Financial risk management on a neutral atom quantum processor

Lucas Leclerc,[1,2,*] Luis Ortiz-Gutiérrez,[1] Sebastián Grijalva,[1] Boris Albrecht,[1] Julia R. K. Cline,[1] Vincent E. Elfving,[1] Adrien Signoles,[1] Loïc Henriet,[1,†] Gianni Del Bimbo,[3,*] Usman Ayub Sheikh,[3,*] Maitree Shah,[4] Luc Andrea,[5] Faysal Ishtiaq,[3] Andoni Duarte,[3] Sam Mugel,[4] Irene Cáceres,[3] Michel Kurek,[5] Roman Orús,[3,6,7] Achraf Seddik,[8] Oumaima Hammami,[8] Hacene Isselnane,[8] and Didier M'tamon[8]

[1]*PASQAL, 7 rue Léonard de Vinci, 91300 Massy, France*

[2]*Université Paris-Saclay, Institut d'Optique Graduate School, CNRS, Laboratoire Charles Fabry, 91127 Palaiseau, France*

[3]*Multiverse Computing, Parque Científico y Tecnológico de Gipuzkoa, Paseo de Miramón 170, 20014 San Sebastián, Spain*

[4]*Centre for Social Innovation, 192 Spadina Avenue, Suite 509, Toronto, Ontario, Canada M5T 2C2*

[5]*WIPSE Paris-Saclay Enterprises, 7 rue de la Croix Martre, 91120 Palaiseau, France*

[6]*Donostia International Physics Center, Paseo Manuel de Lardizabal 4, E-20018 San Sebastián, Spain*

[7]*Ikerbasque Foundation for Science, Maria Diaz de Haro 3, E-48013 Bilbao, Spain*

[8]*Crédit Agricole Corporate and Investment Bank, 12 Place des États-Unis, 92545 Montrouge, France*

(Received 8 June 2023; accepted 9 October 2023; published 6 November 2023)

Machine learning models capable of handling the large data sets collected in the financial world can often become black boxes expensive to run. The quantum computing paradigm suggests new optimization techniques that, combined with classical algorithms, may deliver competitive, faster, and more interpretable models. In this paper we propose a quantum-enhanced machine learning solution for the prediction of credit rating downgrades, also known as fallen-angels forecasting in the financial risk management field. We implement this solution on a neutral atom quantum processing unit with up to 60 qubits on a real-life data set. We report performance that is competitive with the state-of-the-art random forest benchmark, whereas our model achieves better interpretability and comparable training times. We examine how to improve performance in the near term, validating our ideas with tensor-networks-based numerical simulations.

DOI: 10.1103/PhysRevResearch.5.043117

I. INTRODUCTION

In finance, an interesting and relevant problem consists in estimating the probability of debtors reimbursing their loans, which represents an essential quantitative problem for banks. Financial institutions generally attempt to estimate credit worthiness of debtors by sorting them into classes called credit ratings. These institutions not only can build their own credit rating model but also can rely on credit ratings provided by one or more of the three main rating agencies: Fitch, Moody's, and Standard & Poor's (S&P). Borrowers are generally grouped into two main categories according to their credit worthiness: investment-grade borrowers with low credit risk and sub-investment-grade borrowers with higher credit risk. Should a borrower's rating be downgraded from the investment-grade category to the sub-investment-grade category, the borrower is referred to as a *fallen angel*.

The early anticipation of these fallen angels is a problem of utmost importance for financial institutions and one that has gathered significant attention from the machine learning (ML) community in the past years. Indeed, these institutions usually have access to large amounts of data accumulated over several decades. The wide variety of features gathered can be fed to advanced ML models tasked with solving the following classification task: Will a debtor have a high or low risk of becoming a fallen angel in the foreseeable future? Proposals of binary classification methods, or *classifiers*, targeting such tasks have been investigated, and promising results were obtained using the random forest model and XGBOOST [1,2]. Due to their feature extraction flexibility, those tree-based ensemble methods turned out to be more suitable for similar credit risk modeling tasks [3,4], compared with deep learning approaches [5,6]. However, these methods quickly become computationally demanding as the numbers of decision trees grow. Furthermore, denser and denser forests usually become black boxes in terms of interpretability, i.e., hard to understand by their users.

Quantum computing offers a new computational paradigm promising advances in computational efficiency for particular types of tasks. One of the most promising fields where quantum computing could be useful in practical industrial applications is the financial sector, with its wide range of hard computational problems [7]. Quantum and quantum-inspired approaches have already shown many promising

*These authors contributed equally to this work.
†loic@pasqal.com

applications in financial problems [8–15]. In particular, the nascent subfield of quantum machine learning harnesses well-known quantum mechanics phenomena such as superposition and entanglement to enhance machine learning routines. This technology has drawn much attention in the past years with numerous proposed algorithms and a broad array of applications [16–18]. Indeed, many classification, clustering, or regression tasks can be tackled by quantum neural networks [12,19] or quantum kernel models [18,20–22]. Much focus has also been placed on hybrid perspectives that combine classical components and current quantum hardware capabilities, such as variational approaches [23] or hybrid ensemble-type methods [24–29].

This paper investigates the capabilities of today's noisy intermediate-scale quantum hardware in conjunction with quantum-enhanced machine learning approaches for the detection of fallen angels. In this paper, we propose a quantum-enhanced classifier inspired by the quantile boost (QBoost) algorithm [24,25] with hardware implementation on a neutral atom quantum processing unit (QPU). We benchmark the proposed solution against a highly optimized random forest classifier. As we shall see, using the proposed quantum solution, we achieve a performance very similar to the benchmark with comparable training times and better interpretability, i.e., a simpler and smaller predictive model. Furthermore, we provide a clear path on how we expect to beat the threshold set by the benchmarked random forest, providing evidence based on tensor networks simulations.

