IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 18 April 2025, accepted 1 May 2025, date of publication 6 May 2025, date of current version 16 May 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3567352

RESEARCH ARTICLE

Comparative Analysis of a Quantum SVM With an Optimized Kernel Versus Classical SVMs

MATHEUS CAMMAROSANO HIDALGO
Escola Politécnica da Universidade de São Paulo, São Paulo 05508-010, Brazil
e-mail: matheuschidalgo@gmail.com

This work was supported in part by the Brazilian Federal Agency and in part by the Evaluation of Graduate Education (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – CAPES, ROR identifier: 00x0ma614).

ABSTRACT Support Vector Machine (SVM) is a widely used algorithm for classification, valued for its flexibility with kernels that effectively handle non-linear problems and high-dimensional data. Businesses across industries face challenges in improving customer retention and reducing churn, making predictive models essential for identifying at-risk customers and enhancing revenue. This paper investigates an optimized quantum embedding kernel for SVM (Quantum SVM - QSVM) applied to a public bank customer dataset, featuring variables on customer relationships and churn indicators. While focused on the financial sector, the methodology is broadly applicable for reducing churn and boosting revenue across industries. QSVM performance is compared to SVMs with established kernels, including Radial Basis Function (RBF), linear, polynomial, and sigmoid. Experiments varied the number of variables from two to seven to evaluate their impact on model performance and kernel behavior. The experiments were conducted on quantum simulators, which faced scalability challenges addressed using reduced datasets. Even so, this study sheds light on the potential of QSVMs to effectively manage increasing numbers of variables in predictive models, offering valuable insights into their capability to handle complex, high-dimensional data and their applicability in real-world scenarios.

INDEX TERMS Churn prevention, quantum machine learning, quantum support vector machines.

I. INTRODUCTION

Quantum computing is an emerging field that leverages the principles of quantum mechanics to process information in fundamentally new ways. Unlike classical computers, which use bits as the smallest unit of information (represented as 0 or 1), quantum computers use quantum bits, or qubits, which can exist in a superposition of states, enabling them to perform multiple calculations simultaneously [1]. This unique capability allows quantum computers to tackle problems that are infeasible for classical systems, such as optimizing complex systems, simulating molecular interactions, and enhancing machine learning algorithms. By harnessing phenomena like entanglement and interference, quantum computing promises to revolutionize industries ranging from cryptography to pharmaceuticals, although it remains in its nascent stages [2], with challenges like error correction and scalability still to be addressed.

The associate editor coordinating the review of this manuscript and approving it for publication was Muammar Muhammad Kabir.

A prominent area of study in quantum computing is quantum speedup [3], where researchers investigate types of algorithms in which quantum computers perform faster than their classical counterparts. The most notable example of quantum speedup is Shor's algorithm [4], which is exponentially faster than any classical algorithm for finding the prime factors of an integer [5]. Another example is Simon's algorithm [6], which also achieves exponential speedup compared to any classical implementation.

Besides, Quantum Machine Learning (QML) involves applying the principles of quantum computing to machine learning problems. This application is promising, as machine learning models are inherently probabilistic, a characteristic shared by quantum mechanics. Therefore, it is reasonable to hypothesize that quantum computers could outperform classical computers in certain machine learning tasks [7].

Thus, the inherent probabilistic nature of quantum computing, combined with potential speedups, provides compelling reasons to research QML. One important area within QML is Variational Quantum Algorithms (VQA), which involves leveraging both classical and quantum computers to develop machine learning models [8]. In this approach, a parameterized model is executed on a quantum computer, while its parameters are optimized on a classical computer. Several methods exist for applying VQAs to QML problems [9], one of which is designing an optimized quantum kernel through a VQA for an SVM [10].

This approach is particularly interesting because SVM is one of the classical machine learning techniques that can integrate quantum computing concepts directly through the use of quantum kernels. This is promising, as quantum vector spaces are inherently high-dimensional: for example, a system with n qubits can represent 2^n states. This enables the creation of highly complex decision boundaries, potentially making SVMs more powerful. Such a configuration is known as a Quantum Support Vector Machine (QSVM) [10].

A significant number of works are approaching this method, such as [11], [12], [13], [14], [15] and many others.

In [16], a quantum kernel optimized through a VQA is proposed for classification problems, specifically in the form of a Variational Quantum Classifier (VQC) [17]. In this approach, the kernel is parameterized to reduce the distance between data points within the same class while increasing the separation between samples of different classes, thereby simplifying the decision boundary design. This kernel is applied to an SVM with promising results.

This work builds on a similar concept for constructing a quantum kernel as proposed in [16], but with a different number of parameters for kernel optimization.

The QSVM is compared with SVMs that utilize widely used kernels such as Radial Basis Function (RBF), linear, polynomial, and sigmoid, in a bank customer churn prediction problem. The algorithms are trained across various scenarios, starting with two variables from the dataset and progressively increasing to seven. The objective of this analysis is to evaluate how the performance of the QSVM compares to that of classical SVMs as the number of variables increases. This case study is relevant not only to the financial sector but also to businesses across various industries, as shown in [18], [19], and [20]. Furthermore, churn prediction is widely used to identify customers at high risk of leaving, enabling companies to implement retention actions, such as reduced fees, Customer Relationship Management (CRM) campaigns, and other strategies that can increase revenue and enhance customer lifetime value.

The experiments were conducted on quantum simulators using the Pennylane library in Python. Since the current era of quantum computing is classified as the Noisy Intermediate-Scale Quantum (NISQ) era, simulations with noise were intended. However, preliminary tests indicated that conducting such simulations was unfeasible due to computational limitations. Nevertheless, this paper addresses certain scenarios involving error probabilities for the proposed quantum circuit to evaluate the susceptibility of this approach to noise.

Therefore, this work focuses on noise-free simulations as a first step of the research. However, future studies should include experiments on real quantum computers to assess the impact of noise on the results.

This paper has the following structure: Section II presents the concepts used in this work, Section III describes the quantum kernel design, Section IV presents data and the experiments methodology, Section V shows the results and in Conclusion the main findings are summarized.

II. CONCEPTS

This section introduces the key concepts used in this paper, beginning with a brief overview of kernels, followed by an explanation of quantum embedding kernels, and concluding with an introduction to VQA.

A. KERNEL METHOD

In SVMs, specifically for classification problems, the objective is to find the optimal separator between classes by maximizing the margin. This separator is defined by a vector v, which is orthogonal to the separating hyperplane. For the linear case, the decision function is given by (1).

$$D(x) = \operatorname{sgn}(\langle v, x\rangle + a)) \tag{1}$$

where x is a data point, a is the intercept of the decision boundary. Thus, points where $\langle v, x\rangle + a > 0$ belong to one class, otherwise they belong to the other class.

