

# Performance of QSVR Algorithm on NISQ-era Quantum Computers

Mateusz A. Kucharski \*

Department of Control and Quantum Computing, Wrocław University of Science and Technology, Wybrzeże Wyspiańskiego 27, Wrocław, 50-370, Lower Silesia, Poland.

**Abstract.** Quantum Support Vector Regression (QSVR) has emerged as one of the most promising quantum machine learning algorithms for nonlinear regression on Noisy Intermediate-Scale Quantum (NISQ) devices. However, despite numerous simulation-based studies, experimental evaluations on physical quantum hardware remain relatively scarce. This work presents a comparative analysis of three regression approaches: classical Support Vector Regression (SVR) with a Radial Basis Function (RBF) kernel, classical SVR utilizing precomputed quantum kernels, and the native QSVR implementation available in the `qiskit-machine-learning` framework. Experiments were conducted on three benchmark regression datasets using three execution environments: a noiseless Qiskit Aer simulator, the IQM Adonis hardware-aware simulator, and the physical 5-qubit IQM Spark superconducting quantum computer (Odra 5) located at Wrocław University of Science and Technology. Multiple quantum feature maps and circuit depths were investigated, while model hyperparameters were optimized using grid search. The obtained results demonstrate that QSVR can be successfully executed on contemporary superconducting quantum hardware and, for selected datasets, achieves prediction accuracy comparable to or exceeding that of a classical RBF-based SVR. The experiments further reveal the significant influence of quantum hardware noise, feature map selection, and circuit depth on regression quality. Moreover, the implementation based on precomputed quantum kernels generally exhibits greater robustness on physical hardware than the native QSVR implementation. The presented benchmark provides an experimental assessment of QSVR under realistic execution conditions and contributes to the ongoing evaluation of quantum machine learning algorithms on current NISQ-era quantum computers.

**Key words:** quantum computing; quantum machine learning; support vector regression; quantum algorithm

## 1. INTRODUCTION

In recent years, quantum computing has rapidly transitioned from a theoretical concept into a practical paradigm, driven by significant breakthroughs in physical hardware development. However, modern quantum devices operate in the Noisy Intermediate-Scale Quantum (NISQ) era, characterized by limited qubit counts, short coherence times, and non-negligible gate error rates. Despite these hardware constraints, the physical realization of quantum processors has sparked immense research interest in designing algorithms capable of executing on current systems.

Among these emerging applications, Quantum Machine Learning (QML) has gained substantial momentum. While much of the existing QML literature focuses on classification problems, the task of continuous value estimation – the regression problem – is equally crucial across domains such as finance, medical diagnostics, and physical science modeling. Adapting classical kernel-based estimators to leverage quantum feature maps offers a promising avenue for capturing complex data relationships.

To facilitate the development and deployment of QML models, several software ecosystems have emerged. The most widely adopted framework is `qiskit`, an open-source Python library spearheaded by IBM and increasingly supported by hardware vendors such as IQM and AQT. Among its tools, the `qiskit-machine-learning` package offers built-in support for Quantum Support Vector Regression (QSVR). Testing such implementation on physical quantum hardware is essential to

understand how real-world quantum effects influence performance compared to classical counterparts [1].

This work presents an empirical evaluation of QSVR across three distinct datasets: Iris (in a regression form), a Synthetic benchmark, and a Diabetes dataset. The performance of direct `qiskit` implementation is compared to a classical SVR with Radial Basis Function (RBF) kernel, as well as a hybrid approach using precomputed quantum kernels in a classical SVR. The evaluation encompasses noiseless simulations, noisy simulations, and execution on a physical quantum computer – Odra 5 located in Wrocław University of Science and Technology [2].

This study addresses three main research questions:

- **RQ1:** Can QSVR be run on current quantum hardware?
- **RQ2:** How does QSVR compare to the classical (RBF) version?
- **RQ3:** How does physical hardware noise impact the results?

## 2. LITERATURE REVIEW

The emergence of NISQ-era devices has led to a rapid development of Quantum Machine Learning algorithms, and while author's previous QML-focused reviews indicate the highest interest is in the field of classification [3, 4], Quantum Support Vector Regression has also emerged as a promising method for non-linear regression. Current literature primarily utilizes high-performance simulators like IBM's Qiskit Aer to validate these models, though studies have successfully deployed algorithms on real hardware such as the D-Wave Advantage quantum annealer [5], the 27-qubit IBM Quantum System One `ibmq_ehningen` [6], and the IonQ Harmony trapped-

