I.J. Information Technology and Computer Science, **2025, 6, 143-159**
Published Online on December 8, 2025 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2025.06.08

Quantum–inspired Methods for Training Machine Learning Models

Nilesh T. Fonseka
Department of Mathematics, University of Colombo, Colombo 03, Sri Lanka
E-mail: nileshfonseka0@gmail.com
ORCID iD: https://orcid.org/0009-0002-0635-9260

Anuradha Mahasinghe*
Department of Mathematics, University of Colombo, Colombo 03, Sri Lanka
E-mail: anuradhamahasinghe@maths.cmb.ac.lk
ORCID iD: https://orcid.org/0000-0003-2828-6090
*Corresponding author

Received: 21 July 2025; Revised: 26 September 2025; Accepted: 13 November 2025; Published: 08 December 2025

Abstract: Machine learning model training, which ultimately optimizes a model's cost function is usually a time-consuming and computationally intensive process on classical computers. This has been more intense due to the increased demand for large-scale data analysis, requiring unconventional computing paradigms like quantum computing to enhance training efficiency. Adiabatic quantum computers have excelled at solving optimization problems, which require the quadratic unconstrained binary optimization (QUBO) format of the problem of interest. In this study, the squared error minimization in the multiple linear regression model is reformulated as a QUBO problem enabling it to be solved using D-wave adiabatic quantum computers. Same formulation was used to obtain a solution using gate-based algorithms such as quantum approximate optimization algorithm (QAOA) and sampling variational quantum eigensolver (VQE) im- plemented via IBM Qiskit. The results obtained through these approaches in the context of runtime and mean squared error(MSE) were analyzed and compared to the classical approaches. Our experimental results indicate a runtime ad- vantage in the D-wave annealing approach over the classical Scikit learn regression approach. The time advantage can be observed when $N > 524288$ compared to Sklearn Linear Regression and when $N > 65536$ compared to Sklearn SGDRegressor. Support vector machine induced neural networks, where the margin-based entropy loss is converted into a QUBO with Lagrangian approach is also focused in this study concerning the applicability for nonlinear models.

IndexTerms: Quadratic Unconstrained Binary Optimization, Quantum Approximate Optimization Algorithm, Sampling Variational Quantum Eigensolver, Multiple Linear Regression Model, D-wave, IBM Qiskit.

1. Introduction

Regression modeling is the core concept for many machine learning models that follow the least-square optimization method. For a set of given data that involves features (independent variables) and the target (dependent variable), the best-fit line is plotted in a way that minimizes the error between the data points and the line. This remains a computationally tedious task as the size of the data and features increases and conventional optimization methods in regression, such as Ordinary Least Squares (OLS) and Stochastic Gradient Descent(SGD), have failed to efficiently train the models due to their substantial processing power and hardware requirements. Hence, investigators are inclined to find different computing paradigms, primarily quantum computing which has the ability to perform parallel computations tackling many optimization problems that can be seen as a promising alternative.

Quantum computing operates based on the fundamental principles of quantum mechanics. Similar to how in classical computers the processing of information occurs using binary bits (0 and 1), quantum computers use quantum bits or qubits that exist in either of the states (0,1). It is called the superposition of states which carry out parallel calculations leading to exponential speedup in specific kinds of problems. Quantum computing can be classified into three main sub-roots: quantum circuit, quantum annealing, and quantum walk model. The quantum circuit model performs calculations analogous to classical logic gates, applying unitary operations to qubits. Unitary operations maintain probability amplitudes such that the overall probability remains one, supporting reversibility, an essential difference from classical computing.

Companies like IBM and Google have built quantum processors that rely on this model, making their servers available on cloud platforms using API authentication. Quantum annealing, on the contrary, is designed mainly to solve combinatorial optimization problems using quantum mechanical effects such as superposition, tunneling, and entanglement. Quantum annealing is based on the quantum adiabatic theorem [1], which forms the basis for adiabatic quantum computing. A key player in developing and marketing such machines is D-Wave Systems. The strategy is to prepare the system in an easy-to-compute ground state of an elementary Hamiltonian and slowly evolve it to the Hamiltonian for the problem in question, homotopically evolving the system to the global minimum of the problem [2].

Quantum annealers such as D-wave are adept at approximately solving quadratic unconstrained binary optimization (QUBO) problems. To better understand QUBO, it is helpful to first consider some foundational ideas from computational complexity theory. In complexity theory, problems can be classified into two classes, which are polynomial time (P) and non-deterministic polynomial time (NP). Polynomial Time (P) problems can be interpreted as problems that can be solved using deterministic resources in polynomial time, whereas Nondeterministic Polynomial (NP) problems are problems that can be verified in polynomial time. However, solving these problems requires nondeterministic computers [3]. Simply, these are the problems that may not be polynomial efficient problems. Within this NP class, NP-hard problems are those that are at least as difficult as the hardest problems in NP. That is, their solutions may not be easily verifiable. Combinatorial problems such as the traveling salesman problem, and graph partitioning are some of the NP-hard problems. Since annealers perform operations in high-dimensional tensor product spaces and approximate solutions to some NP-hard problems such as quadratic unconstrained binary optimization (QUBO), [2] problems such as least-square optimization must first be translated into the QUBO format before they can be processed by adiabatic quantum systems. This format will be later discussed in the methodology chapter where a detailed formulation is presented.

This paper focuses on two approaches mainly, quantum annealing method and quantum gate based method. The least square optimization in multiple regression model is reformulated into a QUBO problem and then solved using D-wave open source annealers. Also, in the gate based method, the solution is obtained via IBM Qiskit QAOA and Sampling VQE. The obtained solutions from both methods is then benchmarked against the classical solution methods where the performance of these models is analyzed to the scalability of datapoints and features. Contributions of the proposed method are as follows:

- Reformulating least-square optimization in multiple regression model for a synthetic data set into a quadratic un- constrained binary optimization (QUBO) problem.
- Solving the formulated QUBO using D-wave's open source simulated annealing sampler to replicate quantum annealing, and also using IBM Qiskit QAOA and Samplng VQE algorithms.
- Benchmarking of the solutions against classical regression algorithms from Scikit-Learn Linear Regression and Stochastic Gradient Descent (SGD) to the scalability of data points and features.
- Proposing a QUBO formulation for minimizing the margin based loss in support vector machine induced neural network using the Lagrangian approach.

The remaining sections are organized as follows: The second part is related to work and reviews on different quantum approaches for advancements in machine learning. The third part illustrates the methodology of this work which includes phases of the process, whereas the fourth section discusses the benchmarking results between classical and quantum approaches. Finally, the last section summarizes the conclusions and discussion of this work.

2. State of the Art

Quantum annealing has provided promising solutions to many problem instances of decision and optimization problems [4, 5, 2, 6]. It is considered as the practical pathway to adiabatic quantum computing [7] It formulates the problem into an Ising Problem or a QUBO to facilitate its implementation on platforms like D-Wave QPU(Quantum Proccessing Unit) [8]. In 2013 Lloyd, Mohseni, and Rebentrost presents a quantum version of Lloyd's algorithm for performing k-means clustering: using a novel version of the quantum adiabatic algorithm one can classify M vectors into k clusters in time $O(k\log kMN)$ which provides exponential speed up over classical algorithms[9]. In 2017 Mott, Job, Vlimant, et al. explains the solution of the Higgs signal versus background machine learning optimization problem, mapped to the problem of finding the ground state of a corresponding Ising spin model[10]. Date, Arthur, and Pusey-Nazzaro in 2021 explains linear regression, support vector machine (SVM) and balanced k-means clustering as QUBO problems. For linear regression it is said to have a time complexity of $O(Nd^2)$, for N no. of data points and d no. of features which is equivalent to the classical approach. For support vector machine time complexity of $O(N^2)$ which is better than the classical approach $O(N^3)$. The transformations for linear regression include turning euclidean error to QUBO and for support vector machine turning Lagrangian minimization to QUBO.[2]. Similarly [11] explains adiabatic linear regression further which formulates euclidean error to QUBO. N. Ide, M. Ohzeki explains an l0-regularized linear regression for a sparse signal reconstruction implemented based on QUBO formulations [12]. Also van Dommelen, M.R and Phillipson, F. introduces a QUBO formulation for sparse sensor placement for facial image classification [13].

