Wireless Networks (2024) 30:7543–7560
https://doi.org/10.1007/s11276-024-03700-w

Detection of malicious URLs using machine learning

Nuria Reyes-Dorta[1] · **Pino Caballero-Gil**[1] · **Carlos Rosa-Remedios**[1]

Accepted: 7 February 2024 / Published online: 6 March 2024

Abstract
The detection of fraudulent URLs that lead to malicious websites using addresses similar to those of legitimate websites is a key form of defense against phishing attacks. Currently, in the case of Internet of Things devices is especially relevant, because they usually have access to the Internet, although in many cases they are vulnerable to these phishing attacks. This paper offers an overview of the most relevant techniques for the accurate detection of fraudulent URLs, from the most widely used machine learning and deep learning algorithms, to the application, as a proof of concept, of classification models based on quantum machine learning. Starting from an essential data preparation phase, special attention is paid to the initial comparison of several traditional machine learning models, evaluating them with different datasets and obtaining interesting results that achieve true positive rates greater than 90%. After that first approach, the study moves on to the application of quantum machine learning, analysing the specificities of this recent field and assessing the possibilities it offers for the detection of malicious URLs. Given the limited available literature specifically on the detection of malicious URLs and other cybersecurity issues through quantum machine learning, the research presented here represents a relevant novelty on the combination of both concepts in the form of quantum machine learning algorithms for cybersecurity. Indeed, after the analysis of several algorithms, encouraging results have been obtained that open the door to further research on the application of quantum computing in the field of cybersecurity.

Keywords Malicious URL · Machine learning · Confusion matrix · ROC curve · Support vector machine · Decision tree · Logistic regression · Neural network · Quantum computing

1 Introduction

The unique and specific address of each page on the Internet is called URL (Uniform Resource Locator). One of the most typical cyberattacks is based on the use of fraudulent versions of URLs, which are links that appear to lead to legitimate pages but redirect to fake pages that cybercriminals take advantage of to steal personal information such as passwords, bank accounts, etc. Thus, in the current digital age, the detection of fraudulent URLs has become a very important concern due to the increasing number of phishing cyberattacks that seek to deceive users to gain their trust by impersonating a person, company, or service, to to achieve victims to do something they should not, such as clicking on a fraudulent URL and providing sensitive information. Specifically, the annual report of the European Union Agency for Cybersecurity (ENISA) has recently found that phishing has become the most common initial attack vector [1].

Cybercriminals are specializing in using sophisticated techniques to create malicious URLs that look legitimate, making them harder to detect. Therefore, although phishing awareness has improved over the years, phishers are evolving their techniques through different URL phishing techniques that include mixing legitimate links with malicious links, abusing redirects or obfuscating malware with images [2]. However, with the advancement of Artificial Intelligence techniques and in particular Machine Learning (ML), it is possible to develop highly effective models to detect these fraudulent URLs, through the analysis of large amounts of data in order to recognize patterns and make predictions.

The main goal of this research is to analyze the application of different ML techniques for the early detection of fraudulent URLs, paying special attention to the pioneering

✉ Pino Caballero-Gil
pcaballe@ull.edu.es

✉ Carlos Rosa-Remedios
crosarem@ull.edu.es

[1] Department of Computer Engineering and Systems, University of La Laguna, La Laguna, Tenerife, Spain

use of Quantum Machine Learning (QML) to address this problem. In order to expand the set of tools to combat the phishing threat, this work analyses the possible application of QML for the detection of fraudulent URLs, and compare the obtained results with those produced using classical machine learning/deep learning methods. As this is a fairly new field, one of the first steps is to identify the most suitable combination of algorithms to apply a QML model, depending on the quantum conditions, and taking into account both its advantages and disadvantages in order to assess whether this approach can be useful in the context of cybersecurity in general, and in the detection of malicious URLs in particular.

In recent years, several studies have addressed the issue of applying ML techniques for the early detection of fraudulent URLs from different points of view.

In the paper [3], the authors work with a dataset consisting of 121 sets of URLs collected over different days. In total, this public dataset comprises over 2.4 million URLs, each with over 3.2 million features, that are analysed with various ML algorithms.

The authors of [4] propose, in addition to using a blacklist of URLs, to leverage other features such as lexical characteristics, length of the URL and length of the primary domain. Host-based features also include information such as creation date, Whois server, and name servers.

In the work [5], the authors apply Convolutional Neural Networks (CNN) to both characters and words of the URL string to capture several types of semantic information.

The paper [6] provides an extensive literature review highlighting the main techniques used to detect malicious URLs that are based on ML models.

The authors of [7] apply logistic regression, decision trees and SVM combined with majority voting technique for malicious URLs detection.

The work [8] uses decision trees, random forest, SVM, Naive Bayes and CNN algorithms, with a dataset of with legitimate website URLS collected from the site lists of the top 5000 websites in the world.

The paper [9] applies decision trees, K-NN and random forest algorithms on a dataset taken from a specific repository for ML.

The work [10] uses random forest, K-NN, J48 decision tree and BayesNet algorithms on a dataset taken from malicious and benign websites and ML classifiers, using a features like URL length or number of special characters.

The authors of [11] use decision tree, random forest, K-NN, Naive Bayes, SVM and logistic regression algorithms with a dataset from Kaggle, using a features like URL labels and text tokenization.

The paper [12] uses J48 decision tree, logistic regression, Naive Bayes and SVM algorithms with a dataset from Open-Phish, Phishtank, Zone-H, and WEBSPAM-UK2007, and features like having IP address, URL length, Shortening Service, httpSecure, Digit count or Abnormal URL.

In [13], J48 decision tree, logistic regression, Naive Bayes and SVM algorithms are applied with a dataset from Machine Learning Lab and features like ContentLength, compromissionType, serverType, poweredBy or contentType.

The work [14] considers several generic attributes such as length of UR, use of an IP address in URL, hexadecimal character codes in the URL, @ symbol in URL, number of dots in URL, number of sensitive words in URL, etc.

The paper [15] uses a dataset that contains real-world legitimate and malicious Android applications, converting each application into a grayscale image. Besides, they also employ a hybrid quantum CNN, a quantum Neural Network, and other CNN models.

The authors of [16] also apply QML to analyze an intrusion dataset and compare the obtained results obtained with conventional Support Vector Machine (SVM) and quantum SVM, as well as with conventional CNN and quantum CNN.

The work [17] is another of the few papers that deal with a quantum-based neural network classifier to detect malicious web request.

Table 1 shows a schematic comparison between the main aspects of this work in relation to some of the aforementioned publications.

As can be seen, none of the above-mentioned works includes one of the main novelties of the present work, which consists of studying the potential of the application of QML for the early detection of fraudulent URLs, and comparing the obtained results with those produced with different classic ML techniques.

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

This work is structured as follows. Section 2 includes a high level schematic about the overall proposal while Sect. 3 covers some preliminaries on ML. Section 4 is focused towards metrics, while Sect. 5 discusses some issues with ML algorithms. Section 6 comments on some features about the used dataset, while Sect. 7 refers to the data processing. Section 8 details the proposed models and implementation with classical ML, including both the obtained results and a brief evaluation. Section 9 introduces the application of QML to face the phishing problem, containing the adaptation of the dataset for the application of quantum algorithms, the application of quantum algorithms, and a brief evaluation. Finally, Sect. 10 closes the work with some conclusions and future work.

2 Proposed model

The development of this work has followed the model shown in Fig. 1, based on the Deming Cycle or PDCA (Plan, Do, Check, Act) cycle, with the following phases being carried out iteratively throughout the research process, first on classical Machine Learning and then on Quantum Machine Learning:

Phase 1: Exploratory analysis, analysing the different ways of approaching the problem under study, the usual techniques and new lines of research.