Equation (1) is a good decision boundary for linear problems, where a line or a plane serves as the optimal separator. However, for nonlinear cases, this solution might not provide good results. To address this, the concept of kernels is introduced, which is a mathematical abstraction and a generalization of (1). Kernels enable the proper separation of data in more complex cases, allowing SVMs to handle nonlinear decision boundaries effectively.

$$D(x) = \operatorname{sgn}(\langle v', \Phi(x)\rangle + a) \tag{2}$$

where, in (2), Φ is a feature map that maps x into a high dimensional vector space, resulting in $\Phi(x)$. Furthermore, v' is the orthogonal vector in the high dimensional space. Additionally, v' can be expressed as follows, according to the representer theorem [21].

$$v' = \sum_i \alpha_i \Phi(x_i) \tag{3}$$

The main point of (3) is that the orthogonal vector to the separator can be represented as a linear combination of the data points in the feature map. This simplifies (2), resulting in the form given by (4).

$$D(x) = \operatorname{sgn}\left(\sum_i \alpha_i \langle \Phi(x_i), \Phi(x)\rangle + a\right) \tag{4}$$

The kernel function is given by (5).

$$k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j)\rangle^2 \tag{5}$$

where x_i and x_j are two data points. It is also possible to infer that (5) represents a similarity function between two points. This concept can be extended to a kernel matrix, which defines the relationships between each pair of data points.

$$K(x) = \begin{bmatrix} k(x_1, x_1) & \dots & k(x_1, x_n) \\ \vdots & \ddots & \vdots \\ k(x_n, x_1) & \dots & k(x_n, x_n) \end{bmatrix} \tag{6}$$

The key advantage of using a kernel is that, despite working in a high-dimensional vector space, the use of inner products allows the calculations to be performed in the original, lower-dimensional vector space. A good mapping Φ enables better separation between classes through the kernel.

Some examples of common kernels, which are also used in this work as benchmarks for comparison with the quantum kernel are given in (7) [22], [23].

$$k(x_i, x_j) = \begin{cases} \langle x_i, x_j \rangle, & \text{linear} \\ \left(\langle x_i, x_j \rangle + 1 \right)^p, & \text{polynomial} \\ \exp\left(- \left(\dfrac{||x_i - x_j||^2}{2\sigma^2} \right) \right), & \text{RBF} \\ \tanh(\gamma \langle x_i, x_j \rangle + a), & \text{sigmoid} \end{cases} \tag{7}$$

B. QUANTUM EMBEDDING KERNEL

The quantum embedding kernel is a special case of what was presented in the previous subsection, where the feature map is represented as a quantum state.

$$|\Phi(x)\rangle = U(x)|0\rangle \tag{8}$$

where, in (8), $U(x)$ represents a unitary operator, which is a sequence of quantum components that transform the data point x into quantum data through use of quantum gates, $|0\rangle$ is one of the base states in quantum computing and $|\Phi(x)\rangle$ is the resulting quantum state of the feature map. An interesting aspect here is that the data point x is not being converted into an input quantum state $|x\rangle$, as typically done in VQCs [24]. Instead, x is used as a parameter of the unitary.

In this work, the number of qubits used in the quantum circuit is equal to the number of features in the model. Thus, the quantum kernel has from two to seven qubits depending on the number of used variables in the experiment. Consequently, the original vector space has n dimensions, and the feature map transforms the data into a 2^n-dimensional Hilbert vector space that varies from 4 to 128 states. Therefore, the quantum feature map naturally resides in a high-dimensional vector space as much as the number of variables, therefore qubits, increase.

Next, it is necessary to combine the kernel function in (5) with the quantum feature map in (8) to generate a quantum kernel function.

$$k(x_i, x_j) = \left| \langle \Phi(x_i) | \Phi(x_j) \rangle \right|^2 \tag{9}$$

Note that (9) is analogous to (5), but there is an important detail in (9) that is intrinsic to quantum systems: $\Phi(x_j)$ is represented as a column vector (ket) and $\Phi(x_i)$ is represented as a row vector (bra). The inner product between two quantum states is a bra-ket, as shown in (9). However, this expression must be written in a form that allows for the calculation of the resulting kernel value, which is given by (10).

$$k(x_i, x_j) = \left| \langle 0 \left| U^\dagger(x_i) U(x_j) \right| 0 \rangle \right|^2 \tag{10}$$

where $U^\dagger$ is the adjoint of the unitary U.

The representation of (10) in the form of a quantum circuit is shown in Fig. 1, which illustrates a four-qubit configuration. In this work, configurations ranging from two to seven qubits are utilized. For cases involving more or less qubits, the circuit in Fig. 1 can be adjusted by increasing or decreasing the number of qubits in the diagram.

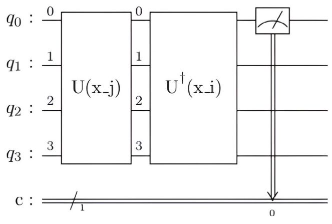

FIGURE 1. Quantum circuit representation of the kernel function, adapted from [16].

In Fig.1, the result of kernel function is obtained by measuring the output of the first qubit in the circuit. The quantum kernel matrix is a particular case of (6), which is obtained by applying (10) to the kernel matrix in (6).

C. VARIATIONAL QUANTUM ALGORITHMS

The current stage of quantum computing presents a number of challenges to developers and scientists, as the number of available qubits is limited and the devices are relatively noisy (Noisy Intermediate-Scale Quantum era - NISQ).

VQAs are a successful use case in Quantum Machine Learning during the NISQ era. They involve using a classical computer to train a parameterized quantum circuit [8].

The two key components of the VQA are the ansatz, which is created on the quantum computer, with parameters w that are adjusted through an optimization process on the classical computer, based on a cost function $C(w)$. The algorithm proceeds by defining an initial set of parameters w_0, running the ansatz multiple times on a quantum computer (to properly calculate the output probabilities), calculating $C(w_0)$ and find a new set of parameters according to the optimization algorithm. This process is repeated iteratively until a minimum is reached or until the algorithm completes its iterations in a hybrid computing scheme, as can be seen in Fig. 2 [25]. In Fig. 2, $f(x, w)$ is the measured output of the ansatz.