Recent Literature on quantum circuits for machine learning mainly includes mapping inputs/data to a quantum Hilbert space. This is similar to the kernel method used in the support vector machine where the input data is mapped to a high-

dimensional feature space where it becomes easier to make the classes linear separable[14]. In the year 2019 Schuld and Killoran proposes two approaches to build a quantum classifier. The first approach, quantum device estimates inner products of quantum states to compute a classically intractable kernel which can be fed into any classical kernel method such as a support vector machine. The second approach, a variational quantum circuit was used as a linear model that classifies data explicitly in Hilbert space [14]. Variational quantum circuits are one of the main classes in NISQ algorithms also known as near-term intermediate-scale quantum algorithms. They are fixed-size circuits with variable parameters that can be optimized to solve a given task. These variational or parametrized circuits are sometimes also known as quantum neural networks and due to the probabilistic nature of quantum measurements, this suggests that variational circuits can be used as an analogue for generative models[15]. Chen, Yang, Qi, et al. in the year 2020 proposed a framework for deep Q-learning inspired by variational quantum circuits. In addition, it reduces the number of model parameters using a quantum information encoding scheme compared to classical neural networks.proposed framework shows the quantum advantage in terms of less memory consumption and the reduction of model parameters. Specifically for the testing environments considered in this paper, the variational quantum deep Q -learning involves parameters as small as $O(n)$, but the tabular Q-learning and neural network-based deep Q-learning have parameters $O(n^3)$and $O(n^2)$, respectively.[16] QAOA which was proposed by Farhi, Goldstone, and Gutmann which approximate solutions for combinatorial optimization problems

[17-19], is computationally universal[20] and has been used to train unsupervised machine learning models [21]. Solving linear systems has been a key element in machine learning. For a system of linear equations $Ax = b$ with $A \in \mathbb{R}^{N\times N}$ and $x, b \in \mathbb{R}^N$, the best classical algorithm has a runtime of $\tilde{O}(N^{2.373})$ [22]. However, The quantum linear system algorithm (QLSA)[22] also known as HHL after the three authors Harrow, Hassidim, and Lloyd, promises to solve the problem in $\tilde{O}(\log(N)k^2s^2/\epsilon)$, where k is the condition number (defined to be the ratio of the largest to the smallest eigenvalue), s is the sparsity or the maximum number of nonzero entries in a row and column of A, and ϵ is the precision to which the solution is approximated.

3. Methodology

3.1. Analysis of Existing Linear Regression Techniques

The main goal of squared error minimisation or any cost function minimisation is to find model parameters also known as weights and biases that best fit the model, which is being trained over the training set. In the context of Linear Regression there are two approaches. Mainly the direct "closed form" solution approach and iterative optimisation approach called gradient descent. In the context of Neural networks this iterative approach is also called as Backpropagation.

The "closed form" solution is mainly used in Scikit-Learn Linear regression class and Stochastic gradient descent is used in their SGDRegressor class. Both methods are efficient algorithms on predicting results however with a large number of features or datapoints Stochastic gradient descent might be useful. A regression model is given by

$$\hat{y} = w_1x_1 + w_2x_2 + w_3x_3 + \ldots + w_dx_d + b \tag{1}$$

where $\hat{y}$ is the predicted value, d is the number of features, w_i is the ith feature weight, x_i is the ith feature value and b which is bias. Therefore, the squared error between the predicted and the actual can be represented in the vector form $||Xw - Y||^2$. T The closed form solution w is then given by $(X^T X)^{-1}X^T Y$. However, if $(X^T X)^{-1}$ doesn't exist, the pseudo inverse is computed. The pseudo inverse X^+ is calculated using a standard matrix factorization technique called *Singular Value Decomposition* where X is represented as $U\Sigma V^T$ and X^+ as $V\Sigma^+U^T$.

The gradient descent approach as an iterative algorithm converges to the solution by calculating the gradient vector at each step. When mean squared error,

$$\text{MSE}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}\left(w^T\mathbf{x}^{(i)} - y^{(i)}\right)^2 \tag{2}$$

is given for the number of N samples in the training set, the partial derivative of the cost function can be calculated as

$$\frac{\partial}{\partial w_j}\text{MSE}(\mathbf{w}) = \frac{2}{N}\sum_{i=1}^{N}\left(w^T\mathbf{x}^{(i)} - y^{(i)}\right)x_j^{(i)} \tag{3}$$

The gradient vector can then be calculated as

$$\nabla_{\mathbf{w}}\text{MSE}(\mathbf{w}) = \begin{pmatrix} \frac{\partial}{\partial w_0}\text{MSE}(\mathbf{w}) \\ \frac{\partial}{\partial w_1}\text{MSE}(\mathbf{w}) \\ \vdots \\ \frac{\partial}{\partial w_n}\text{MSE}(\mathbf{w}) \end{pmatrix} = \frac{2}{N}\mathbf{X}^T(\mathbf{X}\mathbf{w} - \mathbf{Y}) \tag{4}$$

and at the gradient descent step

$$w_{nextstep} = w - \eta \nabla_{\mathbf{w}} \mathrm{MSE}(\mathbf{w}) \tag{5}$$

where η is the learning rate which determine the size of the downhill step towards the optimal solution. If the learning rate is too high it causes the algorithm to diverge, and if it's too low it takes too many iterations to converge. This way, we can find the required weights and biases using both the closed form solution method and the gradient descent method. [23]

3.2. Conversion of the Cost Function to the Quadratic Unconstrained Binary Optimization Problem

As discussed in 3.1 we can represent the cost function for the multiple regression model as

$$E(w) = ||Xw - Y||^2 \tag{6}$$

where, $X \in \mathbb{R}^{N\times(d+1)}$ is the training dataset; $w \in \mathbb{R}^{d+1}$ the regression weights and bias; $Y \in \mathbb{R}^N$ the training labels; N the number of training samples in the training set, and d is the number of features in the data set. Since the closed-form solution as discussed is given by

$$w = (X^T X)^{-1}(X^T Y) \tag{7}$$

we can reformulate the cost function as

$$\min_{w\in\mathbb{R}^{d+1}} E(w) = w^T X^T X w - 2 w^T X^T Y + Y^T Y \tag{8}$$

Then a K dimentional precision vector

$$P = [p_1, p_2, p_3, \ldots, p_k]^T \tag{9}$$

where each entry in P can be an integral power of 2 which can be both positive or negative. Next, a K number of binary variables $\widehat{w}_{ik}$ is introduced for each of the $d+1$ regression weights and bias w_i such that

$$w_i = \sum_{k=1}^{K} p_k \, \widehat{w}_{ik} \quad \forall\, i = 1,2,\ldots,d+1, \tag{10}$$

p_k denotes the kth entry in the precision vector P where $\widehat{w}_{ik}$ is a binary decision variable that selects or ignores p_k based on its value is 1 or 0 respectively. This summation can be re written in the matrix form as follows

where

$$w = \mathcal{P}\widehat{w} \tag{11}$$

$$\mathcal{P} = I_{d+1} \otimes P^T \tag{12}$$

is a $(d+1)\times K(d+1)$ precision matrix obtained using the Kronecker product of I_{d+1} and P^T and

$$\widehat{w} = [\widehat{w}_{11}, \ldots, \widehat{w}_{1K}, \widehat{w}_{21}, \ldots, \widehat{w}_{2K}, \ldots, \widehat{w}_{(d+1)1}, \ldots, \widehat{w}_{(d+1)K}]^T \tag{13}$$

is the vector containing all binary decision variables $\widehat{w}_{ik}$. Since the weights and bias are expressed as a product of $\mathcal{P}$ and $\widehat{w}$ we can express (8) as a QUBO problem below

$$\min_{\widehat{w}\in\mathbb{R}^{(d+1)K}} E(\widehat{w}) = \widehat{w}^T \mathcal{P}^T X^T X \mathcal{P}\widehat{w} - 2\widehat{w}^T \mathcal{P}^T X^T Y. \tag{14}$$

$Y^T Y$ is left out because it's a constant scalar and doesn't affect the optimal solution of the problem of interest.

3.3. QUBO Representation

D-wave adiabatic quantum computers solve QUBO problems which is stated as below

$$\min_{z\in\mathbb{B}^M} z^T A z + z^T b \tag{15}$$

where, M is a natural number; $\mathbb{B} = \{0,1\}$ is the set of binary numbers; $z \in \mathbb{B}^M$ is the binary decision vector; $A \in \mathbb{R}^{M\times M}$ is the real-valued $M \times M$ QUBO matrix and $b \in \mathbb{R}^M$ is the real-valued M-dimensional QUBO vector.

Hence (14) can be represented in the above matrix form $z^T Az + z^T b$ where A and b represent the quadratic and linear terms separately. From (14) $A = \mathcal{P}^T X^T X \mathcal{P}$, $b = -2\mathcal{P}^T X^T Y$ and $z = \widehat{w}$. This representation can be also expressed as $z^T Qz$ where Q_{ij} corresponds to A_{ij} $\forall\, i,j$ where $i \neq j$ and Q_{ii} corresponds to $A_{ii} + b_i$. This can be explained further using the summation form of $z^T Az + z^T b$ and $z^T Qz$. The summation form of $z^T Az + z^T b$ is given as

$$\sum_{i=1}^{M} \sum_{j=1}^{M} A_{ij}\, z_i z_j + \sum_{i=1}^{M} b_i\, z_i \tag{16}$$

where A is a $M \times M$ dimentional matrix and b is a M dimentional vector. Also, the summation form of $z^T Qz$ is given similarly

$$\sum_{i=1}^{M} \sum_{j=1}^{M} Q_{ij}\, z_i z_j \tag{17}$$

Due to the binary nature of the problem $\sum_{i=1}^{M} b_i\, z_i$ corresponds to $\sum_{i=1}^{M} b_i\, z_i z_i$. Therefore, the linear term b_i is added to the diagonal elements of Q. Also, it is common to assume Q matrix is symmetric or in upper triangular form. In the symmetric form for all i and j except $i = j$, Q_{ij} is replaced by $(Q_{ij} + Q_{ji})/2$ whereas the Upper triangular form replace Q_{ij} by $(Q_{ij} + Q_{ji})$ for all i and j where $j > i$ and then replace all Q_{ij} for $j < i$ by 0 [25]. In this experiment the Q matrix is made symmetric using the Upper triangular form.

