International Journal of Environmental Sciences
ISSN: 2229-7359
Vol. 11 No. 21s, 2025
https://theaspd.com/index.php

Artificial Intelligence-Based Malicious Accounts Detection Model Using Machine Learning

Ms. Pankaj Sharma[1], Dr. Tapsi Nagpal[2]
[1]Research Scholar, Department of Computer Science & Engineering, Lingaya's Vidyapeeth, Faridabad (Haryana) – 121002
[2]Associate Professor, Department of Computer Science & Engineering, Lingaya's Vidyapeeth, Faridabad (Haryana)- 121002

Abstract
One of the most important lines of defence against phishing attacks is the detection of fake URLs, which, although seeming to be from legitimate websites, really connect to dangerous websites. In light of the fact that Internet of Things devices often have Internet connections and are thus vulnerable to phishing attacks, this is of utmost significance at the present time. In this article, a summary of the most essential ways for accurately identifying counterfeit URLs is presented. These approaches include the most generally used DL (Deep Learning) and ML (Machine Learning) techniques, as well as the proof-of-concept use of quantum machine learning categorisation models. The first and most important part of the data preparation process, we concentrate on contrasting a number of traditional machine learning models. We put these models to the test on a variety of datasets, and we get encouraging results with true positive rates that are more than 90%. After a brief introduction to the fundamentals of the first technique, the study then moves on to investigate the most recent advancements in quantum machine learning and the promise that it has for spotting potentially hazardous URLs. This study fills a vacuum in the research by identifying malicious URLs and other cybersecurity concerns. Additionally, it gives fresh insights into bringing these two ideas together via cybersecurity algorithms. Quantum machine learning has been mentioned seldom in the literature; this work fills that gap. The findings obtained from the examination of a large number of algorithms are encouraging, and they open the way for more research into the possible uses of quantum computing in the field of cybersecurity.

Keywords: *Website that is harmful, Machine learning, Matrix of confusion, a decision tree, logistic regression, ROC curve, A neural network, and support vector machine The future of computing: quantum.*

INTRODUCTION

A numerical address that is specifically assigned to each web page is referred to as the Uniform Resource Locator, or URL. One of the most popular strategies used in cyberattacks is the utilisation of fabricated URLs. These connections give the impression of leading to legitimate websites, but in reality, they redirect to malicious websites. It is on these websites that the attackers gather critical information such as login passwords, bank account data, and other information. Given the frequency of phishing assaults in the current day, the detection of bogus URLs has become a significant issue for experts working in the field of information security. The purpose of these assaults is to deceive users into divulging important information by giving them the impression that they are interacting with a reputable company or organisation. In accordance with the most current annual report from the European Union Cybersecurity Agency (ENISA), [1] phishing has exceeded all other types of initial attack vector.

With the use of sophisticated tools, fraudsters are constructing malicious URLs that are designed to seem like legitimate ones. This makes it more difficult to identify them. Phishers are always coming up with new tactics, such as URL spoofing, which includes merging legitimate and malicious URLs, altering redirection, or disguising malware with graphics. This is the case despite the fact that people's awareness of phishing has increased over the years [2]. However, thanks to advancements in artificial intelligence (AI) and machine learning (ML), we are now able to construct models that are capable of analysing vast amounts of data, recognising patterns, and making predictions—all with the intention of identifying these fake URLs.

Particularly in regard to the ground-breaking For the purpose of tackling this challenge, this study intends to examine the implementation of Quantum Machine Learning (QML) in conjunction with a variety of machine learning algorithms for the early detection of counterfeit URLs. This study investigates the possibility of using QML for the detection of phishing URLs and compares the results with those obtained from conventional deep learning and machine learning techniques. The purpose of this research is to enhance the methods that are currently being used to eradicate phishing. Due to the

fact that it is a relatively new field of research, one of the first things that has to be done is to determine which algorithms are most effective when used in conjunction with a QML model under certain quantum settings. Following that, we will be able to evaluate the benefits and drawbacks of this technique to determine whether or not it can assist with cybersecurity in general and the identification of dangerous URLs in particular.

Making use of ML techniques for the early identification of fake URLs Recent years have seen a number of studies that have focused on it, each of which has taken a distinct viewpoint on the subject.

When it comes to their research dataset, the authors make use of 121 separate sets of URLs that were gathered on various days [3]. The more than 2.4 million URLs that make up this dataset, each of which has more than 3.2 million attributes, are sorted through using a broad variety of machine learning algorithms. This dataset is accessible to the general public.

Leveraging lexical features, URL length, and primary domain length is something that the authors of [4] recommend doing in addition to employing a URL blacklist. In addition, facts such as the creation date, name servers, and Whois server are included in the host-based features assortment.

In order to extract a wide variety of semantic information, the authors of the work that was quoted [5] make use of CNNs that have been trained on the letters and words that make up the URL string.

The article [6] provides a thorough literature review that focusses on the primary approaches that are dependent on machine learning models. This analysis is presented in order to discover hazardous URLs.

Identifying potentially hazardous URLs is accomplished by the authors of [7] by the use of decision trees, logistic regression, support vector machines (SVMs), and a majority vote, among other techniques.

With the use of a dataset that is comprised of genuine URLs that were selected from the site lists of the top 5,000 websites in the world, the research [8] makes use of a variety of techniques, including decision trees, random forests, support vector machines (SVMs), and others, as well as Naive Bayes and convolutional neural networks (CNNs).

Methods such as decision trees, K-NN, and random forests are used in the study [9] was out on a dataset that was obtained from a specialised machine learning repository.

In the research [10], machine learning classifiers are applied to a dataset that includes both potentially harmful and safe websites. These classifiers make use of characteristics such as the quantity of special characters or the letters that make up a URL. Random forest, K-NN, J48 decision tree, and BayesNet are some of the algorithms that are available for download.

[11] makes use of logistic regression, decision trees, random forests, K-NN and Naive Bayes, support vector machine, and Kaggle datasets. URL labels and text tokenisation are two of the qualities that are used in this process.

In the research [12], which makes use of datasets from Open-Phish, Phishtank, Zone-H, and WEBSPAM-UK2007, several methods are used. These methods include logistic regression, support vector machines, J48 decision tree, & Naive Bayes. Other factors that are taken into consideration include the presence of an IP address, the length of the URL, the presence of httpSecure, the number of digits, and an irregular URL.

The dataset from the Machine Learning Lab is used in combination with logistic regression, J48 decision trees, Naive Bayes, and SVM algorithms in [13].

The variables that are utilised include ContentLength, compromissionType, serverType, poweredBy, and contentType. In their research, the authors took into consideration a wide range of features that are shared by all individuals, including the following: The length of the URL, the presence or omission of the IP address, the number of dots per line, the number of sensitive words, and the @ symbol are all aspects that were taken into consideration.

Through the process of converting every Android application into a greyscale image, the research [15] makes use of a collection that contains both legitimate and malicious applications that are accessible in the wild. In addition, they make use of a number of CNN models, one of which is an integrated quantum convolutional neural network (CNN), as well as a quantum neural network.

In addition, the authors of [16] examine the outcomes of applying QML to an incursion dataset and compare it to the outcomes of utilising conventional Support Vector Machine (SVM) and quantum SVM, conventional CNN and quantum CNN, and conventional SVM. The CNN.

Among the few articles that discuss using a neural network classifier based on quantum mechanics to identify harmful online requests, this one is another gem [17]. The primary features of this study are compared schematically with those of some of the publications in Table 1

FROM: USING MACHINE LEARNING DETECTION OF MALICIOUS URLS

Table 1 Comparative analysis

References	ML fund	Multiple ML	Different datasets	ML/QML	QML parameterizations
[3]	Yes	Yes	No	No	No
[5]	No	No (Only one)	No	No	No
[7]	Yes	Yes	No	No	No
[8]	Yes	Yes	No	No	No
[9]	No	Yes	No	No	No
[10]	Yes	Yes	Yes	No	No
[11]	Yes	Yes	No	No	No
[12]	Yes	Yes	No	No	No
[13]	No	Yes	No	No	No
[14]	Yes	No	No	No	No
[16]	No	Yes	No	Yes	Yes
[17]	No	No	No	No	Yes
This paper	Yes	Yes	Yes	Yes	Yes