Phase 2: Identification of fundamental concepts. Compilation and study of the theoretical and practical foundations in the context of Machine Learning, including:

- Algorithms
- Metrics
- Potential problems

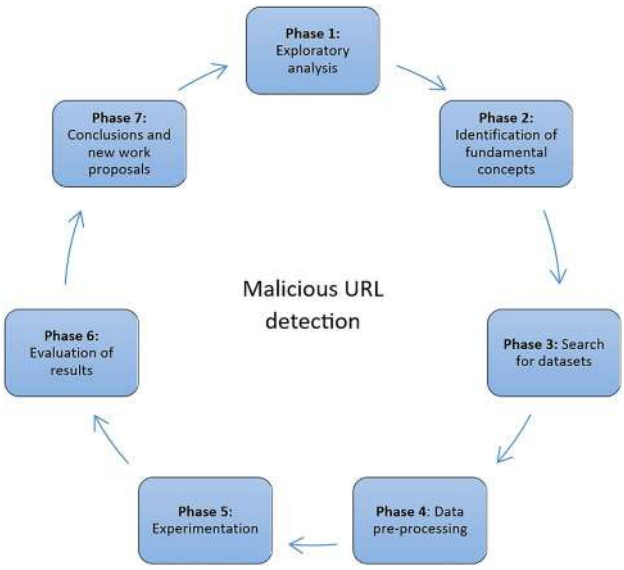


Fig. 1 Workplan

Phase 3: Search for datasets. One of the most problematic points, since most of them are old and in the field of cybersecurity it is critical to have up-to-date data.

Phase 4: Data pre-processing. Adaptation to the data in the reference dataset. Here, the peculiarities of adapting to quantum computing added an extra point of complexity.

Phase 5: Experimentation. This includes all the code development, testing, trial-and-error phase and algorithm execution. The time required to obtain meaningful data in the context of quantum computing is particularly important, which is why it was decided to include execution on Apple's Silicon chips.

Phase 6: Evaluation of results. At this point, based on the used metrics, the numerical results have been contextualised in relation to the problem under study.

Phase 7: Conclusions and new work proposals. Based on the results of each execution, new objectives were set and new work cycles were carried out.

3 Preliminaries of machine learning

This section includes some preliminaries about different ML algorithms.

3.1 Logistic regression

Logistic regression is a statistical data analysis technique used to model the relationship between independent variables and a categorical, usually binary, dependent variable [18].

This method works by calculating the probability that an observation belongs to a particular category. It uses a logistic function to predict the probability that a binary dependent variable has a value of 1 or 0 based on the independent variables. This function transforms the output of the linear regression to a value between 0 and 1, interpreted as the probability of belonging to a specific category.

Logistic regression is one of the most used supervised learning algorithms for binary classification in ML. It models the probability of a binary outcome by employing a logistic function to predict the probability of occurrence of a categorical dependent variable. Thus, it allows estimating the probability that a given input belongs to a certain category by fitting data to a logistic curve.

The model followed by logistic regression is based on the following expression [19], which returns the probability of a class x.

$$g_\theta(z) = \frac{1}{1 + e^{-z}} \tag{1}$$

where

$$z = \theta^T \cdot X \tag{2}$$

θ^T is the vector of the parameters that must be estimated from the data, and X is the vector of the independent variables.

3.2 Decision tree

A decision tree is a graphical representation that illustrates all the possible outcomes of a series of related decisions [20]. In essence, a decision tree resembles an inverted tree where each internal node represents a feature, each branch represents a decision rule, and each leaf node represents the outcome or the final decision.

Decision trees work by recursively partitioning the data into smaller and smaller subsets based on the most significant attributes or features. This partitioning process continues until it reaches a point where the data in each subset belongs to a single class or when the subset becomes too small, according to defined criteria.

These supervised learning algorithms are widely used for both classification and regression due to their adaptability to different data types, interpretability, and ability to capture complex relationships. They are versatile models that create decision boundaries based on features, making them effective for various tasks.

3.3 Support vector machine

A Support Vector Machine is a supervised ML algorithm used for both classification and regression tasks, although it is more commonly known for classification purposes. SVM is effective in finding the best possible decision boundary between data points of different classes.

The primary goal of SVM in classification is to find the optimal hyperplane that maximizes the margin, which is the distance between the hyperplane and the nearest data points of different classes, also known as support vectors [21]. This hyperplane effectively separates the data into distinct classes.

A dataset is said to be linearly separable [22] if the distribution of the observations is such that they can be separated perfectly linearly into two denoted classes (+1 and −1). In most cases, they cannot be separated perfectly linearly, so there is no hyperplane of separation. To solve this problem, kernels are introduced. The fundamental idea of the kernels that are used to treat linear inseparable data, is to create non-linear combinations of the original characteristics to project them towards a space of higher dimensions, through a mapping function ϕ, where they become linear separable. Some of the most commonly used kernels in SVM include:

- Linear:

$$K(x^{(i)}, x^{(j)}) = x^{(i)} \cdot x^{(j)} \tag{3}$$

- Radial Basis Function (RBF) or Gaussian Kernel:

$$K(x^{(i)}, x^{(j)}) = \exp(-\frac{\|x^{(i)} - x^{(j)}\|^2}{2 \cdot \theta^2}) \tag{4}$$

- Polynomial of degree k:

$$K(x^{(i)}, x^{(j)}) = (x^{(i)} \cdot x^{(j)})^k \tag{5}$$

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

$$K(x^{(i)}, x^{(j)}) = (c + x^{(i)} \cdot x^{(j)})^k \tag{6}$$

- Sigmoid:

$$K(x^{(i)}, x^{(j)}) = \tanh(ax^{(i)} \cdot x^{(j)} + b) \tag{7}$$

3.4 Neural network

A neural network is a computational model inspired by the structure and function of the human brain's interconnected network of neurons. It is a powerful ML algorithm used for tasks such as classification, regression or pattern recognition.

At its core, a neural network consists of layers of interconnected nodes called neurons organized in three main layers: input layer, hidden layers, and output layer. Each neuron receives input signals, processes them through an activation function, and then passes an output signal to the next layer.

Perceptron, Adaline and logistic neurons are early types of neural network models that paved the way for more complex network architectures.

The perceptron is a neural network based on the McCulloch-Pitts neuron [23]. This neuron is the first mathematical model used to replicate the electrical activity of a biological neural network, the perceptron has the same McCulloch-Pitts structure, the only difference is that the input variables are multiplied by some w_i, which are the weights. More information about the perceptron can be found at [24].

The Adaline and the logistic neuron are similar to the perceptron, but differ in the activation function. In the Adaline, a linear function is used while the logistic neuron applies a logistic function, which is the Sigmoid function defined in Eq. (1).

Pre-fed or feed-forward neural networks stand out among the fundamental types of neural networks [25]. They are characterized by having hidden layers and because the connections between neurons do not form a cycle. In each of the hidden

layers there are several neurons with the following structure. The neurons of the same layer are not connected to each other and they all share the same activation function. On the other hand, when there are two consecutive layers, it happens that all the neurons in one layer connect with all the neurons in the next layer, which makes the network a dense network.

3.5 Quantum computing

Below some basic concepts of quantum computing are introduced [26].

A qubit, short for quantum bit, is the fundamental unit of quantum information in quantum computing. Unlike classical bits, which can exist in one of two states (0 or 1), a qubit can exist in multiple states simultaneously due to the principles of quantum superposition. The state of a qubit ψ over the computational basis $|0\rangle, |1\rangle$ is defined as:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{8}$$

where $|\alpha|^2 + |\beta|^2 = 1$, and $\alpha, \beta \in \mathbb{C}$ are referred to as state amplitudes.