3.4. Finding Solution to the QUBO Problem

After formulating QUBO, finding solutions follows mainly three approaches. That is, Quantum Annealing, Quantum Approximate Optimization Algorithm, and SamplingVQE algorithm. All these methods have their own quantum computing principles to approach the solution, and in this section each of the methods will be discussed in the subsequent sections.

A. Quantum Annealing Approach

This section explores quantum annealing as a solution technique to solve the formulated QUBO problem. Quantum annealing has been a very useful technique because of its energy and stability, which uses quantum mechanical fluctuations to search for the solution of an optimization problem. It shares the basic idea with the quantum adiabatic evolution actively studied in quantum computation.[1].

Founded in 1999, D-Wave has been a pioneer company in delivering quantum computing solutions and systems mainly quantum annealers. They claim to be the first commercial quantum computer and they also provide solutions as a *Software as as Service (SaaS)* via cloud. Users interact with a web user interface or open-source access *dwave-ocean sdk* which uses a Solver API (SAPI) to connect to the backend of QPU's and additional solvers, located in different regions in the world.*dwave-ocean sdk* provides samplers which are compatible with both classical, quantum and hybrid approaches.

D-Wave provides some common samplers namely *DWaveSampler*, *DWaveCliqueSampler*, *LeapHybridSampler*, *LeapHybridCQMSampler*, *LeapHybridDQMSampler* which accepts the problem in *binary qaudratic model (BQM)* or *discrete quadratic model (DQM)* format and returns variable assignments.[26]. D-wave also provides some classical samplers on their open source *dwave-ocean sdk* platform which includes samplers namely *PlanarGraphSolver*, *SimulatedAnnealingSampler*, *RandomSampler* etc. The classical simulation techniques of the simulated annealing algorithm from *SimulatedAnnealingSampler* using *neal* package in D-wave was used to find solutions to formulated QUBO under the quantum annealing approach. When implementing this on real D-wave hardware, we need to consider the available network topology (Chimera, Pegasus or Zephyr) and embedding of the problem has to be done accordingly.

B. Quantum Approximate Optimisation Algorithm (QAOA) Approach

As an alternative approach QAOA was explored to find the solution to the formulated QUBO. This provides both classical and quantum hardware simulation approach through *IBM Qiskit* Platform. QAOA mainly finds approximate solutions to hard combinatorial optimization problems by encoding the Hamiltonian problem into a quantum circuit and leveraging adiabatic time evolution and layering it optimize the variational parameters of the circuit such that the approximate solution can be constructed by measuring the QAOA circuit. [27]

The solutions were obtained connecting the IBM QPU's backend in IBM quantum platform using a Token API key which provides access to their free QPU's namely *ibm sherbrooke*, *ibm kyiv* and *ibm brisbane*. Optimizers such as *COBYLA* were used in the solution process.

C. Sampling Variational Quantum Eigensolver (VQE) Approach

Variatonal quantum eigensolvers is mainly a quantum-classical hybrid approach which finds eigenvalues of a Hamiltonian proposed as an alternative to fully quantum algorithms like quantum phase estimation which requires much bigger hardware requirement. They have been effective in solving the electronic Schrodinger equation for a variety of small molecules, and the scalability mainly depends on the quantum circuit complexity and classical optimization complexity. Therefore, the choice of variational quantum circuit *ansantz* is crucial, which affects both of these factors[28].

IBM Qiskit optimization provides automatic conversion of QUBO to Ising Hamiltonian using there inbuilt function which then allows the use of sampling minimum eigensolver implementations such as SamplingVQE. After conversion, a suitable *ansatz* is chosen employing optimizers like *SPSA* for the solution process [29, 30].

D. Benchmarking of the Reformulated Solutions

It is crucial to test the effectiveness of the process through a dataset to perform the evaluations. Therefore, at the initial stage of the benchmarking process, a synthetic dataset was generated through the scikit learn standard methods with a noise introduced to replicate a real dataset. The dataset consists of varying datapoints N upto 4194304 and features d differing from 2 to 20.

As a preprocessing step, the data was standardized using *sklearn StandardScaler* as stochastic gradient descent is sensitive to feature scaling [31]. The dataset was then split into training and testing sets in an 80:20 ratio using *sklearn train_test_split*. The QUBO conversion is done for the training set, and the binary solution obtained will be reshaped. Matrix multiplication between the binary solution and the precision matrix was performed to obtain the quantum weights and finally predictions were made on the test set with those weights.

4. Experimental Results

4.1. QUBO Modeling of Linear Regression using D-wave Annealing (neal)

This section discusses the results obtained by running the simulation using *Dwave's SimulatedAnnealingSampler* compared to *sklearn LinearRegression* and *SGDRegressor*. Here, the results were obtained for the scalability of both data points and features. For the datapoints experiment since the precision (K) is 12 bits and the dimensions are 3 $(d + 1)$ the qubit utilization would be 36 qubits $K(d + 1)$.

A. Time and MSE Comparison with the Scalability of Data Points

The parameters used for this section are given below in Table 1.

Table 1. Parameters used for running D-wave simulated annealing for linear regression

Parameter	Value
Number of Features (d)	2
Precision Vector (P)	[-2, -1, -0.5, -0.25, 0.25, 0.5, 1, 2, 4, 8, 16, 32]
QUBO Simulator	SimulatedAnnealingSampler
Classical Algorithms	sklearn.LinearRegression, SGDRegressor

Table 2 refers to the runtime for both classical and D-wave annealing methods to the scalability of data points which doubles the number of data points each time. The values listed in Table 3 summarizes the mean squared error comparison of both methods. Figure 1 and Figure 2 plots the values expressed in the below tables.

Table 2. Time comparison of classical vs Dwave SA to the scalability of datapoints

N	Sklearn LinearRegression (ms)	SGDRegressor time (ms)	Dwave SA time (ms)
64	1.029253	0.000000	12.666941
128	0.999689	0.765966	10.557413
256	1.535416	2.800941	10.846853
512	0.999689	0.000000	11.684179
1024	3.005981	2.005577	9.979010
2048	3.009796	0.000000	9.859085
4096	2.262115	3.877640	10.587454
8192	2.235889	3.715992	9.394646
16384	4.666567	7.139683	8.791685
32768	3.512383	13.588190	13.863087
65536	2.998352	24.169922	10.824919
131072	5.960941	48.830986	16.350508
262144	9.532690	90.140104	14.075756
524288	22.192001	466.681719	19.797802
1048576	49.074411	1124.238253	19.423485
2097152	92.510462	2475.666285	22.844076
4194304	193.397522	4974.416971	31.640291

Table 3. MSE comparison of classical vs Dwave SA to the scalability of datapoints

N	Sklearn LinearRegression MSE	SGDRegressor MSE	Dwave SA MSE
64	27.29163857	27.81671827	27.00650933
128	13.31065956	13.43409221	12.98515383
256	17.32734716	17.30247359	17.54397392
512	22.72084343	22.75847746	22.75847746
1024	23.41554408	23.98329379	23.40580598
2048	25.41194184	26.51188626	25.45283901
4096	25.97420203	26.22361983	857.76707114
8192	24.79032455	24.93267668	24.93267668
16384	25.07708944	25.07550555	25.08215717
32768	24.64638022	24.65516383	115.68797690
65536	25.67379495	25.98703362	25.66228630
131072	24.88664939	25.05029397	24.89526502
262144	25.13878869	25.25261166	25.14636095
524288	24.93393494	24.94721316	24.94094687
1048576	25.20727932	25.28152085	25.21138537
2097152	25.00330739	25.17264298	78.81952999
4194304	24.94680502	24.96297860	24.95595396