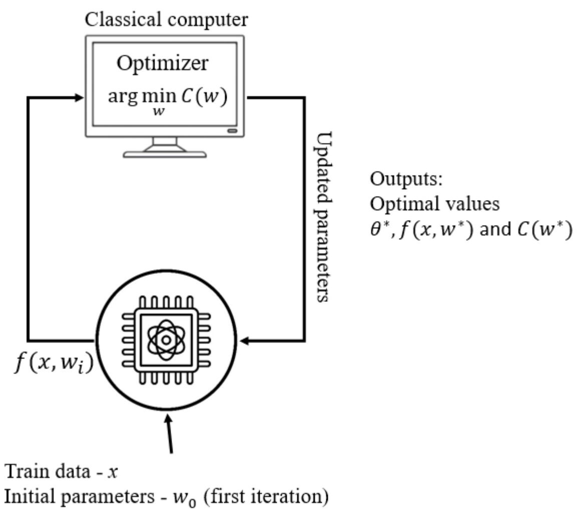

FIGURE 2. Schematic diagram of a Variational Quantum Algorithm (VQA), adapted from [8].

III. QUANTUM KERNEL DESIGN

In this section, the design of the kernel is detailed. First, the quantum circuit is proposed as basis for the kernel, with the optimization process outlined in Subsection III-B.

A. UNITARY CIRCUIT DESIGN

In Subsection II-B, the quantum kernel formulation is defined. The subsequent step involves designing the unitary circuit, a key component that depends on the input variables x and a vector of adjustable parameters w. This unitary is implemented as a two-layered circuit, represented by $U(x, w)$.

As depicted in Fig. 3, the ansatz for a single circuit layer is shown for a four-qubit configuration. For the QSVM with five variables, the ansatz expands to five qubits while preserving the same gate structure. This pattern holds consistently across other configurations.

The first operation in Fig. 3 applies a Hadamard gate to each qubit, placing them into an equal superposition state. Subsequently, a rotation gate (RZ) is applied to each qubit, where the rotation angle varies according to the corresponding feature value. In this study, the circuit utilizes between two and seven qubits, corresponding to the number of selected features in the dataset, with x_{ij} representing the value of the j-th feature of the i-th data point. Following this, RY gates are applied to each qubit, with the rotation angles determined by adjustable parameters.

The final set of operations consists of controlled-RZ (CRZ) gates, which introduce adjustable levels of entanglement between pairs of qubits. These gates are arranged in a ring configuration. The complete unitary, $U(x, w)$, comprises a second layer identical to the first, but with a different set of parameters for the vector w. Consequently, this quantum circuit exhibits a high degree of both superposition and entanglement.

According to [26], this quantum circuit architecture has demonstrated competitive or superior performance compared to well-established classical classification techniques, such as shallow neural networks and Support Vector Machines (SVMs), across a variety of benchmark problems. Consequently, this approach was selected based on its success in previous studies; however, exploring alternative ansatzes in future work is encouraged.

B. OPTIMIZATION PROBLEM

The choice of a parameterizable unitary is aimed at designing an optimized kernel, where the vector w can be adjusted according to a specific objective. In this work, the objective is to achieve a particular kernel function value, as given by (11).

$$k_{obj}(x_i, x_j) = \begin{cases} 1, & \text{if } y_i = y_j \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

where y_i and y_j are the target values for x_i and x_j, representing whether the person has churned or not.

This objective function aims to drive the kernel value between data points that belong to the same class toward 1, while driving the kernel value between points in different classes to approach zero. The optimization task is to find the vector w that minimizes the quadratic error between the elements of the objective kernel matrix and the current kernel matrix.

$$C(w) = \sum_i \sum_j \left(k(x_i, x_j, w) - k_{obj}(x_i, x_j)\right)^2 \tag{12}$$

where, in (12) $C(w)$ is the proposed cost function that needs to be minimized as shown in (13).

$$w = \arg\min_w C(w) \tag{13}$$

This problem can be solved by a VQA and, in each iteration, a batch of data points is used to refine the cost function and, consequently, the quantum kernel.

In comparison to Fig. 2, $f(x, w)$ is equivalent to $k(x_i, x_j, w)$, which represents the output of the kernel function. This output is then evaluated by the cost function in (12).

The chosen optimization method is the Adaptive Gradient Algorithm (AdaGrad) Optimizer, which is a gradient-based optimization algorithm designed to adapt the learning rate for each parameter based on historical gradients, effectively providing a higher learning rate for infrequent features and a lower learning rate for frequent ones. This adaptive property makes Adagrad particularly useful when dealing with sparse data [27]. The algorithm updates parameters according to (14):

$$w_{t+1,i} = w_{t,i} - \frac{\eta}{\sqrt{G_{t,i} + \epsilon}} \nabla_\theta L(w_t) \tag{14}$$

where η is the initial learning rate, $G_{t,i}$ is the sum of the squares of the gradients of the loss function with respect to θ_i up to time step t, and ϵ is a small constant for numerical stability. The learning rate in step t is given by (15) [27].

$$\eta_{t+1,i} = \frac{\eta}{G_{t,i} + \epsilon} \tag{15}$$

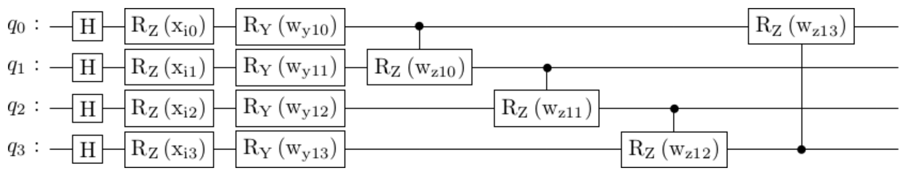

FIGURE 3. Ansatz layer for unitary $U(x)$ used in this work — illustrated for a four-qubit case.

where the sum of squares of the gradients is given by (16).

$$G_{t,i} = \sum_{\tau=1}^{t} (\nabla_w L(w_\tau))^2 \tag{16}$$

The initial learning rate in this work is 0.10 for all VQAs with 110 iterations, each one with a different batch of 31 samples of data. This setup presented goods results in terms of convergence, since the cost function converged smoothly, as can be seen in Figure 4.

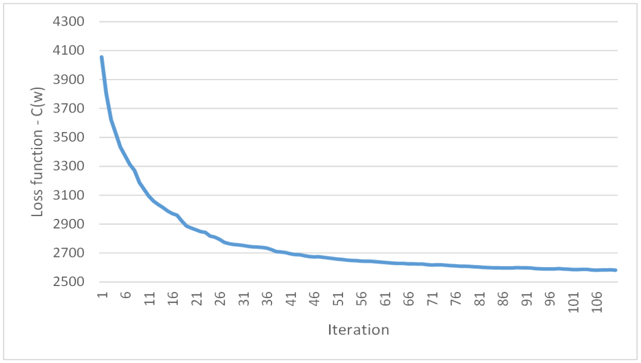

FIGURE 4. Cost function curve during VQA optimization — five-variable model configuration.

IV. DATA AND EXPERIMENTS

Subsection IV-A presents the dataset and its variables and the feature selection procedure and Subsection IV-B depicts the experiments to be held in this work.