A quantum register is a collection of qubits $\psi_1, \cdots, \psi_n$ that are employed for computation. A quantum gate is an operator that acts on qubits and is represented by a unitary matrix. Examples of quantum gates include the NOT gate and the CNOT gate. A quantum circuit is a sequence of quantum gates $P_1, \cdots, P_n$ that are applied to a quantum register.

3.6 Variational quantum classifier

The model named Variational Quantum Classifier (VQC) is a special variant of the neural network classifier, which involves a quantum circuit and a function defined with its outcome, proposed to perform binary classification of classical data in a supervised learning scenario.

Similar to classical supervised ML algorithms, the VQC has a training stage (where data points with labels are provided and learning takes place) and a testing stage (where new data points without labels are provided which are then classified).

Specifically, the VQC is based on SamplerQNN, which is a neural network that accepts a parameterized quantum circuit with specific parameters for input data and/or weights. In the VQC, the measured expectation value is interpreted as the output of a classifier. In particular, it translates the quasi-probabilities estimated by the Sampler primitive into predictions for different classes.

4 Metrics

To assess different ML algorithms, both confusion matrix or ROC curve can be used.

4.1 Confusion matrix

The confusion matrix is a table used in the field of ML to evaluate classification models in general. It is a matrix that allows visualization of the performance of an algorithm by comparing predicted classes against actual classes.

The following information can be extracted from this structure:

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \tag{9}$$

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP+FN}} \tag{10}$$

$$\text{Specificity} = \frac{\text{TN}}{\text{TN+FP}} \tag{11}$$

$$\text{Accuracy} = \frac{\text{TP+ TN}}{\text{TP+FN+TN+FP}} \tag{12}$$

where

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.

Sensitivity, Recall or True Positive Rate are the names for the percentage of well-classified positive cases, which shows the ability of an algorithm to predict a positive result when the real result is positive.

Specificity is the percentage of well-classified negative cases.

Accuracy is the quotient of the well-classified data between the sum of all the data. That is, it is the percentage of correct predictions compared to the total.

4.2 ROC curve

The Receiver Operating Characteristic (ROC) curve [27] is a graphical representation that illustrates the performance of a binary classification model across different threshold settings. The ROC curve represents the rate of True Positives against the False Positive Rate. Therefore, the ROC curve plots the sensitivity against 1-Specificity. Also, one way to compare classifiers is to measure the Area Under

the Curve (AUC). A classifier is said to be perfect when it has an area under the ROC curve equal to 1. An example of a ROC curve can be seen in Fig. 2.

5 Issues with machine learning algorithms

Two typical problems that can be found when using Machine Learning algorithms are overfitting and underfitting [28].

Overfitting occurs when a Machine Learning model learns the training data too well, capturing not only the underlying patterns but also the noise or random fluctuations in the data. As a result, the model performs exceptionally well on the training data but poorly on unseen or new data.

Underfitting occurs when a Machine Learning model is too simple to capture the underlying patterns in the training data. It fails to fit the training data and, as a result, performs poorly on both the training and test data.

In order to resolve the overfitting issue, the following solutions can be applied:

- Use simpler models with fewer parameters or less complicated Machine Learning algorithms.
- Select relevant features and discard irrelevant or redundant ones, and apply techniques like dimensionality reduction like PCA [29].
- Use cross-validation techniques like k-fold cross-validation to assess model performance more accurately, and adjust hyperparameters based on cross-validation results [28].
- Apply regularization techniques like L1 (Lasso) and L2 (Ridge) regularization to penalize large parameter values, which prevent the model from fitting noise in the data [30].

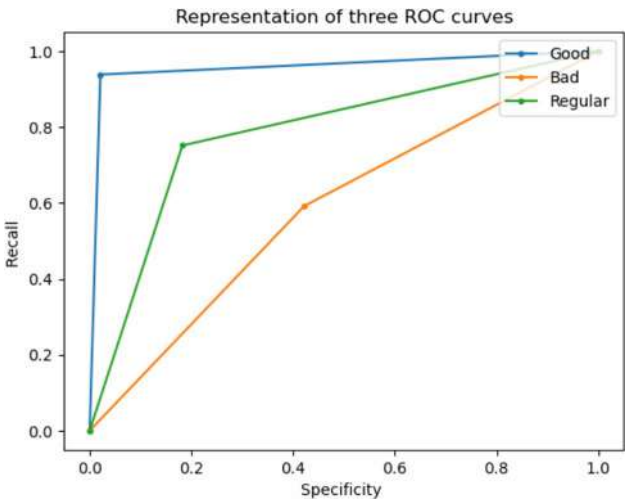


Fig. 2 ROC curve

- In deep learning, use early stopping to halt training when validation performance starts to degrade [31].

In order to resolve the underfitting issue, the following solutions can be applied:

- Choose more complex models with greater capacity to capture the underlying patterns, and use deep neural networks or more sophisticated Machine Learning algorithms.
- Create additional informative features that better describe the data, and experiment with transformations of existing features.
- Adjust hyperparameters to fine-tune the model, and try different learning rates, depths, regularization strengths, etc [32].

6 Used dataset

For the research on ML applied to the detection of fraudulent URLs, the dataset obtained from https://machinelearning.inginf.units.it/data-and-tools/hidden-fraudulent-urls-dataset was used. This dataset was chosen because it is available and labelled, and contains a rich set of data that provides good performance in the study of different ML algorithms, including QML. In addition, it has been used in other researches, which allows comparing results. Below, a comparison between the present work with another work that used the same dataset is included.

In particular, this dataset contains the following information:

- *url* is the current URL.
- *compromissionType* is the variable that indicates if the website is compromised by phishing, defacement, or normal.
- *isHiddenFraudulent* is a dependent variable indicating whether the URL is fraudulent or not.
- *contentLength* is a variable that takes integer values and was obtained by sending an HTTP HEAD request to the URL. It also indicates the size of the message body, in bytes, sent to the recipient.
- *serverType* is a string indicating the server, such as Apache, Microsoft IIS.
- *poweredBy* is a string that indicates the application platform underlying the web server, that is, it is used to specify with which software the response has been generated by the server.
- *contentType* contains charset information that is of the encoding type.
- *lastModified* is a variable indicating when its last modified date was.

7 Data processing

Both the *poweredBy* and *serverType* variables were preprocessed to hold the framework name and the major and minor version number. Besides, several approaches were developed for the treatment of the data and in all of them the *lastModified* column was deleted since that data was found not relevant to the study. In Fig. 3 the first five rows of the used dataset are shown.

Before implementing any model, the NaN value was replaced with a 0 in the *PoweredBy* column. In the same way, the rows in which some data was missing in any column were deleted. Doing so, a dataset of 181,916 rows and 7 columns was obtained, including the dependent variable. Among them, in total were 8,618 fraudulent URLs. Therefore, it was concluded that this is an unbalanced set of data, so this class was taken into account by indicating *class_weight="balanced"* in the models.