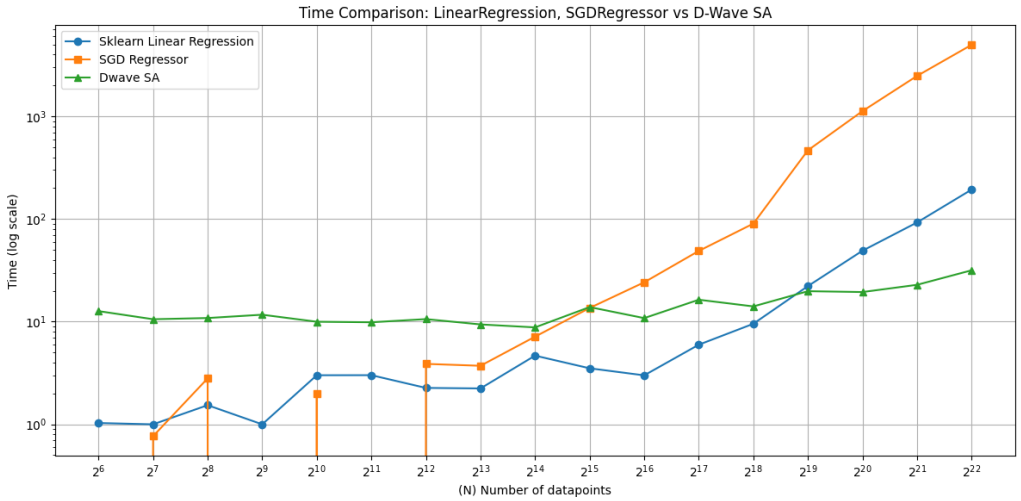

Fig.1. Time comparison: Classical vs Dwave SA

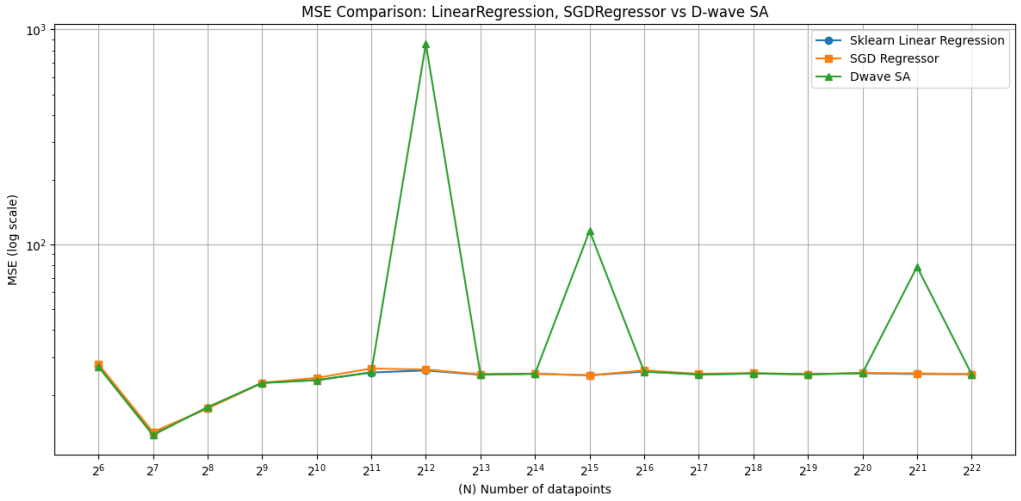

Fig.2. MSE comparison: Classical vs Dwave SA

B. Time and MSE Comparison with the Scalability of Features

This section includes the results obtained from D-wave annealing and classical Sklearn algorithms to the scalability of features. The parameters will be same as Table 1 except the Number of features (d) will be scalable. Below table shows the time comparison of both approaches.

Table 4. Time comparison of classical vs Dwave SA to the scalability of Features

d	Sklearn LinearRegression (ms)	SGDRegressor time (ms)	Dwave SA time (ms)
2	22.192001	466.681719	19.797802
4	24.425268	444.792271	20.324707
6	55.523157	587.148428	37.039518
8	69.458246	619.696140	53.586245
10	93.891859	658.503294	85.642576
12	103.103399	640.120268	100.579500
14	158.275604	692.317486	133.623838
16	153.058529	662.550449	158.120155
18	186.878443	742.199421	203.798532
20	220.087528	721.099138	248.407364

The values listed in Table 5 summarize the mean squared error for the classical and D-wave annealing approaches with respect to the scalability of features followed by a visual representation of the values expressed in the Table 4 and Table 5

Table 5. MSE comparison of classical vs Dwave SA to the scalability of Features

d	Sklearn LinearRegression MSE	SGDRegressor MSE	Dwave SA MSE
2	24.93393494	24.94721316	24.94094687
4	24.99825561	25.31666564	25.00292108
6	25.03648134	25.28461758	25.04564480
8	24.97544825	25.21257531	78.26776797
10	25.10007187	25.36684900	25.10001506
12	24.98887887	25.18918208	24.98761110
14	25.03629081	25.46154023	25.09570698
16	25.05420607	25.40812186	25.17229539
18	25.00853830	25.54330819	25.01736561
20	24.85946967	25.73314244	24.97807346

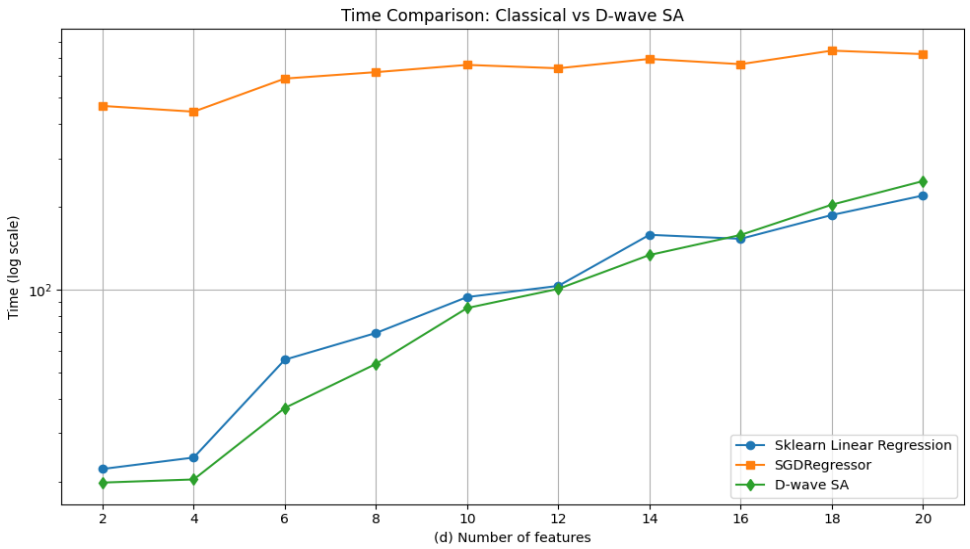

Fig.3. Time comparison: Classical vs D-wave SA with scalability of features

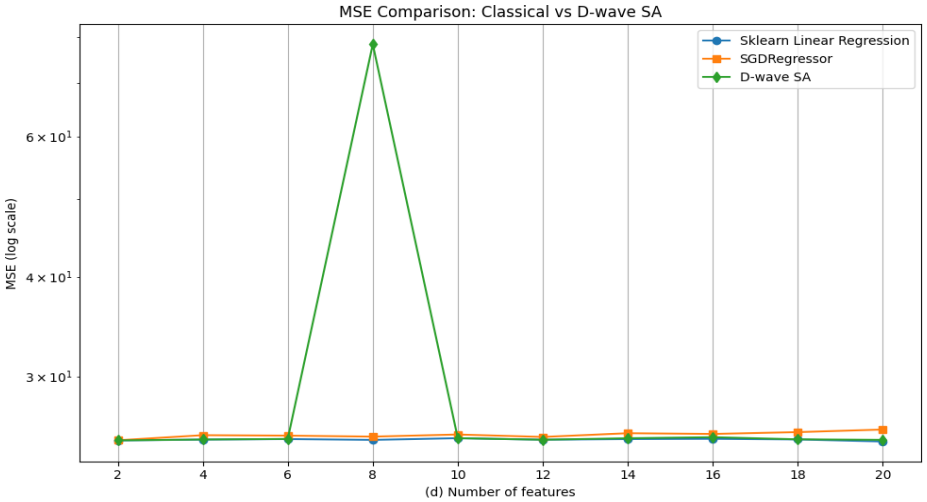

Fig.4. MSE comparison: Classical vs D-wave SA with scalability of features

4.2. QUBO Modeling of Linear Regression using QAOA and SamplingVQE (IBM Qiskit)

This section discusses the results obtained from the multiple linear regression performed using IBM Qiskit's *QAOA* and *SamplingVQE* compared with Scikit Learn *LinearRegression* and *SGDregressor*.The optimization can be done using both simulators and actual hardware. With the 10 minute free access given by the IBM Quantum Platform, we conducted the optimization limited to 27 qubits from QPU's namely ibm sherbrooke, ibm kyiv and ibm brisbane.Due to the qubit limitations, we had to reduce the number of precision bits (K) to 7, which resulted in the usage of 21 qubits. More than 27 qubits were required for features (d) greater than 2, i.e. for example, if there are 3 features the resulting number of qubits would be $K(d+1)$, which equals 28. Therefore, the simulation could not be run on the IBM Quantum Platform for higher number of features and the results only include Time and MSE comparison with the scalability of Datapoints with only 2 features in the dataset. The parameters used for this method are listed in the table below.

