software

MDPI

Article

Implementation and Performance Evaluation of Quantum Machine Learning Algorithms for Binary Classification

Surajudeen Shina Ajibosin and Deniz Cetinkaya *

Department of Computing & Informatics, Bournemouth University, Poole BH12 5BB, UK
* Correspondence: dcetinkaya@bournemouth.ac.uk; Tel.: +44-1202-961241

Abstract: In this work, we studied the use of Quantum Machine Learning (QML) algorithms for binary classification and compared their performance with classical Machine Learning (ML) methods. QML merges principles of Quantum Computing (QC) and ML, offering improved efficiency and potential quantum advantage in data-driven tasks and when solving complex problems. In binary classification, where the goal is to assign data to one of two categories, QML uses quantum algorithms to process large datasets efficiently. Quantum algorithms like Quantum Support Vector Machines (QSVM) and Quantum Neural Networks (QNN) exploit quantum parallelism and entanglement to enhance performance over classical methods. This study focuses on two common QML algorithms, Quantum Support Vector Classifier (QSVC) and QNN. We used the Qiskit software and conducted the experiments with three different datasets. Data preprocessing included dimensionality reduction using Principal Component Analysis (PCA) and standardization using scalers. The results showed that quantum algorithms demonstrated competitive performance against their classical counterparts in terms of accuracy, while QSVC performed better than QNN. These findings suggest that QML holds potential for improving computational efficiency in binary classification tasks. This opens the way for more efficient and scalable solutions in complex classification challenges and shows the complementary role of quantum computing.

Keywords: quantum machine learning; binary classification; quantum algorithms

check for updates

Citation: Ajibosin, S.S.; Cetinkaya, D. Implementation and Performance Evaluation of Quantum Machine Learning Algorithms for Binary Classification. *Software* **2024**, *3*, 498–513. https://doi.org/10.3390/software3040024

Academic Editor: Francisco José García-Peñalvo

Received: 31 October 2024
Revised: 21 November 2024
Accepted: 26 November 2024
Published: 28 November 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Quantum Computing (QC) exploits the principles of quantum mechanics to process information and solve problems that are too complex for classical computers. Unlike classical bits, qubits possess the unique ability to represent numerous possible combinations of zero and one at the same time. This simultaneous existence in multiple states is a phenomenon referred to as superposition. This property enables the processing of information in a parallel and exponentially expanded manner compared to classical computers. Although still in its early stages, quantum computing holds great promise for solving problems that are currently infeasible with classical computers [1].

Machine Learning (ML) has significantly advanced human capabilities and contributed to societal progress across various fields by automating complex tasks, improving decision-making processes, and providing personalized experiences [2]. ML uses algorithms and statistical models to analyze and interpret complex data, assisting in planning, decision support, predictive analysis, intelligent automation, and many other activities across multiple domains including healthcare, finance, education, transportation, defense, etc. [3–5].

ML is a rapidly evolving and active field that is continuously being expanded with new models and approaches [6]. Training and deploying ML models can be computationally expensive and time-consuming depending on the volume of data and the complexity of the methods. Research efforts to address this challenge include improving ML algorithms and architectures to be more efficient and scalable and using hardware advancements

such as Graphics Processing Units (GPUs) and Tensor Processing Units (TPUs), as well as distributed machine learning, to accelerate model training and inference [7].

QC, a multifaceted field based on quantum theory that has different potential applications in software engineering, is an alternative paradigm used to reduce the time required to solve complex problems with improved performance [8,9]. Quantum Machine Learning (QML) combines the power of ML and QC to solve challenges in various domains [10–12]. It uses the principles of quantum mechanics to improve computational efficiency and the performance metrics of the available ML algorithms. As QC technology evolves, the synergy between quantum algorithms and ML is expected to drive innovative solutions to previously unsolved computational problems.

QML offers potential solutions to these problems by utilizing quantum mechanical properties such as superposition and entanglement to process and analyze data more efficiently. However, the practical implementation of quantum algorithms and models is a relatively new research field, so many application areas still remain underexplored. In this study, we focused on the binary classification problem in machine learning. Our main research question was about understanding whether QML algorithms improve the performance of trained ML models by capturing more complex patterns or correlations in the datasets. We implemented a set of commonly used classical and quantum machine learning algorithms for binary classification by using three different datasets and compared the performances of the two computing paradigms.

This section presented an introduction to the problem domain and an overview of the study. The remainder of this manuscript is organized as follows: Section 2 presents background information and a review of the current literature regarding QC and QML. Section 3 explains the materials and methods adopted for building and training the models for this study. Section 4 describes more details about the experiments and presents the results. Section 5 includes the discussion and conclusions, as well as suggestions for future work.

2. Background Information and Literature Review

In this section, the fundamental concepts and principles of QC and QML will be briefly explained with a review of related work in the literature. Concepts such as superposition, entanglement, and quantum gates will be described. Related work about QML and specifically using quantum classifiers for binary classification is presented.

2.1. Quantum Computation and Quantum Information

Unlike classical computation, in which the smallest unit of information is represented in bits (represented as zeros and ones), quantum computation uses quantum bits, or *qubits* [13]. Quantum computers are not simply faster or better versions of today's classical computers, but instead, they represent a fundamentally new paradigm for processing information. While classical bits can take a value of 0 or 1, qubits possess the unique ability to represent and store various possible combinations of 0 and 1 at the same time [14]. This ability to simultaneously be in multiple states is called *superposition*. This property enables the processing of information in a parallel and exponentially expanded manner compared to classical computers.

A *quantum state* is any possible state of a quantum mechanical system or quantum hardware. There are numerous examples of quantum mechanical two-level systems in nature that potentially could serve as qubits. The electronic states of an ion and the electron spin of an atom implanted in silicon serve as examples. If the quantum system is based on a multi-level computational unit instead of the conventional two-level qubit, then each unit is called a *qudit*. Compared to qubits, qudits provide a larger state space that can reduce the circuit complexity and enhance the algorithm efficiency [15].

In quantum computing, we can generate pairs of qubits that are entangled, which means that changing the state of one of the qubits will instantly change the state of the other one. *Entanglement* has two very special properties: it is inherently private and allows maximal coordination. This happens even if the qubits are separated by very long distances.

Quantum computers are built using various hardware technologies, but the main types are superconducting circuits, photonic networks, trapped ions, quantum dots, etc. Each type has its advantages and disadvantages, such as greater entanglement or longer coherence times. *Quantum supremacy* is the goal, representing a quantum computer that could perform calculations that are not possible with classical computers or beyond the reach of even the most powerful supercomputer. In the future, quantum computers may not rely on a single hardware technology but could instead be based on combining different technologies for greater effectiveness [16]. Table 1 presents a comparison of classical vs. quantum computing according to various features.

Table 1. Classical vs. quantum computing.

Feature	Quantum	Classical
Theory	Quantum mechanics	Classical physics
Computation	Probabilistic	Deterministic
Operations	Linear algebra operations	Boolean algebra operations
Information storage	Qubits, qudits	Bits
System state	Continuous possible states in superposition	Discrete number of possible states
Technology	Superconducting loops, trapped ions, quantum dots, etc.	Transistors
Applications	Complex problems, optimization, simulation	General purpose
Error rate	High	Low
Environment	Ultracold	Room temperature
Computing power	Exponential growth	Linear growth
Processing	QPU	CPU

A universal fault-tolerant quantum computer that can efficiently solve complex tasks, such as integer factorization or unstructured database search, requires millions of qubits with optimized coherence times and error rates [17]. Practical realization of such devices could take a long time; however, Noisy Intermediate-Scale Quantum (NISQ) computation has been achieved, and near-term devices are presently available for use [18]. For example, IBM's Quantum Lab, Google Quantum AI, Microsoft's Azure Quantum, and Amazon's Bracket services offer cloud-based solutions for the implementation of quantum algorithms and are being used for real-world applications [19]. Other companies, such as D-Wave, Rigetti Computing, IonQ, Intel, Quantiuum, Xanadu, etc., are developing quantum computers and technologies as well [20]. These cumulative efforts will likely propel QC towards the intermediate-scale quantum era [21] and perhaps the fault-tolerant quantum era [22].