\*e-mail: mateusz.kucharski@pwr.edu.pl

ion processor [7]. These methods are applied to diverse domains including price forecasting [8, 7], hydrogen storage in nanoporous materials [9], corrosion inhibition [10], and biophysical variable estimation [5]. In most implementations, the "quantum" component is the kernel estimation, where classical data is mapped into a high-dimensional Hilbert space to calculate fidelities [11, 12]. A notable exception is the work of Pasetto et al., which utilizes quantum annealing for the optimization phase by reframing regression as a Quadratic Unconstrained Binary Optimization (QUBO) problem [5]. Due to hardware constraints, research typically employs 2 to 12 qubits, often requiring dimensionality reduction through Principal Component Analysis (PCA) or autoencoders to map high-dimensional features to a limited qubit count. Real-world experiments highlight the significant impact of quantum noise; for instance, gate errors can cause predictive performance to drop to the level of random guessing unless error mitigation or low-rank kernel approximations are used. While simulated results often show QSVR outperforming classical SVR with RBF kernels in terms of error metrics, some studies on high-dimensional materials datasets find that classical SVR still maintains a slight edge in accuracy [9, 5].

### 3. DATASETS

To evaluate and compare the performance of classical SVR and QSVR, three distinct regression datasets are utilized. To accommodate execution constraints on quantum hardware and noisy quantum simulators, each dataset is truncated to  $N = 30$  samples, and the feature space is constrained to three dimensions to match a 3-qubit quantum feature map.

#### 3.1. Datasets Description

1. **Iris (Regression Variant):** Derived from the classical Iris classification dataset [13], this variant uses the first three geometric features (sepal length, sepal width, and petal length) as inputs to predict the continuous fourth feature, petal width. Because it inherently possesses three input features, no dimensionality reduction is applied.
2. **Diabetes:** A standard benchmark dataset comprising 10 original clinical features used to measure disease progression [14].
3. **Synthetic Non-Linear Dataset:** A custom synthetic regression benchmark generated using three features with added Gaussian noise ( $\sigma = 0.1$ ). To evaluate the ability of quantum kernels to capture complex functional dependencies, a highly non-linear target variable  $y$  is defined as:

$$y = \sin(X_0) + X_1^2 + \exp(0.5 \cdot X_2), \quad (1)$$

where  $X_0, X_1$ , and  $X_2$  represent the input feature dimensions.

#### 3.2. Preprocessing Pipeline

To ensure data consistency and hardware compatibility, all datasets undergo a unified preprocessing workflow prior to model training. For high-dimensional dataset *Diabetes*, PCA is applied to compress the input feature space down to three principal components, thereby ensuring direct alignment with

the fixed  $n$ -qubit topology of the target quantum backend. Following dimensionality reduction, each dataset is partitioned into an 80% training set (24 samples) and a 20% testing set (6 samples) using a fixed random seed to guarantee strict reproducibility across experimental runs.

Subsequently, feature scaling is performed to map the continuous input features ( $X$ ) into the interval  $[0, \pi]$ . This specific normalization range is crucial for quantum machine learning models, as the scaled values directly serve as rotation angles ( $\theta \in [0, \pi]$ ) for parameterized quantum gates within the quantum feature map circuits. Finally, the continuous target variable ( $y$ ) is standardized to achieve zero mean and unit variance ( $\mu = 0, \sigma = 1$ ) based on training set statistics. Standardizing  $y$  prevents numerical instability during hyperparameter optimization in both SVR and QSVR solvers.

## 4. CLASSICAL AND QUANTUM SUPPORT VECTOR REGRESSION

This section provides an overview of the regression algorithms considered in this work. First, the classical Support Vector Regression algorithm is introduced, followed by its quantum counterpart, Quantum Support Vector Regression, which extends the classical approach through the use of quantum kernels.

### 4.1. Classical Support Vector Regression

Support Vector Regression is the regression counterpart of the Support Vector Machine (SVM) algorithm and is widely used for predicting continuous target variables. Unlike traditional regression methods that minimize the squared prediction error, SVR constructs a function that deviates from the observed target values by no more than a predefined tolerance  $\epsilon$ , while simultaneously maximizing the flatness of the regression function. This formulation makes SVR particularly effective for nonlinear regression tasks and robust against overfitting.

Given a training dataset

$$\{(\mathbf{x}_i, y_i)\}_{i=1}^N, \quad (2)$$

where  $\mathbf{x}_i \in \mathbb{R}^d$  denotes the  $d$ -dimensional input feature vector and  $y_i \in \mathbb{R}$  is the corresponding target value, SVR seeks a regression function of the form

$$f(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b, \quad (3)$$

where  $\phi(\mathbf{x})$  maps the input data into a higher-dimensional feature space,  $\mathbf{w}$  is the weight vector, and  $b$  is the bias term.