Table 6. Parameters used for running IBM QAOA and SamplingVQE for Linear regression

Parameter	Value
Number of Features (d)	2
Precision Vector (P)	[0.5, 1, 2, 4, 8, 16, 32]
Algorithms used	Qiskit QAOA, SamplingVQE
Optimizers for Qiskit	COBYLA, SPSA
Classical Algorithms	sklearn LinearRegression, SGDRegressor

Table 7. Time Comparison: Classical vs IBM qiskit algorithms to the scalability of datapoints

N	LinearRegression time (ms)	SGDRegressor time (ms)	QAOA time (ms)	SamplingVQE time (ms)
64	0.997782	1.999378	5716.434717	13119.860888
128	1.023293	1.981020	5815.423489	11916.832209
256	0.999928	1.999855	5716.434717	13119.860888
512	1.017570	1.996517	5861.490488	11725.635529
1024	2.277374	1.999855	5387.269020	11520.322561
2048	0.998020	2.001047	6749.864578	12544.059753
4096	1.511812	2.999306	6220.631123	11726.328135
8192	0.999928	4.251242	5815.544128	11455.620766
16384	0.998497	8.488417	5965.760469	12184.648514
32768	2.002478	13.187885	6455.990553	11471.103907
65536	4.352093	35.815716	6174.324751	11440.911055
131072	5.001068	50.654650	6254.344702	11935.555935
262144	12.593031	143.298149	5655.900955	12169.527292
524288	24.649382	419.079065	6692.414284	11569.677591
1048576	48.601151	895.991325	5857.837200	11813.598156
2097152	93.412876	2377.471924	7069.299698	11587.595463
4194304	184.718370	4963.366270	6314.347982	12545.646429

The values listed in the following tables showcase the run-time and mean squared error comparison between classical algorithms and IBM Qiskit algorithms with the scalability of data points.

Table 8. MSE comparison: Classical vs IBM qiskit algorithms to the scalability of datapoints

N	LinearRegression MSE	SGDRegressor MSE	QAOA MSE	SamplingVQE MSE
64	27.29163857	27.47545073	86.08624274	52.53842676
128	13.31065956	12.93790641	35.06379259	13.93090115
256	17.32734716	16.31355613	26.01594802	17.36692370
512	22.72084343	22.63374963	86.00160892	30.12939811
1024	23.41554408	24.07507802	94.23445023	25.86737406
2048	25.41194184	25.41302738	113.20052245	29.508600483
4096	25.97420203	26.45835206	1261.08349979	910.96399677
8192	24.79032455	25.01171658	49.30758383	37.50614198
16384	25.07708944	25.12303590	52.22272272	36.31969540
32768	24.64638022	24.75624649	227.10281521	122.387261607
65536	25.67379495	25.84373587	26.23174917	27.23071767
131072	24.88664939	24.89030361	35.14213980	27.10653224
262144	25.13878869	25.21896017	27.95934490	38.63497510
524288	24.93393494	24.96249537	51.34072918	45.19400965
1048576	25.20727932	25.28817425	61.94634517	34.46840882
2097152	25.00330739	25.06629175	159.27504990	267.98721416
4194304	24.94680502	24.99968097	40.67441908	28.74653042

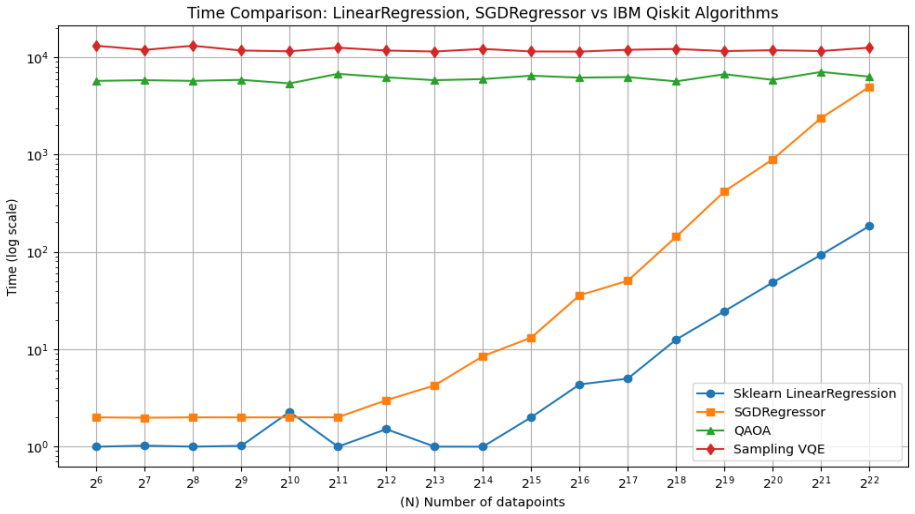

Fig. 5. Time comparison: Classical vs IBM qiskit

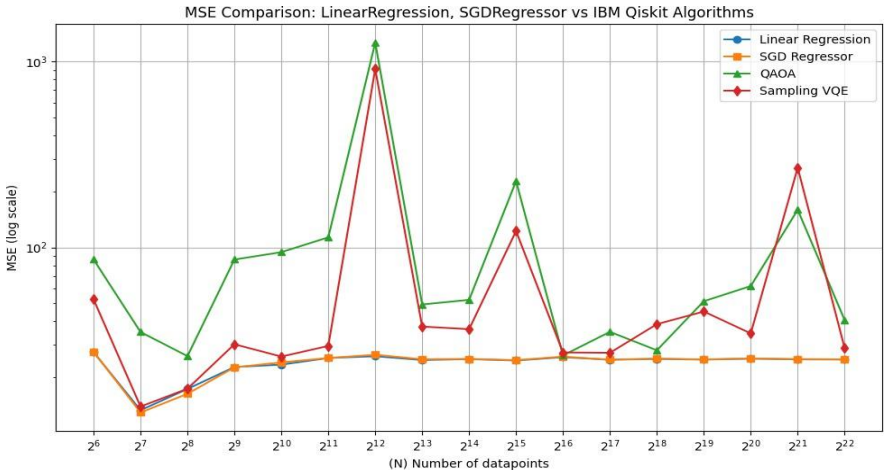

Fig.6. MSE comparison: Classical vs IBM Qiskit

4.3. Prediction curves for D-wave Annealing and IBM Qiskit

This section includes the prediction curves obtained from both approaches on synthetic testing data plotted against the classical counterpart with the by choosing only 2 features. As per the curves, it can be observed that D-wave annealers closely resemble the scikit-learn curve in many instances whereas Qiskit fails aligning due to the precision bit constraint.