2.2. QC Implementation Models

We use Dirac notation to represent the quantum states, as it is widely used in quantum mechanics. A quantum state of a system can be represented by a column vector whose components are probability amplitudes for different states in which the system might be found when measured, i.e., in correspondence with the classical states of that system. The probability amplitudes are complex numbers, and the sum of the absolute values squared of the probability amplitudes is equal to 1 [14].

We assume that a qubit represents an abstraction of the fundamental unit of information without regard to what it is. It might be a single physical qubit or a logical qubit that consists of multiple physical qubits (e.g., 10–100). Qubits can have the value $|0\rangle$ and $|1\rangle$ or be in a state other than $|0\rangle$ or $|1\rangle$. A particular quantum state can be represented by a wave function $\psi(x)$ as follows:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \quad (1)$$

$$\text{where } \alpha,\ \beta \in \mathbb{C} \text{ and } \alpha^2 + \beta^2 = 1 \quad (2)$$

In the literature, there are various methods for representing and examining quantum systems, though the most common ones to have been successfully implemented are quantum gate circuit and adiabatic quantum computing [23].

Quantum gates are similar to classical gates, and *quantum circuits* can be designed by using existing sets of quantum gates that operate on a constant number of qubits [24]. This approach is commonly used and universal for QC, as gates are linked to each other like in the classical circuits and each gate performs some unitary operator [14].

Symbols are often used to denote the gates during the design of quantum circuits. For example, the quantum analogue of a classical NOT gate is the X-gate, also known as the Pauli-X gate. Note that not all classical gates have a direct quantum analogue. Multi-qubit gates act on two or more qubits simultaneously and enable the creation and manipulation of quantum states. Simulations can be executed by using single or two qubit gates on a universal quantum computer.

A qubit state can be illustrated via the Bloch sphere [14]. $|\psi\rangle$ points from the origin to a point on the surface of the unit sphere. The direction of $|\psi\rangle$ is specified by polar angle θ and azimuthal angle φ. Commonly used single-qubit gates are the Pauli-X, -Y, and -Z gates, which correspond to rotations by π radians about the x, y, and z axes, respectively, on the Bloch sphere, and the Hadamard gate, which is a π rotation about the X + Z axis and has the effect of putting the state into superposition. Once measured, the qubit will be in either one of its computational basis states.

The most common two-qubit gate is the quantum-controlled NOT or CNOT gate, which flips the second qubit (the target qubit) if and only if the first qubit (the control qubit) is $|1\rangle$. All unitary circuits can be decomposed into single-qubit gates and CNOT gates. Because the two-qubit CNOT gate requires much more time to execute on real hardware than single-qubit gates, circuit cost or complexity can be measured by the number of CNOT gates. There are also three-qubit gates such as the Toffoli gate, also known as the CCNOT gate.

Adiabatic quantum computing is an alternative universal approach, but in terms of computational complexity it is equivalent to gate-based quantum computing at polynomial time [25]. It is founded upon the quantum adiabatic theorem, which describes the evolution of the ground state of a quantum system as follows:

$$H(t) = s(t)H_0 + (1 - s(t))H_f \tag{3}$$

A time-varying Hamiltonian evolves from an initial ground state H_0 to a form H_f that encodes the problem to be solved. $s(t)$ represents an adiabatic evolution path, a function that decreases from 1 to 0 for an elapsed time t_f. If the time evolution of the Hamiltonian is sufficiently slow, the state is likely to remain in the ground state [26]. Adiabatic quantum computing, and more specifically quantum annealing, are mainly used for optimization problems, while gate-based quantum computing can be used for a broader range of problems.

2.3. Quantum Algorithms and Quantum Data Encoding Methods

For a two-qubit system, all possibilities could be encoded into the state of the two qubits via superposition of the four basis states $|00\rangle$, $|01\rangle$, $|10\rangle$ and $|11\rangle$. A superposition of four states requires four probability amplitudes to fully describe the quantum state. Multiple qubits in a quantum computer can be conceptually grouped together in a quantum register. Each qubit will have an index within this register and each qubit can be addressed in qubit operations by using this qubit index.

Quantum parallelism is based on the ability of a quantum register to exist in a superposition of base states. It is the possibility of performing a large number of operations in parallel without the need for extra resources. For example, while with three classical bits there are $2^n = 2^3 = 8$ cases, all these cases can be represented using three qubits in superposition, meaning they are in a single quantum state simultaneously.

Quantum interference is a phenomenon in quantum mechanics when subatomic particles interact with and influence themselves and other particles while in a probabilistic superposition state. It can influence the probability of the outcomes when the quantum state is measured.

Quantum parallelism and quantum interference form the foundation of how a quantum computer processes information simultaneously. Quantum computers have the potential to exceed the performance of conventional computers for complex problems such as cryptography, chemistry, pharmaceuticals, etc. Quantum advantage is the point at which quantum algorithms deliver a significant, practical benefit beyond what classical computers alone are capable of for computationally complex problems.

Quantum algorithms are usually described via a circuit model, but they can be defined using other mathematical models too. They can use other techniques or algorithms as sub-parts of the algorithm, such as quantum phase estimation, quantum Fourier transform, amplitude amplification, etc.

One of the important steps in QML is encoding classical data into quantum states suitable for quantum computation [27]. There are several techniques for quantum data encoding, such as basis encoding, amplitude encoding, angle encoding, etc.

Basis encoding is a technique where classical data are encoded directly into the computational basis states of qubits. Each classical bit of data is mapped to a qubit, and the classical value is represented by the qubit being in either the $|0\rangle$ or $|1\rangle$ state. This is also known as binary encoding. Basis encoding is straightforward and directly maps classical binary data to quantum states, making it easy to implement in quantum circuits for classical data in binary form.

Amplitude encoding involves encoding the data into amplitudes of the quantum states. For a system with n qubits, amplitude encoding uses 2^n amplitudes to represent classical data. However, preparing a quantum state with specific amplitudes is non-trivial and may require complex quantum circuits or additional resources.

Angle encoding, also known as phase or rotation encoding, is another technique that maps classical data into the phase angles of qubits. A qubit state can be represented with Bloch sphere parameters using θ and φ.

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\varphi}\sin\left(\frac{\theta}{2}\right)|1\rangle \tag{4}$$

Angle encoding is straightforward to implement using basic quantum gates, making it accessible in quantum circuits; however, a qubit can only be used to encode a single data feature, making it more expensive to use for high-dimension datasets unless a dimensionality reduction technique is used.

2.4. Related Work

This section presents related work on using QML for classification problems. We grouped the studies into three subsections based on the underlying model.

2.4.1. Studies Using Variational Quantum Classifier (VQC)

VQC is a promising QML model, particularly for tackling classification problems. Built upon the principles of variational quantum algorithms, VQCs use parameterized quantum circuits to perform tasks in high-dimensional quantum spaces, potentially achieving a computational advantage over classical classifiers in certain scenarios [28].

To improve the detection rate of heart failure, Munshi et al. (2024) proposed a QML-based framework using a standard heart failure dataset consisting of different features related to cardiovascular diseases [29]. VQC was one of the QML algorithms used for the research. The Principal Component Analysis (PCA) dimensionality technique was used to reduce the dimension of the dataset. This study focused on comparing the QML methods rather than comparing QML performance with classical ML algorithms on the same dataset. After a comparative analysis, they concluded that QSVC outperformed VQC in all the metrics for the given dataset.