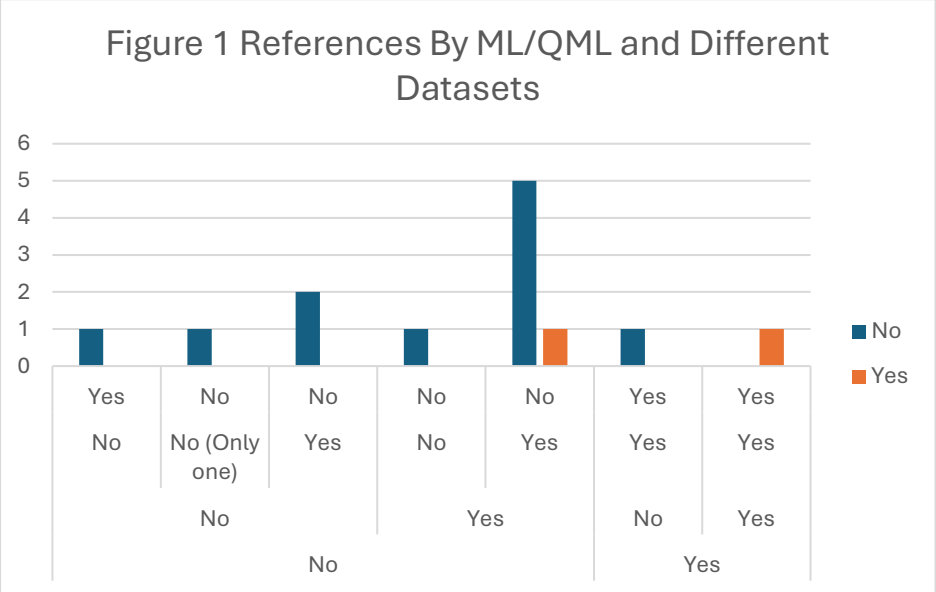

Figure 1 References By ML/QML and Different Datasets

One of the key innovations of this work is that it compares the results obtained with various classic ML techniques to those of other studies that have investigated the possibility of using QML to identify fake URLs quickly.

Interestingly, none of the works incorporate this potential.

The following is the organization of this work. While Section 3 delves into the preliminary work on ML, Section 2 provides a high-level overview of the whole idea. While Section 5 addresses some concerns with ML algorithms, Section 4 focuses on measurements. Section 6 discusses the dataset that was utilized, whereas Section 7 discusses how the data was processed. The suggested models and their application with traditional ML are described in Section 8, along with the findings and a short assessment. In Section 9, the use of QML has been presented to tackle the phishing issue. This includes preparing the dataset for quantum algorithms, applying the algorithms, and doing a quick assessment. Finally, Section concludes with some findings and suggestions for further research 10, which finishes the study.

1. Proposed Model

2. Iteratively carrying out the following stages, this study's development approach has been based on standard Machine Learning and on Quantum Machine Learning. The Deming Cycle, often known as PDCA (Plan, Do, Check), is the foundation of this concept, Act) cycle, as illustrated in Figure 1:

Phase 1: Analyzing both established methods and potential new avenues of inquiry, exploratory research examines several perspectives on the issue at hand.

Phase 2: The recognition of basic ideas. Theoretical and practical underpinnings of Machine Learning compiled and studied, including.

- Algorithms
- Metrics
- Potential problems

Phase 3: Collect datasets. Most of them are rather outdated, which is a major issue since current data is essential in the cybersecurity industry.

Phase 4: Data preparation. Changing to fit the specifications of the reference dataset. An additional layer of difficulty was introduced here by the idiosyncrasies of adjusting to quantum computing.

Phase 5: Trying new things. Everything from writing code to testing it to running algorithms via trial-and-error falls under this category. Because of how crucial the time needed to get useful data is in the context of quantum computing, execution on Silicon chips manufactured by Apple was chosen.

Phase 6: Results assessment. Here, the numerical findings have been placed in context with respect to the subject being studied, using the metrics that were utilized.

Phase 7: Final thoughts and new ideas for future projects. What happens after every run informed the establishment of new goals and the subsequent implementation of new work cycles.

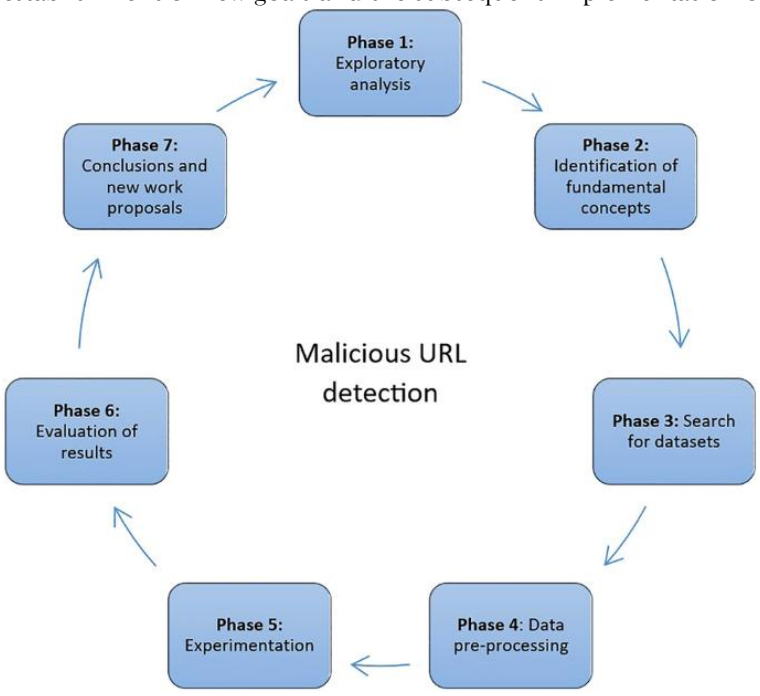

Figure 2 Workplan

3. Preliminaries of Machine Learning

3.1Logistic regression

Regression modelling using logistic variables allows one to examine the relationship between independent variables and dependent variables that is categorical, often binary, using statistical data analysis [18].

The technique is based on determining the likelihood that a given observation falls into a certain category. To forecast the likelihood that the independent variables have a value of 1 or 0 for a binary dependent variable, it employs a logistic function. This method takes the linear regression result and converts it to a probability of belonging to a certain category, which may be expressed as a number between zero and one.

When it comes to machine learning's binary classification tasks, logistic regression is a popular supervised learning approach. By using a logistic function to foretell the possibility of an event including a

dependent variable with a categorical nature, it foretells the chance of a binary result. So, it's possible to fit data to a logistic curve and estimate the likelihood that an input belongs to a specific category.

The following phrase [19] yields the probability of a class, and it forms the basis of an analysis using logistic regression. x

returns the probability of a class x.

$$g(z)=11+e-z$$

(1)

were

$$z=\theta T\cdot X$$

(2)

θT is the data-dependent parameter estimation vector, and X is the independent variable vector.

3.2Decision Tree

An example of a graphical representation that shows all the potential consequences of a set of connected actions is a decision tree [20]. The basic structure of a decision tree is like an inverted tree: the core nodes stand for features, the branches for decision rules, and the leaf nodes for the ultimate choice or conclusion.

To make decisions, decision trees iteratively divide data into subsets defined by the most important characteristics. This data splitting procedure keeps on until all of the subsets include just one kind of information or until the subsets are too tiny based on predetermined standards.

Classification and regression make heavy use of these supervised learning methods because of their flexibility, interpretability, and capacity to grasp intricate linkages. The fact that these models can provide feature-based decision limits makes them useful for a wide range of applications.

3.3Support Vector Machine

Although it is more often used for classification and regression are two techniques of supervised machine learning techniques known as Support Vector Machines (SVMs). Finding the sweet spot where data points intersect is essential when trying to make a choice belonging to various classes, SVM works well.

Finding the best hyperplane that maximizes the margin is the main objective of support vector machines (SVMs) in classification [21]. Margin is the distance between the hyperplane and the closest data points of various classes. The data is efficiently divided into several classes by this hyperplane.

If the distribution of the observations allows for a complete linear separation into two classes, indicated as +1 and −1, then the the dataset can be easily partitioned into smaller sets [22]. There is no hyperplane of separation since they cannot be properly separated linearly in most circumstances. Kernels are created to address this issue. The basic concept for linearly inseparable data, one kernel approach is to create non-linear combinations of the original features, which are then transferred to a higher dimensional space and turned into linearly separable data, using a mapping function ϕ. In support vector machines, some of the most popular kernels include.

- Linear:

$$K(x(i),x(j))=x(i)\cdot x(j)$$

(3)

- Radial Basis Function (RBF) or Gaussian Kernel:

$$K(x(i), x(j)) =\exp(-\|x(i)-x(j)\|22\cdot\theta 2)$$

(4)

- Polynomial of degree k:

$$K(x(i),x(j))=(x(i)\cdot x(j))k$$

(5)

- Polynomial of degree up to k: For some $c>0$

$$K(x(i),x(j))=(c+x(i)\cdot x(j))k$$

(6)

- Sigmoid:

$$K(x(i), x(j)) =\tanh(ax(i)\cdot x(j)+b)$$

(7)

3.4Neural Network

The intricate web of connections in the human brain serves as inspiration for a kind of computer model known as a neural network. Useful for ML jobs like pattern recognition, classification, and regression, it is a potent method.

There are essentially three distinct levels that make up a neural network all the way to the output layer after passing via the input and hidden layers. With each layer comes linked neurons. After taking in data, each neuron analyzes it using an activation function and sends out a signal to the layer below it.