The optimization problem is formulated as

$$\min_{\mathbf{w}, b, \xi_i, \xi_i^*} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N (\xi_i + \xi_i^*), \quad (4)$$

subject to

$$y_i - f(\mathbf{x}_i) \leq \epsilon + \xi_i, \quad (5)$$

$$f(\mathbf{x}_i) - y_i \leq \epsilon + \xi_i^*, \quad (6)$$

$$\xi_i, \xi_i^* \geq 0. \quad (7)$$

Here,  $C$  is the regularization parameter controlling the trade-off between model complexity and prediction accuracy,  $\varepsilon$  defines the width of the insensitive loss region, and  $\xi_i, \xi_i^*$  are slack variables allowing prediction errors larger than  $\varepsilon$ .

Instead of explicitly computing the mapping  $\phi(\mathbf{x})$ , SVR employs the kernel trick by replacing inner products in the transformed feature space with a kernel function

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j). \quad (8)$$

This enables efficient nonlinear regression without explicitly constructing the high-dimensional feature vectors. Commonly used kernels include the linear, polynomial, sigmoid, and radial basis function (RBF) kernels. The regression function obtained after solving the dual optimization problem is

$$f(\mathbf{x}) = \sum_{i=1}^N (\alpha_i - \alpha_i^*) K(\mathbf{x}_i, \mathbf{x}) + b, \quad (9)$$

where  $\alpha_i$  and  $\alpha_i^*$  are the learned Lagrange multipliers corresponding to the support vectors. Only training samples with non-zero coefficients contribute to the final prediction, making SVR computationally efficient during inference.

In this work, the classical regression model is implemented using the SVR class from the Scikit-learn library, which provides multiple kernel functions and uses the LIBSVM optimization framework.

#### 4.2. Quantum Support Vector Regression

Quantum Support Vector Regression (QSVR) extends the classical SVR framework by replacing the classical kernel function with a quantum kernel computed using parameterized quantum circuits. Rather than defining a mathematical similarity function explicitly, QSVR estimates similarities between input samples by encoding classical data into quantum states and evaluating their overlaps.

For an input vector  $\mathbf{x}$ , a quantum feature map prepares the quantum state

$$|\phi(\mathbf{x})\rangle = U_\phi(\mathbf{x})|0\rangle^{\otimes n}, \quad (10)$$

where  $U_\phi(\mathbf{x})$  is a parameterized quantum circuit that encodes the classical feature vector into an  $n$ -qubit quantum state.

The corresponding quantum kernel is defined as

$$K_Q(\mathbf{x}_i, \mathbf{x}_j) = |\langle \phi(\mathbf{x}_i) | \phi(\mathbf{x}_j) \rangle|^2, \quad (11)$$

which represents the fidelity between the two encoded quantum states. The resulting kernel matrix replaces the classical kernel matrix during training, while the optimization problem remains identical to that of classical SVR.

The resulting regression function is therefore

$$f(\mathbf{x}) = \sum_{i=1}^N (\alpha_i - \alpha_i^*) K_Q(\mathbf{x}_i, \mathbf{x}) + b. \quad (12)$$

The coefficients  $\alpha_i$  and  $\alpha_i^*$  are obtained using the same convex optimization procedure employed by the classical SVR algorithm.

In practice, the quantum kernel is evaluated by executing quantum circuits on either quantum simulators or quantum hardware. Each kernel value corresponds to an estimate of the fidelity between two quantum states prepared from different input samples. The computed kernel matrix is subsequently passed to a conventional SVR optimizer.

The implementation used in this work employs the QSVR class from the Qiskit Machine Learning library. Internally, QSVR inherits the optimization procedure from Scikit-learn's SVR while replacing the classical kernel computation with a quantum kernel. Consequently, both algorithms solve the same optimization problem and differ only in the method used to compute pairwise similarities between samples.

One of the principal motivations for QSVR is that quantum feature maps can embed classical data into Hilbert spaces whose dimensionality grows exponentially with the number of qubits. Such feature representations may be difficult to reproduce using classical kernels and therefore have the potential to improve regression performance for certain nonlinear problems. Nevertheless, the practical effectiveness of QSVR depends on factors including the selected quantum feature map, circuit depth, hardware noise, and the computational cost associated with constructing the quantum kernel matrix. Consequently, current applications of QSVR are primarily focused on relatively small datasets and low-dimensional feature spaces that are compatible with NISQ devices.

#### 5. METHODOLOGY

The experimental workflow was designed to compare three implementations of Support Vector Regression under identical preprocessing and evaluation conditions. The investigated methods were:

1. Classical SVR with the RBF kernel implemented in `scikit-learn`,
2. Classical SVR using a precomputed quantum kernel generated by `qiskit`,
3. QSVR implemented using the `qiskit-machine-learning` package.