Maheshwari et al. (2022) presented the application of VQC for binary classification [30]. The three datasets used were: a synthetic dataset with randomly generated values between zero and one, a publicly available sonar dataset consisting of mining data, and a proprietary diabetes dataset related to diabetes with acute diseases and diabetes without acute disease. The feature importance technique was used to reduce the dimensions of the datasets by

dropping some features with least importance. The study reported VQC accuracies of 75%, 71.4%, and 68.73% for the synthetic, sonar, and diabetes datasets, respectively, when using basis data encoding. In contrast, VQC using amplitude data encoding achieved superior results, with accuracies of 98.40%, 67.3%, and 74.50% for the same datasets. These findings show the potential of quantum approaches in improving performance in classification tasks, suggesting promising directions for future research in QML.

2.4.2. Studies Using Quantum Support Vector Classifier (QSVC)

QSVC is an adaptation of the classical Support Vector Machine (SVM) within the framework of quantum computing. Classical SVMs operate by mapping data into a high-dimensional feature space where classes can be separated by an optimal hyperplane, but this mapping can become computationally intensive as data dimensionality grows. QSVCs address this challenge by using quantum feature maps and quantum kernels, potentially allowing for exponential speedup in certain classification scenarios. As the field advances, understanding the capabilities and limitations of QSVCs will be essential for evaluating their practical relevance within QML.

In a recent study, Kavitha and Kaulgud (2024) employed various datasets to benchmark the performance of QSVCs, demonstrating their potential and highlighting the areas where further research is needed [31]. A key factor in the study was the selection of suitable feature maps, which are critical for encoding data into quantum states. Effective feature maps can enhance the classifier's ability to distinguish between different classes in the dataset. The results showed that by using the right quantum feature maps, a quantum advantage was demonstrated for all three datasets compared to the classical alternatives. Comparisons to other QML approaches could provide broader insight on QSVC's strengths.

Suzuki et al. (2024) explored the practical applicability of both QSVC and Quantum Support Vector Regression (QSVR) by evaluating their performance on a variety of datasets [32]. QSVC was evaluated using fraudulent credit card transactions and image datasets, while the regression performance of QSVR was evaluated using financial and materials datasets. The experiments were implemented both on a quantum circuit simulator and a real quantum computer. The results show the resilience of quantum kernels on noisy devices, even when implemented on shallow circuits, and their adaptability across different types of datasets and tasks. These findings demonstrate the potential of QML to offer significant advantages over classical counterparts, particularly in handling noisy environments, which are prevalent in near-term quantum devices.

By adopting a QSVC-based approach to quantum annealing principles, Yuan et al. (2023) were able to enhance the classification of flow separation scenarios [33]. The QSVC algorithm demonstrated superior performance compared to classical SVM approaches. Specifically, it achieved an 11.1% increase in accuracy for binary classification tasks. Additionally, the study extended its investigation to multiclass classification, focusing on multiple angles of attack on aircraft wings. The developed multiclass QSVC, utilizing a one-against-all strategy, showed a notable 17.9% accuracy improvement over classical methods.

2.4.3. Studies Using Quantum Neural Networks (QNN)

QNNs represent a novel fusion of quantum computing and artificial neural networks where quantum parallelism and entanglement provide more efficient ways to process information [34]. A QNN has an input, output, and a number of hidden layers. The smallest building block of a QNN is the quantum perceptron, which is the quantum analogue of the perceptron used in ML [35]. Most QNNs are developed as feed-forward networks; i.e., they take input from one layer of qubits, evaluate this information, and passe on the output to the next layer. However, other models such as Quantum Recurrent Neural Networks (QRNN) have also been proposed in the literature, contributing to an emerging and rapidly developing field of research [36].

Several studies used the QNN approach in the literature to advance QML. For example, Simoes et al. (2023) performed an experimental analysis of QNN algorithm by evaluating its

performance on five different datasets using different combinations of quantum encoding techniques [37]. The study was able to demonstrate a quantum advantage, with the results showing that the QNN outperformed the classical neural network by 7%. In the hybrid approach used, QNN was implemented as a variational quantum circuit, while the optimizer was implemented on classical hardware.

Although it is a relatively new research field, the number of publications has increased recently in this area, and researchers have started to present systematic review studies to compare the QML methods and explore their usage for specific problems [38–40]. Overall, the existing literature on QML highlights significant advancements in developing quantum-enhanced classification algorithms. While practical implementations are limited by current quantum hardware constraints such as noise and low coherence, the results are promising with regard to better accuracy and efficiency.

3. Materials and Methods

This section presents the selected methods in this study, as well as the datasets. We used Python programming language (version 3.10.12) with Jupyter Notebook (version 7.2.2) and standard libraries pandas (version 2.2.3), numpy (version 2.1.2), matplotlib (version 3.9.2), and scikit-learn (version 1.5.2). The quantum computation processes were implemented by using the IBM Quantum Qiskit software package (initially version 1.1.2 then migrated to 1.2), which provides open access to quantum computing services for various QML algorithms. We used a quantum simulator due to the limited usage time of real quantum computers and the fact that simulators provide a flexible and accessible alternative for exploring quantum computation.

Currently available quantum processors, which are relatively small and noisy, lack the capacity to disentangle and generalize quantum data independently. To be effective, these NISQ processors must operate alongside classical co-processors [41]. As such, we employed a hybrid quantum–classical model that combines classical and quantum computing to adopt the strengths of both paradigms. The basic workflow is shown in Figure 1. During the initialization step, the building of the models, analysis, and preprocessing of the datasets were conducted on a classical computer. The training and optimization of the models were conducted on the classical and quantum hardware for the classical and quantum algorithms, respectively. During the evaluation step, the results were analyzed on a classical computer.

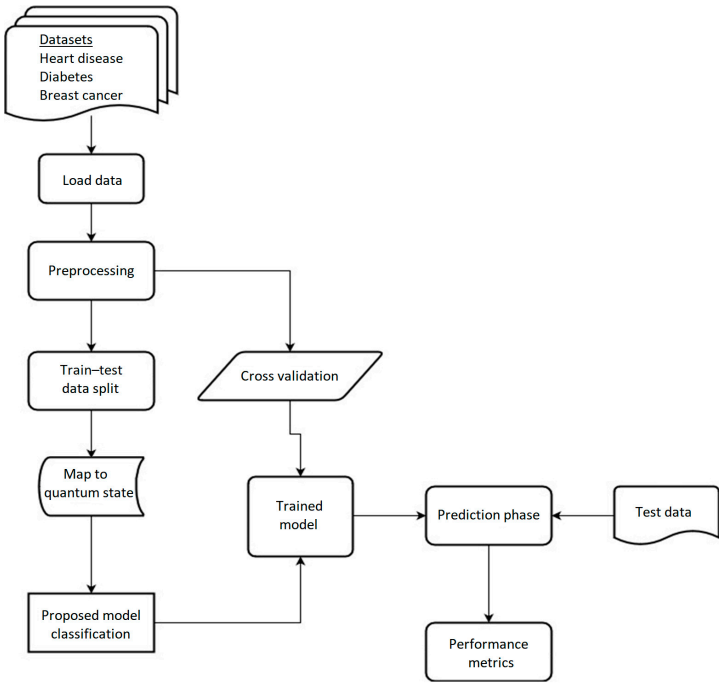


Figure 1. Flowchart of the model implementation.