The foundation for more advanced network topologies may be found in the early models of neural networks, such as perceptrons, adalines, and logistic neurons.

The McCulloch-Pitts neuron serves as the foundation for the perceptron neural network [23]. The perceptron is the first mathematical model to simulate It shares the same McCulloch-Pitts structure as a genuine neural network and is the first mathematical model to mimic its electrical activity as real networks, with the sole modification being the addition of weights to the input variables. Refer to [24] for more details on the perceptron.

While both the Adaline and logistic neurons have similarities with perceptrons, their activation functions are different. In contrast to the logistic neuron's usage.

Out of all the basic kinds of neural networks, pre-fed or feed-forward ones stand out [25]. Their buried layers and non-cyclical neuronal connections are their defining features. Multiple neurons with this structure are present in every buried layer. There is no communication between neurons in the same layer; instead, they all use the same activation function. The converse is true for dense networks, which are formed when two successive layers allow all neurons in one layer to communicate with all neurons in the next.

3.5Quantum Computing

Here we will go over the fundamentals of quantum computing [26].

The basic building block of quantum information in quantum computing is a qubit, which stands for quantum bit. A qubit, according to the laws of quantum superposition, may exist in more than two states at once, in contrast to classical bits, which can only exist in two states (0 and 1). The quantum bit's condition on the computational foundation

$|0\rangle,|1\rangle$ is defined as:

$|\psi\rangle=\alpha|0\rangle+\beta|1\rangle$

(8)

where $|\alpha|2+|\beta|2=1$, and $\alpha, \beta \in C$ are known as the amplitudes of states.

A set of qubits $\psi 1, \cdots, \psi n$ used for calculation is called a quantum register. A unitary matrix represents a quantum gate, an operator acting on qubits. The NOT gate and the CNOT gate are two examples of quantum gates. A quantum circuit consists of a quantum register and a series of quantum gates $P1, \cdots, Pn$.

3.5.1 Variational Quantum Classifier

In a supervised learning setting, the Variational Quantum Classifier (VQC) model is suggested to carry out binary classification of classical data. This model is a unique variation of the neural network classifier that incorporates a quantum circuit and a function created using its output.

The VQC follows the same structure as traditional supervised ML algorithms: it starts with training data points that already have labels applied, and then moves on to testing data points that do not, to classify them.

In particular, the VQC follows the principles of Sampler, a kind of neural network that takes as input data and/or weights a parameterized quantum circuit. As a classifier's output, the measured expectation value is used in the VQC. The Sampler primitive estimates quasi-probabilities, and this function converts them into predictions for various classes.

4. Metrics

Both the confusion matrix and the ROC curve may be used to evaluate various ML methods.

4.1 Confusion Matrix

Machine learning researchers often use a table called the confusion matrix to assess the performance of various categorization algorithms. It is a matrix that compares real classes with anticipated ones, allowing one to see how well an algorithm is doing.

The following information can be extracted from this structure:

Precision=TPTP+FP

(9)
Sensitivity=TPTP+FN
(10)
Specificity=TNTN+FP
(11)
Accuracy=TP+ TNTP+FN+TN+FP
(12)
were
TP are True Positives
FN are False Negatives
FP are False Positives
TN are True Negatives
Precision is the percentage of well-classified cases within a class.
The proportion of well-classified positive instances, also known as sensitivity, recall, or true positive rate, demonstrates an algorithm's capacity to forecast a positive outcome when the actual outcome is positive.
The proportion of correctly categorized negative situations is known as specificity.
To get the accuracy, divide the total data by the fraction of well-classified data. In other words, it's the ratio of accurate forecasts to the total.

4.2 ROC Curve

One way to see how well a binary classification model does with varying thresholds is to look at its Receiver Operating Characteristic (ROC) curve [27]. A comparison of the True Positive and False Positive rates is shown by the ROC curve. Hence, the sensitivity is shown versus 1-Specificity on the ROC curve. Area Under the Curve (AUC) is another metric that may be used to compare classifiers. When a classifier achieves an area under the ROC curve of 1, it is said to be flawless. You may see a ROC curve in action in Figure 3.

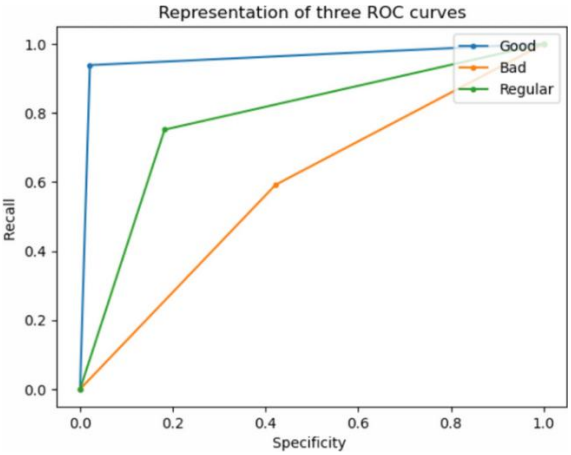

Figure 3 ROC Curve

5. Issues with Machine Learning Algorithms

Overfitting and underfitting are two common issues that may arise when using Machine Learning algorithms [28].
If a machine learning model memorizes all of the training data points—including the patterns and noise—too well, it will overfit the data. Consequently, the model excels at analyzing the training data but struggles when presented with novel or unfamiliar data.
When an ML model is too simplistic and fails to grasp the hidden patterns in the training data, this phenomenon is known as underfitting. Its low performance on training and test data is a direct consequence of its inability to match the training data.
The following remedies may be taken to fix the overfitting issue:

- Machine Learning techniques that are less sophisticated should be used with fewer model parameters.
- Use dimensionality reduction methods, such as principal component analysis (PCA), to pick out useful characteristics and remove superfluous or unnecessary ones [29].
- To get a better read on how well a model is doing, use cross-validation methods like k-fold cross-validation, and then tweak the hyperparameters according to the findings [28].

- To avoid the model fitting noise in the data, use regularization approaches such as L1 (Lasso) and L2 (Ridge) regularization to penalize big parameter values [30].
- Early stopping is a technique used in deep learning to cease training when validation performance begins to decline [31].
- The underfitting problem may be addressed by using the following measures:
- Use deeper neural networks or more advanced Machine Learning algorithms; use more complicated models; they may better capture the underlying patterns.
- Define the data more accurately by making new useful features and try out different ways of transforming current characteristics.
- To make the model work better, play about with the hyperparameters and experiment with various learning depths, regularization strengths, learning rates, etc. [32].

6. Used Datasets

This dataset was this dataset was obtained from https://machinelearning.inginf.units.it/data-and-tools/hidden-fake-urls-dataset to do research on the use of ML for the identification of URLs that are fraudulent. The availability and labelling of this dataset, together with its vast data set, made it an ideal choice for studying several ML algorithms, including QML, due to its high performance. It has also been utilized in other studies, so the findings may be compared. What follows is a comparison of one study to another that also made use of the same dataset.
Specifically, the following are contained inside this dataset details:

- "url" stands for the current URL.
- The field "compromission Type" shows whether the website is normal, hacked, or defaced.
- The URL's legitimacy is indicated by the dependent variable is Hidden Fraudulent.
- To get the content length, we sent an HTTP HEAD request to the URL, which returns a variable that accepts integer values. In bytes, it shows how big the message body was that was transmitted to the receiver.
- The name of the server, which might be Apache or Microsoft Internet Information Server, is known as server Type.
- To find out what software the web server is using to create responses, you may look at the "powered by" string. It specifies the application platform.
- The content type includes encoding type charset information.
- The variable last Modified indicates the date of its last modification.

7. Data processing

The framework name and major/minor version numbers were pre-processed into the powered By and server Type variables, respectively. Also, the last Modified column was removed from all the data treatment options as it was determined that the data was irrelevant to the research. Figure 4 displays those five rows that make up the dataset that was used.

	url	compromissionType	isHiddenFraudulent	contentLength	serverType	poweredBy	contentType	lastModified
0	http://www.sinduscongoias.com.br/index.html	defacement	False	2474	Apache/2.2	NaN	text/html	Sat, 05 Jan 2013 19:38:29 GMT
1	http://www.sinduscongoias.com.br/index.php/ins...	defacement	False	0	Apache/2.2	NaN	text/html; charset=utf-8	Mon, 21 Jan 2013 19:30:53 GMT
2	http://www.sinduscongoias.com.br/index.php/ins...	defacement	False	0	Apache/2.2	NaN	text/html; charset=utf-8	Mon, 21 Jan 2013 19:30:58 GMT
3	http://www.sinduscongoias.com.br/index.php/ins...	defacement	False	0	Apache/2.2	NaN	text/html; charset=utf-8	Mon, 21 Jan 2013 19:31:01 GMT
4	http://www.sinduscongoias.com.br/index.php/ins...	defacement	False	0	Apache/2.2	NaN	text/html; charset=utf-8	Mon, 21 Jan 2013 19:31:05 GMT

Figure 4 Dataset (First Five Columns)

The Powered By column had its NaN value changed with a 0 before any model was applied. Similarly, records where any column with missing data were also eliminated. This process yielded a dataset including the dependent variable as well as 181,916 rows and 7 columns. There were 8,618 phony URLs among them. After determining that the data set is imbalanced, the class was included in the models with the class_weight="balanced" attribute.