For the quantum approaches, three execution environments were investigated in order to evaluate the influence of hardware noise on the obtained regression performance:

- a noiseless Qiskit Aer simulator without hardware topology constraints,
- the IQM Adonis simulator, reproducing the topology and noise characteristics of the IQM Spark superconducting quantum processor,
- the physical 5-qubit IQM Spark superconducting quantum computer (Odra 5) located at Wrocław University of Science and Technology.

Although the target hardware provides five logical qubits, all quantum experiments were performed using 3-qubit feature maps. This decision was made to minimize circuit depth introduced during transpilation, thereby reducing the accumulation of gate errors on NISQ hardware [15]. Consequently, all datasets were reduced to three input features, as described in

Section 3.2. Investigation of higher-dimensional feature maps and larger quantum circuits, including execution on quantum processors with a greater number of qubits, is left for future work.

Quantum feature encoding was realized using three feature maps available in Qiskit: ZFeatureMap, ZZFeatureMap, and PauliFeatureMap. Each feature map was evaluated with one and two circuit repetitions, resulting in six quantum circuit configurations for every dataset and execution backend.

For every quantum backend, feature map circuits were transpiled using the native Qiskit transpiler with optimization level 3 to satisfy the topology and native gate constraints of the target backend. Quantum kernel matrices were estimated using 1000 measurement shots for every kernel evaluation.

The hyperparameters of the regression models were selected using a grid search over multiple combinations of the regularization parameter  $C$  and the insensitive loss parameter  $\epsilon$ . The parameter combination yielding the highest coefficient of determination ( $R^2$ ) was selected for each method. For fairness, identical hyperparameter values were used when comparing the precomputed-kernel SVR and the native QSVR implementation.

All datasets were processed using the preprocessing pipeline described in Section 3.2. The coefficient of determination ( $R^2$ ) was adopted as the primary evaluation metric throughout all experiments.

The experiments were implemented in Python version 3.12 using `iqm-client` version 33.0.2, Qiskit version 1.2.4, `qiskit-machine-learning` version 0.8.0, and `scikit-learn` version 1.7.1. Classical simulations were executed on a workstation equipped with an AMD Ryzen 9 PRO 8945HS processor with Radeon 780M Graphics (4.00 GHz) and 32 GB RAM, whereas quantum experiments were executed on the 5-qubit IQM Spark superconducting quantum computer Odra 5 available at Wrocław University of Science and Technology.

IQM Spark operates on a native gate set consisting of single-qubit Phased  $RX$  ( $PRX$ ) arbitrary rotations and two-qubit Controlled-Z ( $CZ$ ) entangling gates. The median gate fidelities are 99.9% for native single-qubit gates ( $\epsilon_{1Q} = 0.1\%$ ) and 99.0% for native two-qubit  $CZ$  gates ( $\epsilon_{2Q} = 1.0\%$ ), with minimum reported median fidelities reaching 99.7% and 98.0%, respectively [2].

As summarized in Table 1, cumulative circuit error is overwhelmingly dominated by the accumulation of native two-qubit  $CZ$  gates required during transpilation [16]. This is heavily influenced by the chip’s physical topology (Figure 1), which features a star (central-hub) architecture where a single central qubit (QB3) is directly coupled to four surrounding qubits (QB1, QB2, QB4, QB5). Because direct two-qubit connections between outer qubits are unavailable, non-adjacent entangling operations must be routed through the central node, incurring additional gate overhead.

Consequently, non-entangling feature maps (such as ZFeatureMap) decompose solely into native single-qubit  $PRX$  gates, preserving shallow depths ( $\leq 5$ ) and keeping estimated total gate error under 1.5%. In contrast, entangling feature

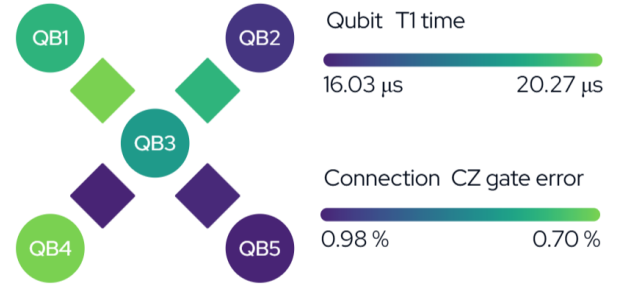


**Fig. 1.** Star-like qubit layout of the 5-qubit IQM processor, illustrating the central hub qubit (QB3) connected to the four peripheral qubits (QB1, QB2, QB4, and QB5) via  $CZ$  couplers. Note: The metric ranges shown ( $T_1$  times and  $CZ$  gate errors) represent an illustrative calibration snapshot from the control interface and do not reflect the exact system parameters at the time of the experiments.