3.1. ML Approach and Models

The quantum gate circuit model was adopted for this research because it is more popular and has the most easily accessible quantum hardware. The adaptation of classical ML algorithms in designing the corresponding quantum analogues is an ongoing research area. While it enables much easier transition to QC by using existing methods, it also allows for direct comparison by providing a benchmark for measuring the performance and effectiveness of quantum adaptations. We used QSVC and QNN as they have both achieved a level of success in their design and implementation, as discussed in the previous section.

QSVC is an adaptation of Support Vector Classifier (SVC), which is a sub type of SVM that works well with small- and medium-sized datasets [42]. It is used for both linear and non-linear classification tasks, though for non-linear classification tasks, it uses a kernel trick that maps the input data into a higher-dimensional feature space. For comparison, the classical model was trained and evaluated using SVC, for which four different models were trained using the linear, poly, rbf, and sigmoid kernels for each dataset. The *cross_validate()* function was used for the training and evaluation of the classical models.

To adapt SVC into a quantum algorithm as QSVC, the quantum kernel function can be represented with a quantum Hilbert space using a quantum feature map [43]. A kernel function can be represented as a matrix, and the kernel matrix K that represents the overlap of two quantum states is defined as follows:

$$K_{i,j} = \left| \left\langle \phi\left(\vec{x}_i\right) \middle| \phi\left(\vec{x}_j\right) \right\rangle \right|^2 \tag{5}$$

where a quantum feature map $\phi\left(\vec{x}\right)$ maps a classical feature vector $\vec{x}$ to a Hilbert space.

QNN uses parameterized quantum circuits to represent the layers. For comparison, the classical model was trained and evaluated using the Multilayer Perceptron (MLP) from the scikit-learn library, which is a feed-forward artificial neural network.

We used PCA for dimensionality reduction and data visualization. This method reduces the dataset into its most critical features, referred to as its principal components [44]. These components are linear combinations of the original variables that capture the maximum variance within the dataset. Through this process, PCA offers an approximation of the original data matrix, relying on a reduced number of principal components while preserving the most significant variance present in the data [45].

We used Quantum Feature Map (QFM) to map our classical datasets into quantum states. The quantum data estimation approach adopted by the two selected algorithms differs slightly. While we used the same quantum feature maps for both, we used the Quantum Kernel Estimation (QKE) method for QSVC and the Parameterized Quantum Circuit (PQC) method for QNN. These methods are explained in Section 3.4. The code is available upon request from the corresponding author.

3.2. Datasets

We used three health-related datasets that are publicly available. This section briefly explains the datasets and provides the links to the datasets. The breast cancer dataset is a part of the Python scikit-learn package and also publicly available via https://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnostic (accessed on 31 October 2024). It is a multivariate dataset with real feature characteristics. It contains 569 instances and 31 features. The features were computed from digitized images of fine-needle aspirations of breast masses. The dataset was loaded via the scikit-learn library directly during implementation.

The diabetes dataset can be found on the Kaggle data repository and is available via https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database (accessed on 31 October 2024). This dataset comprises 768 instances and 9 features, with the prediction feature included. It contains information such as the number of pregnancies, glucose level, blood pressure, skin thickness, insulin level in the blood stream, body mass index, age

and diabetes pedigree function. There are no missing values or duplicated information in the dataset.

The heart disease dataset is also available from the Kaggle data repository via https://www.kaggle.com/datasets/andrewmvd/heart-failure-clinical-data (accessed on 31 October 2024). It includes heart failure clinical data with 299 instances and 13 features.

The three different datasets are defined as D_{BC}, D_D, and D_{HD} of order $R^{n \cdot m}$, representing the breast cancer, diabetes and heart disease datasets, respectively. Each contains a set of features $\{f_1, f_2, \ldots, f_m\}$, where m is the number of features and n is the number of instances in each dataset. After importing these datasets, some exploratory data analysis was conducted to gain more insights into the contents of the datasets. Appendix A includes the target class balance analysis and PCA visualization for the three datasets. The next section provides brief information about the data preprocessing step.

3.3. Data Preprocessing

The first step was identifying and isolating the target/class feature for prediction. Then, the datasets were split into training and testing datasets using either the *cross_validate()* or the *train_test_split()* function with a splitting ratio of 80:20. This is a standard supervised ML procedure where 80% of a dataset is used for training a model and the remaining 20% is used to evaluate the performance of the trained model on an unseen dataset. The splitting ratio can be varied.

$$\begin{aligned} D_{BC} : \mathcal{R}^{569\times 31} &\xrightarrow{Isolating\ Target\ Column} D_{BC1} : \mathcal{R}^{569\times 30} \\ D_{D} : \mathcal{R}^{768\times 9} &\xrightarrow{Isolating\ Target\ Column} D_{D1} : \mathcal{R}^{768\times 8} \\ D_{HD} : \mathcal{R}^{299\times 13} &\xrightarrow{Isolating\ Target\ Column} D_{HD1} : \mathcal{R}^{299\times 12} \end{aligned}$$

A standard scaler was applied to the datasets. This technique removes the mean of each feature in the datasets and scales them by unit variance. This process ensures that each feature contributes equally to the model, preventing any feature from dominating due to its scale. This has been shown to improve the trainability of ML models [46]. It takes care of the outlier datapoints.

$$\begin{aligned} D_{BC1} : \mathcal{R}^{569\times 30} &\xrightarrow{StandardScaler} D_{BC2} : \mathcal{R}^{569\times 30} \\ D_{D1} : \mathcal{R}^{768\times 8} &\xrightarrow{StandardScaler} D_{D2} : \mathcal{R}^{768\times 8} \\ D_{HD1} : \mathcal{R}^{299\times 12} &\xrightarrow{StandardScaler} D_{HD2} : \mathcal{R}^{299\times 12} \end{aligned}$$

Next, the PCA technique was applied to reduce the dimensionality of the datasets without reducing the information or pattern therein. The dimension of each dataset was reduced to a linear combination of two principal components. The dimensionality of a dataset impacts the speed and training efficiency of QML models. The limitation and operational constraints of quantum hardware is also a justification for this dimensionality reduction.

$$\begin{aligned} D_{BC2} : \mathcal{R}^{569\times 30} &\xrightarrow{PCA} D_{BC3} : \mathcal{R}^{569\times 2} \\ D_{D2} : \mathcal{R}^{768\times 8} &\xrightarrow{PCA} D_{D3} : \mathcal{R}^{768\times 2} \\ D_{HD2} : \mathcal{R}^{299\times 12} &\xrightarrow{PCA} D_{HD3} : \mathcal{R}^{299\times 2} \end{aligned}$$

The final step was the application of the MinMaxScaler for the quantum model to restrict the range of values in the dataset to between 0 and 1.

3.4. QML Model Implementation

In this study, we adopted the ZZFeatureMap, which was designed with the angle encoding technique. It is assumed to be difficult to simulate classically [47]. It is available as one of the built-in QFMs provided by IBM in the qiskit_machine_learning 0.8.0 library.

Figure 2 shows the ZZFeaturemap circuit for the QSVC model with two feature dimensions (for the two principal components), two repetitions, and a linear entanglement. It has a depth of 10. The x[0] and x[1] parameters are the placeholders for the input features.

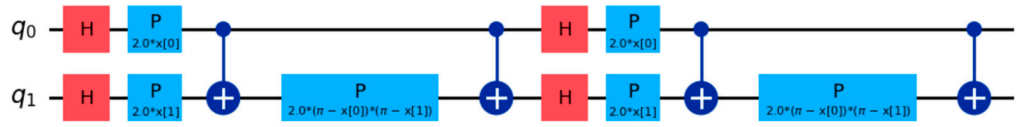


Figure 2. ZZFeatureMap with 2 qubits for the QSVC model.