A. DATASET AND MODEL FEATURES

The problem at hand is to develop a predictive model capable of identifying potential bank customers likely to churn. The dataset utilized in this work is public and can be accessed in [28]. It consists of 10000 records, with 10 features and an additional column indicating whether each customer churned.

The possible variables in the dataset are [28]:

- CreditScore: The credit score of the customer.
- Geography: The geographical location of the customer (e.g., country or region).
- Gender: The gender of the customer.
- Age: The age of the customer.
- Tenure: The number of years the customer has been with the bank.
- Balance: The account balance of the customer.
- NumOfProducts: The number of bank products the customer has.
- HasCrCard: Indicates whether the customer has a credit card (binary: yes/no).
- IsActiveMember: Indicates whether the customer is an active member (binary: yes/no).
- EstimatedSalary: The estimated salary of the customer.

The dataset includes 10 variables that can potentially predict churn propensity. However, due to the computational limitations of this study, it is not feasible to utilize all of them. The design of the ansatz maps each variable onto a qubit, meaning that a model with 10 variables would require 10 qubits, resulting in 2^{10} quantum states. This level of complexity is prohibitive for quantum simulators with the available computing capacity.

Thereby, feature selection is necessary to build a model compatible with the existing computing restrictions. Firstly, Pearson correlation was calculated to verify possible multicolinearities between continuous variables through (17) [29].

$$\rho(a, b) = \frac{E(ab)}{\sigma_a \sigma_b} \tag{17}$$

where ρ is the correlation between variables a and b, $E(ab)$ is the expected value of ab and σ_a and σ_b are the standard deviations of a and b, respectively. The Pearson correlation provides how linearly strong is the relation between two variables. If its value is close to 1, a and b have positive linear correlation, if its value is close to -1, the variables have a negative linear correlation and if the output is 0, then a and b have no correlation.

In this dataset, there is no significant correlation between the continuous variables, suggesting that each variable may provide distinct information for predicting churn.

Mutual Information (MI) is used to select the variables, which is a key concept in information theory that quantifies the shared information between two random variables X and Y. Mathematically, MI is defined by (18) [30].

$$I(X; Y) = \int_x \int_y p(x, y) \log \frac{p(x, y)}{p(x)p(y)}\, dx\ dy \tag{18}$$

where $p(x, y)$ is the joint probability density function of X and Y, and $p(x)$ and $p(y)$ are their marginal probability density functions.

Equation (18) can be interpreted as follows: if X and Y are independent, then $p(x, y) = p(x)p(y)$. In this case, the logarithm becomes zero, resulting in zero mutual information between X and Y. Conversely, if there is some degree of dependency between them, MI will have a non-zero value. This measure quantifies how much knowing one variable reduces uncertainty about the other, capturing both linear and non-linear dependencies.

Thus, the MI is calculated between each variable in the dataset (X) and the Exited variable (Y), which is the model target.

TABLE 1. Mutual information scores of the dataset variables and the resulting feature selection decision.

Variable	MI	Models
Age	0.07660	All
NumOfProducts	0.06720	All
GeographyGermany	0.01400	3 to 7 vars.
IsActiveMember	0.01230	4 to 7 vars.
Balance	0.00736	5 to 7 vars.
GenderFemale	0.00565	6 to 7 vars.
GeographyFrance	0.00554	7 vars.
CreditScore	0.00372	No
EstimatedSalary	0.00271	No
HasCrCard	0.00003	No
Tenure	0.00001	No

Age exhibits the highest Mutual Information (MI) in Table 1, followed by NumOfProducts and GeographyGermany, which has the third-highest MI. In contrast, Tenure, HasCrCard, and EstimatedSalary have very low MI values. The last column of Table 1 indicates which models include each variable: Age and NumOfProducts are present in all models, while GeographyGermany appears in models with three to seven variables. EstimatedSalary, HasCrCard, and Tenure are excluded from all models due to their low MI values.

B. EXPERIMENTS

With the kernel design defined and the model variables selected, the next step involves configuring the experiments. This stage presents notable challenges due to the unavailability of quantum hardware, requiring the use of quantum simulators. However, simulators introduce relevant scalability limitations. All experiments were conducted on a machine with 16 GB of RAM and a 4-core processor without GPU. Additionally, memory consumption for SVM training grows quadratically with the number of training samples, i.e., $\mathcal{O}(n^2)$, where n is the number of samples. These constraints led to the following experimental limitations:

- The dataset size was limited to 10000 samples to ensure simulations were feasible. Larger datasets resulted in excessive memory usage, making multiple experimental scenarios impractical.
- Simulation time increased with the number of qubits. To maintain reasonable computational times, the number of qubits was restricted to a maximum of 7.

The experiments were conducted using the Pennylane library (version 0.36.0) with Python 3.10, employing the "default.qubit" device, which does not support noise modeling. Preliminary tests with noise were performed using the "default.mixed" device; however, its memory consumption proved excessively high, rendering comprehensive simulations with noise infeasible. Nevertheless, Section V provides an analysis of the probability of encountering at least one error during the execution of the quantum circuit, considering configurations ranging from two to ten qubits.

Additional Python libraries used in this work include Pandas (2.2.3), NumPy (1.26.4), and Scikit-learn (1.6.1). The experimental procedure is divided into two main phases. In the first phase, the dataset is partitioned into subsets for kernel optimization, SVM training, and testing. Optimal parameters for the quantum kernel are obtained using the AdaGrad optimizer. In the second phase, the QSVM is trained using the optimized kernel via the SupportVectorClassifier object from Scikit-learn. Once trained, the model is evaluated on the test set using standard metrics available in the Scikit-learn library. Since Scikit-learn and the SupportVectorClassifier are used, the dataset is formatted as a table, where the selected variables for each model serve as features, and an additional column indicates whether the customer has exited.

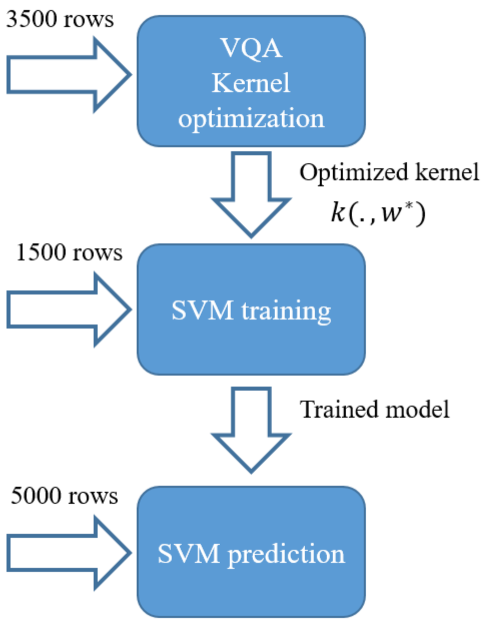

FIGURE 5. Pipeline for training and prediction in quantum support vector machine (QSVM).