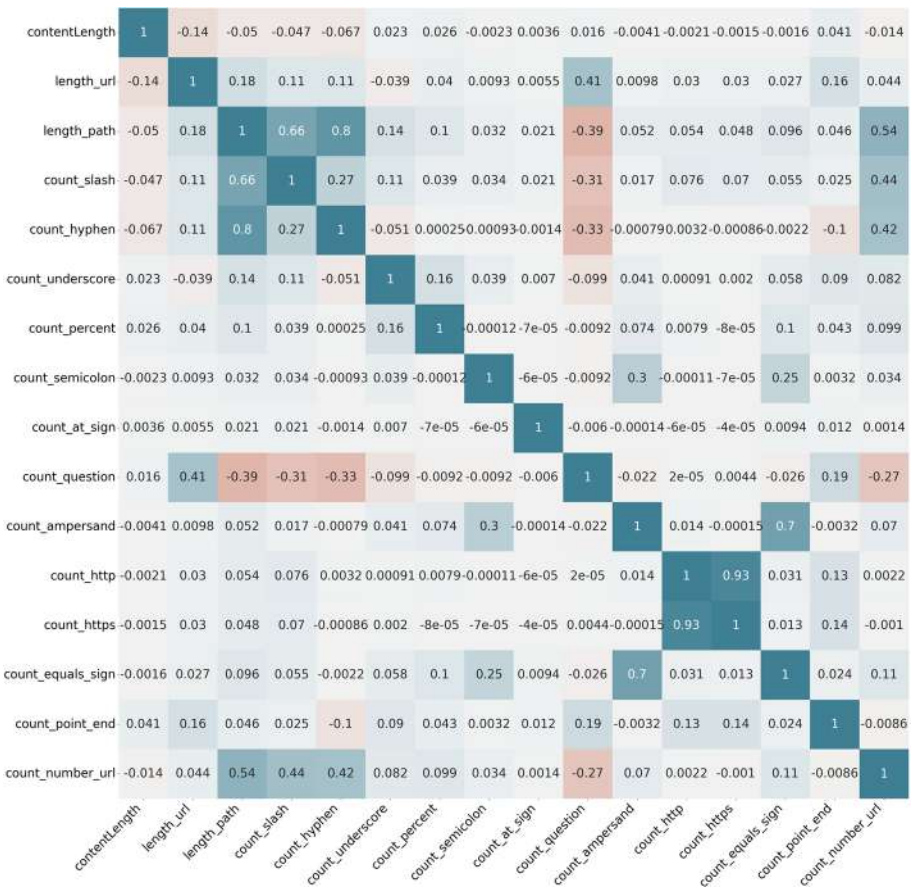
Besides, in order to study the correlation, the Pearson method is applied to the independent variables that do not take classes (see Fig. 4).

As can be seen in Fig. 5, there is a high correlation among some variables. Therefore, it was decided to remove the count_http and count_hyphen variables. However, after doing so, the dataset was saved with all variables, including

	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 19:36:29 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

Fig. 3 First five columns of the used dataset

Fig. 4 Dataset correlation



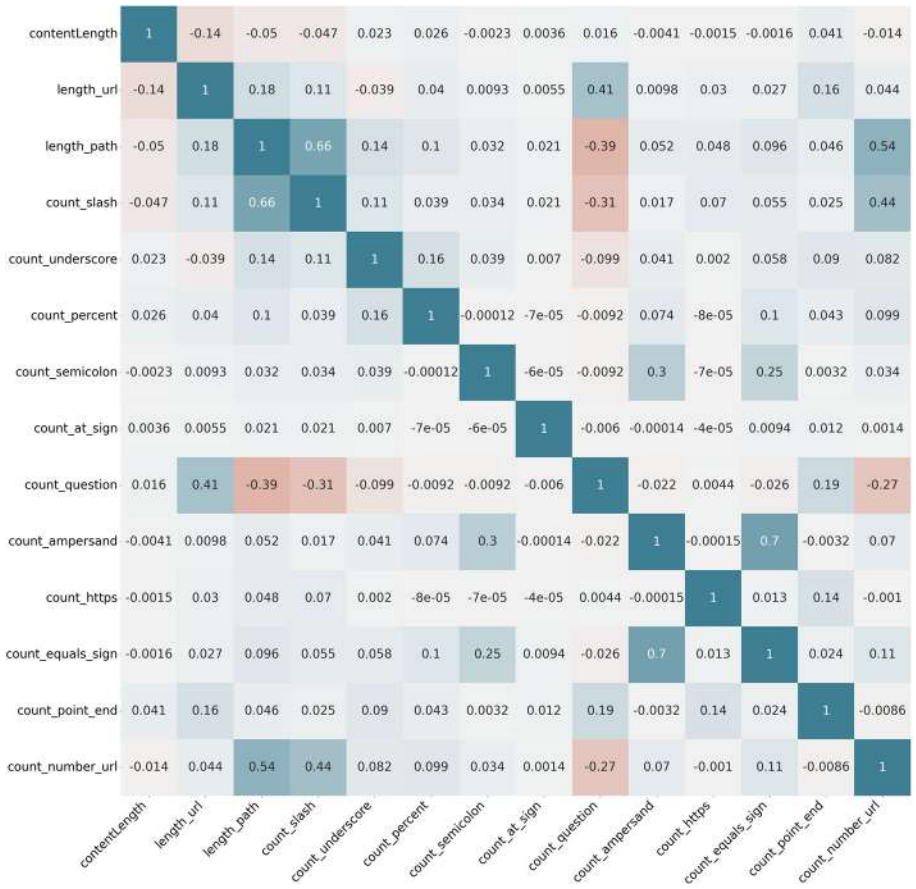


Fig. 5 Correlation of the dataset for the other algorithms

count_http and count_hyphen, because they were utilized in certain algorithms.

8 Application of classical ML algorithms

The first step to apply classical ML algorithms is to check the goodness of the data without the URLs and without the newly created variables. Using the ML decision tree algorithm, 145,532 random data points from the dataset were used for training. In total, the training set comprised 6,965 fraudulent URLs. With this, an accuracy of 87.77% was achieved (see Fig. 6).

In Fig. 6 it can be seen that from the training set only 1,343 fraudulent URLs out of 1,653, that is, only 81.25%, were correctly identified. On the other hand, of the non-fraudulent URLs, 88.08% were identified. This leads to the conclusion that if the URLs are analyzed, perhaps more precision could be achieved since right now 310 fraudulent URLs are not being classified well.

Thus, after analyzing only the URLs, a decision tree was created by eliminating the variables *serverType*, *compromissionType*, *poweredBy* and *contentType*. New variables were then defined, such as the length of the URL, counting how many times '/' appears in the URL path, and

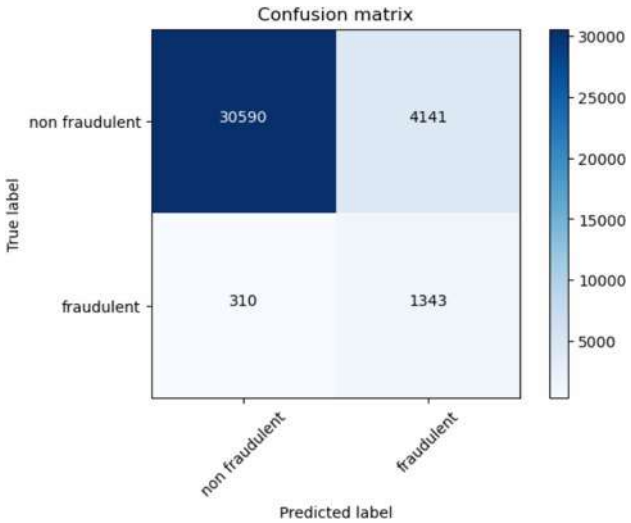


Fig. 6 Resulting confusion matrix

more such variables. Figure 4 shows all the new definitions that were created. In this way, the dataset has the same number of rows, and 16 columns counting the dependent variable.

The addition of the new variables returned an accuracy of 93,18%. In Fig. 7 the new confusion matrix is shown.