The paper is organized as follows: Sec. I presents the application of the classical solution, random forest, to the classification task. Section II presents the proposed quantum-enhanced solution and methodology. Section III comments on the results and scaling of the devised quantum-enhanced classifier implemented on the proposed QPU, followed by conclusions and an outlook.

II. CLASSICAL METHOD FOR FALLEN-ANGELS DETECTION

This section delves into the random forest model classically used in risk management to assess the creditworthiness of counterparts and predict future fallen angels. Random forest is a well-known ensemble method based on bootstrap aggregation, also called *bagging*, applicable for regression and classification alike [30]. Training a random forest of m trees on an n-size training set entails sampling with replacement of the latter to generate m new data sets with n elements each. To ensure low correlation between the trees, each tree is trained on a different subset of randomly selected features. The trained classifiers are then collectively used to predict the class of unseen data through majority vote over m decisions.

A. Data set

The data set used in this paper originates from public data over a period of 20 yr (2001–2020). It comprises more than 90 000 instances characterized by around 150 features, representing the historical evolution of credit ratings as well as numerous financial variables. Predictors include rating, financial, and equity market variables and their trends calculated on a biannual, quarterly, and 5-yr basis. The examples considered are based on over 2000 companies from ten different industrial fields (e.g., energy, healthcare, utility) and 100 subsectors (e.g., infrastructure, oil and gas exploration, mining), located in 70 different countries. Each of the records is labeled as either a fallen angel (i.e., critical downgrade; class 1 or positive) or a non-fallen-angel (i.e., stable or upgraded credit score; class 0 or negative) based on standard credit rating scales.

The training set consists of around 65 000 examples from the 2001–2016 period. The testing set comprises around 26 000 examples from the 2016–2020 period. The class distribution is highly unbalanced with only 9% of fallen angels in the training set and 12% in the test set.

Since our goal is to benchmark the performance of the proposed quantum framework against the classical random forest's solution over the same data set, no further data processing or feature engineering was performed. To deal with the highly skewed distribution of classes as mentioned above, both random undersampling of the majority class and random oversampling of the minority class were tested to balance the training set.

B. Metrics

In order to assess the quality of the studied classification models, we settle on two metrics, i.e., *precision* and *recall*, defined as

$$P = \frac{T_p}{T_p + F_p} \quad \text{and} \quad R = \frac{T_p}{T_p + F_n}, \tag{1}$$

where T_p and T_n represent the number of true positives and the number of true negatives, respectively (i.e., accurate prediction of the model), and F_p and F_n represent the number of false positives and the number of false negatives, respectively (i.e., inaccurate prediction of the model), as illustrated in Fig. 1(a). The precision P is the ability of a classifier to not mistake a negative sample for a positive one; it thus represents the quality of a positive prediction made by the model. Similarly, the recall R can be understood as the ability of the model to find all the positive samples.

Our primary goal consists in increasing the precision of predicting fallen angels while keeping the recall over $R = 80\%$. Accommodating this criterion requires tuning the decision threshold, a parameter that governs conversion of class membership probabilities to the corresponding hard predictions (e.g., 0 or 1). This threshold is usually set at 0.5 for normalized predicted probabilities.

Here, the optimal probability threshold value is determined using a precision-recall curve (PR curve), shown in Fig. 2(a). Specifically, P and R are computed at several decision thresholds. A linear interpolation between neighboring points of the PR curve [see Fig. 2(b)] enables us to determine the precision value corresponding to $R = 83\%$.

C. Benchmark baseline

Training the random forest model and optimizing its hyperparameters through random search lasts more than 3 h on a classical computer. The 1200 decision trees of the obtained model enable us to achieve $R = 83\%$ and $P = 28\%$, as showcased in Fig. 1(b).

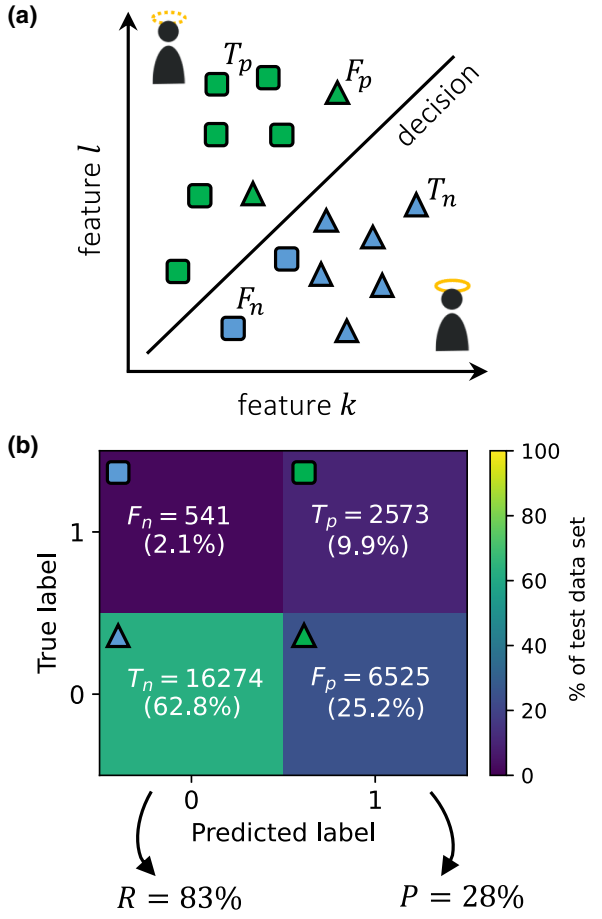

FIG. 1. (a) A binary classification problem deals with two classes (e.g., squares and triangles). The classifier is trained on the labeled training data and tasked to classify the instances, i.e., to find a decision boundary (black line) separating the test data. Considering test data comprising two features k and l, the model either predicts correctly the class of an instance, thus increasing T_p and T_n (green squares and blue triangles), or predicts incorrectly, thus increasing F_p and F_n (green triangles and blue squares). (b) Classification performance of the classical solution on the test set shown as a confusion matrix. From the proportions of T_p, F_p, and F_n obtained by the random forest model described in Sec. I C, both recall R and precision P scores can be derived.