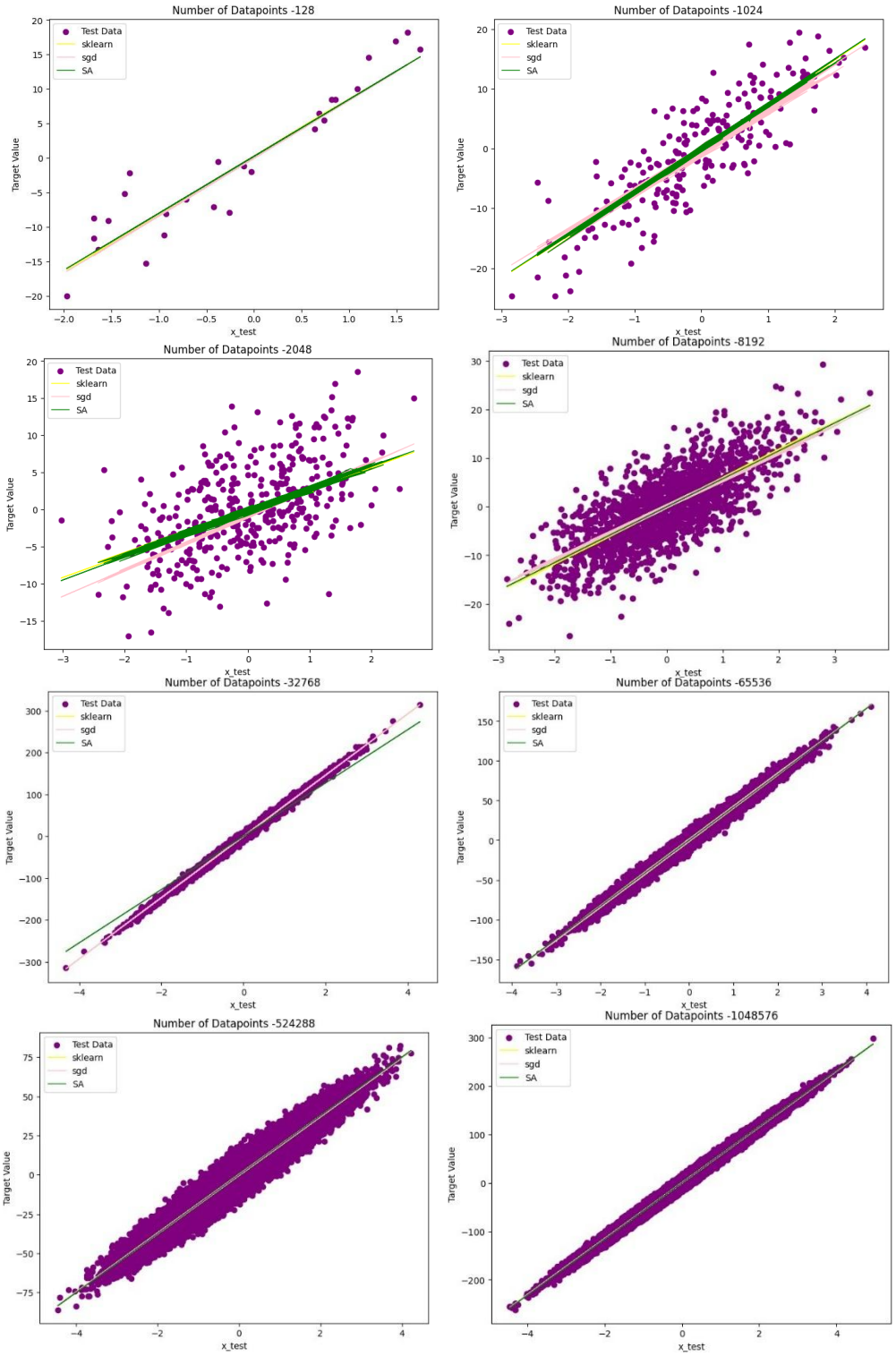

Fig.7. Prediction curves: D-wave Annealing vs classical

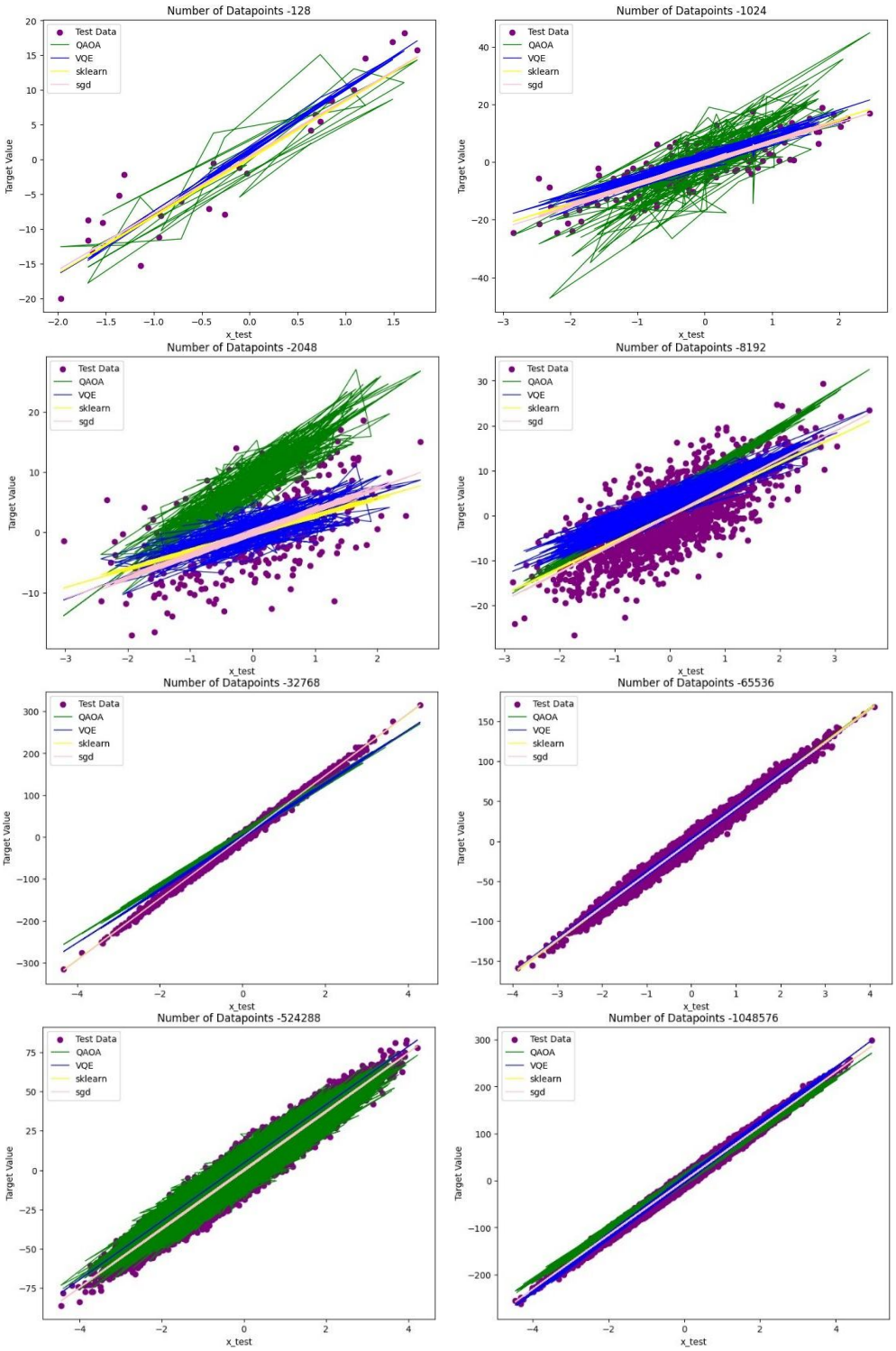

Fig.8. Prediction curves: IBM Qiskit vs classical

5. Discussion

5.1. Comparison with Existing Linear Regression Methods

As discussed in the previous chapters, linear regression is widely used across numerous domains, such as Finance, Economics, and Healthcare, which includes applications mainly in forecasting sales, Customer Retention, Stock market predictions, Risk assessment, and so on. However, performing these applications for a larger number of data using conventional methods such as Ordinary Least squares and stochastic gradient descent regression requires extensive amount of resources and time which could take several hours or even days to perform.

Therefore, this section primarily focuses on the time error comparison between the quantum and classical approaches discussing its robustness, interpretation, and suitability for larger datasets.

A. Classical vs D-wave Annealing Comparison

According to Date, Arthur, and Pusey-Nazzaro [2] the time solving the regression problem classically takes $\mathcal{O}(Nd^2)$ but the total time to convert the regression problem into QUBO and solve on adiabtic quantum computers would take $\mathcal{O}(Nd^2K^2)$. It can be seen that the quantum approach is worse than the classical counterpart but since the precision is not fixed for quantum computers analogous to classical computers K can be fixed and interpreted as a constant. Therefore, the resulting qubit footprint would be $\mathcal{O}(d^2)$ and the time complexity would be $\mathcal{O}(Nd^2)$, which is equivalent to the classical approach. However, it can be observed that from Table 2 and Figure 1 the D-wave annealing has a considerable time advantage after data points $N > 524288$ where D-wave is 2.3941 milliseconds faster compared to *Sklearn LinearRegression* and reaches upto 6x faster when $N > 4194304$ .When $N > 65536$ D-wave is 21.17157 milliseconds faster compared to *Sklearn SGDRegressor* and reaches upto 157x times faster when $N > 4194304$. Also according to Table 4 and Figure 3 it can be observed that the time advantage on the D- wave diminishes when the number of features $d > 16$. The D-wave annealing method in some instances, has significantly higher error compared to the classical method when scaling up data points which can be observed from Table 3 and Figure 2. Analogously the error is significant in some instances when scaling up the features also. This can be observed in Table 5, and Figure 4

B. Classical vs IBM Qiskit Comparison

In this section time and error for the solutions obtained using IBM Qiskit *QAOA* and *SamplingVQE* from the previous chapter will be compared against the classical algorithms. It can be observed that compared to the classical algorithms, qiskit algorithms takes a significant amount of time to provide a solution. However, it can also be seen that the time doesn't vary much in *QAOA* and *SamplingVQE* whereas *Sklearn SGDRegressor* and *Sklearn LinearRegression* have a significant increment in time when the number of data points increase as in Figure 5. It can be observed that the error in *Sklearn LinearRegression* and *Sklearn SGDRegressor* remains relatively stable whereas *QAOA* and *SamplingVQE* have many fluctuations. These fluctuations occur mainly due to the precision bit constraint in Qiskit

5.2. Potential Enhancements to the Model

One potential enhancement that could be made is increasing the number of precision bits used to represent the regression weights and bias. Rather than using 12 bit precision for D-wave and 7 bit precision in Qiskit, if the number of bits representing the regression weights can be increased, the error could be minimized significantly, requiring higher number of qubits meaning higher use of resources. Another approach is partitioning the QUBO instance into sub QUBO's solve them and combining the results to form a solution to the original instance. This can make large QUBOs effectively solve by a solver using quantum annealing hardware of limited size and connectivity [32]. As a preprocessing technique feature selection and dimentional reduction techniques such as Principal Component Analysis (PCA) can be used to reduce the number of features in the problem. This will reduce the dimentions of the QUBO matrix which then can be efficiently solved in quantum annealing hardware.