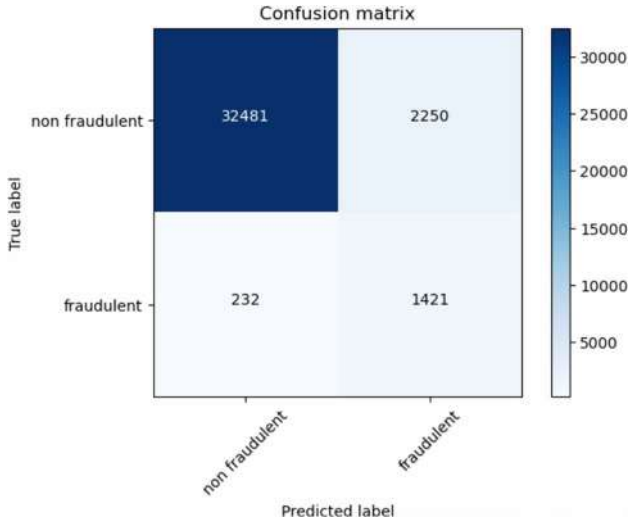


Fig. 7 New confusion matrix

In Fig. 7 it can be seen that this training set identified more data correctly than the previous one. Of those that are fraudulent, it identified either 85.96% while of the non-fraudulent URLs it identified either 93.52%. With this program it was concluded that 232 fraudulent URLs are not well classified, which leads to the hypothesis of whether joining both programs could obtain more specificity.

Once this was done, the next step was to prepare the data. To do this, it was decided to leave the definitions created to extract the information from the URLs and keep the independent variables *serverType*, *compromissionType*, *poweredBy*, *contentType* and *contentLength*.

In addition, the normalization of the data was used in all the algorithms used in ML and Deep Learning except in the decision tree. The algorithms used to finally analyze the dataset were:

- Logistic regression
- Decision tree
- Support Vector Machine
- Neural networks

In particular, logistic regression, decision trees and Support Vector Machine were chosen since they have been successfully applied in other works like [7, 13, 14]. Another example of a study that uses these algorithms and also uses neural networks is the work [33].

For the decision tree and the neural networks, the dataset that can be seen in Fig. 4 was used.

For the other ML algorithms, the dataset that can be seen in Fig. 5 was used.

8.1 Results

The results obtained with all the methods are compared below.

In the SVM algorithm, different kernels were used: Sigmoid, RBF and Poly. In particular, the RBF kernel used in the work [34] was chosen. Tables 2 and 3 show the comparison of the applied methods, except the neural networks. In those Tables, the results obtained with Support Vector Clustering (SVC) are shown. SVC is a method similar to SVM, which also builds on kernel functions but is appropriate for unsupervised learning.

In the case of neural networks, three different ones were built. In the first neural network a hidden layer was created in which the Sigmoid activation function is used. In addition, the 'binary_crossentropy' loss function, the 'adam' optimizer and the 'binary_accuracy' metric were used. Its confusion matrix is shown in Fig. 8.

The second neural network was created with three hidden layers, all of which used the Sigmoid activation function. Also the same loss function and the same metric was used as in the first neural network, the optimizer that was used was 'rmsprop'. Figure 9 shows the confusion matrix of this neural network.

In the third neural network, three hidden layers were created. In the first two layers the 'relu' activation function was used, while in the last layer the Sigmoid activation function was used. The same loss function, optimizer, and metric were also used as in the first neural network. Figure 10 shows the corresponding confusion matrix.

Table 4 shows a summary of the confusion matrices of the three neural networks.

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

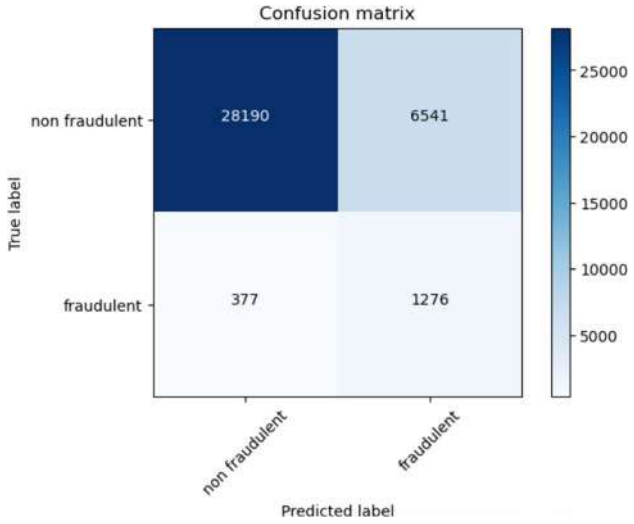


Fig. 8 First neural network

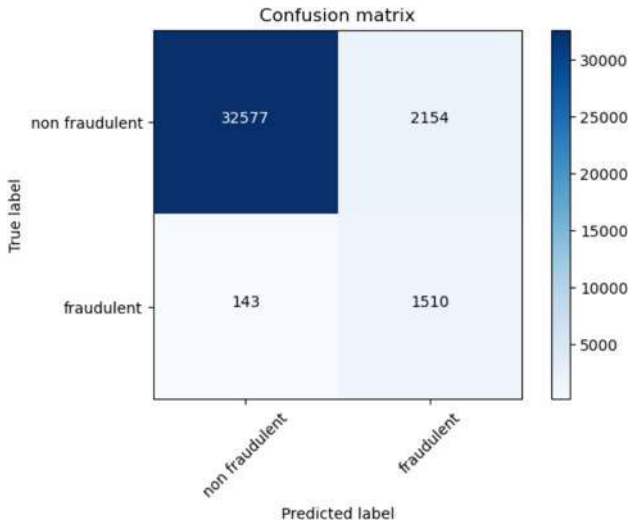


Fig. 9 Second neural network

Once the comparisons were made, it was reconsidered whether the models used are overfitting or underfitting. To do that, the *binary_accuracy* and *val_binary_accuracy* were analysed. Fig 11 shows the representation of both variables. A model is said to be underfit when a low precision is obtained and the validation precision is also low. In the case of overfitting, high precision is obtained in training, but low precision is obtained in validation.

As seen in Fig. 11, the representation of the variable *val_binary_accuracy* and the variable *binary_accuracy* are almost at the same level.

Table 5 shows the area under the ROC curve.

From what can be seen, if it is classified by Area, first would be the SVC (RBF), then the third neural network and finally the decision tree.