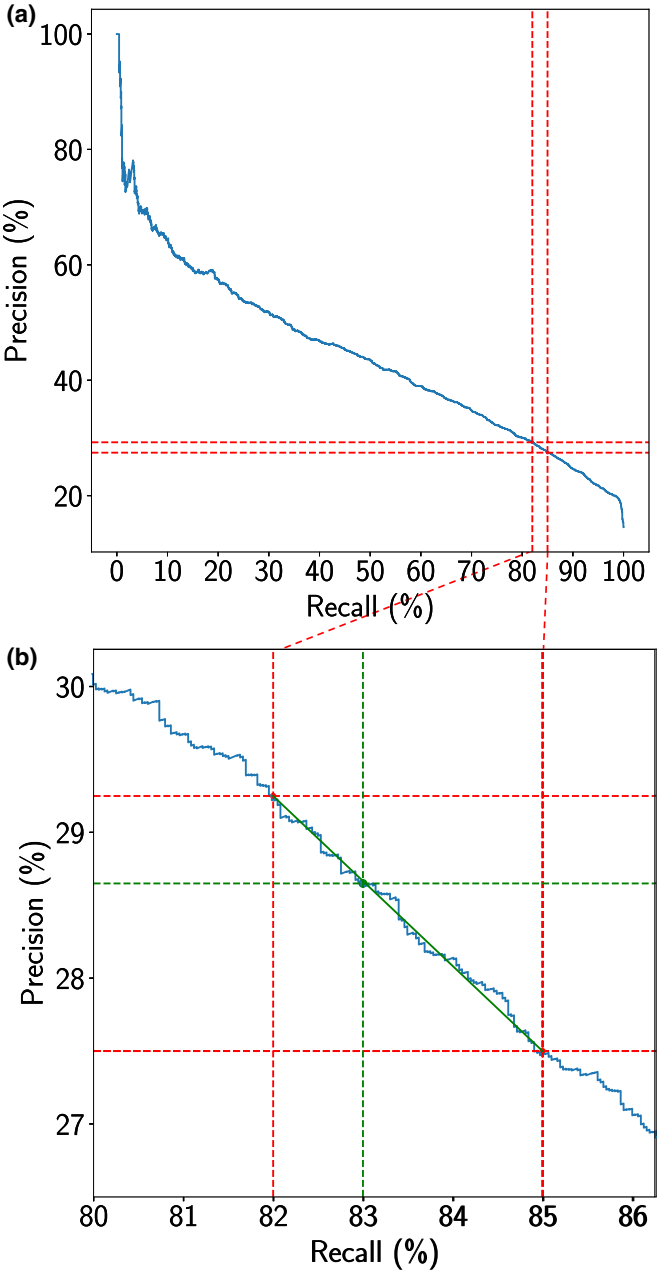

FIG. 2. (a) Precision-recall curve (blue curve) calculated using different decision thresholds for the predictions. A close-up of the region of interest is shown in (b) (dashed red lines). (b) Precision value corresponding to the recall of 83% obtained through linear interpolation (dashed green lines) between the neighboring points $R = 82$ and 85% on the PR curve.

This result, far from being optimal, especially in terms of precision, is due to several factors, representative of the complexity of the problem.

(1) The data set is highly unbalanced, which is a notoriously hard problem for classical machine learning models.

(2) Processing a significant number of features can be resource consuming, and it remains impossible to exhaustively search the space of solutions at too large sizes. Therefore the classical method uses a suboptimal shortcut to select relevant features.

(3) Finding the optimal weight for each predictor is an exponentially complex optimization problem as the number of predictors increases. Hence the random forest model uses majority voting for classification, which is quite restrictive in terms of performance.

We address the above-mentioned points with a proposed quantum-enhanced machine learning approach, taking advantage of quantum combinatorial optimization to efficiently explore the space of solutions.

III. QUANTUM-ENHANCED CLASSIFIER

A. QBoost

First proposed by Neven *et al.* [24], the QBoost algorithm is an ensemble model comprising a set of weak, i.e., simple

low-depth, decision tree (DT) classifiers, also called learners, optimally combined to build a strong classifier.

The workflow of the algorithm starts from a boosting procedure, based on the standard adaptive boosting (AdaBoost) algorithm [31,32]. A set of N weak learners $h_{i=1\cdots N}$ is classically trained in a sequential fashion on the training set. Initially, the *first* weak learner is trained such that all the data points are weighted uniformly, using the constant distribution $D_{i=1}(s) = \frac{1}{S}$, with S being the size of the data set. Each weak learner h_i is then iteratively trained on the same training set, where the data points are, however, weighted differently based on an updated distribution $D_i(s)$. This latter distribution is recomputed after the training of each weak learner. More precisely, it depends on the quantity ε_i that considers the misclassified points by the previous weak learner:

$$\varepsilon_i = \sum_s \mathbb{1}[h_i(\vec{x}_s) \neq y_s] D_i(s). \tag{2}$$

From ε_i, one computes the quantity

$$\alpha_i = \frac{1}{2} \ln\left(\frac{1-\varepsilon_i}{\varepsilon_i}\right), \tag{3}$$

which enters the exponential coefficient to update the data distribution as

$$D_{i+1}(s) = \frac{1}{Z_i} D_i(s) \cdot e^{-\alpha_i y_s h_i(\vec{x}_s)}. \tag{4}$$

Here, we introduce the normalization factor Z_i such that D_{i+1} is a probability distribution.