The QSVM training and prediction pipeline is shown in Fig. 5. In the first step, the quantum kernel is optimized using the VQA depicted in Subsection III-B. This step uses a 3500 rows sample as input and whose output is the quantum embedding kernel with optimized parameters w^*. The optimized kernel serves as an input for SVM training, which is performed using a 1500 samples data. There is no overlap between the data used for kernel optimization and SVM training, in order to avoid possible data leakage and overfitting in the QSVM.

Furthermore, cross-validation could not be performed in this work, as methods such as K-fold cross-validation and grid search require training multiple models, which is not feasible due to the extensive time required to train the QSVM (ranging from three to nine days, depending on the number of qubits involved). Therefore, all experiments were conducted with a regularization parameter of $C = 1.0$.

Once the model is trained, predictions are held on a 5000 row sample, using the entire dataset to train and evaluate. There is no intersection between train and test data. The decision to use only half of the dataset for training is driven by computational limitations. A larger training sample would require significantly more computational power and result in much longer processing times, which was not feasible for this work.

The decision to allocate more data to kernel optimization than to SVM training is rooted in computational efficiency. SVM training typically has a memory complexity of $\mathcal{O}(n^2)$, where n is the number of training samples [31]. This arises because the SVM requires the construction of an $n \times n$ kernel matrix. While kernel optimization also has a memory complexity of $\mathcal{O}(n^2)$, it is performed in batches of size $n = 31$, which is significantly smaller than the 1500 samples used for SVM training. Furthermore, since this work uses quantum simulators the relation between training data and use of memory is even worse. Thus, to optimize the use of available resources, a larger portion of the data was allocated to kernel optimization, as it is less memory-intensive than the SVM training process.

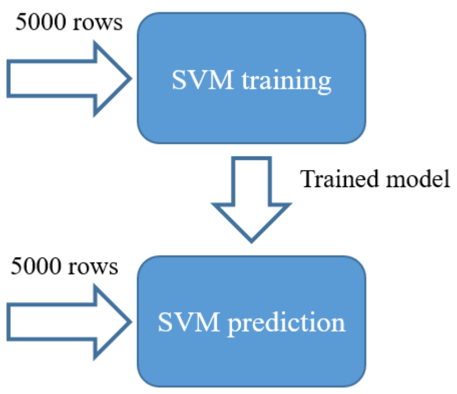

FIGURE 6. Training and prediction pipeline for classical SVMs with standard kernels.

Fig. 6 presents the training and predict pipeline for the SVM with linear, RBF, polynomial and sigmoid kernels. In this case, there is no need to train these kernels separately, as they are already built into scikit-learn, the Python package used for all SVM implementations in this study. The training set consists of 5000 data points, the same data used for kernel optimization and QSVM training. The prediction data remains consistent across all experiments.

The evaluation metrics in this work include accuracy, precision, recall, F1, and the Area under the Receiver Operating Characteristic (AUC).

C. ANALYSIS OF NOISE-INDUCED ERRORS

Before inferring the potential effects of noise on the quantum circuit, certain assumptions must be made to facilitate the estimation process. According to [32], one-qubit quantum gates can be considered ideal, whereas two-qubit quantum gates must be treated as non-ideal. The ansatz used in this work includes Hadamard, RY, and RZ gates as one-qubit operations, and CRZ as a two-qubit operation. In a single layer of an n-qubit arrangement, there are n CRZ gates. Moreover, the ansatz comprises two layers, and constructing the kernel matrix requires building the adjoint ansatz, resulting in a total of $4n$ CRZ gates.

Additionally, errors induced by noise can be modeled using Kraus operators. Common error types include depolarizing channels, amplitude damping, and phase damping [32]. It is assumed that the overall error rate in each quantum gate follows a binomial distribution $X_i \sim B(1, p)$ [33], and that these errors are independent for each gate [34]. For simplicity, it is also assumed that the probability of error in a quantum gate is given by p and is the same for every CRZ gate. Therefore, the total probability of an error occurring in the quantum circuit can be described by the sum of these distributions [35]. Consequently, the overall error probability follows a binomial distribution, given by (19).

$$X \sim B(4n, p) \tag{19}$$

Furthermore, current quantum computing technologies generally exhibit error rates ranging from 0,1% to 1% per quantum gate [36]. Therefore, the probability of having at least one error in the quantum circuit is given by (20).

$$P(N \geq 1) = \sum_{i=1}^{4n} \binom{4n}{i} p^i (1-p)^{4n-i} \tag{20}$$

V. RESULTS

The test dataset consists of 79.64% of customers who did not churn, making it an unbalanced sample. Analyzing accuracy in Fig. 7, the QSVM achieved the best results for models with three or more variables, performing slightly better than the SVM with RBF and polynomial kernels. The linear SVM showed an accuracy of 79.64%, which is misleadingly high since it predicted only non-churn cases. This is evident in the 0% precision and recall observed in Figs. 8 and 9. Moreover, the 50% AUC score in Fig. 11 highlights that the linear SVM lacks any predictive power, a consequence of its inability to effectively separate the two classes due to the limitations of the linear kernel.

The SVM with a polynomial kernel had slightly lower accuracy than the RBF kernel for models with three to four variables but performed similarly to the RBF kernel for five to seven variables. The SVM with a sigmoid kernel consistently exhibited the lowest accuracy for all variable configurations,

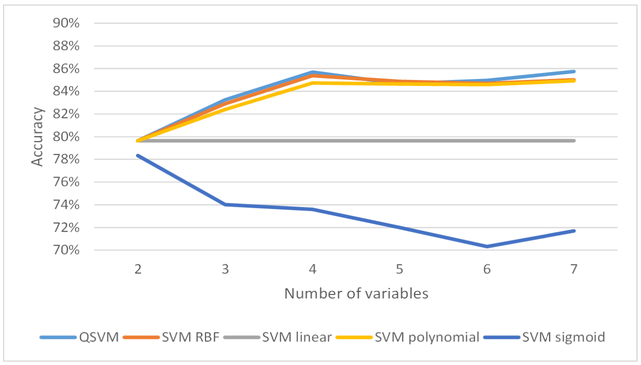

FIGURE 7. **Accuracy scores for QSVM and classical SVMs across different experiments.**