**Table 1.** Transpiled circuit depth, native gate counts ( $PRX$  and  $CZ$ ), estimated circuit fidelity ( $F_{\text{circuit}}$ ), and total estimated gate error ( $E_{\text{total}}$ ) on the 5-qubit IQM processor based on typical median gate fidelities ( $F_{1Q} = 99.9\%$ ,  $F_{2Q} = 99.0\%$ ).

| Feature Map      | Depth | $PRX$ | $CZ$ | Est. $F_{\text{circuit}}$ | Est. $E_{\text{total}}$ |
|------------------|-------|-------|------|---------------------------|-------------------------|
| Z (reps = 1)     | 3     | 9     | 0    | 99.10%                    | 0.90%                   |
| Z (reps = 2)     | 5     | 15    | 0    | 98.51%                    | 1.49%                   |
| ZZ (reps = 1)    | 25    | 25    | 9    | 89.10%                    | 10.90%                  |
| ZZ (reps = 2)    | 51    | 46    | 18   | 79.70%                    | 20.30%                  |
| Pauli (reps = 1) | 25    | 25    | 9    | 89.10%                    | 10.90%                  |
| Pauli (reps = 2) | 51    | 46    | 18   | 79.70%                    | 20.30%                  |

maps (ZZFeatureMap and PauliFeatureMap) transpile into significantly deeper operational circuits (up to depth 51) requiring up to 18 native  $CZ$  operations at reps = 2, leading to an estimated cumulative gate error rate of 20.30% under typical median hardware performance.

## 6. RESULTS

This section presents the evaluation results of the QSVR model in two implementation variants: using a precomputed kernel matrix in the `scikit-learn` library and a direct implementation using `qiskit-machine-learning`. The results are compared with a classical SVR grid with an RBF kernel (computed on CPU) and presented in Table 2 and Table 3. The coefficient of determination  $R^2$  was adopted as the primary quality metric.

### 6.1. Comparison on the Iris Dataset

For the Iris dataset, the classical CPU SVR model with an RBF kernel achieved an  $R^2$  score of 0.920. Both quantum methods demonstrated an advantage over the classical approach. In the case of the precomputed kernel (Table 2), the noiseless simulation AER\_clean achieved  $R^2 = 0.959$  using the Z feature map with two repetitions. The real quantum processor (Odra\_QPU) achieved a very high score of  $R^2 = 0.946$  (also for Z, reps = 2), outperforming the noisy simulation Odra\_sim ( $R^2 = 0.935$ ). The `qiskit` implementation allowed achieving the highest score on the Odra\_sim simulator ( $R^2 = 0.966$ ), but on the physical Odra\_QPU system, the score dropped to 0.892.

**Table 2.** The best  $R^2$  value results achieved with Scikit-learn SVR with precomputed quantum kernel.

| Dataset          | Backend   | Kernel | Reps | $R^2$             |
|------------------|-----------|--------|------|-------------------|
| <b>Iris</b>      | CPU_SVR   | RBF    | –    | 0.920             |
|                  | AER_clean | Z      | 2    | 0.959             |
|                  | Odra_sim  | Z      | 1    | $0.935 \pm 0.002$ |
|                  | Odra_QPU  | Z      | 2    | $0.946 \pm 0.002$ |
| <b>Synthetic</b> | CPU_SVR   | RBF    | –    | 0.962             |
|                  | AER_clean | Z      | 2    | 0.955             |
|                  | Odra_sim  | Z      | 1    | $0.979 \pm 0.001$ |
|                  | Odra_QPU  | Z      | 1    | $0.531 \pm 0.015$ |
| <b>Diabetes</b>  | CPU_SVR   | RBF    | –    | 0.636             |
|                  | AER_clean | Pauli  | 1    | 0.623             |
|                  | Odra_sim  | Z      | 2    | $0.333 \pm 0.021$ |
|                  | Odra_QPU  | ZZ     | 2    | $0.647 \pm 0.014$ |

### 6.2. Comparison on the Synthetic Dataset

For the artificially generated dataset, the classical benchmark was  $R^2 = 0.962$ . Quantum simulations (AER\_clean and Odra\_sim) maintained very high precision, reaching a maximum of  $R^2 = 0.979$  (Odra\_sim, precomputed kernel, Z,  $reps = 1$ ). However, a drastic drop in performance was observed on the physical quantum processor: for the precomputed kernel approach, the  $R^2$  score fell to 0.531, whereas for the qiskit implementation, it was 0.562. In both cases, the Z feature map with one repetition proved to be the optimal feature map.