After the entire ensemble of weak learners $h_{i=1\cdots N}$ has been trained, a strong classifier C is built by combining the weak learners. The optimal combination is obtained through the optimization of binary weights $w \in \mathbb{B}^N$ that minimize the following cost function:

$$\mathcal{H}(w) = \sum_s \left(\frac{1}{N}\sum_i^N w_i h_i(\vec{x}_s) - y_s\right)^2 + \lambda \|w\|_0, \tag{5}$$

where w_i is the ith binary weight, $h_i(\vec{x}_s) \in [-1, 1]$ is the prediction of the ith weak learner for the data point $\vec{x}_s$, and $y_s \in [-1, 1]$ are the classification labels. A regularization term parametrized by λ helps to favor better generalization of the model on new data by penalizing too-complex ensembles with many weak learners. As the number of learners increases, the space of possible binary weights expands exponentially. Minimizing $\mathcal{H}$ is in fact an NP-hard problem.

Expanding the squared term in the above equation and dropping the constant terms, which are irrelevant to the minimization problem, we can reformulate the cost function as

$$\mathcal{H}(w) = \sum_{i,j}^N w_i w_j \mathrm{Corr}(h_i, h_j) + \sum_i^N w_i(\lambda - 2\mathrm{Corr}(h_i, y)), \tag{6}$$

with $\mathrm{Corr}(h_i, h_j) = \sum_s h_i(\vec{x}_s) h_j(\vec{x}_s)$ and $\mathrm{Corr}(h_i, y) = \sum_s h_i(\vec{x}_s) y_s$. On the one hand, the weak classifiers whose outputs correlate well with the labels cause the second term to be lowered via $\mathrm{Corr}(h_i, y)$. On the other hand, via the quadratic part $\mathrm{Corr}(h_i, h_j)$ describing the correlations between the weak classifiers, pairs of strongly correlated classifiers increase the value of the cost function, thereby increasing the chance for one of them to be switched off. This is in line with the general paradigm of ensemble methods for promoting a diversification of the ensemble in order to improve the model generalization on unseen data.

Once the optimization of Eq. (6) is performed (see Sec. II C), the strong classifier C can be built using w^{opt}, the weights minimizing $\mathcal{H}$. Given a new data point $\vec{x}$, we infer a classification prediction by

$$C(\vec{x}) = \mathrm{sgn}\left(\frac{1}{N}\sum_i^N w_i^{\mathrm{opt}} h_i(\vec{x}) - T\right), \tag{7}$$

where T is an optimal threshold that enhances results as proposed in Refs. [24,25] and is computed as a postprocessing step

$$T = \left(\frac{1}{S}\sum_s^S \frac{1}{N}\sum_i^N w_i^{\mathrm{opt}} h_i(\vec{x}_s)\right). \tag{8}$$

B. QBoost-inspired quantum classifier

An important challenge in designing a successful ensemble is to ensure that the base learners are highly diverse, i.e., that their predictions do not correlate too much with each other. The initial idea of QBoost [24] was to accomplish this by using weak learners of the same type, specifically decision tree (DT) classifiers, and train them sequentially using the boosting procedure. Another way is to use different types of base learners [33], creating a heterogeneous ensemble with a mix of different learners including, e.g., DTs, logistic regression (LR), k-nearest neighbors (kNNs), and Gaussian naive Bayes (GNB) [34]. Having inherently different mathematical foundations, these learners can exhibit significantly different views of the data landscape.

For this specific problem, we find that a classifier based on a heterogeneous ensemble comprising different types of learners can lead to better generalization performance than the plain-vanilla model with DT only. The choice of type and mixing of such a heterogeneous ensemble is motivated by comparing results from extensive simulations, both with one type of learner and with combinations thereof. Each of these models is trained on the undersampled version of the training set, and the corresponding prediction performance is obtained on a separate cross-validation set, using precision and recall. As displayed in Fig. 3, DTs perform better in recall, while kNNs perform better in precision. Heuristically, combining these two types of base learners results in the actual best-performing model over any other tested combinations.

Furthermore, in order to take advantage of the historical structure of the data, where multiple historical data points for the same companies are available, we propose to train each of the learners of the heterogeneous ensemble on different historical periods of the data set. Specifically, using date features in the data set, the raw training set is split into subsets, and then subgroups of learners are trained on them. This ensemble-training procedure based on subsampling is expected to further diversify the ensemble, where the weak learners are trained independently on the different economic recession and expansion periods underlying the training data

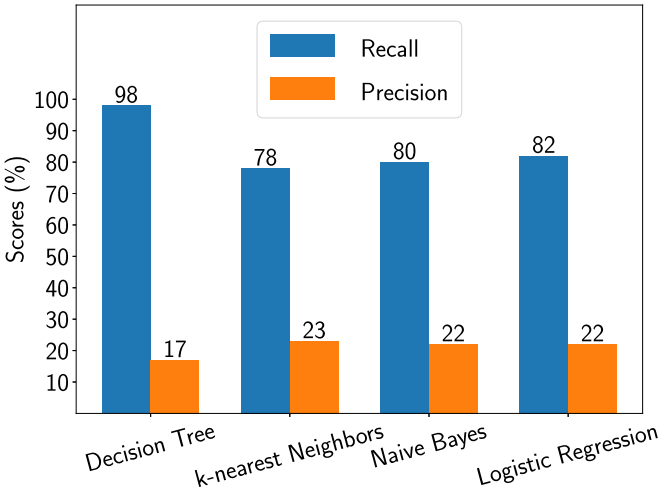

FIG. 3. Comparison of QBoost performances, i.e., recall (blue) and precision (orange), with different base learners including decision trees, k-nearest neighbors, Gaussian naive Bayes, and logistic regression.

set. Additionally, it reduces the training time significantly as each subgroup of learners is trained on a subset of data. The learners trained in this way can potentially capture different views of the data, resulting in a better diversification of the ensemble.

