M. Babaeianjelodar *et al.*, "Interpretable and high-performance hate and offensive speech detection," in *International Conference on Human-Computer Interaction*, 2022, pp. 233–244, doi: 10.1007/978-3-031-21707-4_18.

[20] S. Lenkala, R. Marry, S. R. Gopovaram, T. C. Akinci, and O. Topsakal, "Comparison of automated ML (AutoML) tools for epileptic seizure detection using electroencephalograms (EEG)," *Computers*, vol. 12, no. 10, Sep. 2023, doi: 10.3390/computers12100197.

[21] V. Aishwarya, "Binary classification model for fraudulent credit card transactions," *IOSR Journal of Computer Engineering*, vol. 22, no. 3, pp. 38–45, 2020, doi: 10.9790/0661-2203023845.

[22] N. N. Sakhare, J. L. Bangare, R. G. Purandare, D. S. Wankhede, and P. Dehankar, "Phishing website detection using advanced ML techniques," *International Journal of Intelligent Systems and Applications in Engineering*, vol. 12, no. 12s, pp. 329–346, 2024.

[23] C. Johnson, B. Khadka, R. B. Basnet, and T. Doleck, "Towards detecting and classifying malicious urls using deep learning," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, vol. 11, no. 4, pp. 31–48, 2020, doi: 10.22667/JOWUA.2020.12.31.031.

[24] A. Odeh, Q. A. Al-Haija, A. Aref, and A. A. Taleb, "Comparative study of CatBoost, XGBoost, and LightGBM for enhanced URL phishing detection: a performance assessment," *Journal of Internet Services and Information Security*, vol. 13, no. 4, pp. 1–11, Dec. 2023, doi: 10.58346/JISIS.2023.I4.001.

[25] B.-C. Mocanu *et al.*, "NextEDR-next generation agent-based edr systems for cybersecurity threats," in *2024 32nd Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, Mar. 2024, pp. 183–190, doi: 10.1109/PDP62718.2024.00033.

[26] Y. A. Kustiawan and K. I. Ghauth, "Evaluating the impact of feature engineering in phishing url detection: a comparative study of URL, html, and derived features," *IEEE Access*, vol. 13, pp. 126756–126768, 2025, doi: 10.1109/ACCESS.2025.3579223.

[27] G. Dharmaraju, T. N. Kumar, P. P. Mohan, R. R. Pbv, and A. Lakshmanarao, "Phishing website detection through ensemble ML techniques," in *2024 2nd International Conference on Computer, Communication and Control (IC4)*, Feb. 2024, pp. 1–5, doi: 10.1109/IC457434.2024.10486530.
[28] M. H. A. Suras, R. G. Utomo, and M. Irsan, "A comparison of ML methods for phishing detection: XGBoost and logistic regression," in *2025 International Conference on Data Science and Its Applications (ICoDSA)*, Jul. 2025, pp. 553–558, doi: 10.1109/ICoDSA67155.2025.11157169.
[29] S. Singh and S. Kaur, *Latest trends in engineering and technology*. London: CRC Press, 2024, doi: 10.1201/9781032665443.
[30] A. Hannousse and S. Yahiouche, "Web page phishing detection," *Mendeley Data*, V3, doi: 10.17632/c2gw7fy2j4.3.
[31] N. Erickson *et al.*, "AutoGluon-tabular: robust and accurate automl for structured data," *7th ICML Workshop on Automated ML*, 2020, pp. 1-16.
[32] P. Grover *et al.*, "Fraud dataset benchmark and applications," *arXiv:2208.14417,* Sep. 2023.
[33] R. Purwanto, A. Pal, A. Blair, and S. Jha, "Man versus machine: AutoML and human experts' role in phishing detection," *arXiv:2108.12193,* Aug. 2021.
[34] H. Bouijij, A. Berqia, and H. S.-Hassan, "Phishing URL classification using Extra-Tree and DNN," in *2022 10th International Symposium on Digital Forensics and Security (ISDFS)*, Jun. 2022, pp. 1–6, doi: 10.1109/ISDFS55398.2022.9800795.
[35] H. Singh, *Practical ML with aws*. Berkeley, CA: Apress, 2021, doi: 10.1007/978-1-4842-6222-1.
[36] X. Liang, S. Li, and J. Fei, "Adaptive fuzzy global fast terminal sliding mode control for microgyroscope system," *IEEE Access*, vol. 4, pp. 9681–9688, 2016, doi: 10.1109/ACCESS.2016.2636901.
[37] V. Perrone *et al.*, "Amazon sagemaker automatic model tuning: scalable gradient-free optimization," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, Aug. 2021, pp. 3463–3471, doi: 10.1145/3447548.3467098.

BIOGRAPHIES OF AUTHORS

Saba Hussein Rashid holds a master's degree in Computer Science from the Department of Computer Science, College of Computer Science and Mathematics, Tikrit University, Iraq, in 2023. She received her B.Sc. degree in Computer Science from Tikrit University, Iraq in 2010. Her research interests involve cloud ML and cybersecurity. She can be contacted at email: sabahussein88@tu.edu.iq.

Saba Alaa Abdulwahhab received master's degree from Department of Computer Science, College of Computer Science and Mathematics, Tikrit University, Iraq in 2022. She received her B.Sc. degree in Computer Science from Tikrit University, Iraq in 2010. Her research interests are cryptography, quantum cryptography, post quantum cryptography and information security. She can be contacted at email: saba.programmer12@tu.edu.iq.

Farah Amer Abdulaziz received master's degree from Department of Computer Science, College of Computer Science and Mathematics, Tikrit University, Iraq in 2022. She received her B.Sc. degree in Computer Science from Tikrit University, Iraq in 2009. Her research interests are ML, deep learning, and internet of things. She can be contacted at email: farah.amer@tu.edu.iq.