We employed the QKE method for the QSVC model to optimize the quantum kernel's parameters, utilizing a quantum kernel trained model and an optimizer. Specifically, we used the Quantum Kernel Alignment (QKA) method to accomplish this task [48]. In QKA, a quantum kernel that is parameterized to fit a dataset is iteratively adjusted, aiming for the largest possible margin in SVMs.

A custom rotational layer is created and composed with the ZZFeatureMap, as shown in Figure 3. A quantum kernel is then instantiated from the TrainableFidelityQuantumKernel class with the feature map passed a parameter. In training the quantum kernel, a gradient-free optimizer SPSA (Simultaneous Perturbation Stochastic Approximation) was used [49]. It is particularly efficient for noisy quantum circuits. The *fit()* method was then used to train the QKT with the dataset.

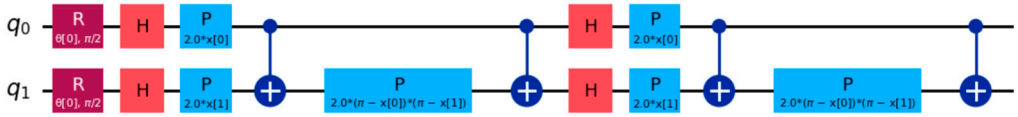


Figure 3. ZZFeatureMap with a custom rotational layer.

Similar to the QSVC model, the ZZFeatureMap and an ansatz were configured for the QNN model, as shown in Figures 4 and 5. A custom quantum circuit was created with a number of qubits equivalent to the feature dimension of our datasets. The ansatz was instantiated from the RealAmplitudes class library. A QNN was created from the SamplerQNN class and passed into the configured circuit as a parameter. We used the PQC method for the QNN model. A quantum classifier from the NeuralNetworkClassifier class that integrates an optimization algorithm COBYLA (Constrained Optimization by Linear Approximation) was used for training the QNN.

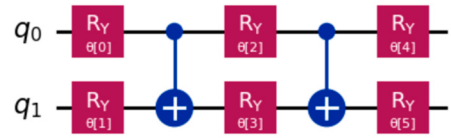


Figure 4. Ansatz generated for the ZZFeatureMap.

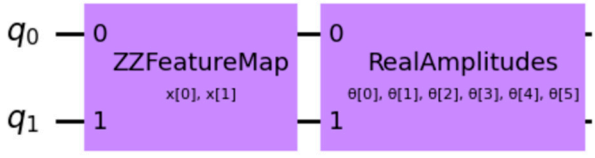


Figure 5. Two-qubit quantum circuit configured with ZZFeatureMap and ansatz.

4. Results

This section presents the experimental results for both the classical and quantum models for each dataset and compares their performance. The IBM quantum development environment and Qiskit software library were easy to use, and the available documentation

was comprehensive [42,50]. We used the standard metrics: accuracy, F1 score, precision, recall, and the area under the ROC (ROC-AUC). Accuracy measures the number of instances correctly classified, and it can be defined as below by using the confusion matrix values:

$$\frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

Precision is the proportion of correctly predicted positive observations to the total predicted positives (TP/(TP + FP)). Recall is the proportion of correctly labelled positive observations to all actual positives (TP/(TP + FN)), aka the True Positive Rate (TPR). F1 score is the harmonic mean of precision and recall, providing a single metric that represents a model's total class-wise accuracy. ROC-AUC is a metric that measures the performance of a classification model by plotting the TPR against the False Positive Rate (FPR) at various threshold settings.

4.1. Breast Cancer Dataset Performance Metrics

Table 2 shows the breast cancer dataset results for the seven models trained for this study, which include a classical SVC with four different kernels (linear, poly, rbf, and sigmoid), MLP, and the two quantum algorithms QSVC and QNN. In terms of accuracy, the SVC_Linear and MLP models performed the best with the highest accuracy of 95.26%, while QSVC had a competitive score of 93.86%. QNN performed the worst with the lowest accuracy of 77.19%. Other metrics were not significantly different.

Table 2. Results for the breast cancer dataset.

Models	SVC_Linear	SVC_Poly	SVC_RBF	SVC_Sigmoid	MLP	QSVC	QNN
Accuracy	95.26	90.17	94.03	90.34	95.26	93.86	77.19
Precision	96.17	87.67	94.32	91.67	96.13	93.86	77.19
Recall	96.36	98.31	96.35	93.27	96.36	93.73	76.44
F1 score	96.23	92.65	95.29	92.38	96.23	94.41	77.02
ROC-AUC	98.82	98.11	98.39	96.05	98.80	91.86	73.44

4.2. Diabetes Dataset Performance Metrics

Table 3 shows the diabetes dataset results for each model. Among the models, the MLP model performed the best with an accuracy score of 72.4%. SVC_Linear and SVC_RBF showed comparable performance with accuracy scores of 72.39% and 72.14%, respectively.

Table 3. Results for the diabetes dataset.

Models	SVC_Linear	SVC_Poly	SVC_RBF	SVC_Sigmoid	MLP	QSVC	QNN
Accuracy	72.39	69.40	72.14	62.63	72.40	70.78	67.53
Precision	64.89	74.48	65.40	46.5	65.27	70.78	67.53
Recall	45.90	17.92	42.54	47.01	45.92	69.26	60.35
F1 score	53.68	28.67	51.31	46.72	53.44	69.66	68.04
ROC-AUC	76.27	75.45	74.71	63.02	77.25	64.75	56.16

On the other hand, SVC_Poly and QNN showed lower accuracy scores of 69.4% and 67.53%, respectively. The SVC_Sigmoid model had the lowest accuracy at 62.63%, suggesting it is the least suitable for this task among the models tested. QSVC performed relatively well, with an accuracy of 70.78%, making it a competitive alternative in the quantum machine learning category. On the other hand, the F1 scores were higher for quantum models, which indicates that the quantum models had a good overall performance in binary classification.

4.3. Heart Disease Dataset Performance Metrics

Table 4 shows the heart disease dataset results for each model. For this dataset, the MLP model achieved the highest accuracy of 75.57%, closely followed by SVC_RBF, SVC_Linear, and SVC_Poly. The QNN model demonstrated the lowest accuracy of 58.33%, indicating that it was less useful in this application. QSVC exhibited moderate performance, with an accuracy score of 63.33%, which is better than QNN but still lags behind the other models. However, the other performance metrics were different, specifically recall and F1 score. QSVC performed best regarding F1 score, with a score of 77.49%.

Table 4. Results for the heart disease dataset.

Models	SVC_Linear	SVC_Poly	SVC_RBF	SVC_Sigmoid	MLP	QSVC	QNN
Accuracy	74.57	73.90	74.90	69.55	75.57	63.33	58.33
Precision	75.11	69.67	77.47	53.02	74.34	63.33	58.33
Recall	43.42	25.74	45.53	51.89	50.74	53.31	42.98
F1 score	47.26	33.14	48.88	51.24	52.95	77.49	34.03
ROC-AUC	83.79	82.51	77.32	72.69	83.76	56.00	50.00

5. Discussion

This section presents our interpretation of the results and limitations of the study. Conclusions are summarized and future research directions are highlighted as well.

5.1. Discussion and Limitations

Across the three datasets, while the classical algorithms showed similar or slightly better performance, the QSVC showed strength by outperforming some of the classical kernels in some metrics and doing much better than the QNN. Several factors could be responsible for the poor performances of the quantum algorithms in this study. Firstly, the complexities of the datasets may not be suitable enough for the requirements of QC; i.e., datasets are small, with a limited number of instances and feature dimensions as it was only two after PCAas little as two. We plan to continue the experiments with larger datasets and change the parameters to better understand the quantum processes.