Here we propose this approach with two variations:

(*1*) *Boosting*. Following Ref. [24], we train each of the ensembles on the different subsamples of data with the sequential boosting procedure described in Sec. II A. Generally, the learners can exhibit negative correlations between each other.

(*2*) *Subsampling*. We train each of the ensembles without sequential boosting, relying only on subsampling for diversification. Generally, the learners exhibit only positive correlations between each other.

C. Optimization of the ensemble via quadratic unconstrained binary optimization solving

As explained in Sec. II A, the weak learners obtained during the ensemble training are then optimally combined to form a stronger classifier. Finding the best binary weights w for this combination amounts to the minimization of the cost function given in Eq. (6). Since the weights w_i are binary variables, that is, $w_i^2 = w_i$, we reformulate $\mathcal{H}$ as $\mathcal{H}_Q$:

$$\mathcal{H}_Q(w) = w^T Q w = \sum_{i,j}^{N} Q_{ij} w_i w_j, \tag{9}$$

where

$$Q_{ij} = \begin{cases} \mathrm{Corr}(h_i, h_j) & \text{if } i \neq j \\ \frac{S}{N^2} + \lambda - 2\mathrm{Corr}(h_i, y) & \text{otherwise.} \end{cases} \tag{10}$$

This second formulation is written in the form of a *quadratic unconstrained binary optimization* (QUBO) problem [35]. Solving a QUBO problem amounts to finding the minimum of a quadratic polynomial of bit variables, i.e., the optimal bit string minimizing the cost function $\mathcal{H}_Q$, with $Q \in M_N(\mathbb{R})$, the symmetric matrix encompassing the correlations between the weights to optimize.

As the number of learners grows, the classical optimization of the weights becomes exponentially hard, thus opening the door to potentially more efficient quantum methods. For instance, what if we encode the binary variables into qubit states on a quantum computer, traversing the search space as a Hilbert space? A common misconception is that quantum computers could solve in polynomial time NP-complete problems; however, this has not been proven, and in fact the consensus is that it would be extremely unlikely. However, there is mounting evidence [36] that they could better *approximate* "sufficiently good" (as defined in Sec. III A) solutions in a short(er) time compared with classical computers in some cases. This expectation partly stems from the fact that quantum computers may offer shortcuts through the optimization landscape inaccessible to traditional classical simulated annealing methods [37]. In our case, if one can produce a state such that the probability amplitudes peak in low-cost bit strings, sampling from it becomes an efficient way to optimize the weights. Such a quantum state may be potentially highly entangled and cannot be efficiently stored by classical means. Quantum computers based on neutral atoms offer unprecedented scalability, up to hundreds of atoms [38], as well as a global addressing scheme (analog mode), allowing a large set of qubits to be easily entangled. In contrast, quantum circuit-based calculations become quite greedy regarding the number of gates required to achieve such levels of complexity. We focus here on solving QUBO problems using an analog neutral atom setup.

D. Information processing on a neutral atom platform

In analog neutral atom quantum processing devices [39], each atom is considered with reasonable approximation as a simpler system described by only two of its electronic states. Each atom can thus be used as a qubit with basis states $|0\rangle$ and $|1\rangle$, these being a low-energy *ground* state and a highly excited Rydberg state, respectively [40]. The evolution of the qubit's state can be parametrized by time-dependent control fields $\Omega(t)$ and $\delta(t)$. These parameters are ultimately related to the physical properties of lasers acting on the atoms. Moreover, the quantum state of atom i can significantly alter the state of atom j, depending on their pair distance r_{ij}. Indeed, the excitation of i to a Rydberg state $|1\rangle$ shifts the energies of the corresponding Rydberg states of nearby j by an amount $U(r_{ij})$. The latter quantity can be considered impactful compared with the action of the control fields when $r_{ij} \leqslant r_b$, with r_b being the *blockade radius*. This blockade effect [41] constitutes the building block of the entangling process in neutral atom platforms.

When a laser pulse sequence acts on an entire array of N atoms, located at positions $\mathbf{r}$, the time evolution of the quantum state $|\psi\rangle$ can be expressed by the Schrödinger equation $i\hbar\frac{\partial}{\partial t}|\psi\rangle = \hat{H}(t)|\psi\rangle$, where $\hat{H}(t)$ is the system Hamiltonian. Neutral atom devices are capable of implementing the so-called *Ising* Hamiltonian, consisting of a time-dependent control part as well as a position-dependent interaction