International Journal of Environmental Sciences
ISSN: 2229-7359
Vol. 11 No. 21s, 2025
https://theaspd.com/index.php

Furthermore, class-free independent variables are subjected to the Pearson technique for correlation analysis (see Figure 5).

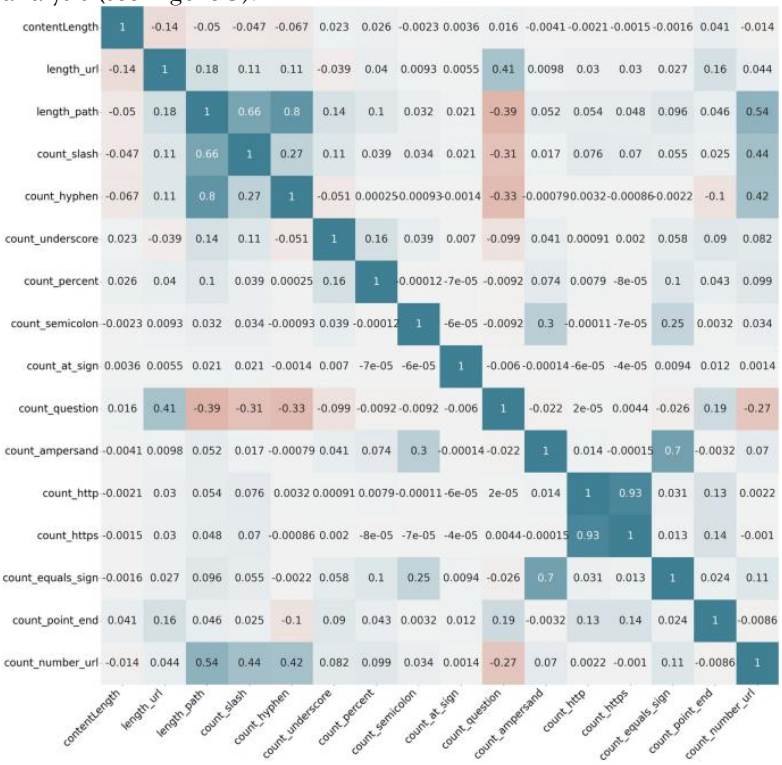

Figure 5 Dataset Correlation

Figure 6 shows that many factors are highly correlated with one another. Consequently, the variables We got rid of count_http and count_hyphen. Nevertheless, counting http and counting hyphen were all retained in the dataset going forward since they were utilized in certain algorithms.

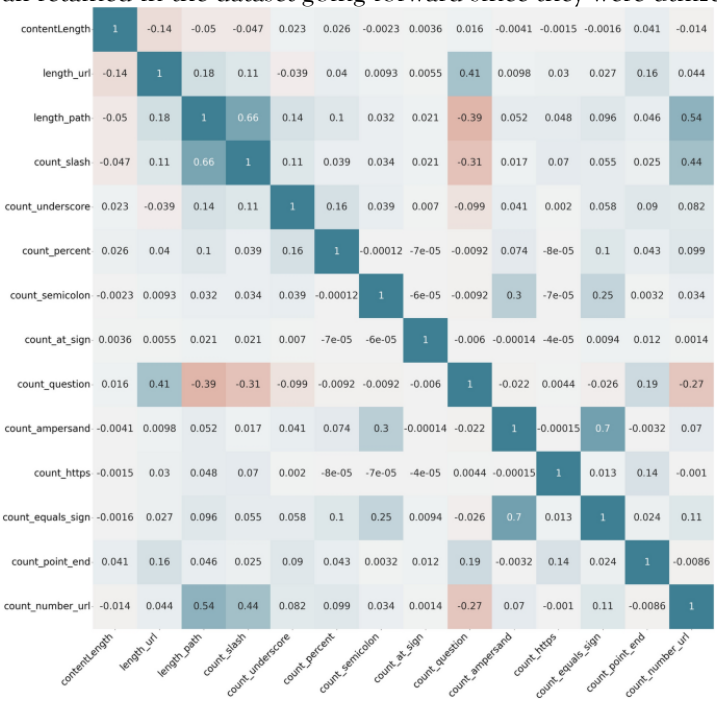

Figure 6 Dataset Correlation for the Alternative Algorithms

8. Application of Classical Machine Learning Algorithms

Before using standard ML algorithms, it is necessary to ensure that the data is good without URLs & newly formed variables. The training, 145,532 randomly selected data points were used from the dataset using the ML decision tree technique. Six thousand nine hundred and sixty-five malicious URLs made up the training set. Achieving an accuracy of 87.77% was made possible with this (see Figure 7).

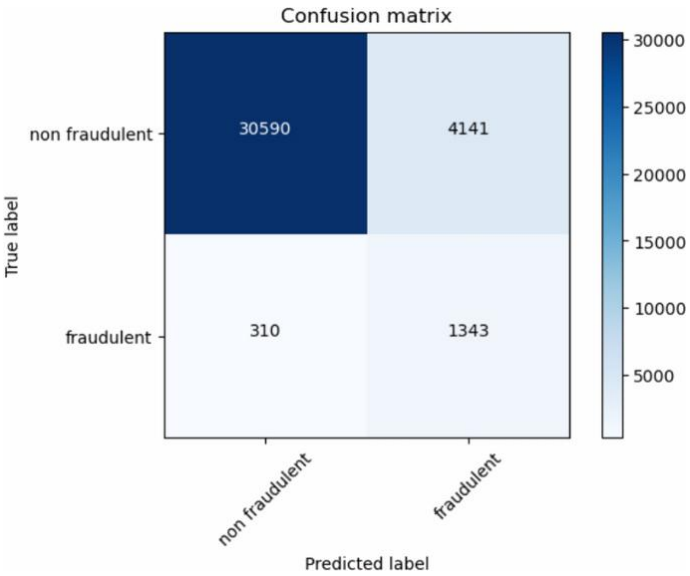

Figure 7 Classical ML Algorithm

9. Resulting Confusion Matrix

From the training set, only 1,343 (or 81.25% of the total) bogus URLs were successfully recognized (Figure 7). In contrast, 88.08% of the URLs that were determined to be legitimate were blocked. Since 310 bogus URLs are currently being poorly categorized, this suggests that more analysis of the URLs might lead to improved accuracy.

After removing the variables serverType, compromissionType, poweredBy, and contentType from the analysis, a decision tree was constructed based on the URLs alone. Next, other variables were established, including the URL's length and the number of times the '/' character occurs in the URL's route, among others. Figure 4 displays all of the newly-created definitions. The result is a dataset with 16 columns (including the dependent variable) and an equal number of rows.

An accuracy of 93.18% was achieved by using the additional variables. Figure 8 displays the updated confusion matrix.

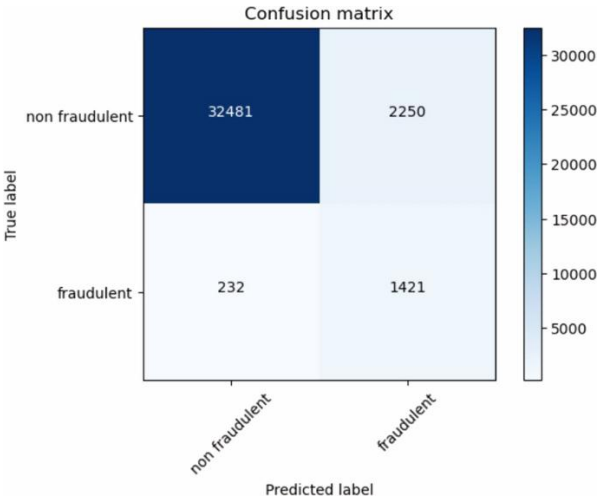

Figure 8 New Confusion Matrix

Compared to the prior training set, this one successfully recognized more data points (Fig. 7). Of those URLs, 93.52% were determined to be legitimate, while 84.96% were determined to be fake. The software found that 232 bogus URLs are poorly identified, which makes one wonder whether the two systems might work together to provide even more detailed results.

After this was finished, the data needed to be prepared. This was accomplished by retaining the following independent variables: content Type, content Length, server Type, compromission Type, powered By, and by disregarding the previously defined terms used to extract data from URLs.

All but one of the algorithms for machine learning and deep learning—the decision tree included—also made advantage of data standardization. Finally, the dataset was analyzed using algorithms that were.