Secondly, the limitations imposed by restricted access to quantum hardware and our use of a simulator could have an impact on the results. Access to real quantum hardware would have further enriched this study. We used IBM quantum platform and Qiskit, but more detailed analysis could be performed with other software packages and by comparing them, e.g., using Cirq by Google or Q# by Microsoft as an alternative quantum programming language.

Finally, IBM's ongoing updates to the quantum services and Qiskit package made model implementation a bit challenging as a user. For example, IBM announced the new Qiskit 1.2 around August 2024, and some classes were deprecated, e.g., BaseSamplerV1 and BaseEstimatorV1. This required continuous updates to the code, and some QML functions did not work with the updated version. Due to the emerging nature of the quantum programming language constructs, backward compatibility has been somewhat poor. For example, V2 primitives are not supported in version 0.7.2 of the Qiskit ML package, but the new 0.8.0 version, which was released in November 2024, fixed some of these issues.

These results highlight the fact that quantum computing, however promising, may not be suitable for all datasets and should not be considered a direct replacement for the classical computing paradigm; rather, its role is complementary. In other words, it would be an overkill to apply QC to simple tasks that can be accomplished by existing classical alternatives, but QC can be a promising solution when the layers of complexity increase, i.e., to tackle exponentially complex problems. We note that although quantum computing is more powerful than classical computing, for most trivial operations and small datasets, classical ML can be enough. Considering the required computing power and current

challenges with quantum hardware, one may prefer to use hybrid or classical models if there are alternative classical solutions with comparable performance.

5.2. Conclusions and Future Work

In this paper, we presented a study evaluating the performance of QML algorithms for binary classification and compared the results with classical counterparts using three datasets. The main objective of this study was to demonstrate the practical implementation and applicability of QML algorithms and compare the performance results to classical counterparts. The relevant literature was reviewed, and our methods were explained. The quantum processes of building a quantum feature map, quantum kernel estimation for QSVC, and parameterized quantum circuit for QNN were implemented. Evaluations of the models and comparative analysis of the performance metrics were presented. The results showed that QML algorithms can improve the performance of trained ML models, but there is not a significant quantum advantage for small datasets.

Although QC promises to revolutionize many fields, significant technical and theoretical challenges remain before practical and widespread use of quantum computers becomes a reality. QC in the NISQ era is prone to errors and noise. Quantum noise describes the unwanted disturbances that affect the quantum systems and lead to errors in quantum computation [51]. Even small amounts of noise can lead to decoherence, causing qubits to lose their superposition and entanglement properties. Quantum noise poses a significant barrier to the development of large-scale, fault-tolerant quantum computers. Advances in quantum error correction, qubit stability, scalability, and quantum algorithm development will be critical for the progression of this technology.

Using a classical computer or a supercomputer will potentially be the easiest and most economical solution for tackling ordinary problems, but on the other hand, many complex mathematical problems and machine learning challenges can benefit from the exponential power and quantum advantage provided by QC and QML. For a future study, we are planning to apply the QML to other classification problems with medium datasets and investigate how data size influences the performance of QML algorithms in both binary and multi-class classification tasks. This emerging field holds promise for solving complex classification problems, particularly in cases involving high-dimensional data. Additionally, we would like to explore QML for multi-labelled data, as multinomial classification adds another layer of complexity.

Author Contributions: Conceptualization, S.S.A. and D.C.; methodology, S.S.A. and D.C.; software, S.S.A.; validation, S.S.A. and D.C.; research and analysis, S.S.A.; writing—original draft preparation, review and editing, S.S.A. and D.C.; supervision, D.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: This study was conducted according to the ethical guidelines of Bournemouth University in the UK. This study did not involve humans or animals.

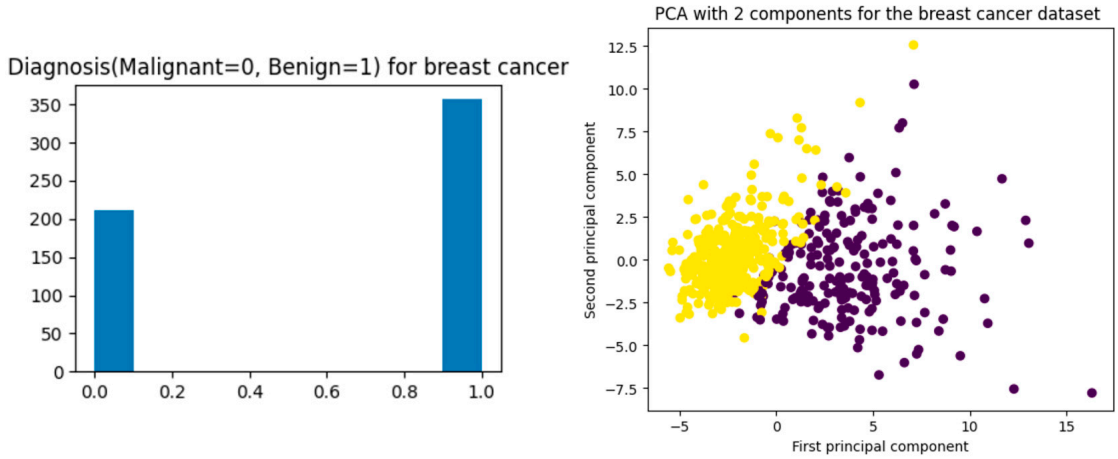
Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were used in this study, as explained in the methods section.

Conflicts of Interest: The authors declare no conflicts of interest.

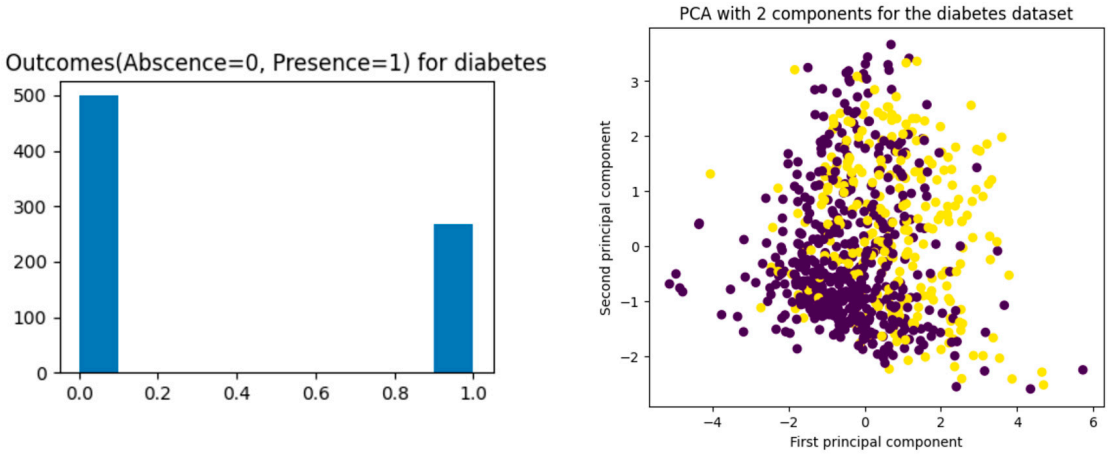
Appendix A

This appendix includes the target class balance analysis and PCA visualization of the three datasets. Figure A1 is for the breast cancer dataset, Figure A2 is for the diabetes dataset, and Figure A3 is for the heart disease dataset. Yellow and purple dots in PCA analysis show the classification.