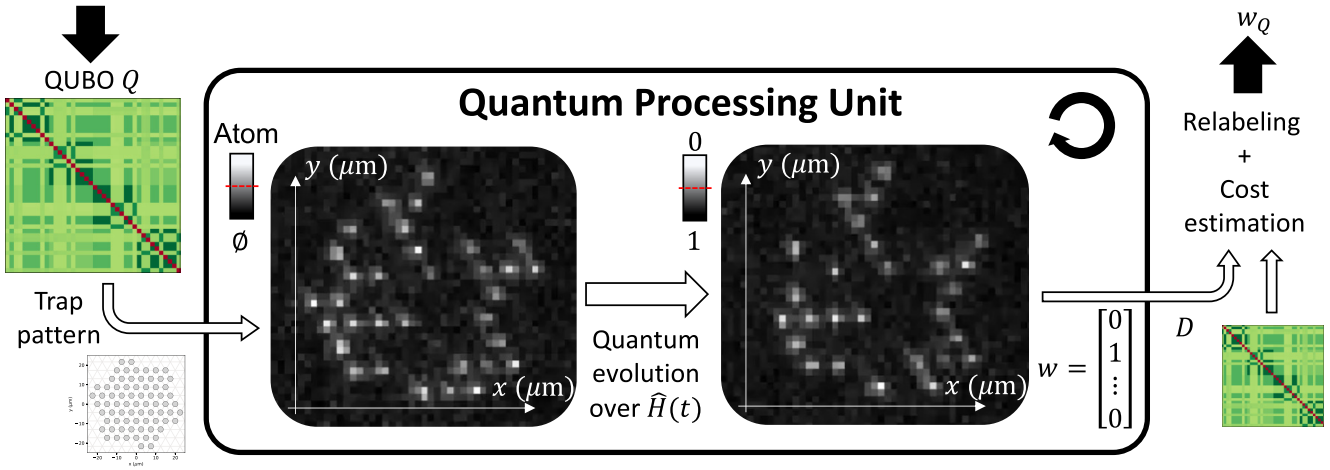

FIG. 4. Random graph sampling pipeline for solving a QUBO Q on a neutral-atom-based QPU. First, a QUBO, here with negative weights on the diagonal (red) and positive weights outside (shades of green), is taken as input. From this QUBO, a trap pattern is devised and sent to the QPU, as well as the wanted number of repetitions and the pulse sequence used. At the beginning of each cycle, a first fluorescence picture enables one to identify which traps were filled by an atom (bright spots). The system evolves according to $\hat{H}(t)$, and a final picture is taken to measure the collapsed state of the system, outputting a bit string w. Using Q, we select w_Q among D, the distributions of bit strings obtained from repeating this process several times.

part:

$$\hat{H}(t) = \hat{H}_{\text{ctrl}}(t) + \hat{H}_{\text{int}}(\mathbf{r}) = \hbar \sum_{i=1}^{N} \left(\frac{\Omega(t)}{2} \hat{\sigma}_i^x - \delta(t) \hat{n}_i \right) + \frac{1}{2} \sum_{i \neq j} U_{ij} \hat{n}_i \hat{n}_j, \quad (11)$$

where $\hat{\sigma}_i^\alpha$ are the Pauli matrices applied on the ith qubit [42], $\hat{n}_i = (1 + \hat{\sigma}_i^z)/2$ is the number of Rydberg excitations (with eigenvalues 0 or 1) on site i, and $U_{ij} = U(r_{ij}) > 0$ represents the distance-dependent interaction between qubit i and j. The state of the system is initialized to $|0 \cdots 0\rangle$. Once the pulse sequence drives the system towards its final state $|\psi\rangle = \sum_{w \in \mathbb{B}^N} a_w |w\rangle$, a global measurement is performed through fluorescence imaging: The system is projected to a basis state $|w\rangle$ with probability $|a_w|^2$. The obtained picture reveals which atoms were measured in $|0\rangle$ (bright spot) and which were in $|1\rangle$ (dark spot); see Fig. 4. Repeating the cycle (loading atoms, applying a pulse sequence, and measuring the register) multiple times constructs a probability distribution that approximates $|a_w|^2$ for $w \in \mathbb{B}^N$, allowing us to get an estimator of $|\psi\rangle$ and use it as a resource for higher-level algorithms.

E. Quantum algorithms for QUBO problems

Proposals to solve combinatorial problems, such as QUBOs, using neutral atom quantum processors abound [43–45], for example, using a quantum adiabatic algorithm (QAA) [46] or quantum approximate optimization algorithm (QAOA) [36]. One crucial ingredient of these proposals is the ability to implement a custom cost Hamiltonian $\hat{H}_Q$ on the quantum processor which should be closely related to the cost function $\mathcal{H}_Q$. When this Hamiltonian is generated exactly, the mentioned iterative methods can ensure that after k iterations the evolution of a quantum system subjected to $\hat{H}_{\text{ctrl}}^{(k)} + \hat{H}_Q$ will tend to produce low-energy states $|w\rangle$, i.e., solutions with low values $\mathcal{H}_Q(w)$. There are ways to compute the evolution over $\hat{H}_Q$ using circuit-based quantum computers [36], or special-purpose superconducting processors such as D-Wave machines [47].

For analog neutral atom technology, innately replicating the cost Hamiltonian with $\hat{H}_{\text{int}}$ would require one to satisfy $\{U_{ij} = Q_{ij} \| 1 \leqslant i \neq j \leqslant N\}$ and thus to find specific coordinates of the atoms respecting all these constraints (their number increases quadratically with the number of variables used to represent N atoms in the plane). As a consequence, only part of the constraints can be fulfilled by a given embedding, resulting in only an approximation of $\hat{H}_Q$ in a general case. Therefore the quality of the solutions sampled from the final quantum state will be limited under a straightforward implementation of a QAA or a QAOA. In addition, the optimization of variational quantum algorithms [23] usually requires diagnosing the expressibility and trainability of several circuits (or pulse sequences at the hardware level) in order to trust that low-energy states are being constructed. Moreover, at each iteration, obtaining a statistical resolution of the energy of the prepared state with precision ε, one requires $O(1/\varepsilon^2)$ samples. Given the currently low repetition rate of quantum processing units (QPUs) based on neutral atom technology (of the order of 1–5 Hz), the implementation of such approaches requires several tens of hours of operation on robust hardware. With current technology, it is therefore crucial to employ methods involving only a small budget of cycles that can quickly provide significant solutions to the QUBO problem.

F. Random graph sampling

Stepping away from the variational paradigm of the QAOA and QAA, we devised a sampling algorithm that exploits the stochastic loading probability of a neutral atom QPU in order to probe efficiently the solution space of a QUBO. This algorithm is faster to implement as it does not require iterative processes such as closed-loop communication between a classical optimizer and the quantum hardware. We propose

the random graph sampling (RGS) method, which builds upon the randomness of the atomic loading process. This procedure allows us to sample solutions of QUBOs of sizes up to 60.