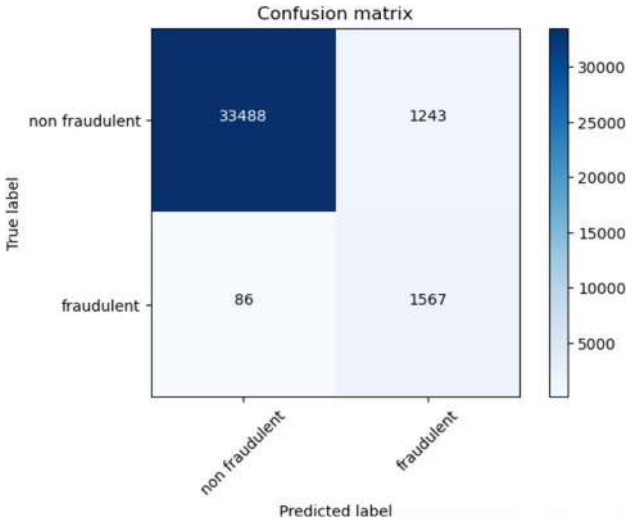


Fig. 10 Third neural network

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

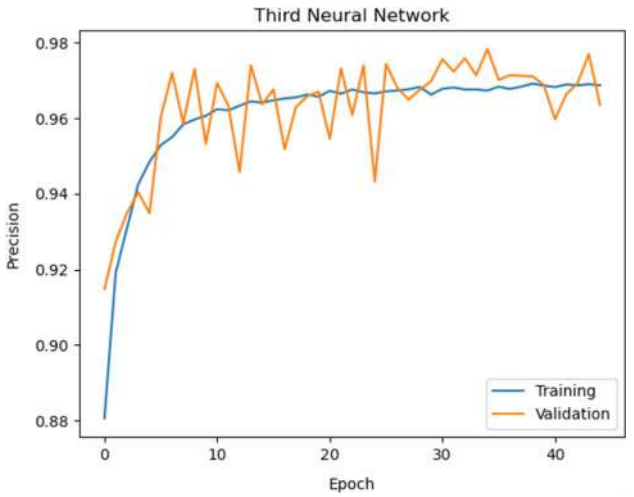


Fig. 11 Representation of *binary_accuracy* and *val_binary_accuracy* variables

By using a public dataset, it is possible to compare the results obtained with it by other authors. For instance, [13] shows a more complex analysis that takes into account the information on the web page. For example, one of their variables within the dataset was *TCP_conversation_exchange*, counting the number of packets between the honeypot and the website over the TCP protocol. On the contrary, the

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

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

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 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

present work, when processing the data, focuses more on the information provided by the URL itself. In addition, that work compares according to accuracy while here the analysis is based on recall. Thus, to compare data, only their second Table can be used as here a different approach to the data has been followed. Since only two methods coincide between both works, those two methods are compared below.

Table 6 shows the accuracies obtained by both programs, where Accuracy (1) denotes the accuracy obtained in the program described in this paper, while Accuracy (2) denotes the accuracy obtained in [13]. It can be seen that the accuracy obtained there in the logistic regression is better than the one obtained here, probably due to the data processing they used.

8.2 Evaluation

To complete the evaluation, the trained models were tested with a different dataset. Those trained data models were then evaluated against a new dataset in order to find out how good the model is and whether it serves to generalize. This new dataset was obtained from https://github.com/ESDAUNG/PhishDataset/blob/main/data_bal%20-%2020000.xlsx.

This new dataset only contains phishing URLs and the *compromissionType*. Therefore, it was decided to delete the rest of the columns of the dataset: *serverType*, *contentLength*, etc. By retraining three of the models with the original dataset, Table 7 was obtained.

Table 7 shows the results obtained from the prediction of the new dataset with some of the models mentioned in Tables 3 and 4, being trained with the original dataset and eliminating the columns mentioned above.

Table 8 shows the results when the models try to predict the fraudulent URLs from the new dataset.

Table 8 shows that the program, when trying to predict a new dataset, the maximum that it is capable of identifying of fraudulent URLs is 87%. This percentage is probably due to the fact that the program found new fraudulent URLS that it did not know were fraudulent because there are no similar ones in its database.

9 Application of quantum machine learning

In order to study the possible practical usefulness of ML algorithms linked to quantum computing, hereinafter called Quantum Machine Learning, an analysis of a quantum approach to the solution of the analyzed problem has been carried out to measure the degree of efficiency that this new paradigm can provide through the use of quantum neural networks.

9.1 QML algorithms

In order to apply QML algorithms, four possible approaches can be distinguished depending on how the type of data to be used and the hardware on which the algorithms are executed are combined.

- CC: Classical data with ML algorithms running on Classic hardware
- CQ: Classical data with ML algorithms running on Quantum hardware
- QC: Quantum data with ML algorithms running on Classical hardware

- QQ: Quantum data with ML algorithms running on Quantum hardware

In this work, CQ is mainly used, starting from classical data, encoded by the corresponding algorithm in quantum information, to subsequently perform the simulations on classical hardware. All the work has been developed in Python, using the IBM quantum computing framework called Qiskit. Specifically, a Variational Quantum Classifier has been used, requiring the following steps prior to training [35]:

- Data coding, process that consists of transferring the original data to qubits and is done through feature mapping, choosing different algorithms for it:
  - ZZFeatureMap
  - ZFeatureMap
  - PauliFeatureMap
- Application of a parameterized quantum circuit or Ansatz, quantum circuit whose main characteristic is that it has a set of adjustable weights that must minimize an objective function. The chosen Ansatz have been:
  - RealAmplitudes
  - EfficientSU2
  - ExcitationngPreserving
- Choice of optimization algorithm, with a function equivalent to that of a classic Deep Learning model, selecting three local optimizers (a function that tries to locate an optimal value within the neighboring set of a candidate solution):
  - COBYLA (Constrained Optimization By Linear Approximation optimizer)
  - GradientDescent (Gradient Descent minimization routine)
  - SLSQP (Sequential Least Squares Programming optimizer)

9.2 Adaptation of the dataset

Since the application of QML models requires datasets where all their characteristics are numeric, it has been necessary to codify the categorical variables. For this, several alternatives were considered, such as:

- Ordinal encoding: suitable for establishing a hierarchical order between the values of the variable. However, in the analyzed case, the values of the categorical variables do not correspond to this casuistry.
- One-hot encoding: a vector of numerical characteristics is linked to each category in such a way that the vector will have as many components as possible categories the variable has, all of them being "0" except for the position that corresponds to the category of that observation, which will contain a 1. The drawback in this case is that some of the analyzed features have more than 100 possible categories, which would significantly increase the size of the dataset.
- Binary coding: hybrid method combining the two previous ones so that first, ordinal encoding is applied and then each integer is converted into binary and as many columns are generated as there are digits in the resulting binary. This method is more optimal than the “one hot” encoding, but it still complicates the dataset.
- Hashing: a hashing function is required that transforms each category of the variable into an integer value within a certain range. However, collisions (different inputs generating the same output) must be controlled because they can affect the quality of the dataset.

After evaluating the impact on the dataset and on the model (SVM/Q-VQC), ordinal encoding was chosen. In particular, to choose the algorithm for converting text variables to numerical variables, different advantages and disadvantages of each algorithm were analysed, opting for ordinal encoding mainly due to its simplicity, efficiency and ease of implementation, which is fundamental in quantum processes. Besides, unlike other methods such as one-hot encoding, ordinal encoding does not increase the dimensionality of the dataset, which is crucial for current quantum models that are very limited in the number of usable qubits. Moreover, its use generally leads to less information loss (e.g. by preserving order) compared to algorithms such as random coding. Therefore, although it also has some disadvantages (possible artificial numerical relationship between categories that might not be inherently ordered), the advantages identified and the preliminary tests we were able to do, tipped the balance towards this option.

However, one of the future lines of work involves comparing different encoding methods and evaluating their impact on the final results. In particular, it is considered especially interesting to analyse the use of one-hot in combination with dimensionality reduction algorithms such as PCA (Principal Component Analysis), in cases where the application of one-hot increases the number of variables to be handled excessively.

The selected fields and the processes carried out on the original dataset are detailed below to allow the application of QML algorithms considering that all the characteristics must be numeric.

- *url*: stores the total number of characters in the URL.
- *compromissionType*: variable that indicates if the website is compromised by phishing, defacement, or is normal.

It is transformed by assigning a numerical value to each type (1, 2 or 0 respectively).
- *isHiddenFraudulent*: changed to 1 for fraudulent URLs or 0 for benign ones.
- *contentLength*: already a variable that takes integer values.
- *serverType*: numeric value assigned to each server type.
- *poweredBy*: numeric value assigned for each application platform underlying the web server.
- *contentType*: numeric value assigned for each encoding type.