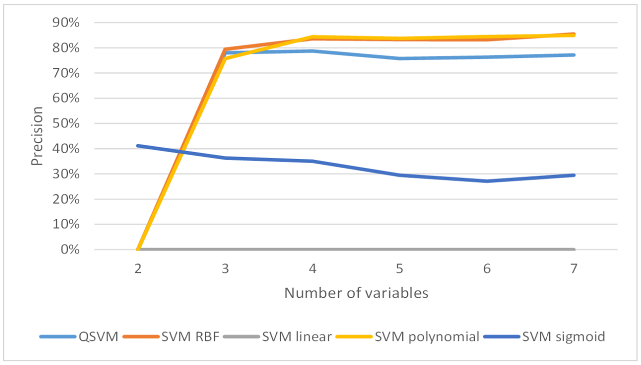

FIGURE 8. **Precision scores for QSVM and classical SVMs across different experiments.**

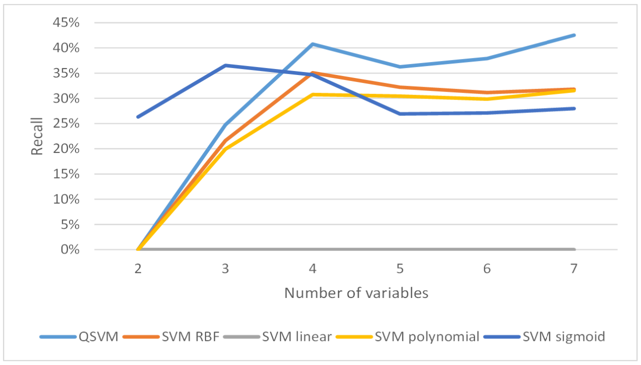

FIGURE 9. **Recall scores for QSVM and classical SVMs across different experiments.**

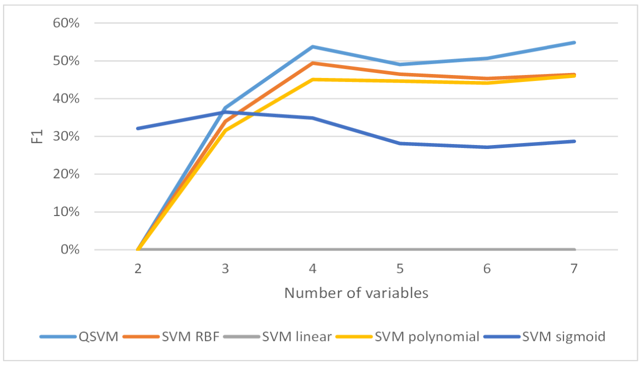

FIGURE 10. **F1-scores for QSVM and classical SVMs across different experiments.**

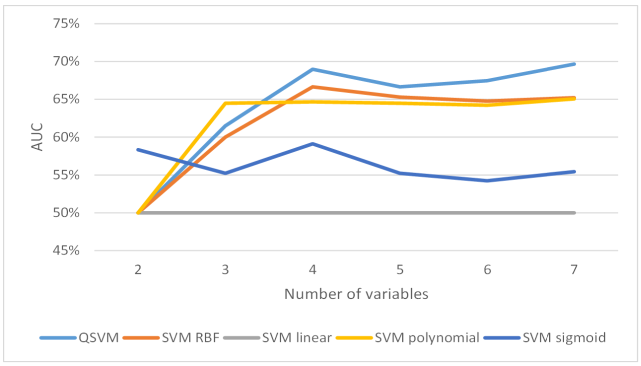

FIGURE 11. **AUC-scores for QSVM and classical SVMs across different experiments.**

but it outperformed the linear SVM as it predicted some churn cases, as shown in Figs. 8 and 9.

In Fig. 8, the RBF and polynomial SVMs achieved the highest precision for models with four to seven variables, with similar performance between the two. The QSVM showed slightly lower precision in this range but performed comparably for models with three variables. The sigmoid kernel SVM exhibited a decline in precision as the number of variables increased, except between six and seven variables, where the performance slightly increased.

Fig. 9 highlights that the QSVM demonstrated the highest recall for models with four to seven variables, consistently outperforming the RBF and polynomial SVMs in these scenarios. The sigmoid kernel SVM showed the best recall for two and three variables, while the RBF kernel SVM outperformed the polynomial kernel SVM for models with three or more variables. The performance gap between the RBF and polynomial SVMs narrowed as the number of variables increased.

The F1-score, depicted in Fig. 10, balances precision and recall. The QSVM achieved the best F1-scores for models with three or more variables. The sigmoid kernel SVM performed well for models with two and three variables, but its performance declined for models with four or more variables. In contrast, the RBF kernel SVM had the second-best F1-scores, followed closely by the polynomial kernel SVM. Notably, the QSVM, RBF, and polynomial SVMs exhibited a performance peak with four variables, a slight decline with five variables, and an increasing performance gap favoring the QSVM for six and seven variables. Increasing the number of variables beyond this range is not feasible due to prohibitive computational demands and diminishing returns from variables with low IVs and MIs.

Finally, the AUC values in Fig. 11 reveal that the SVM with polynomial kernel has a relatively flat performance for three or more variables, achieving the best AUC score for three variables and the third-best for configurations with four variables or more. For models with seven variables, it showed a small increase in performance, having very close results to the RBF kernel SVM. The QSVM delivered the best results for models with four or more variables and. Besides, similarly to Fig. 10, the QSVM has a better performance improvement from five to seven variables, indicating effective utilization of the high-dimensional quantum kernel. The sigmoid kernel SVM was the only model capable of predicting churn cases with two variables, achieving the best F1 and AUC scores in this scenario. However, for models with more than two variables, it ranked second-worst in terms of AUC scores, demonstrating limited discrimination capability.

Based on the analysis presented in Subsection IV-C, the probabilities of noise-induced errors in the quantum circuit as a function of the number of qubits are illustrated in Fig. 12.

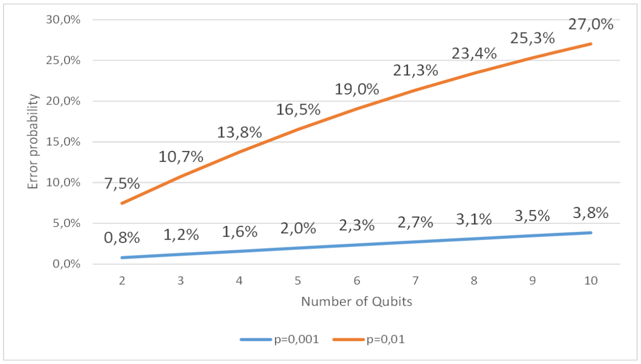

FIGURE 12. Estimated probability of at least one error occurring in the quantum circuit as a function of the number of qubits.

As discussed in Subsection IV-C, the probability of noise-induced errors per gate ranges from 0.1% to 1%. Accordingly, Fig. 12 presents two curves: one representing the probability of at least one error occurring in the circuit during simulation with a 0.1% error probability per quantum gate, and the other with a 1% error probability. Therefore, the probability of encountering any type of error for the 2-qubit arrangement presented in this work lies between 0.8% and 7.5%, while for the 7-qubit arrangement, it ranges from 2.7% to 21.3%.