5.3. Practical Implications of the Proposed Model

The proposed QUBO-based linear regression model introduces several practical implications across different domains, particularly in computationally intensive and optimizeable applications. Unlike classical regression techniques such as gradient-based optimization or closed-form solution method, this framework allows the users to solve high-dimensional regression problems more efficiently using quantum annealers. In business and finance, this model can be used to optimize risk assessment, stock price prediction and portfolio management where a large number of factors and constraints exist. Traditional methods may struggle when handling thousands of correlated variables whereas QUBO method will improve performance in such cases. In the healthcare sector, this method could improve patient risk analysis and personalised treatment recommendations by handling feature selection more efficiently. Classical models may require this as a preprocessing step, whereas QUBO method will give highly correlated features more importance based on the quantum weights and biases.While the proposed method offers potential advantage over classical methods, limitations on quantum hardware mainly, higher qubit utilisation and the associated cost for hardware may negatively affect practically setting up this model. But with hybrid quantum-classical solvers, the applicability of these methods will likely expand and may openly accessible to users in the near future.

5.4. Assumptions of the Proposed Model

The QUBO-based method operates under several assumptions, which ensure the validity of the regression results, which are aligned with classical regression principles and the QUBO optimization framework.

One of the key assumptions in the model is that the synthetically generated data will adequately represent real-world patterns and distributions. The validity of results depends on up to which extent the synthetic data mimic actual relationships found in the day-to-day datasets. The failure to capture the real world variability through synthetic data may incur limited model generalization.

Feature scaling using *sklearn StandardScaler* to scale the data introduces another assumption that the features in the dataset will have a better influence on the dependent variable and model performance. In classical regression, feature scaling is important when features have vastly different ranges as it prevents certain features from disproportionately influencing the target variable. Therefore, feature scaling helps maintain balanced optimized weights and biases.

Another assumption is that encoding continuous regression variables into discrete variables ensures less information loss. Since QUBO models require variables that operate in discrete space, this assumption ensures that the relationships between variables is preserved while maintaining predictive accuracy.

This model also assumes no severe multicollinearity among independent variables.In the real world empirical datasets it is common to have highly correlated features, whereas here since the data is generated synthetically the assumption is made that highly correlated features may not exist. If highly correlations exist between features, dimension reduction techniques may have to be used appropriately.

Finally, it is assumed that the minimization of QUBO using the open source D-wave solvers find a near-optimal solution within reasonable computational time. Since D-wave and Qiskit have data loading inefficiencies due to being Noisy Intermediate-Scale Quantum devices (NISQ), we neglected this issue in computation and the constraints due to Since the quality of the solution may highly depend on the solver's efficiency, it is assumed that the obtained solutions are close enough to the theoretical best-fit parameters.

5.5. Future Research Directions

The QUBO-based linear regression model suggested in this research presents several opportunities for future research areas to be explored. One key area would be validating the model with real-world datasets across different domains such as finance, banking, healthcare etc. These empirical studies on large-scale datasets would provide more insights on practical limitations and advantages of QUBO-based linear regression models.

Another avenue for research involves extending the model to handle non-linear and neural network problems. Y.Tang proposes a method for replacing the soft-max layer of a neural network problem with a linear support vector machine that minimizes a margin-based loss instead of the cross-entropy loss [33]. The loss function can be denoted as the following constrained optimization (primal problem):

$$\min_{w,\xi_i} \quad \frac{1}{2}w^\top w + C\sum_{i=1}^{N}\xi_i \tag{18}$$

$$\text{s.t.} \quad y_i(w^\top x_i + b) \geq 1 - \xi_i, \quad \forall i = 1,2,\ldots..N$$

$$\xi_i \geq 0, \quad \forall i = 1,2,\ldots..N$$

Where, $X \in \mathbb{R}^{N\times d}$; $Y \in \{-1,+1\}^N$ the training labels; $w \in \mathbb{R}^{\mathbb{d}}$ the regression weights; $b \in \mathbb{R}$ the bias and ξ_i the slack variables which penalizes data points which violate the margin requirements.

Introducing λ and μ as Lagrange multipliers for the constraints, the primal problem can be turned to the Lagrangian as follows.

$$\mathcal{L}(w,b,\xi,\lambda,\mu) = \frac{1}{2}w^\top w + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\lambda_i\,[y_i(w^\top x_i + b) - 1 + \xi_i] - \sum_{i=1}^{N}\mu_i\,\xi_i. \tag{19}$$

As part of the KKT conditions, we set the partial derivatives of $\mathcal{L}(w,b,\xi,\lambda,\mu)$ with respect to w,b,ξ_i equal to 0. In doing so, we get the following.

$$\frac{\partial\mathcal{L}(w,b,\xi,\lambda,\mu)}{\partial w} = w - \sum_{i=1}^{N}\lambda_i\,y_i x_i = 0 \quad \Rightarrow \quad w = \sum_{i=1}^{N}\lambda_i\,y_i x_i. \tag{20}$$

$$\frac{\partial\mathcal{L}(w,b,\xi,\lambda,\mu)}{\partial b} = -\sum_{i=1}^{N}\lambda_i\,y_i = 0 \quad \Rightarrow \quad \sum_{i=1}^{N}\lambda_i\,y_i = 0. \tag{21}$$

$$\frac{\partial\mathcal{L}(w,b,\xi,\lambda,\mu)}{\partial\xi_i} = C - \lambda_i - \mu_i = 0 \quad \Rightarrow \quad C = \lambda_i + \mu_i \tag{22}$$

Substituting these equations obtained through partial derivatives into the Lagrangian, we can obtain the following maximization problem:

$$\mathcal{L}(\lambda) = \sum_{i=1}^{N}\lambda_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\,\lambda_j x_i x_j y_i \tag{23}$$

This maximization problem will need to be rearranged as below into a minimization problem, in order to convert into a QUBO problem.

$$\min_{\lambda}\mathcal{L}(\lambda) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\,\lambda_j x_i x_j y_i y_j - \sum_{i=1}^{N}\lambda_i \quad \lambda_i,\lambda_j \geq 0 \quad \forall i,j \tag{24}$$

Given 1_N and 0_N which represent N-dimensional vectors of ones and zeros respectively, the above minimization problem can be rewritten as follows.

$$\min_{\lambda} \mathcal{L}(\lambda) = \frac{1}{2}\lambda^T(XX^T \odot YY^T)\lambda - \lambda^T 1_N \quad \lambda \geq 0_N \tag{25}$$

Next, a K-dimensional precision vector $P = [p_1, p_2, \ldots, p_k]^T$ is introduced as described in section 3.2 which yields λ_i such that:

$$\lambda_i = \sum_{k=1}^{K} p_k \hat{\lambda}_{ik} \quad \forall i = 1,2, \ldots, N \tag{26}$$

where p_k is the kth entry in the precision vector P. Next, the binary variables are stacked vertically where:

$$\hat{\lambda} = [\hat{\lambda}_{11}, \ldots, \hat{\lambda}_{1K}, \hat{\lambda}_{21}, \ldots, \hat{\lambda}_{2K}, \ldots, \hat{\lambda}_{N1}, \ldots, \hat{\lambda}_{NK}]^T \tag{27}$$

and the precision matrix $\mathcal{P}$ is denoted as

$$\mathcal{P} = I_N \otimes P^T \tag{28}$$

where λ is denoted as:

$$\lambda = \mathcal{P}\hat{\lambda} \tag{29}$$

Finally, this λ can be substituted to the matrix form of Lagrangian minimization problem and express as,

$$\min_{\hat{\lambda} \in \mathbb{R}^{NK}} \mathcal{L}(\hat{\lambda}) = \frac{1}{2}\hat{\lambda}^T\mathcal{P}^T(XX^T \odot YY^T)\mathcal{P}\hat{\lambda} - \hat{\lambda}^T\mathcal{P}^T 1_N \tag{30}$$

This way, a QUBO model for SVM induced neural networks can be derived and solve using Adiabatic Quantum computers which could have a potential gain compared to classical approach. Date, Arthur, and Pusey-Nazzaro states that that the resulting qubit footprint would be $O(N^2)$ which is better than the classical algorithm $O(N^3)$ [2]

Another research direction could be observing on the approximation errors due to QUBO discretization. In future, confidence intervals for errors can be considered so that this method is more reliable.

Finally, as quantum technology advances future research areas should explore the practical feasibility of running these models on real hardware, where a benchmarking could be done to find out the efficiencies in hardware, making QUBO Machine learning models a viable alternative to classical approach.