- Neural networks
- Decision trees

- Support Vector Machines
- Logistic regression

Because of their proven effectiveness in previous studies, we select to use Decision trees, Support Vector Machine, and logistic regression [7, 13, 14]. The study is just one more example of research that use these methods in conjunction with neural networks [33].

- Fig. 4's dataset was used with relation to neural networks and decision trees.
- The dataset shown in Figure 5 was used by the other ML algorithms.

10.RESULTS

What follows is a comparison of the outcomes produced by each of the approaches Sigmoid, RBF, and Poly kernels were used in the SVM method. Specifically, the RBF kernel from the study [34] was selected. Except for neural networks, the compared approaches can be found in Tables 2 and 3. Tables 1 and 2 show the outcomes of the Support Vector Clustering (SVC) analysis. While support vector machines (SVMs) and SVCs both rely on kernel functions, SVCs are more suited to unsupervised learning.

Table 2 Comparison of the results obtained as non-fraudulent

ML algorithm	Precision (%)	Recall (%)	F1-score (%)
Logistic regression	99	82	89
SVC (Sigmoid)	97	58	72
SVC (Poly)	100	96	98
SVC (RBF)	100	98	99
Decision tree	100	99	100

Table 3 Comparison of the results obtained as fraudulent

ML algorithm	Precision (%)	Recall (%)	F1-score (%)
Logistic regression	16	75	27
SVC (Sigmoid)	6	59	11
SVC (Poly)	55	93	69
SVC (RBF)	68	94	79
Decision tree	89	91	90

When it came to neural networks, three distinct types were constructed. The first neural network made use of the Sigmoid activation function in its hidden layer. Furthermore, the 'binary accuracy' metric, both the 'binary_crossentropy' loss function and the 'adam' optimizer were utilized. Its confusion matrix is shown in Figure 9.

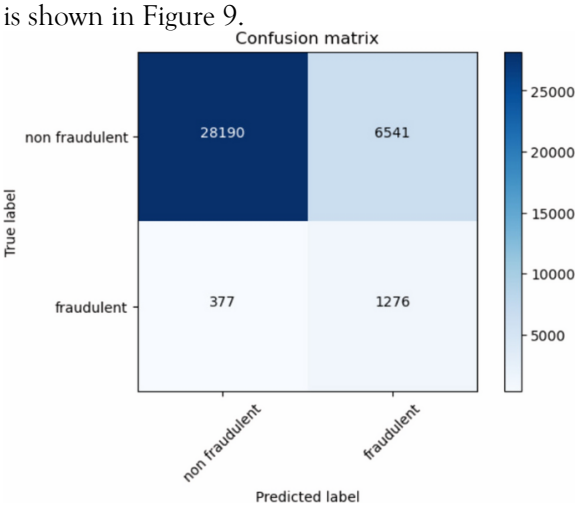

Figure 9 First Neural Network

The Sigmoid activation function was used by each of the three hidden layers that comprised the supplementary neuronal system. Utilizing the identical optimizer, 'rmsprop,' as well as the identical loss function and metric, this neural network replicated the prior one. Figure 10 displays the neural network's confusion matrix.

International Journal of Environmental Sciences
ISSN: 2229-7359
Vol. 11 No. 21s, 2025
https://theaspd.com/index.php

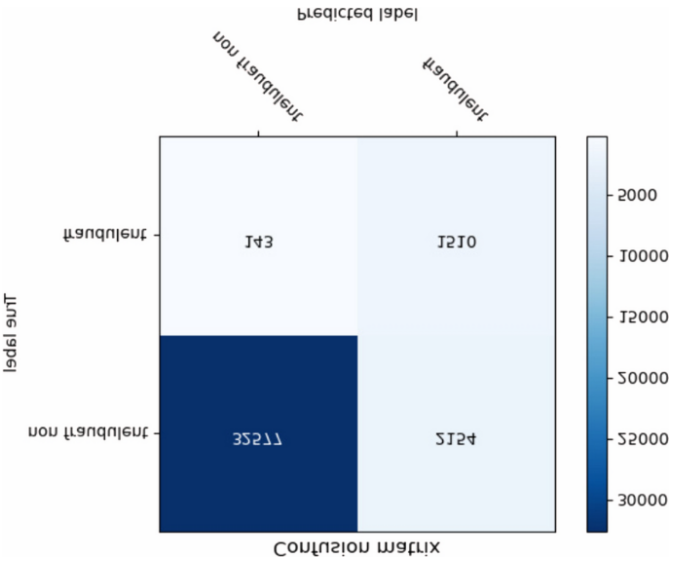

Figure 10 Second Neural Network

The activation functions'relu' and Sigmoid were used in the first two levels, respectively. Like the first neural network, this one made use of the same optimizer, measure, and loss function. You can see the matching confusion matrix in Figure 11.

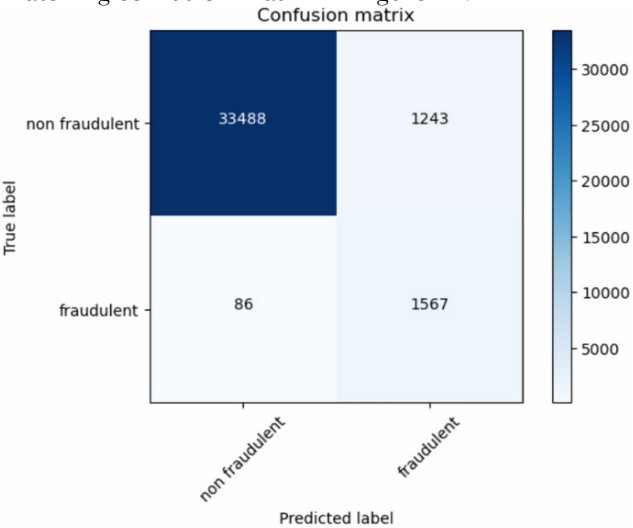

Figure 11 The Third Kind of Neural Network

Table 4 Overview of the Three Neural Networks' Confusion Matrices

Table 4 Comparison of the three neural networks

ML algorithm	Precision (%)	Recall (%)	Accuracy (%)
First neural network	16	77	80, 99
Second neural network	41	91	93, 69
Third neural network	56	95	96, 35

It was reevaluated if after the differentitations were made, it became clear that the models utilized were either overfitting or underfitting. Binary accuracy and val binary accuracy were investigated for this reason. Figure 12 shows that there are two variables' representations. When both the obtained and validated precisions of a model are low, we say that it is underfit. accuracy while training is high but accuracy during validation is poor when overfitting is present.

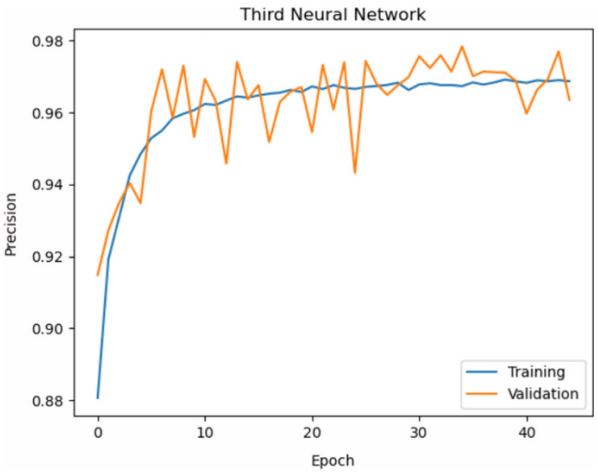

Figure 12 The Binary Accuracy and Valuable Binary Accuracy Constants Described View in its entirety

Figure 12 shows that the values of value_binary_accuracy and binary_accuracy are nearly identical.

Table 5 Illustrates the ROC Area

Table 5 Area under the ROC curve

ML algorithm	Area (%)
Logistic regression	78.46
SVC (Sigmoid)	58.50
SVC (Poly)	94.49
SVC (RBF)	95.86
Decision tree	94.98
First neural network	79.18
Second neural network	92.57
Third neural network	95.61

As far as the eye can tell, the decision tree comes last in an area classification, followed by third-generation neural networks and SVC (RBF).

It is feasible to compare the outcomes achieved by other writers by using a publicly available dataset. As an example, a more involved study that considers the data presented on the website may be seen in [13]. They kept track in terms of the total number of TCP packets transmitted and received by the honeypot from the website, for instance, using a variable called TCP conversation exchange. In fact, the data processing in this study is primarily concerned with the details supplied by the URL. On top of that, recall is used for analysis here, while in that study accuracy is used for comparison. Since they took a different tack with the data in this case, we can only utilize their second Table for comparisons. Below, we compare those two strategies as they are the only ones that are consistent throughout the two works. Table 6 displays the accuracies achieved by, and Accuracy (1) stands for the program's accuracy in this research; Accuracy (2) stands for the accuracy in [13]. both programs. Evidently, their data processing allowed them to achieve more accuracy in the logistic regression than we were able to do here.