In a neutral atom QPU, atoms are spatially arranged by combining the trapping capacity of optical tweezers with the programmability of a spatial light modulator (SLM). By means of these two devices, atoms can be individually trapped in arbitrary geometries. Once an atomic cloud has been loaded, each of the N_t traps is randomly filled with success probability p according to a binomial law $\mathcal{B}(N_t, p)$. Thus one must set up around $N_t = N/p$ traps to maximize the probability of trapping N atoms. A rearrangement algorithm then moves atoms one at a time to the wanted positions using a moving tweezer. The excess atoms are released midstroke to end up with a register of N correctly positioned atoms. However, by skipping the rearrangement step, we obtain samples from an essentially random subconfiguration of the underlying pattern of traps. For $N_t = 2N$, the number of possible configurations of size N scales as $\binom{2N}{N} \sim 4^N/\sqrt{N}$, offering a large variety of $\hat{H}_{\text{int}}$ for a devised pattern. In order to produce the quantum distributions from which we sample the QUBO solutions, we repeatedly apply a parametrized sequence with constant pulses to the atoms. The latter evolve under $\hat{H}(t)$ according to their interactions, which are set by the atom positions at each cycle.

The QUBO to solve, Q, first acts as a resource to design the trap pattern (N_t, shape, spacing) sent to the QPU (see Fig. 4). Once a chosen budget of samples has been acquired on the QPU, we are left with a bit-string distribution D. Using again the QUBO, we apply a relabeling procedure (described in Appendix A 1) to each bit string according to both Q and the related atom positions. This optimization procedure is designed to scale only linearly with N and is tasked to search for a way of labeling the atoms from 1 to N which minimizes for each repetition the difference between $\hat{H}_Q$ and $\hat{H}_{\text{int}}$. Finally, we compute the bit string corresponding to the optimized weights for the ensemble of learners considered.

While RGS offers no theoretical guarantee of sampling a global minimum of the cost function, we can still expect to output bit strings with low function value. We can compare RGS performances to several state-of-the-art quantum-inspired methods. Quantum-inspired algorithms are run on classical devices and allow with some approximation a fast emulation of the quantum phenomena happening in a QPU. Two numerical methods are introduced to benchmark QUBO solving: a naive analog QAOA with three pulse durations as optimizable parameters (see Appendix A 1) and simulated annealing (SA) [48,49]. The methods are always compared for similar budgets of cycle repetitions, and we can assess the quality of the bit-string distribution or the scalability of each approach. We also propose in the following a numerical tensor-networks-based algorithm to handle large QUBOs without any restriction on their structure.

G. Tensor network optimization

Tensor networks (TNs) are a mathematical description for representing quantum many-body states based on their entanglement amount and structure [50,51]. They are used to decompose highly correlated data structures, i.e., high-dimensional vectors and operators, in terms of more fundamental structures and are especially efficient in classically simulating complex quantum systems.

To be more specific, one can consider an N-qubit system and its wave function naively described by its $O(2^N)$ coefficients a_w in the computational basis. Formally, these coefficients can be represented by a tensor with N indices, each of them having two possible values (0 and 1), which quickly become costly to store and process with increasing N. However, we can replace this huge tensor by a network of interconnected tensors with fewer coefficients, defining a TN. Each subsystem corresponds in practice to the Hilbert space of one qubit. By construction, the TN depends on $O(\text{poly}(N))$ parameters only, assuming that the rank of the interconnecting indices is upper-bounded by a parameter, called the *bond dimension*. Because of polynomial scaling, TNs constitute useful tools for emulating quantum computing, and many of the current state-of-the-art simulators of quantum computers are precisely based on them.

In addition, TNs have also proven to be a natural tool for solving both classical and quantum optimization problems [9]. They have been used as an ansatz to approximate low-energy eigenstates of Hamiltonians. In our case, which involves a classical cost function, we propose an optimization algorithm based on time-evolving block decimation (TEBD) [52]. In this approach, we simulate an imaginary-time evolution driven by the classical Hamiltonian $\hat{H}_Q$ and simulate the state of the evolution at every step by a particular type of tensor network called a matrix product state (MPS). By using this approach, the algorithm reaches an optimum final configuration of the bit string minimizing the cost function. The optimal configuration of the ensemble thus achieved was used to make predictions on the unseen test set.

IV. RESULTS

In this section, we present the classification results based on the RGS quantum optimization procedure performed on a QPU for QUBOs of sizes up to 60 qubits. In addition, we present the performance of the boosting variation of the quantum classifier, already capable of beating the benchmark by reaching a precision of 29% for 90 qubits or learners, based on tensor network methods.

A. QPU optimizer results

We experimentally implement the RGS method described in Sec. II F to solve six sets of QUBOs ranging in size from $N = 12$ to $N = 60$ qubits. QUBOs of one set being produced by repeatedly applying the subsampling approach on the same data set, they only exhibit minor discrepancies between their structure and range of values. Therefore we can, within reasonable approximation and for faster implementation on the quantum hardware, use only one trap pattern per set. In addition, we can reuse statistics acquired at large sizes and extract from them distributions of bit strings of smaller size as explained in Appendix A 2. In essence, by neglecting the interaction between pairs of atoms separated by more than one site, an N-atom regular array can be divided into smaller