These results have several implications: Firstly, when applying this approach to real quantum computers with noise, the results will be increasingly compromised as the number of variables (and thus, qubits) increases. Additionally, this indicates that the proposed approach faces scalability challenges when applied to NISQ-era quantum computers.

Given this scenario, a variation of the proposed kernel was developed with a single layer. The results for the five-qubit QSVM, using the same input variables as in the two-layer configuration, are presented in Table 2.

The one-layer QSVM yielded overall good results, performing slightly below the two-layer QSVM and the SVM with RBF kernel. Nevertheless, it outperformed the SVM models with linear, polynomial, and sigmoid kernels when using five variables. In this configuration, the estimated noise-induced error probability ranges from 1.0% to 9.1%, which is lower than in the two-layer scenario but still significant enough to potentially affect outcomes in real quantum hardware.

TABLE 2. Classification performance of the one-layer QSVM compared to other cases with five input variables.

Model	Accuracy	Precision	Recall	F1	AUC
QSVM 1 layer	85.02%	85.87%	31.63%	46.23%	65.15%
QSVM 2 layers	84.66%	75.77%	36.25%	49.04%	66.64%
SVM RBF	84.88%	83.25%	32.22%	46.46%	65.28%
SVM linear	79.64%	0.00%	0.00%	Div. 0	50.00%
SVM poly	84.64%	83.78%	30.45%	44.67%	64.47%
SVM sigmoid	72.00%	29.46%	26.92%	28.13%	55.22%

Therefore, quantum error detection (QED) and quantum error correction (QEC) are essential to achieve reliable results when employing this type of quantum kernel. Notable applications of QED are presented in [37], while QEC strategies incorporating quantum machine learning can be found in various works, such as [38] and [39].

VI. CONCLUSION

Overall, the QSVM demonstrated the most well-balanced results across all experiments. The observed increase in performance as the number of variables grew was expected, given that the use of an optimized kernel in a high-dimensional space should provide an expected performance improvement, even considering the optimization challenges in this scenario. This behavior is particularly promising for complex problems with a large number of available variables, where the QSVM could potentially outperform its classical counterparts. Additionally, the QSVM performed well even with fewer variables (with the exception of the extreme case involving only two variables), where the quantum advantage diminishes. As anticipated, in these cases, its performance became comparable to, and in some instances slightly worse than, the classical SVM models.

However, even not being able to perform simulations with noise, it was possible to infer that noise interference is costly in terms of results for this case, even with a one layer quantum circuit.

For future research, it would be valuable to investigate alternative, simpler quantum circuits to evaluate whether comparable results can be achieved with reduced susceptibility to noise. Another promising direction is to test the approach proposed in this work in conjunction with QEC algorithms.

ACKNOWLEDGMENT

The Article Processing Charge (APC) for the publication of this research was funded by the Brazilian Federal Age.

REFERENCES

[1] A. Steane, "Quantum computing," *Rep. Prog. Phys.*, vol. 61, no. 2, p. 117, 1998.

[2] X. Zhao, X. Xu, L. Qi, X. Xia, M. Bilal, W. Gong, and H. Kou, ‘‘Unraveling quantum computing system architectures: An extensive survey of cutting-edge paradigms,’’ *Inf. Softw. Technol.*, vol. 167, Mar. 2024, Art. no. 107380.

[3] T. F. Rønnow, Z. Wang, J. Job, S. Boixo, S. V. Isakov, D. Wecker, J. M. Martinis, D. A. Lidar, and M. Troyer, ‘‘Defining and detecting quantum speedup,’’ *Science*, vol. 345, no. 6195, pp. 420–424, Jul. 2014.

[4] A. Ekert and R. Jozsa, ‘‘Quantum computation and Shor’s factoring algorithm,’’ *Rev. Modern Phys.*, vol. 68, no. 3, pp. 733–753, Jul. 1996.

[5] J. Bub, ‘‘Quantum computation: Where does the speed-up come from,’’ in *Philosophy of Quantum Information and Entanglement*. Cambridge Univ. Press, 2010, pp. 231–246.

[6] G. Leander and A. May, ‘‘Grover meets Simon–quantumly attacking the FX-construction,’’ in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, Hong Kong. Cham, Switzerland: Springer, 2017, pp. 161–178.

[7] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, ‘‘Quantum machine learning,’’ *Nature*, vol. 549, no. 7671, pp. 195–202, Sep. 2017.

[8] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, Ł. Cincio, and P. J. Coles, ‘‘Variational quantum algorithms,’’ *Nature Rev. Phys.*, vol. 3, no. 9, pp. 625–644, Aug. 2021.

[9] J. Qin, ‘‘Review of ansatz designing techniques for variational quantum algorithms,’’ *J. Phys., Conf. Ser.*, vol. 2634, no. 1, Nov. 2023, Art. no. 012043.

[10] N. Innan, M. A. Z. Khan, B. Panda, and M. Bennai, ‘‘Enhancing quantum support vector machines through variational kernel training,’’ *Quantum Inf. Process.*, vol. 22, no. 10, p. 374, Oct. 2023.

[11] Z. Shan, J. Guo, X. Ding, X. Zhou, J. Wang, H. Lian, Y. Gao, B. Zhao, and J. Xu, ‘‘Demonstration of breast cancer detection using QSVM on IBM quantum processors,’’ *Res. Square*, 2022. [Online]. Available: https://doi.org/10.21203/rs.3.rs-1434074/v1

[12] G. Bologna and Y. Hayashi, ‘‘QSVM: A support vector machine for rule extraction,’’ in *Proc. 13th Int. Work-Conf. Artif. Neural Netw.*, Palma de Mallorca, Spain, Jan. 2015, pp. 276–289.

[13] D. Shin, Y. Park, H. Jeong, H.-C.-V. Tran, E. Jang, and S. Jeong, ‘‘Exploring the potential of colloidal quantum dots for near-infrared to short-wavelength infrared applications,’’ *Adv. Energy Mater.*, vol. 15, no. 2, Jan. 2025, Art. no. 2304550.

[14] M. Shahid, M. A. Hassan, F. Iqbal, A. Altaf, S. W. H. Shah, A. V. Elizaincin, and I. Ashraf, ‘‘Enhancing movie recommendations using quantum support vector machine (QSVM),’’ *J. Supercomput.*, vol. 81, no. 1, p. 78, Jan. 2025.

[15] L. Simon and M. Radons, ‘‘On neural quantum support vector machines,’’ *Quantum Mach. Intell.*, vol. 6, no. 1, p. 3, Jun. 2024.