Table 6 Evaluating the Accuracy of Two Programs

Table 6 Comparison of the accuracy of both programs

ML algorithm	Accuracy (1) (%)	Accuracy (2) (%)
SVM	97, 70	97, 41
Logistic regression	81, 48	90, 58

11.Evaluation

Finally, a new dataset was employed to assess the trained models, focusing on their accuracy and generalization performance. The source of this dataset is outlined below:

https://github.com/ESDAUNG/PhishDataset/blob/main/data_bal%20-%2020000.xlsx.

Only phishing URLs & compromissiontType are included in this new dataset. Hence, the remaining dataset columns (serverType, contentLength, etc.) were chosen to be deleted. To get Table 7, we retrained three models using the initial dataset.

Table 7 Training with The Original Set

Table 7 Training with the original set

ML algorithm	Precision (%)	Recall (%)	Accuracy (%)
Decision tree	50	91	95, 43
Third neural network	38	95	92, 85
SVM (Poly)	30	94	90, 01
SVM (RBF)	37	95	92, 48

Table 7 displays the outcomes of the new dataset's prediction using a subset for the models shown in Tables 3 and 4, following their training on the initial dataset and subsequent removal of the columns indicated before.

When the algorithms attempt to forecast the fraudulent URLs using the fresh information, the outcomes are shown in Table 8.

Table 8 Prediction of New Dataset

Table 8 Prediction of the new dataset

ML algorithm	Precision (%)	Recall (%)	Accuracy (%)
Decision tree	46	75	42, 95
Third neural network	46	85	43, 29
Support vector machine (poly)	47	75	44, 81
Support vector machine (RBF)	49	87	47, 33

According to Table 8, the program's greatest capability for detecting bogus URLs while attempting to anticipate a fresh dataset is 87%. Since the software did not have any previous knowledge of such URLs being false, this proportion is likely attributable to the fact that it discovered new, similar ones.

12.Adoption of Quantum Algorithms in Machine Learning

The degree of efficiency that this new paradigm may deliver employing quantum neural networks was measured by conducting an examination of a quantum method to addressing the examined problem. The purpose of this research was to examine the ML algorithms associated with quantum computing and their possible practical applications, hereafter referred to as Quantum Machine Learning.

13.QML Algorithms

There are four distinct ways to implement QML algorithms, each tailored to a different combination of data type and algorithm execution hardware.

CC: Traditional data processed using machine learning techniques on traditional hardware.

CQ: Quantum hardware processing classical data with ML techniques

QC: Utilizing quantum data in conjunction with classical hardware and machine learning methods

QQ: Big Data using ML techniques implemented on Quantum hardware

The principal use of CQ in this study is to conduct simulations on classical hardware using classical data that has been encoded in quantum information using the associated technique. Python and IBM's quantum computing framework Qiskit were the only languages used to develop the project.

The following steps must be taken prior to training a Variational Quantum Classifier, since it has been utilized [35].

Data coding, which involves selecting several techniques for feature mapping and translating the original data to qubits.

Using a parameterized quantum circuit, also known as ansatz, is a great way to minimize an objective function. These quantum circuits are characterized by a set of changeable weights. There has been a selection of Ansatz.

- Finding the best optimization strategy by combining a function similar to a classic Deep Learning model with three local optimizers, which aim to discover the optimal value inside a candidate solution's surrounding set):
- The COBYLA optimizer, which stands for " Implementing Optimization Constraints via Linear Approximatio."
- To minimize gradient descent, you may use the GradientDescent procedure.
- Sequential Least Squares Programming enhancer, or SLSQP

14.Adaptation of the Dataset

Codifying the category variables became important since QML model applications need datasets with only numerical attributes. In this regard, several options were explored, including:

Optimal for creating a hierarchical structure using the variable's values is ordinal encoding. Nonetheless, this casuistry does not match to the values of the categorical variables in the instance that was examined.

One-hot encoding: each category is connected to a numerical characteristic vector in such a manner that the vector contains as many elements as the variable's categories, with all elements representing "0" except for the position associated with the observation's category, which includes a 1. The dataset would grow substantially as some of the investigated attributes had over a hundred possible classifications. —a negative in this situation.

Binary coding is a hybrid approach that combines the two methods described above. It starts with ordinal encoding and then converts each integer to binary. An increasing number of columns is generated for every integer in the binary output. Even though this approach simplifies the dataset, it is still less efficient than the "one hot" encoding.

Hashing: you'll an integer value within a specific range is generated for each variable category using a hashing method. Nevertheless, it is crucial to regulate collisions, when many inputs produce identical outputs, since they have the potential to impact the dataset's quality.

Ordinal encoding was used after considering its effects over other algorithms for text-to-number variable conversions after careful consideration of their relative merits; this was largely due to the algorithms inherent. Additionally, ordinal encoding maintains the dataset's dimensionality, unlike other approaches like one-hot encoding. This is essential for present-day quantum models with a restricted number of useable qubits. In comparison to methods like random coding, its use often results in less data loss (for example, by maintaining order). Therefore, we leaned towards this option due to the benefits we recognized and the early testing we were able to do, even if it does have some drawbacks (such as the possibility of an artificial numerical link between groups that aren't necessarily ordered).

However, comparing various encoding techniques and analyzing their influence on the final outcomes is one of the next areas of inquiry. If the number of variables that need to be handled becomes too large after using one-hot, it is important to examine how it is used in conjunction with dimensionality reduction methods like PCA (Principal Component Analysis).

Following is a description of the original dataset and the fields that were processed in order to apply QML methods, with the stipulation that all attributes be numerical.

url: keeps track of how many characters are in the URL.

compromissionType: variable that indicates if the website is legitimate, compromised, or phished. Everything is given a numerical value, whether it's 1, 2, or 0 accordingly) to alter it.

isHiddenFraudulent: shifted to 1 for malicious URLs and 0 for safe ones.

contentLength: currently an integer-valued variable.

serverType: numerical identifier for every kind of server.

poweredBy: integer representing the web server's underlying application platforms.

contentType: given numerical value for every form of encoding.

15.Application of VQC

A preliminary estimate was created using a balanced dataset of 200 observations, one hundred harmful URLs and one hundred benign ones, due to the dataset's volume and processing delays. As a result, we compared using the whole dataset in the traditional SVM model to using the condensed version. To begin, the following values were acquired via using the standard SVM model on the whole dataset:

- On the training dataset, the classical support vector machine achieved a value of 0.97.
- Results from using classical support vector machines on the test dataset: 0.97

The following resulted from applying it to the reduced dataset:

- For the training dataset, the classical support vector machine achieved a value of 0.91.
- Results from using classical support vector machines on the test dataset: 0.97

In Figure 13, we can see the confusion matrix that is linked to the reduced dataset.

International Journal of Environmental Sciences
ISSN: 2229-7359
Vol. 11 No. 21s, 2025
https://theaspd.com/index.php

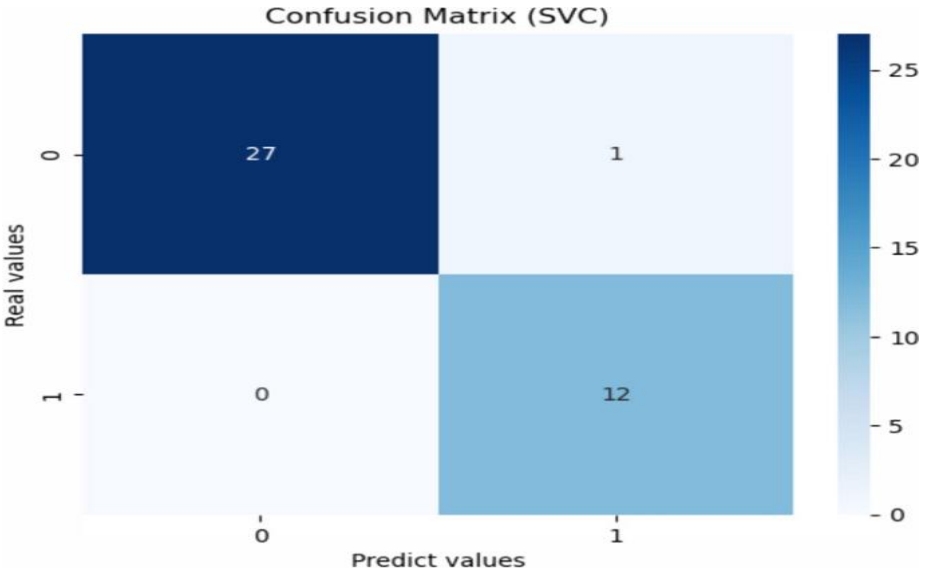

Figure 13 Confusion Matrix

16.Confusion matrix SVC

We tried a lot of different permutations of feature map generating algorithms, using optimization techniques, ansatz methods, and fastest and most efficient parameterization possible, considering the following criteria:

All computations were carried out in Python inside the context of Jupyter Notebook, using the pandas, numpy, matplotlib, seaborn, sklearn, and Libraries for IBM qiskit.