6. Conclusions

This work proposes a quantum inspired framework for solving multiple linear regression by converting the least squares optimization problem into a QUBO formulation. A synthetic dataset was generated to evaluate the performance of three quantum approaches, quantum annealing using D-wave annealers, IBM Qiskit QAOA and sampling VQE. The performance of those approaches was benchmarked against classical regression algorithms such as scikit-learn's linear regression and SGDRegressor. D-wave's *SimulatedAnnealingSampler* demonstrated consistent runtime advantage over classical methods, particularly when the number of data points $N > 65536$ and $N > 524288$ in *sklearn SGDRegressor* and *sklearn LinearRegression* respectively. However, when scaling features, the D-wave's time efficiency diminished after 16 dimensions. The MSE values remained close to the scikit-learn approaches in many cases but fluctuated occasionally producing outliers. In contrast, IBM Qiskit's *QAOA* and *Sampling VQE* which were restricted by qubit limitations were tested on lower-dimensional datasets. However, these methods showed significantly higher runtimes, but remained relatively stable as data volume increases. Due to the qubit constraint these approaches produced higher and inconsistent error compared to classical approaches. The prediction curves visually confirm that D-wave closely resembles the classical regression curve, while Qiskit shows larger deviations. The study also introduced a QUBO based formulation for SVM induced neural networks using lagrangian minimization which can potentially expand quantum applicability to neural networks and nonlinear models. Based on the results and the findings a conclusion can be drawn that quantum annealing approach, though it's performed through D-wave open source annealers can have a significant runtime advantage compared to classical approaches. Therefore, in the future finetuning those models, mainly the precision along with performing on real hardware can have a much more lesser runtime and error compared to classical regression methods.

References

[1] S. Morita and H. Nishimori, "Mathematical foundation of quantum annealing," *Journal of Mathematical Physics*, vol. 49, no. 12, 2008.

[2] P. Date, D. Arthur, and L. Pusey-Nazzaro, "Qubo formulations for training machine learning models," *Scientific reports*, vol. 11, no. 1, p. 10029, 2021.

[3] W. Ha¨ma¨la¨inen, "Class np, np-complete, and np-hard problems," *Sort*, pp. 1–7, 2006.

[4] A. Mahasinghe, R. Hua, M. J. Dinneen, and R. Goyal, "Solving the hamiltonian cycle problem using a quantum computer," in *Proceedings of the Australasian Computer Science Week Multiconference*, pp. 1–9, 2019.
[5] M. J. Dinneen, A. Mahasinghe, and K. Liu, "Finding the chromatic sums of graphs using a d-wave quantum com- puter," *The Journal of Supercomputing*, vol. 75, pp. 4811–4828, 2019.
[6] A. Mahasinghe and Y. Jayasinghe, "An initial step toward a quantum annealing approach to the discrete logarithm problem," *Security and Privacy*, vol. 5, no. 4, p. e234, 2022.
[7] A. Mahasinghe, V. Fernando, and P. Samarawickrama, "Qubo formulations of three np problems," *Journal of Infor- mation and Optimization Sciences*, vol. 42, no. 7, pp. 1625–1648, 2021.
[8] R. K. Nath, H. Thapliyal, and T. S. Humble, "A review of machine learning classification using quantum annealing for real-world applications," *SN Computer science*, vol. 2, no. 5, p. 365, 2021.
[9] S. Lloyd, M. Mohseni, and P. Rebentrost, "Quantum algorithms for supervised and unsupervised machine learning," *arXiv preprint arXiv:1307.0411*, 2013.
[10] A. Mott, J. Job, J.-R. Vlimant, D. Lidar, and M. Spiropulu, "Solving a higgs optimization problem with quantum annealing for machine learning," *Nature*, vol. 550, no. 7676, pp. 375–379, 2017.
[11] P. Date and T. Potok, "Adiabatic quantum linear regression," *Scientific reports*, vol. 11, no. 1, p. 21905, 2021.
[12] N. Ide and M. Ohzeki, "Sparse signal reconstruction with qubo formulation in l0-regularized linear regression," in *2022 International Symposium on Information Theory and Its Applications (ISITA)*, pp. 19–23, 2022.
[13] M. R. van Dommelen and F. Phillipson, "Qubo formulation for sparse sensor placement for classification," in *Inter- national Conference on Innovations for Community Services*, pp. 17–35, Springer, 2024.
[14] M. Schuld and N. Killoran, "Quantum machine learning in feature hilbert spaces," *Physical review letters*, vol. 122, no. 4, p. 040504, 2019.
[15] W. Guan, G. Perdue, A. Pesah, M. Schuld, K. Terashi, S. Vallecorsa, and J.-R. Vlimant, "Quantum machine learning in high energy physics," *Machine Learning: Science and Technology*, vol. 2, no. 1, p. 011003, 2021.
[16] S. Y.-C. Chen, C.-H. H. Yang, J. Qi, P.-Y. Chen, X. Ma, and H.-S. Goan, "Variational quantum circuits for deep reinforcement learning," *IEEE access*, vol. 8, pp. 141007–141024, 2020.
[17] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," *arXiv preprint arXiv:1411.4028*, 2014.
[18] E. Farhi, D. Gamarnik, and S. Gutmann, "The quantum approximate optimization algorithm needs to see the whole graph: A typical case," *arXiv preprint arXiv:2004.09002*, 2020.
[19] E. Farhi, J. Goldstone, S. Gutmann, and L. Zhou, "The quantum approximate optimization algorithm and the sherrington-kirkpatrick model at infinite size," *Quantum*, vol. 6, p. 759, 2022.
[20] S. Lloyd, "Quantum approximate optimization is computationally universal," *arXiv preprint arXiv:1812.11075*, 2018.
[21] J. S. Otterbach, R. Manenti, N. Alidoust, A. Bestwick, M. Block, B. Bloom, S. Caldwell, N. Didier, E. S. Fried, S. Hong, *et al.*, "Unsupervised machine learning on a hybrid quantum computer," *arXiv preprint arXiv:1712.05771*, 2017.
[22] C. Ciliberto, M. Herbster, A. D. Ialongo, M. Pontil, A. Rocchetto, S. Severini, and L. Wossnig, "Quantum machine learning: a classical perspective," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 474, no. 2209, p. 20170551, 2018.
[23] A. W. Harrow, A. Hassidim, and S. Lloyd, "Quantum algorithm for linear systems of equations," *Physical review letters*, vol. 103, no. 15, p. 150502, 2009.
[24] A. Ge´ron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*." O'Reilly Media, Inc.", 2022.
[25] F. Glover, G. Kochenberger, and Y. Du, "A tutorial on formulating and using qubo models," *arXiv preprint arXiv:1811.11538*, 2018.
[26] "Samplers—dwave-system 1.10.0 documentation." Online, 2021. https://dwave-systemdocs.readthedocs.io/en/latest/reference/samplers.html (accessed Feb 28,2025).
[27] K. Blekos, D. Brand, A. Ceschini, C.-H. Chou, R.-H. Li, K. Pandya, and A. Summer, "A review on quantum approximate optimization algorithm and its variants," *Physics Reports*, vol. 1068, pp. 1–66, 2024.
[28] D. A. Fedorov, B. Peng, N. Govind, and Y. Alexeev, "Vqe method: a short survey and recent developments," *Materials Theory*, vol. 6, no. 1, p. 2, 2022.
[29] "Minimum eigen optimizer - qiskit optimization 0.6.1." Online, 2019. https://qiskit-community.github.io/qiskit-optimization/tutorials/03 minimum eigen optimizer (accessed Feb. 28, 2025).
[30] M. Shoaib, "How to solve qubo problems using qiskit - mohammad shoaib - medium." Online, 06 2023. https://medium.com/@shoaib6174/how-to-solve-qubo-problems-using-qiskit-f4eab6cc3061 (accessed Feb 28, 2025).
[31] scikit learn, "1.5. stochastic gradient descent — scikit-learn 0.23.2 documentation." Online, n.d. https://scikit-learn.org/stable/modules/sgd.html (accessed Feb 28, 2025).
[32] M. Booth, S. Reinhardt, and A. Roy, "Partitioning optimization problems for hybrid classical/quantum execution technical report." Online. https://github.com/dwavesystems/qbsolv/blob/master/qbsolv techReport.pdf (accessed March 3rd, 2025).
[33] Y. Tang, "Deep learning using support vector machines," *CoRR, abs/1306.0239*, vol. 2, no. 1, 2013.

Authors' Profiles

Nilesh T. Fonseka completed his B.Sc.(Hons) in Computational Mathematics at University of Colombo. He is currently working as a Junior Machine Learning Engineer for an Australian based Company and has over one and a half year experience in the field of machine learning. His research interests are mainly into Deep learning, Natural language processing and Quantum machine learning applications

Anuradha Mahasinghe is a Professor in Computational Mathematics at University of Colombo, Sri Lanka. Also, he is serving as the Deputy Director of the Centre for Mathematical Modelling of University of Colombo. He has more than 12 years' experience in quantum computing from his doctoral and post–doctoral studies at University of Western Australia and University of Auckland. He pioneered in introducing quantum computing to Sri Lankan research community. His other research interests include optimization and recreational mathematics.

How to cite this paper: Nilesh T. Fonseka, Anuradha Mahasinghe, "Quantum–inspired Methods for Training Machine Learning Models", International Journal of Information Technology and Computer Science(IJITCS), Vol.17, No.6, pp.143-159, 2025. DOI:10.5815/ijitcs.2025.06.08