### 6.3. Comparison on the Diabetes Dataset

The Diabetes dataset proved to be the most demanding for the analyzed models, as reflected by the low score of the classical SVR ( $R^2 = 0.636$ ). The noisy Odra\_sim simulations recorded a significant drop in quality ( $R^2 = 0.333$  for precomputed and  $R^2 = 0.301$  for qiskit). Interestingly, on the physical Odra\_QPU processor in the precomputed variant with the ZZ map ( $reps = 2$ ), the highest score among all tested models was achieved ( $R^2 = 0.647$ ), slightly outperforming the classical algorithm. In the qiskit variant on Odra\_QPU, an  $R^2$  of 0.538 was obtained using the Pauli map ( $reps = 2$ ).

## 7. DISCUSSION

The obtained results allow us to formulate answers to the research questions posed in this work regarding the capabilities, efficiency, and noise resilience of QSVR algorithms on current NISQ-era quantum computers.

### 7.1. RQ1: Can QSVR be run on modern quantum hardware?

The conducted experiments demonstrate the technical feasibility of executing small-scale QSVR pipelines on modern 5-qubit superconducting quantum hardware (Odra\_QPU). Using an approach based on a quantum feature map and kernel matrix estimation allows for seamless integration with classical

**Table 3.** The best  $R^2$  value results achieved with qiskit-machine-learning QSVR.

| Dataset          | Backend   | Kernel | Reps | $R^2$             |
|------------------|-----------|--------|------|-------------------|
| <b>Iris</b>      | CPU_SVR   | RBF    | –    | 0.920             |
|                  | AER_clean | Z      | 2    | 0.962             |
|                  | Odra_sim  | Z      | 2    | $0.966 \pm 0.001$ |
|                  | Odra_QPU  | Z      | 2    | $0.892 \pm 0.003$ |
| <b>Synthetic</b> | CPU_SVR   | RBF    | –    | 0.962             |
|                  | AER_clean | Z      | 1    | 0.947             |
|                  | Odra_sim  | Z      | 2    | $0.976 \pm 0.001$ |
|                  | Odra_QPU  | Z      | 1    | $0.562 \pm 0.014$ |
| <b>Diabetes</b>  | CPU_SVR   | RBF    | –    | 0.636             |
|                  | AER_clean | Pauli  | 1    | 0.398             |
|                  | Odra_sim  | ZZ     | 1    | $0.301 \pm 0.025$ |
|                  | Odra_QPU  | Pauli  | 2    | $0.538 \pm 0.018$ |

SVR estimators. It is worth highlighting that the variant with the precomputed kernel matrix in scikit-learn proved to be more stable on physical QPU than the direct implementation from the qiskit-machine-learning library, suggesting greater flexibility and numerical robustness of traditional SVM optimizers when processing noisy quantum matrices.

### 7.2. RQ2: How does QSVR compare to the classical (RBF) version?

Comparing QSVR with the classical RBF kernel largely depends on the characteristics of the dataset:

- For simpler, lower-dimensional datasets (Iris), quantum kernels (particularly the simple Z map) not only match but outperform the classical RBF kernel, achieving  $R^2 > 0.94$  on real hardware compared to 0.920 for RBF.
- For synthetic data, simulated quantum kernels outperformed classical RBF (0.979 vs 0.962), confirming the higher expressivity of the quantum feature space.
- On more challenging real-world datasets (Diabetes), QSVR on physical hardware ( $R^2 = 0.647$ ) managed to outperform the classical benchmark (0.636) by leveraging more complex feature maps (ZZ).

This suggests that the linear Z feature map performs well for simpler correlations, whereas ZZ and Pauli maps are required to capture non-trivial dependencies in complex data.

### 7.3. RQ3: How does physical hardware noise impact the results?

The impact of quantum noise on regression quality is non-uniform and strongly depends on data structure as well as circuit depth (number of repetitions  $reps$ ):

1. **Performance degradation on sensitive data:** For the Synthetic dataset, noise on Odra\_QPU caused the  $R^2$  metric to drop from 0.979 (simulation) to  $\approx 0.53 - 0.56$ . This indicates high sensitivity to phase perturbations on physical

qubits for this type of data.

2. **Discrepancy between noisy simulation and physical hardware:** An unexpected divergence was observed for the Diabetes dataset, where the noisy simulation (Odra\_sim) yielded poor predictive performance ( $R^2 \approx 0.30 - 0.33$ ), whereas the physical processor (Odra\_QPU) achieved  $R^2 = 0.647$ . Rather than indicating an inherent algorithmic advantage, this discrepancy likely highlights limitations in standard simulated noise models, which may apply overly pessimistic depolarizing error relative to physical hardware. Alternatively, physical hardware fluctuations may introduce a form of random kernel smoothing, mitigating overfitting on small training sets. However, isolating the precise mechanism behind this effect would require further empirical evaluation across varying noise channels.
3. **Importance of the number of repetitions (*reps*):** A higher number of repetitions ( $reps = 2$ ) increases quantum circuit depth, compounding gate error accumulation; however, for complex datasets (Diabetes, Iris), it is necessary to achieve adequate model expressivity.