9.3 Application of VQC

Given the volume of the dataset and the required processing times, a first approximation was made with a reduced (200 observations), balanced (100 malicious URLs and 100 non-malicious ones) dataset. Thus, the difference between working in the classical SVM model with the full dataset and the reduced version was analyzed. First, by applying the classic SVM model to the complete dataset, the following values were obtained:

- *Classical SVC on the training dataset:* 0.97
- *Classical SVC on the test dataset:* 0.97

Then, applying it to the simplified dataset, the following was obtained:

- *Classical SVC on the training dataset:* 0.91
- *Classical SVC on the test dataset:* 0.97

The confusion matrix associated with the simplified dataset is shown in Fig. 12.

To identify the most optimal parameterization in speed and efficiency, numerous combinations were made depending on the algorithms for the generation of the feature map, the Ansatz algorithms and the optimizers to apply based on the criteria:

- All calculations were developed in Python using Jupyter Notebook with libraries pandas, numpy, matplotlib, seaborn, sklearn and the set of IBM qiskit libraries.
- The results obtained on a MacBook Pro (M2 Pro and 16 Gb of RAM) are shown, but the calculations were also carried out in parallel on a PC (Intel i7 4K Ghz and 32 Gb of RAM) to compare the time (in seconds) it took to train the model on each type of hardware (TPC or TMAC).
- For feature map generation, the algorithms ZZFeatureMap, ZfeatureMap and PauliFeatureMap were selected.

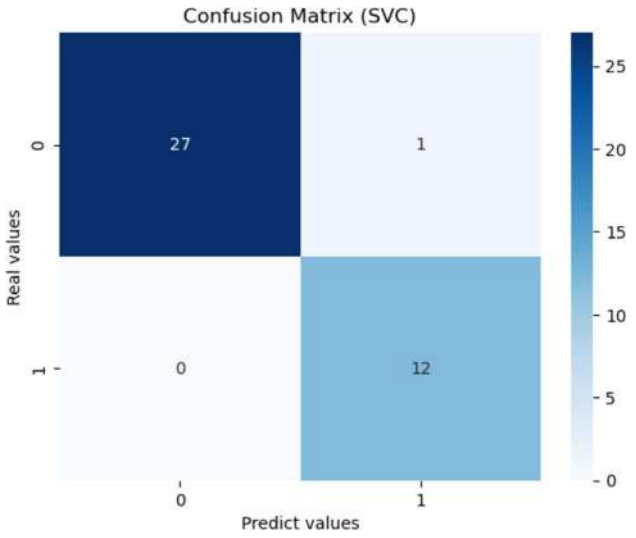


Fig. 12 Confusion matrix SVC

- Regarding Ansatz, the RealAmplitudes (RealAmp), EfficientU2 and ExcitingPreserving (ExcitPreserving) algorithms were used.
- Regarding optimizers, COBYLA, GradientDescent and SLSQP were used, all with 20 iterations.
- To measure the performance of the model, the precision obtained through vqc.score was used for the training (TrainS) and test (TestS) data.

Tables 9, 10 and 11 show a summary of the results obtained by applying the VQC model on the simplified dataset.

The results obtained with the following combination of parameters stand out:

- Generation of the feature map *ZFeatureMap* shown in Fig. 13.
- Ansatz: *RealAmplitudes* (2 repetitions)
- Optimizer: *SLSQP* with 20 iterations

The Ansatz representation in Fig. 14 shows the 17 parameters used, $\theta[1]...\theta[17]$ and the two repetitions indicated in the Python code.

In this case, the results obtained in the simulation of the VQC model are shown in Fig. 15:
Quantum VQC on the training dataset: 0.89
Quantum VQC on the test dataset: 0.97

9.4 Evaluation

From the analysis of the results some interesting conclusions can be drawn:

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 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 PauliFeatureMap

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

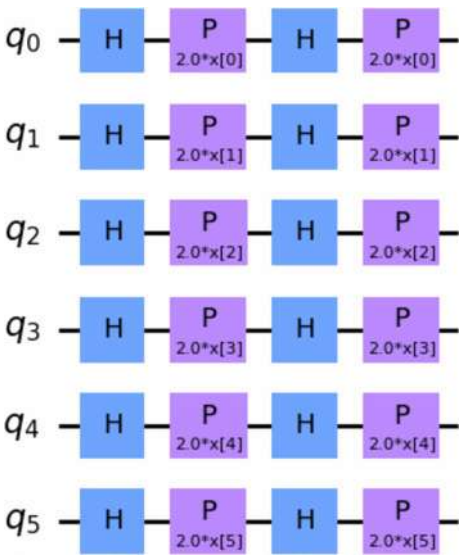


Fig. 13 Feature mapping with ZFeatureMap

- Regarding the combination of parameters, it is observed that the choice of the optimizer is decisive, much more than the Ansatz and the feature mapping algorithm. The SLSQP optimizer stands out from the others despite the low iterations used.
- In training speed, the COBYLA optimizer is by far the fastest since it does not use gradient descent. It is an option to consider when the time factor is decisive, using the combination of ZFeatureMap, RealAmplitudes and COBYLA.
- Although it may seem that the GradientDescent optimizer should be discarded due to the poor results obtained, the low number of used iterations must be considered, requiring in this case a minimum of 100 iterations to improve the results.

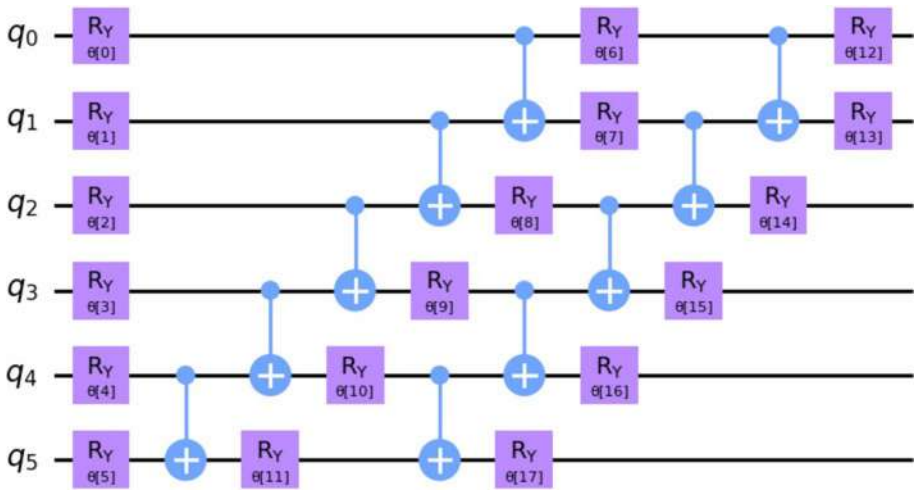


Fig. 14 Representation Ansatz RealAmplitudes

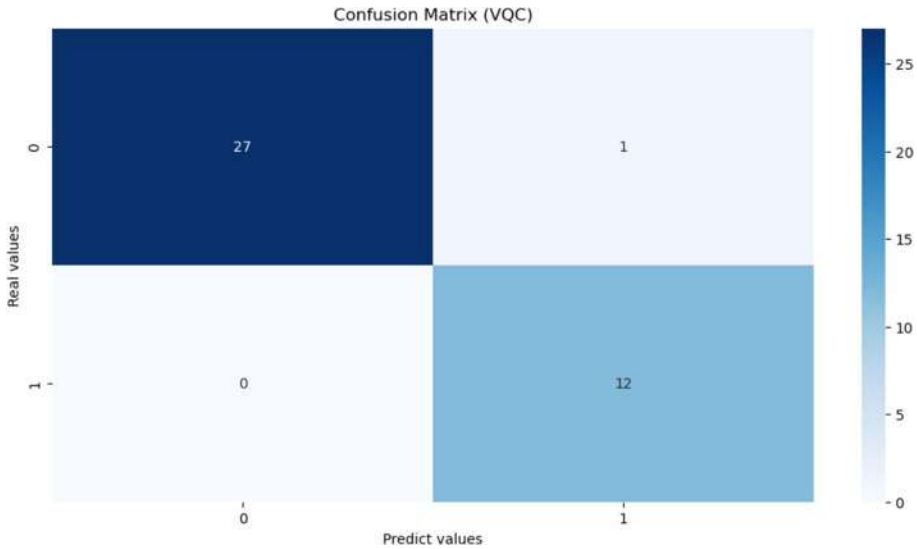


Fig. 15 Confusion matrix VQC

- Regarding the algorithm for mapping features, the ZFeatureMap stands out, with which the best results have been obtained globally.
- In the execution of classic hardware, the performance of the M2 PRO processor stands out, which obtains a clear advantage in training times in almost all cases.