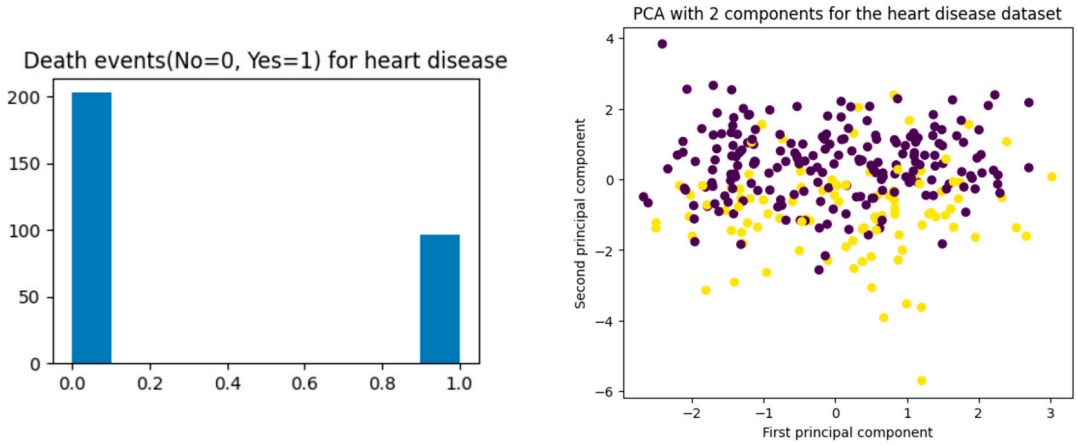
(**a**) (**b**)

Figure A1. Breast cancer dataset (**a**) target class balance; (**b**) PCA visualization.



(**a**) (**b**)

Figure A2. Diabetes dataset (**a**) target class balance; (**b**) PCA visualization.



(**a**) (**b**)

Figure A3. Heart disease dataset (**a**) target class balance; (**b**) PCA visualization.

References

1. Gill, S.S.; Kumar, A.; Singh, H.; Singh, M.; Kaur, K.; Usman, M.; Buyya, R. Quantum computing: A taxonomy, systematic review and future directions. *Softw. Pract. Exp.* **2022**, *52*, 66–114. [CrossRef]
2. Wu, X.; Xiao, L.; Sun, Y.; Zhang, J.; Ma, T.; He, L. A survey of human-in-the-loop for machine learning. *Future Gener. Comput. Syst.* **2022**, *135*, 364–381. [CrossRef]
3. Sahu, M.; Gupta, R.; Ambasta, R.K.; Kumar, P. Artificial intelligence and machine learning in precision medicine: A paradigm shift in big data analysis. *Prog. Mol. Biol. Transl. Sci.* **2022**, *190*, 57–100. [CrossRef] [PubMed]
4. Oztas, B.; Cetinkaya, D.; Adedoyin, F.; Budka, M.; Aksu, G.; Dogan, H. Transaction monitoring in anti-money laundering: A qualitative analysis and points of view from industry. *Future Gener. Comput. Syst.* **2024**, *159*, 161–171. [CrossRef]
5. Sabeur, Z.; Bruno, A.; Johnstone, L.; Ferjani, M.; Benaouda, D.; Arbab-Zavar, B.; Cetinkaya, D.; Sallal, M. Cyber-physical behaviour detection and understanding using artificial intelligence. In Proceedings of the 13th International Conference on Applied Human Factors and Ergonomics (AHFE'22), Cognitive Computing and Internet of Things, New York, NY, USA, 24–28 July 2022; Volume 67.
6. Bertolini, M.; Mezzogori, D.; Neroni, M.; Zammori, F. Machine learning for industrial applications: A comprehensive literature review. *Expert Syst. Appl.* **2021**, *175*, 114820. [CrossRef]
7. Verbraeken, J.; Wolting, M.; Katzy, J.; Kloppenburg, J.; Verbelen, T.; Rellermeyer, J.S. A survey on distributed machine learning. *ACM Comput. Surv.* **2020**, *53*, 30. [CrossRef]
8. Ali, S.; Yue, T.; Abreu, R. When software engineering meets quantum computing. *Commun. ACM* **2022**, *65*, 84–88. [CrossRef]
9. Hassija, V.; Chamola, V.; Saxena, V.; Chanana, V.; Parashari, P.; Mumtaz, S.; Guizani, M. Present landscape of quantum computing. *IET Quantum Commun.* **2020**, *1*, 42–48. [CrossRef]
10. Biamonte, J.; Wittek, P.; Pancotti, N.; Rebentrost, P.; Wiebe, N.; Lloyd, S. Quantum machine learning. *Nature* **2017**, *549*, 195–202. [CrossRef]
11. Ramezani, S.B.; Sommers, A.; Manchukonda, H.K.; Rahimi, S.; Amirlatifi, A. Machine learning algorithms in quantum computing: A survey. In Proceedings of the International Joint Conference on Neural Networks, Glasgow, UK, 19–24 July 2020.
12. Bayerstadler, A.; Becquin, G.; Binder, J.; Botter, T.; Ehm, H.; Ehmer, T.; Erdmann, M.; Gaus, N.; Harbach, P.; Hess, M.; et al. Industry quantum computing applications. *EPJ Quantum Technol.* **2021**, *8*, 25. [CrossRef]
13. Hughes, C.; Isaacson, J.; Perry, A.; Sun, R.F.; Turner, J. *Quantum Computing for the Quantum Curious*; Springer Nature: Cham, Switzerland, 2021.
14. Nielsen, M.A.; Chuang, I.L. *Quantum Computation and Quantum Information*, 10th ed.; Cambridge University Press: Cambridge, UK, 2010.
15. Wang, Y.; Hu, Z.; Sanders, B.C.; Kais, S. Qudits and high-dimensional quantum computing. *Front. Phys. Sec. Quantum Eng. Technol.* **2020**, *8*, 589504. [CrossRef]
16. Menon, S.G.; Glachman, N.; Pompili, M.; Dibos, A.; Bernien, H. An integrated atom array-nanophotonic chip platform with background-free imaging. *Nat. Commun.* **2024**, *15*, 6156. [CrossRef] [PubMed]
17. Grumbling, E.; Horowitz, M. (Eds.) *Quantum Computing: Progress and Prospects*; The National Academies of Sciences, Engineering and Medicine, The National Academies Press: Washington, DC, USA, 2019.
18. Bharti, K.; Cervera-Lierta, A.; Kyaw, T.H.; Haug, T.; Alperin-Lea, S.; Anand, A.; Degroote, M.; Heimonen, H.; Kottmann, J.S.; Menke, T.; et al. Noisy intermediate-scale quantum (NISQ) algorithms. *Rev. Mod. Phys.* **2022**, *94*, 015004. [CrossRef]
19. Bova, F.; Goldfarb, A.; Melko, R.G. Commercial applications of quantum computing. *EPJ Quantum Technol.* **2021**, *8*, 2. [CrossRef] [PubMed]
20. Smith, C.S. Forbes Top 10 Quantum Computing Companies Making Change—December 2023. Available online: https://www.forbes.com/sites/technology/article/top-quantum-computing-companies/ (accessed on 29 October 2024).
21. Elben, A.; Vermersch, B.; van Bijnen, R.; Kokail, C.; Brydges, T.; Maier, C.; Joshi, M.K.; Blatt, R.; Roos, C.F.; Zoller, P. Cross-platform verification of intermediate scale quantum devices. *Phys. Rev. Lett.* **2020**, *124*, 010504. [CrossRef] [PubMed]
22. Shor, P.W. Fault-tolerant quantum computation. In Proceedings of the 37th Conference on Foundations of Computer Science, Burlington, VT, USA, 14–16 October 1996; IEEE Computer Society Press: Washington, DC, USA, 1996.
23. Nimbe, P.; Weyori, B.A.; Adekoya, A.F. Models in quantum computing: A systematic review. *Quantum Inf. Process.* **2021**, *20*, 80. [CrossRef]
24. DiVincenzo, D.P. Quantum gates and circuits. *Proc. R. Soc. A Math. Phys. Eng. Sci.* **1998**, *454*, 261–276. [CrossRef]
25. McGeoch, C. *Adiabatic Quantum Computation and Quantum Annealing: Theory and Practice*; Part of the book series: Synthesis Lectures on Quantum Computing (SLQC); Springer: Berlin/Heidelberg, Germany, 2014.
26. Albash, T.; Lidar, D.A. Adiabatic quantum computation. *Rev. Mod. Phys.* **2018**, *90*, 015002. [CrossRef]
27. Rath, M.; Date, H. Quantum data encoding: A comparative analysis of classical-to-quantum mapping techniques and their impact on machine learning accuracy. *EPJ Quantum Technol.* **2024**, *11*, 72. [CrossRef]
28. Schuld, M.; Bocharov, A.; Svore, K.; Wiebe, N. Circuit-centric quantum classifiers. *Phys. Rev. A* **2020**, *101*, 032308. [CrossRef]
29. Munshi, M.; Gupta, R.; Jadav, N.K.; Polkowski, Z.; Tanwar, S.; Alqahtani, F.; Said, W. Quantum machine learning-based framework to detect heart failures in Healthcare 4.0. *Softw. Pract. Exp.* **2024**, *54*, 168–185. [CrossRef]
30. Maheshwari, D.; Sierra-Sosa, D.; Garcia-Zapirain, B. Variational quantum classifier for binary classification: Real vs synthetic dataset. *IEEE Access* **2022**, *10*, 3705–3715. [CrossRef]