## 8. CONCLUSIONS

This work presented an experimental evaluation of Quantum Support Vector Regression on contemporary NISQ-era quantum hardware. Three regression approaches were compared: a classical SVR with an RBF kernel, a classical SVR using a precomputed quantum kernel, and the native QSVR implementation available in the `qiskit-machine-learning` library. The evaluation was conducted using noiseless simulation, a hardware-aware noisy simulator, and the physical 5-qubit IQM Spark superconducting quantum computer (Odra 5).

The obtained results demonstrate that QSVR can be successfully executed on present-day superconducting quantum hardware and that meaningful regression performance can be achieved despite the presence of hardware noise. For selected datasets, quantum kernels produced prediction accuracy comparable to or exceeding that of the classical RBF kernel, while other datasets exhibited a noticeable degradation caused by noise and limited hardware resources. The experiments also indicate that the implementation based on precomputed quantum kernels generally provides greater robustness on current quantum hardware than the native QSVR implementation, while preserving the advantages of quantum feature mapping.

The study further confirms that the choice of quantum feature map plays a significant role in the final regression performance. Simpler feature maps proved sufficient for low-complexity regression tasks, whereas more expressive feature maps became advantageous for more challenging datasets despite their increased circuit depth. These observations emphasize that selecting an appropriate feature map remains one of the key design decisions in practical QSVR applications.

Although the experiments were intentionally limited to 3-qubit feature maps to reduce transpilation overhead and circuit depth on NISQ hardware, the presented methodology establishes a reproducible framework for evaluating quantum regression algorithms on physical quantum processors. As quan-

tum hardware continues to improve, such benchmark studies will become increasingly important for understanding the practical capabilities and limitations of quantum machine learning algorithms.

## Future Work

Several directions for future research naturally emerge from this work. First, the experiments should be extended to larger quantum feature maps utilizing four or more qubits as larger superconducting quantum processors become available. This would enable investigation of whether the increased expressive power of higher-dimensional Hilbert spaces compensates for the additional circuit depth and accumulated hardware noise.

Second, the proposed benchmarking framework should be evaluated across different quantum computing platforms. In particular, comparing superconducting processors with trapped-ion systems (like AQTs 12-qubit IBEX Q1) and other qubit technologies would provide valuable insight into how hardware connectivity, native gate sets, coherence times, and noise characteristics influence QSVR performance [17]. Likewise, evaluating processors with different qubit topologies and larger qubit counts would help determine the scalability of quantum kernel methods beyond the constraints imposed by current 5-qubit devices.

Finally, an interesting application of QSVR lies beyond conventional regression benchmarks. One promising research direction is to employ QSVR as a surrogate model for predicting optimal chain strengths in graph embedding problems executed on quantum annealers. Such a model could learn the relationship between graph characteristics and the chain strength required for solving MaxCut instances on D-Wave systems, potentially reducing the expensive parameter search currently required before annealing [18]. This represents an attractive example of combining gate-based quantum machine learning with quantum annealing within a single hybrid optimization workflow.

## THE USE OF GENERATIVE AI AND AI-ASSISTED TECHNOLOGIES

AI models have been used to check and improve language and writing to improve readability.

## REFERENCES

- [1] M. Jędruszek, R. Idzikowski, and M. A. Kucharski, “Experimental analysis of quantum neural networks on odra 5 quantum computer,” in *International Conference on Dependability and Complex Systems*. Springer, 2026, pp. 186–195. [Online]. Available: [https://doi.org/10.1007/978-3-032-27927-9\\_18](https://doi.org/10.1007/978-3-032-27927-9_18)
- [2] J. Rönkkö, O. Ahonen, V. Bergholm, A. Calzona, A. Geresdi, H. Heimonen, J. Heinsoo, V. Milchakov, S. Pogorzalek, M. Sarsby *et al.*, “On-premises superconducting quantum computer for education and research,” *EPJ Quantum technology*, vol. 11, no. 1, p. 32, 2024. [Online]. Available: <https://doi.org/10.1140/epjqt/s40507-024-00243-z>