10 Conclusions and future work

In this work, the cybersecurity problem of detecting fraudulent URLs has been analysed from different perspectives, using machine learning in both its traditional version and its quantum version. The main goal has been to explore the possibilities of quantum computing applied to machine learning in the context of cybersecurity.

Several conclusions have been drawn from the analysis, both about the dataset itself and about the usefulness of QML in cybersecurity.

On the one hand, regarding the dataset, during the study it was concluded that the first used dataset was unbalanced, which helped to identify the most optimal algorithms and processes to optimise the results. In this way, this work has highlighted the importance of this prior analysis of the data, before starting to apply different algorithms in a generalised way.

On the other hand, the study concluded that the typology of the analysed problem also conditions the focus on which results are the most interesting in practice. For example, this is the case of the confusion matrix, where working on a cybersecurity problem involves paying special attention to False Negatives because they mean that malicious URLs are being taken as valid and can generate significant service

or economic losses. That is why the objective in this case should be to minimise False Positives as much as possible, to see how Accuracy decreases but Recall increases. Specifically, since the main goal is to achieve the lowest number of False Positives, but without greatly increasing the number of False Negatives, the final conclusion is that the best measure to compare the models is the F1-score.

Another conclusion in the field of classical machine learning models is that one of the three neural networks proposed in this work was clearly identified as the most optimal for the analysed dataset. The next best model was the Support Vector Machine with RBF kernel, and the third best model was the Support Vector Machine with a Poly kernel.

An important objective of this work has been to evaluate the usefulness of QML models in cybersecurity, in an attempt to identify the most appropriate algorithm combinations used in this context. In this sense, several interesting conclusions were obtained about the relevance of the optimisers and the feature mapping algorithm, identifying certain combinations that produce results similar to classical models with a simplified dataset.

When comparing the results obtained with ML and QML, it is clear that classic models produce better results, especially considering that they do so by working on the entire dataset. This conclusion is natural considering that these models have been studied and optimised exhaustively for years, and so the amount of academic literature on the topic is extensive. However, it is also clear that with QML, despite being poorly optimised and relatively recent in its application, very promising results have been achieved in this work, for example with the combination of ZFeatureMap, RealAmp and SLSQP.

From the research carried out on the existing literature, it is directly clear that the application of QML is a very recent field and consequently its results are still very theoretical. In particular, its use in the field of cybersecurity is even more restricted, especially due to the current limitations of quantum hardware. Thus, in this work numerous situations have emerged that pave the way for future studies, such as:

- Shortage of up-to-date cybersecurity datasets suitable for quantum computing work.
- Encoding alphanumeric variables to purely numeric values, striking a balance between information loss and limiting the excessive growth of variables to be processed in QML algorithms.
- Optimal parameterisations of QML for cybersecurity.
- Application of quantum hardware.
- Frameworks to be used in the context of QML, different from the Qiskit libraries, such as QML PennyLane, the Cirq libraries (Google), or Microsoft Quantum Development Kit (QDK).

The aforementioned lines point the directions of several problems detected during this study, as well as new research focuses that give continuity to this work.

As a final conclusion, this work opens the door to numerous future studies on the optimal parameters for the use of QML and how to integrate these algorithms in the analysis of different cybersecurity problems, thus incorporating new possibilities for the early detection of fraudulent actions.

Acknowledgements This research has been partially supported by the Cybersecurity Chair of the University of La Laguna and the project PID2022-138933OB-I00: ATQUE funded by MCIN/AEI/10.13039/501100011033/FEDER, EU.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

References

1. ENISA: ENISA threat landscape 2023. https://www.enisa.europa.eu/publications/enisa-threat-landscape-2023
2. Fortinet: What is URL phishing? (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.
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.
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.
16. Kalinin, M., & Krundyshev, V. (2023). Security intrusion detection using quantum machine learning techniques. *Journal of Computer Virology and Hacking Techniques, 9*, 125–136.
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.
19. Hosmer, D. W., Jr., Lemeshow, S., & Sturdivant, R. X. (2013). *Applied logistic regression*. Wiley.
20. Quinlan, J. R. (2014). *C4.5: programs for machine learning*. Elsevier.
21. Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning, 20*, 273–297.
22. Cristianini, N., & Ricci, E. (2008). *Support vector machines*. Springer.
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.
24. Moldwin, T., & Segev, I. (2020). Perceptron learning and classification in a modeled cortical pyramidal cell. *Frontiers in Computational Neuroscience, 14*, 33.
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.
27. Raschka, S., & Mirjalili, V. (2019). *Python machine learning: Machine learning and deep learning with python, scikit-learn, and tensorflow 2*. Packt Publishing Ltd.
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), 3692–3695.
32. Jasper Snoek, R.P.A. (2012). Hugo Larochelle: Practical bayesian optimization of machine learning algorithms. In: Advances in Neural Information Processing Systems, vol. 25.
33. Abu-Nimeh, S., Nappa, D., Wang, X., & Nair, S. (2007). A comparison of machine learning techniques for phishing detection. In: Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit, pp. 60–69.
34. Li, T., Kou, G., & Peng, Y. (2020). Improving malicious URLs detection via feature engineering: Linear and nonlinear space transformation methods. *Information Systems, 91*, 101494.
35. Qiskit.org: Quantum machine learning course. https://learn.qiskit.org/course/machine-learning

Nuria Reyes-Dorta received her Bachelor's degree in Mathematics from the University of La Laguna, Spain, where she is currently finishing the Master's degree in Cybersecurity and Data Intelligence. She is focusing her main research work on the applications of Machine Learning and Quantum Machine Learning in the field of cybersecurity.

Pino Caballero-Gil completed her B.Sc. and Ph.D. in Mathematics at the University of La Laguna in Spain, where she currently holds the position of full professor of Computer Science and Artificial Intelligence at the Department of Computer Engineering and Systems. Her area of expertise includes stream ciphers, cryptographic protocols, security of wireless networks and mobile applications, and quantum-resistant cryptography. She is the leader of the CryptULL research group on Cryptology, which is dedicated to the development of cutting-edge projects in the field. She has made significant contributions to the academic community through numerous refereed conference and journal papers, as well as books.

Carlos Rosa-Remedios received his B.Sc. in Mathematics from the University of La Laguna and is accredited as Director of Security by the Ministry of the Interior. He currently combines his work as head of technology at 112 in the Canary Islands with his Ph.D. studies at the University of La Laguna and teaching at the Faculty of Computer Engineering and in the Master's Degree in Cybersecurity and Data Intelligence at the same university. He is a member of the CryptULL research group, a research group in Cryptology, focusing his work on the study of the applications of Machine Learning algorithms and Quantum Computing in the field of Cybersecurity and Critical Infrastructures.