31. Kavitha, S.S.; Kaulgud, N. Quantum machine learning for support vector machine classification. *Evol. Intell.* **2024**, *17*, 819–828. [CrossRef]
32. Suzuki, T.; Hasebe, T.; Miyazaki, T. Quantum support vector machines for classification and regression on a trapped-ion quantum computer. *Quantum Mach. Intell.* **2024**, *6*, 31. [CrossRef]
33. Yuan, X.-J.; Chen, Z.-Q.; Liu, Y.-D.; Xie, Z.; Liu, Y.-Z.; Jin, X.-M.; Wen, X.; Tang, H. Quantum support vector machines for aerodynamic classification. *Intell. Comput.* **2023**, *2*, 0057. [CrossRef]
34. Aly, M.; Fadaaq, S.; Warga, O.A.; Nasir, Q.; Talib, M.A. Experimental benchmarking of quantum machine learning classifiers. In Proceedings of the 6th International Conference on Signal Processing and Information Security (ICSPIS), Dubai, United Arab Emirates, 8–9 November 2023; pp. 240–245. [CrossRef]
35. Beer, K.; Bondarenko, D.; Farrelly, T.; Osborne, T.; Salzmann, R.; Scheiermann, D.; Wolf, R. Training deep quantum neural networks. *Nat. Commun.* **2020**, *11*, 808. [CrossRef]
36. Li, Y.; Wang, Z.; Han, R.; Shi, S.; Li, J.; Shang, R.; Zheng, H.; Zhong, G.; Gu, Y. Quantum recurrent neural networks for sequential learning. *Neural Netw.* **2023**, *166*, 148–161. [CrossRef]
37. Simoes, R.D.M.; Huber, P.; Meier, N.; Smailov, N.; Fuchslin, R.M.; Stockinger, K. Experimental evaluation of quantum machine learning algorithms. *IEEE Access* **2023**, *11*, 6197–6208. [CrossRef]
38. Desai, U.; Kola, K.S.; Nikhitha, S.; Nithin, G.; Raj, G.P.; Karthik, G. Comparison of machine learning and quantum machine learning for breast cancer detection. In Proceedings of the International Conference on Smart Systems for Applications in Electrical Sciences, Tumakuru, India, 3–4 May 2024. [CrossRef]
39. Reka, S.S.; Karthikeyan, H.L.; Shakil, A.J.; Venugopal, P.; Muniraj, M. Exploring quantum machine learning for enhanced skin lesion classification: A comparative study of implementation methods. *IEEE Access* **2024**, *12*, 104568–104584. [CrossRef]
40. Peral-García, D.; Cruz-Benito, J.; García-Peñalvo, F.J. Systematic literature review: Quantum machine learning and its applications. *Comput. Sci. Rev.* **2024**, *51*, 100619. [CrossRef]
41. Broughton, M.; Verdon, G.; McCourt, T.; Martinez, A.J.; Yoo, J.H.; Isakov, S.V.; Massey, P.; Halavati, R.; Niu, M.Y.; Zlokapa, A.; et al. TensorFlow Quantum: A software framework for quantum machine learning. *arXiv* **2020**, arXiv:2003.02989. Available online: http://arxiv.org/abs/2003.02989 (accessed on 31 October 2024).
42. Cervantes, J.; Garcia-Lamont, F.; Rodríguez-Mazahua, L.; Lopez, A. A comprehensive survey on support vector machine classification: Applications, challenges and trends. *Neurocomputing* **2020**, *408*, 189–215. [CrossRef]
43. IBM Qiskit Machine Learning Documentation. Available online: https://qiskit-community.github.io/qiskit-machine-learning/ (accessed on 30 October 2024).
44. Hasan, B.M.S.; Abdulazeez, A.M. A review of principal component analysis algorithm for dimensionality reduction. *J. Soft Comput. Data Min.* **2021**, *2*, 20–30. [CrossRef]
45. Greenacre, M.; Groenen, P.J.F.; Hastie, T.; D'Enza, A.I.; Markos, A.; Tuzhilina, E. Principal component analysis. *Nat. Rev.-Methods Primers* **2022**, *2*, 100. [CrossRef]
46. Ahsan, M.; Mahmud, M.; Saha, P.; Gupta, K.; Siddique, Z. Effect of data scaling methods on machine learning algorithms and model performance. *Technologies* **2021**, *9*, 52. [CrossRef]
47. Havlíček, V.; Córcoles, A.D.; Temme, K.; Harrow, A.W.; Kandala, A.; Chow, J.M.; Gambetta, J.M. Supervised learning with quantum-enhanced feature spaces. *Nature* **2019**, *567*, 209–212. [CrossRef]
48. Vasques, X.; Paik, H.; Cif, L. Application of quantum machine learning using quantum kernel algorithms on multiclass neuron M-type classification. *Sci. Rep.* **2023**, *13*, 11541. [CrossRef] [PubMed]
49. Bonet-Monroig, X.; Wang, H.; Vermetten, D.; Senjean, B.; Moussa, C.; Bäck, T.; Dunjko, V.; O'Brien, T.E. Performance comparison of optimization methods on variational quantum algorithms. *arXiv* **2021**, arXiv:2111.13454. Available online: http://arxiv.org/abs/2111.13454 (accessed on 31 October 2024). [CrossRef]
50. Javadi-Abhari, A.; Treinish, M.; Krsulich, K.; Wood, C.J.; Lishman, J.; Gacon, J.; Martiel, S.; Nation, P.D.; Bishop, L.S.; Cross, A.W.; et al. Quantum computing with Qiskit. *arXiv* **2024**, arXiv:2405.08810. [CrossRef]
51. Ball, H.; Biercuk, M.J.; Carvalho, A.R.R.; Chen, J.; Hush, M.; De Castro, L.A.; Li, L.; Liebermann, P.J.; Slatyer, H.J.; Edmunds, C.; et al. Software tools for quantum control: Improving quantum computer performance through noise and error suppression. *Quantum Sci. Technol.* **2021**, *6*, 044011. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.