- [3] M. A. Kucharski, "A survey on quantum machine learning in speech acoustics," in *2025 IEEE 25th International Symposium on Computational Intelligence and Informatics (CINTI)*. IEEE, 2025, pp. 235–240. [Online]. Available: <https://doi.org/10.1109/CINTI67731.2025.11311733>
- [4] R. Idzikowski, M. A. Kucharski, K. Pempera, and M. Jaroszczuk, "A survey on quantum machine learning applications in medicine and healthcare," *Applied Sciences*, vol. 16, no. 3, p. 1630, 2026. [Online]. Available: <https://doi.org/10.3390/app16031630>
- [5] E. Pasetto, A. Delilbasic, G. Cavallaro, M. Willsch, F. Melgani, M. Riedel, and K. Michielsen, "Quantum support vector regression for biophysical variable estimation in remote sensing," in *IGARSS 2022-2022 IEEE international geoscience and remote sensing symposium*. IEEE, 2022, pp. 4903–4906. [Online]. Available: <https://doi.org/10.1109/IGARSS46834.2022.9883963>
- [6] K. Tschärke, S. Issel, and P. Debus, "Semisupervised anomaly detection using support vector regression with quantum kernel," in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, vol. 1. IEEE, 2023, pp. 611–620. [Online]. Available: <https://doi.org/10.1109/QCE57702.2023.00075>
- [7] T. Suzuki, T. Hasebe, and T. Miyazaki, "Quantum support vector machines for classification and regression on a trapped-ion quantum computer," *Quantum Machine Intelligence*, vol. 6, no. 1, p. 31, 2024. [Online]. Available: <https://doi.org/10.1007/s42484-024-00165-0>
- [8] H. Stühler, D. Pranjic, and C. Tutschku, "Evaluating quantum support vector regression methods for price forecasting applications," in *ICAART (3)*, 2024, pp. 376–384.
- [9] C. Chowdhury, "Exploring quantum support vector regression for predicting hydrogen storage capacity of nanoporous materials," *Advanced Intelligent Discovery*, p. e202500015, 2025. [Online]. Available: <https://doi.org/10.1002/aidi.202500015>
- [10] A. P. Santosa and M. Akrom, "Quantum support vector regression for predicting corrosion inhibition of drugs," *Journal of Multiscale Materials Informatics*, vol. 1, no. 2, pp. 30–34, 2024. [Online]. Available: <https://doi.org/10.62411/jimat.v1i2.11427>
- [11] X. Zhou, J. Yu, J. Tan, and T. Jiang, "Quantum kernel estimation-based quantum support vector regression," *Quantum Information Processing*, vol. 23, no. 1, p. 29, 2024. [Online]. Available: <https://doi.org/10.1007/s11128-023-04231-7>
- [12] G. Abdulsalam and I. Ahmad, "Comparative investigation of quantum and classical kernel functions applied in support vector machine algorithms: G. abdulsalam, i. ahmad," *Quantum Information Processing*, vol. 24, no. 4, p. 109, 2025. [Online]. Available: <https://doi.org/10.1007/s11128-025-04728-3>
- [13] E. Anderson, "The species problem in iris," *Annals of the Missouri Botanical Garden*, vol. 23, no. 3, pp. 457–509, 1936.
- [14] M. Kahn, "Diabetes," UCI Machine Learning Repository, DOI: <https://doi.org/10.24432/C5T59G>.
- [15] A. Gnatowski and T. Niżyński, "Quantum circuits compilation for superconducting quantum processing units: a systematic mapping study," in *International Conference on Intelligent Systems in Production Engineering and Maintenance*. Springer, 2025, pp. 269–284. [Online]. Available: [https://doi.org/10.1007/978-3-032-14163-7\\_19](https://doi.org/10.1007/978-3-032-14163-7_19)
- [16] A. Gnatowski, J. Rudy, T. Niżyński, and K. Świącicki, "Exact routing algorithm for quantum circuit compilation for star architectures," in *International Conference on Computational Collective Intelligence*. Springer, 2026, pp. 215–229. [Online]. Available: [https://doi.org/10.1007/978-3-032-37936-8\\_15](https://doi.org/10.1007/978-3-032-37936-8_15)
- [17] N. Mayo, T. Mor, and Y. Weinstein, "Benchmarking quantum computers via protocols, comparing superconducting and ion-trap quantum technology," *arXiv preprint arXiv:2603.27397*, 2026. [Online]. Available: <https://doi.org/10.48550/arXiv.2603.27397>
- [18] W. Bożejko, F. Górajewski, Y. Baloshenko, O. Siemieniuk, and J. Wielgus, "On quantum annealing chain strength related parameters control in d-wave for the single machine total weighted tardiness problem," in *International Conference on Dependability and Complex Systems*. Springer, 2026, pp. 43–54. [Online]. Available: [https://doi.org/10.1007/978-3-032-27927-9\\_5](https://doi.org/10.1007/978-3-032-27927-9_5)