[16] T. Hubregtsen, D. Wierichs, E. Gil-Fuster, P.-J.-H. S. Derks, P. K. Faehrmann, and J. J. Meyer, ‘‘Training quantum embedding kernels on near-term quantum computers,’’ *Phys. Rev. A, Gen. Phys.*, vol. 106, no. 4, Oct. 2022, Art. no. 042431.

[17] H. Yano, Y. Suzuki, K. M. Itoh, R. Raymond, and N. Yamamoto, ‘‘Efficient discrete feature encoding for variational quantum classifier,’’ *IEEE Trans. Quantum Eng.*, vol. 2, pp. 1–14, 2021.

[18] J Shobana, C. Gangadhar, R. K. Arora, P. N. Renjith, J. Bamini, and Y. D. Chincholkar, ‘‘E-commerce customer churn prevention using machine learning-based business intelligence strategy,’’ *Meas., Sensors*, vol. 27, Jun. 2023, Art. no. 100728.

[19] U. D. Prasad and S. Madhavi, ‘‘Prediction of churn behavior of bank customers using data mining tools,’’ *Bus. Intell. J.*, vol. 5, no. 1, pp. 96–101, 2012.

[20] M. Milošević, N. Živić, and I. Andjelković, ‘‘Early churn prediction with personalized targeting in mobile social games,’’ *Expert Syst. Appl.*, vol. 83, pp. 326–332, Oct. 2017.

[21] B. Schölkopf and A. J. Smola, *Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2002.

[22] L. Muflikhah and D. J. Haryanto, ‘‘High performance of polynomial kernel at SVM algorithm for sentiment analysis,’’ *J. Inf. Technol. Comput. Sci.*, vol. 3, no. 2, pp. 194–201, Nov. 2018.

[23] S. Panja, A. Chatterjee, and G. Yasmin, ‘‘Kernel functions of SVM: A comparison and optimal solution,’’ in *Proc. Int. Conf. Adv. Inform. Comput. Res.*, Shimla, India, Dec. 2018, pp. 88–97.

[24] D. Maheshwari, D. Sierra-Sosa, and B. Garcia-Zapirain, ‘‘Variational quantum classifier for binary classification: Real vs synthetic dataset,’’ *IEEE Access*, vol. 10, pp. 3705–3715, 2022.

[25] M. Bilkis, M. Cerezo, G. Verdon, P. J. Coles, and L. Cincio, ‘‘A semi-agnostic ansatz with variable structure for variational quantum algorithms,’’ *Quantum Mach. Intell.*, vol. 5, no. 2, p. 43, Dec. 2023.

[26] A. Pérez-Salinas, A. Cervera-Lierta, E. Gil-Fuster, and J. I. Latorre, ‘‘Data re-uploading for a universal quantum classifier,’’ *Quantum*, vol. 4, p. 226, Feb. 2020.

[27] J. C. Duchi, E. Hazan, and Y. Singer, ‘‘Adaptive subgradient methods for online learning and stochastic optimization,’’ *J. Mach. Learn. Res.*, vol. 12, no. 61, pp. 2121–2159, Feb. 2011.

[28] S. Badole. (2024). *Bank Customer Churn*. [Online]. Available: https://www.kaggle.com/datasets/saurabhbadole/bank-customer-churn-prediction-dataset

[29] I. Cohen, ‘‘Pearson correlation coefficient,’’ in *Noise Reduction in Speech Processing*. Springer, Jan. 2009, p. 132.

[30] P. A. Estevez, M. Tesmer, C. A. Perez, and J. M. Zurada, ‘‘Normalized mutual information feature selection,’’ *IEEE Trans. Neural Netw.*, vol. 20, no. 2, pp. 189–201, Feb. 2009.

[31] Z. Wen, J. Shi, B. He, J. Chen, and Y. Chen, ‘‘Efficient multi-class probabilistic SVMs on GPUs,’’ *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 9, pp. 1693–1706, Sep. 2019.

[32] S. Cheng, C. Cao, C. Zhang, Y. Liu, S.-Y. Hou, P. Xu, and B. Zeng, ‘‘Simulating noisy quantum circuits with matrix product density operators,’’ *Phys. Rev. Res.*, vol. 3, no. 2, Apr. 2021, Art. no. 023005.

[33] S. Gicev, L. C. L. Hollenberg, and M. Usman, ‘‘Quantum computer error structure probed by quantum error correction syndrome measurements,’’ *Phys. Rev. Res.*, vol. 6, no. 4, Dec. 2024, Art. no. 043249.

[34] Y. R. Sanders, J. J. Wallman, and B. C. Sanders, ‘‘Bounding quantum gate error rate based on reported average fidelity,’’ *New J. Phys.*, vol. 18, no. 1, Dec. 2015, Art. no. 012002.

[35] K. Butler and M. Stephens, ‘‘The distribution of a sum of binomial random variables,’’ Dept. Statist., Stanford Univ. Office Naval Res., Tech. Rep. 467, 1993.

[36] Microsoft. (2024). *Quantum Error Conrrection*. [Online]. Available: https://quantum.microsoft.com/en-us/insights/education/concepts/quantum-error-correction

[37] E. Adermann, H. Suzuki, and M. Usman, ‘‘Variational quantum machine learning with quantum error detection,’’ 2025, *arXiv:2504.06775*.

[38] Z. Wang and H. Tang, ‘‘Artificial intelligence for quantum error correction: A comprehensive review,’’ 2024, *arXiv:2412.20380*.

[39] A. Y. He and Z.-W. Liu, ‘‘Discovering highly efficient low-weight quantum error-correcting codes with reinforcement learning,’’ 2025, *arXiv:2502.14372*.

MATHEUS CAMMAROSANO HIDALGO received the B.S. degree in electrical engineering with an emphasis on control and automation and the M.S. and Ph.D. degrees in systems engineering from the Escola Politécnica da Universidade de São Paulo, São Paulo, Brazil, in 2012, 2015, and 2021, respectively.

From 2013 to 2019, he contributed to modeling, simulation, and control projects with the University of São Paulo for Brazilian Navy and Transpetro, a subsidiary of Petrobras, Brazilian state-owned oil company. Since 2020, he has been the Lead Data Scientist in various key sectors in Brazil, including food retail, insurance, banking, healthcare, and services. Since 2022, his research interests have expanded to include quantum machine learning. His research focused on friction compensation in control valves using nonlinear controllers, resulting in publications in high-impact journals. Additionally, he conducted studies on the modeling and simulation of marine oil vessels and propulsion control techniques, with results published in prominent journals and conferences.

Dr. Hidalgo has served as a Reviewer for journals, such as *Ocean Engineering*, *Journal of Control, Automation and Electrical Systems*, and IEEE Access.