The outcomes were recorded using a MacBook Pro equipped with an M2 Pro graphics card and 16 GB of RAM. Nevertheless, to compare the model training times on both types of hardware, the calculations were also executed in parallel on a PC with an Intel i7 4K Ghz CPU and 32 GB of RAM.

Our team has chosen ZZFeatureMap, ZfeatureMap, and PauliFeatureMap as our feature map generating algorithms.

The algorithms ExcitingPreserving (ExcitPreserving), EfficientU2, and RealAmplitudes (RealAmp) were used in relation to Ansatz.

COBYLA, GradientDescent, and SLSQP were used as optimizers, each having a 20-iteration cycle.

The accuracy acquired from vqc.score was applied to both the training (TrainS) and test (TestS) datasets in order to gauge the model's performance.

The outcomes of running the VQC model on the reduced dataset are summarized in Tables 9, 10, and 11.

Table 9 ZZ Feature Map

Table 9 ZZFeatureMap

Ansatz	Opt	Train (s)	Test (s)	TPC (s)	TMAC (s)
RealAmp	COBYLA	0.79	0.82	87	127
RealAmp	GradientDescent	0.21	0.28	1328	536
RealAmp	SLSQP	0.80	0.85	1284	536
EfficientU2	COBYLA	0.74	0.72	97	26
EfficientU2	GradientDescent	0.52	0.65	2900	1189
EfficientU2	SLSQP	0.84	0.82	2942	1192
ExcitPreserving	COBYLA	0.72	0.78	117	33
ExcitPreserving	GradientDescent	0.28	0.17	7439	5323
ExcitPreserving	SLSQP	0.82	0.80	7442	3129

Table 10 Z Feature Map

Table 10 ZFeatureMap

Ansatz	Opt	Train (s)	Test (s)	TPC (s)	TMAC (s)
RealAmp	COBYLA	0.81	0.95	26	11
RealAmp	GradientDescent	0.47	0.72	571	234
RealAmp	SLSQP	0.89	0.97	472	193
EfficientU2	COBYLA	0.55	0.30	38	15
EfficientU2	GradientDescent	0.53	0.78	1470	584
EfficientU2	SLSQP	0.80	0.78	2185	585
ExcitPreserving	COBYLA	0.49	0.75	53	22
ExcitPreserving	GradientDescent	0.46	0.70	4651	7769
ExcitPreserving	SLSQP	0.90	0.95	4871	1807

Table 11 Pauli Feature Map

Table 11 PauliFeatureMap

Ansatz	Opt	TrainS	TestS	TPC (s)	TMAC (s)
RealAmp	COBYLA	0.65	0.85	87	22
RealAmp	GradientDes-cent	0.28	0.25	1228	11019
RealAmp	SLSQP	0.80	0.85	1258	541
EfficientU2	COBYLA	0.74	0.72	98	27
EfficientU2	GradientDes-cent	0.30	0.25	2749	12922
EfficientU2	SLSQP	0.84	0.82	2824	1204
ExcitPreserv-ing	COBYLA	0.72	0.78	118	33
ExcitPreserv-ing	GradientDes-cent	0.28s	0.17s	7410	7887
ExcitPreserv-ing	SLSQP	0.82	0.80	7516	3094

Particularly noteworthy are the outcomes achieved with the following parameter combinations:

- Making the feature map ZFeatureMap, as shown in Figure 13.
- Approach: Real Amplitudes (two times)
- SLSQP-based optimizer with 20 rounds

The 17 parameters utilized, [1]...θ[17], and the two repeats mentioned in the Python code are shown in the Ansatz representation in Figure 15.

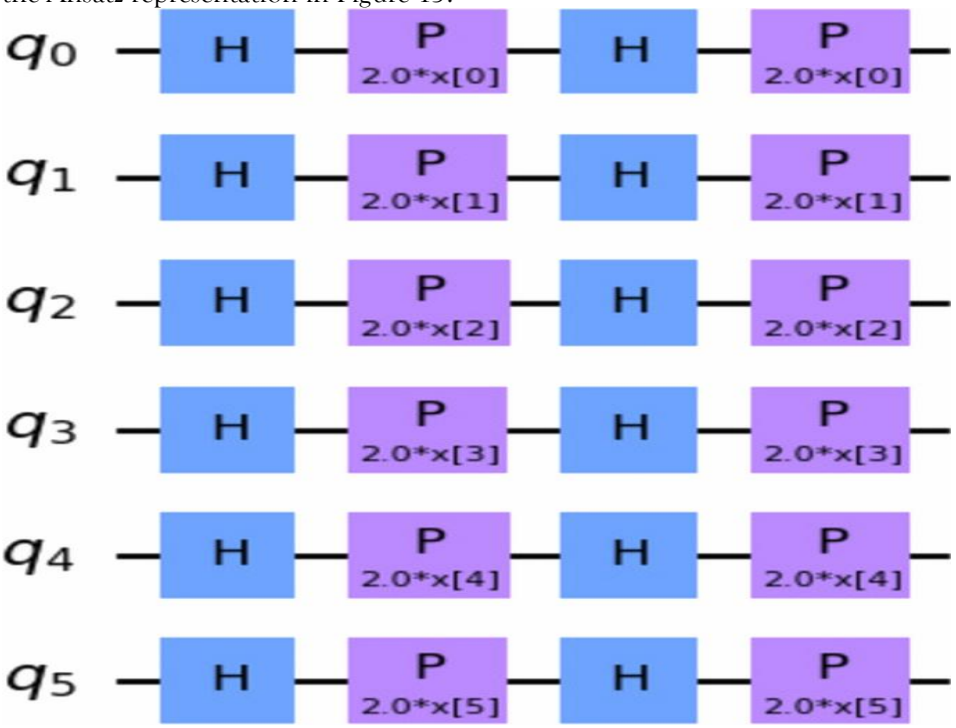

Figure 14 Feature mapping with ZFeatureMap

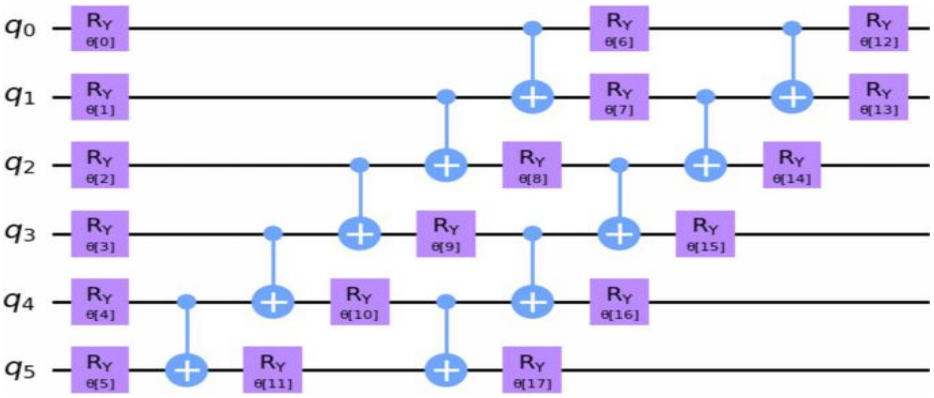

Figure 15 Representation Ansatz Real Amplitudes

Specifically, Figure 16 displays the outcomes of the VQC model simulation:
on the training dataset, the quantum VQC is 0.89.
Test dataset quantum VQC: 0.97

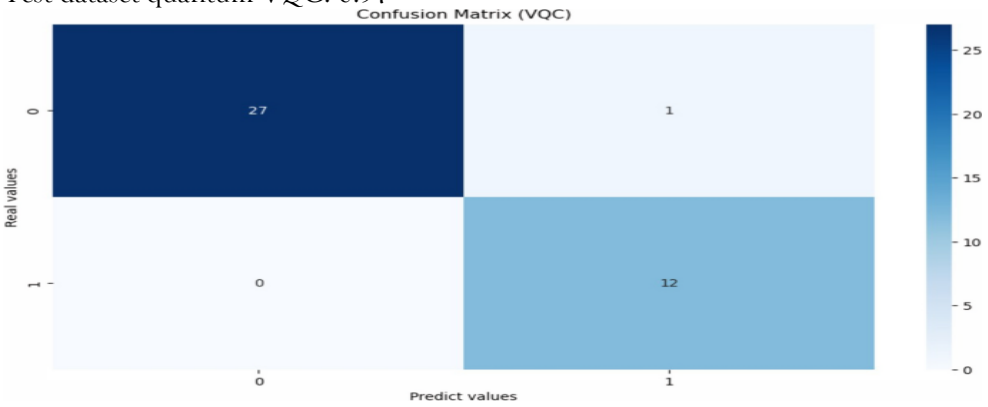

Figure 16 Confusion matrix VQC

17.Evaluation

- Some intriguing inferences may be taken from the data analysis:
- The optimizer selection is more important than the Ansatz and feature mapping algorithms when it comes to parameter combinations. Despite using few iterations, the SLSQP optimizer shines noteworthy.
- Since it does not employ gradient descent, the COBYLA optimizer boasts the quickest training speed. Using Z Feature Map, Real Amplitudes, and COBYLA together is an option to think about when time is of the essence.
- The Gradient Descent optimizer needs at least 100 iterations to improve the results, which is low considering how few iterations were used. Nevertheless, it is worth considering.
- Z Feature Map stands out as the feature mapping method that has achieved the greatest results on a worldwide scale.
- The M2 PRO processor's performance is noticeable while using older hardware; it achieves a distinct edge in training times in almost all instances.

18.CONCLUSION AND FUTURE WORK

Using both classical and quantum machine learning techniques, this study examines the cybersecurity issue of identifying bogus URLs from many angles. The primary objective has been to investigate the potential of cybersecurity-related machine learning applications of quantum computing.

One positive aspect of the dataset is that it was determined during the research that the initial dataset was imbalanced, which allowed for the identification of the best methods and procedures to maximize the findings. This study has so shown the significance of doing such data analysis beforehand, before beginning to use various algorithms in a generalized manner.

Conversely, the research found that the answers that are most relevant to real-world applications are likewise influenced by the issue typology. In the confusion matrix, for instance, addressing a cybersecurity issue necessitates paying close mistaken for legitimate ones, leading to substantial service or financial losses. So, to see how Accuracy drops but Recall rises, it is important to minimize False Positives to the greatest extent feasible in this instance. Our research has shown that the F1-score is the most useful

metric for comparing models in this context, as minimizing False Positives while keeping False Negatives relatively constant is the primary objective.

One of the three neural networks suggested in this study was obviously determined to be the best fit for the dataset that was analyzed, which brings us to another conclusion about conventional machine learning models. Support Vector Machines using RBF kernels and Poly kernels were the runner-up and third-best models, respectively.

Assessing the value of QML models in cybersecurity has been a primary focus of this effort, with the hope of determining the optimal algorithm combinations for this setting. By determining which permutations of optimisers and feature mapping method provide outcomes comparable to those of classical models trained on a reduced dataset, several intriguing findings on the importance of these tools were uncovered.

Classical models outperform ML and QML when comparing their outputs, which is not surprising given that classic models process the complete dataset. Given the vast quantity of scholarly literature on the subject and the years of study and optimization devoted to these models, it is only logical to reach this conclusion. Nevertheless, it is evident that despite QML's lack of optimization and its recent implementation, this study has shown promising results using QML, particularly when combining ZFeatureMap, RealAmp, and SLSQP.

Findings from studies of the current literature make it very evident that QML application is a young discipline, with relatively theoretical outcomes to date. Because quantum hardware is still in its early stages, its use in cybersecurity is very limited. Many new avenues for research have opened because of this discovery, including: • A dearth of current cybersecurity datasets appropriate for quantum computing tasks.

All the above lines lead to new areas of investigation that build on this work and address various issues that were found over the course of this study.

Finally, this study paves the way for a plethora of future research on the best ways to employ.

19. REFERENCES

1. ENISA:ENISAthreat landscape 2023. https://www.enisa.europa.eu/publications/enisa- threat-landscape-2023

2. Fortinet:WhatisURLphishing?
(2023). https://www.fortinet.com/resources/cyberglossary/url-phishing

3. Vanhoenshoven, F., Nápoles, G., Falcon, R., Vanhoof, K., & Köppen, M. (2016). Detecting malicious urls using machine learning techniques. In: IEEE Symposium series on computational intelligence (SSCI), pp. 1–8

4. Sahoo, D., Liu, C., & Hoi, S.C. (2017). Malicious url detection using machine learning: A survey. arXiv preprint arXiv:1701.07179

5. Le, H., Pham, Q., Sahoo, D., & Hoi, S.C. (2018). Urlnet: learning a url representation with deep learning for malicious url detection. arXiv preprint arXiv:1802.03162.

6. Aljabri, M., Altamimi, H.S., Albelali, S.A., Maimunah, A.-H., Alhuraib, H.T., Alotaibi, N.K., Alahmadi, A.A., Alhaidari, F., Mohammad, R.M.A., & Salah, K. (2022). Detecting malicious urls using machine learning techniques: review and research directions. IEEE Access.

7. Patil, D. R., & Patil, J. B. (2018). Malicious URLs detection using decision tree classifiers and majority voting technique. Cybernetics and Information Technologies, 18(1), 11–29.
Article Google Scholar

8. Hieu Nguyen, H., & Thai Nguyen, D. (2016). Machine learning based phishing web sites detection. In: AETA 2015: Recent advances in electrical engineering and related sciences, pp. 123–131.

9. Yahya, F., Isaac W., Mahibol, R., Kim Ying, C., Bin Anai, M., Frankie, A., Sidney, Ling Nin Wei, E., & Guntur Utomo, R. (2021). Detection of phising websites using machine learning approaches. In 2021 International conference on data science and its applications (ICoDSA).

10. Alkhudair, F., Alassaf, M., Khan, U. R., & Alfarraj, S. (2020). Detecting malicious url. In 2020 International conference on computing and information technology 1, 97–101.

11. A. Waheed, M., Gadgay, B., DC, S., P., V., & Ul Ain, Q. (2022). A machine learning approach for detecting malicious url using different algorithms and NLP techniques. In: 2022 IEEE North Karnataka Subsection Flagship International Conference (NKCon).

12. Ha, M., Shichkina, Y., Nguyen, N., Phan, T.-S. (2023). Classification of malicious websites using machine learning based on url characteristics. In Computational Science and Its Applications - ICCSA 2023 Workshops, pp. 317–327

13. Urcuqui, C., Navarro, A., Osorio, J., & García, M. (2017). Machine learning classifiers to detect malicious websites. Proceedings of the Spring School of Networks, 1950, 14– 17.
Google Scholar

14. Chiramdasu, R., Srivastava, G., Bhattacharya, S., Reddy, P.K., & Gadekallu, T.R. (2021). Malicious url detection using logistic regression. In: 2021 IEEE International conference on omni-layer intelligent systems (COINS), pp. 1–6.

15. Mercaldo, F., Ciaramella, G., Iadarola, G., Storto, M., Martinelli, F., & Santone, A. (2022). Towards explainable quantum machine learning for mobile malware detection and classification. Applied Sciences, 12(23), 12025.
Article CAS Google Scholar
16. Kalinin, M., & Krundyshev, V. (2023). Security intrusion detection using quantum machine learning techniques. Journal of Computer Virology and Hacking Techniques, 9, 125–136.
Google Scholar
17. Patel, O., Tiwari, A., Patel, V., & Gupta, O. (2015). Quantum based neural network classifier and its application for firewall to detect malicious web request. In 2015 IEEE Symposium Series on Computational Intelligence. IEEE, pp. 67–74
18. Gelman, A., Hill, J., & Vehtari, A. (2020). Regression and other stories. Cambridge University Press.
Book Google Scholar.
19. Hosmer, D. W., Jr., Lemeshow, S., & Sturdivant, R. X. (2013). Applied logistic regression. Wiley.
Book Google Scholar
20. Quinlan, J. R. (2014). C4.5: programs for machine learning. Elsevier. Google Scholar
21. Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine Learning, 20, 273–297.
Article Google Scholar
22. Cristianini, N., & Ricci, E. (2008). Support vector machines. Springer. Google Scholar
23. McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity.
The Bulletin of Mathematical Biophysics, 5, 115–133.
Article MathSciNet Google Scholar
24. Moldwin, T., & Segev, I. (2020). Perceptron learning and classification in a modeled cortical pyramidal cell. Frontiers in Computational Neuroscience, 14, 33.
Article PubMed PubMed Central Google Scholar
25. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep feedforward network. In: Deep Learning, pp. 164–223.
26. Nielsen, M. A., & Chuang, I. L. (2010). Quantum computation and quantum information. Cambridge University Press.
Google Scholar
27. Raschka, S., & Mirjalili, V. (2019). Python machine learning: Machine learning and deep learning with python, scikit-learn, and tensorflow 2. Packt Publishing Ltd. Google Scholar
28. Osval Antonio Montesinos López, J.C. & Abelardo Montesinos López. (2022). Overfitting, model tuning, and evaluation of prediction performance. In Multivariate statistical machine learning methods for genomic prediction, pp. 109–139.
29. Haozhe Xie, H.X. & Jie Li. (2017). A survey of dimensionality reduction techniques based on random projection. arXiv:1706.04371.
30. Cerulli, G. (2023). Model selection and regularization. In: Fundamentals of supervised machine learning, pp. 61–64.
31. Pothuganti, S. (2018). Review on over-fitting and under-fitting problems in machine learning and solutions. International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, 7(9), 369