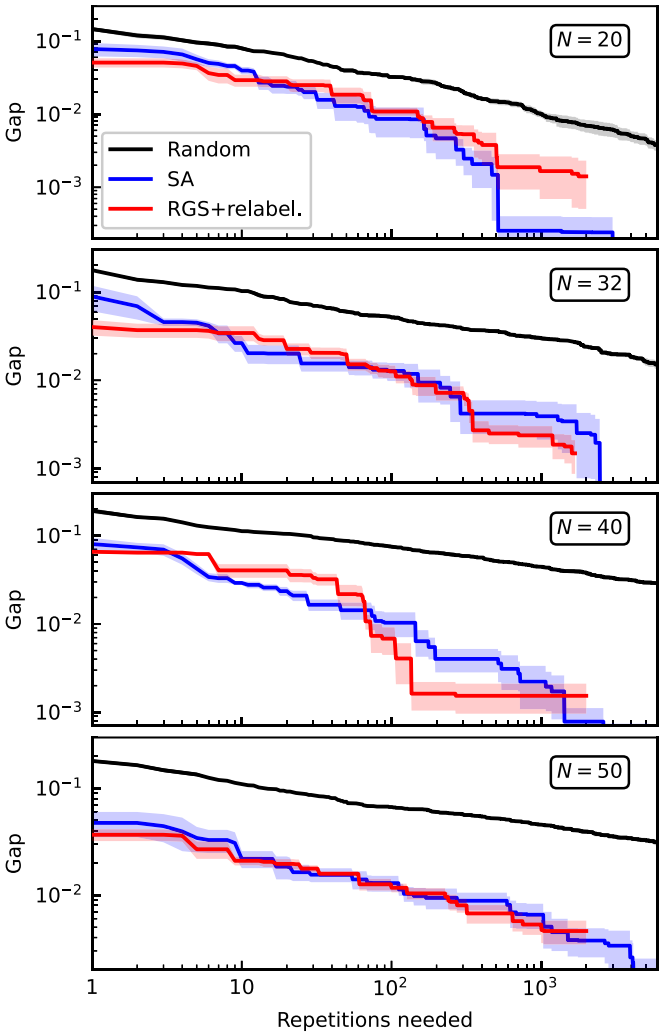

FIG. 5. Gap convergence obtained with classical uniform sampling (black), simulated annealing sampling (blue), and RGS with optimized relabeling (relabel., red) for increasing size of QUBOs. The best gap found after some cycle repetitions is averaged over sets of five QUBOs (solid curves).

clusters with similar regular shape and isolated from each other.

Considering a loading probability $p = 0.55$, we design a triangular pattern with $N_t = 40/0.55 \approx 73$ traps for QUBOs of size 40 (see Fig. 4) and similarly with 91 traps for QUBOs of size 50. While this choice is motivated by both available trapping laser power and maximization of the number of samples at sizes 40 and 50, it restricts the number of statistics gathered at size 60. Results at this size are thus subject to large uncertainty bars. The spacing of the regular pattern and thus the atomic interaction in the array is chosen in combination with the maximum value of $\Omega(t)$ reached during the pulse sequence. Having Hamiltonian terms Ω and U of comparable magnitude in $\hat{H}(t)$ enables us to explore the strongly interacting regime.

Finally, we introduce the *gap* of a bit string w defined as

$$\text{gap} = \left| \frac{\mathcal{H}_Q(w) - \mathcal{H}_Q(w_Q^0)}{\mathcal{H}_Q(w_Q^0)} \right|, \tag{12}$$

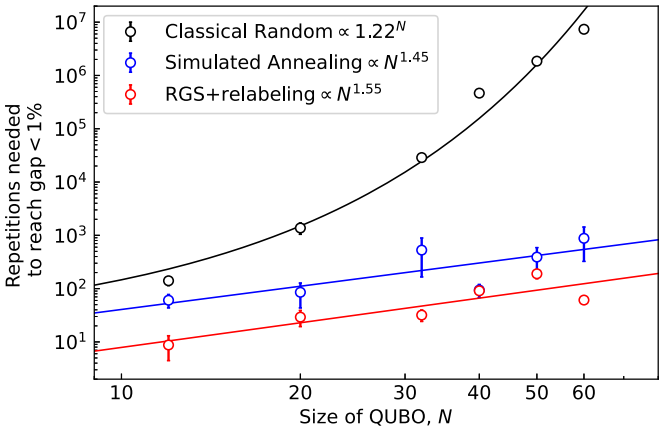

FIG. 6. Scaling analysis of the number of repetitions needed to reach a gap below a threshold of 1% with respect to problem size. The results obtained by the three mentioned methods at sizes $N = 12, 20, 32, 40, 50$, and 60 (open circles) are fitted either exponentially or polynomially (curves) depending on the best match.

where w_Q^0 is the best solution returned by a state-of-the-art SA algorithm given a large number of repetitions (200 000 here). This solution, w_Q^0, is not guaranteed to be the best possible, but acts as such for benchmark purposes. Reaching a gap of 0 amounts to having found the best solution provided by the benchmark. Note that for small sizes of N, it is possible to use an exhaustive search as a benchmark and w_Q^0 is in that case the theoretical best solution. Finding a bit string with a gap below 1%, for instance, means finding a solution with a cost close to 1% of the optimal one, which, in many operational problems such as our case study, is often sufficient. We check that for the various sizes considered, the difference between selecting a 1% solution and the optimal one, i.e., with a gap of 0, is reflected in the classification model with variation of precision P smaller than the standard deviation obtained on the QUBO set. We thus consider as a good enough solution a bit string with a gap below 1%.

The results obtained by RGS with relabeling are showcased both in terms of convergence to low-cost-value $\mathcal{H}_Q(w)$ solutions (see Fig. 5) and scalability of the method with respect to the complexity of the problem, i.e., the QUBO size (see Fig. 6). The classical random method consists in uniformly sampling with replacement bit strings from $\mathbb{B}^N$; it scales exponentially with N. In contrast, the RGS algorithm shows better performance, already finding solutions with a gap smaller than 10% after only a few repetitions. Looking at the number of repetitions needed to go below 1% with respect to N, a log-log linear fit returns a scaling in $0.2 \times N^{1.55}$. Since the QPU runtime scales linearly with the number of cycles, the quantum optimization duration is also expected to scale polynomially. Comparing RGS with the SA algorithm, we observe better performance of the latter at small sizes but more and more comparable performance at increasing sizes. In the case of $N = 40$, this specific implementation of RGS finds on average a gap below 0.2% after 150 repetitions, while SA needs around four times more cycles. For $N = 60$, the mean gap achieved after hundreds of cycles is around 1.5%. Overall, RGS with relabeling applied to QUBOs produced by

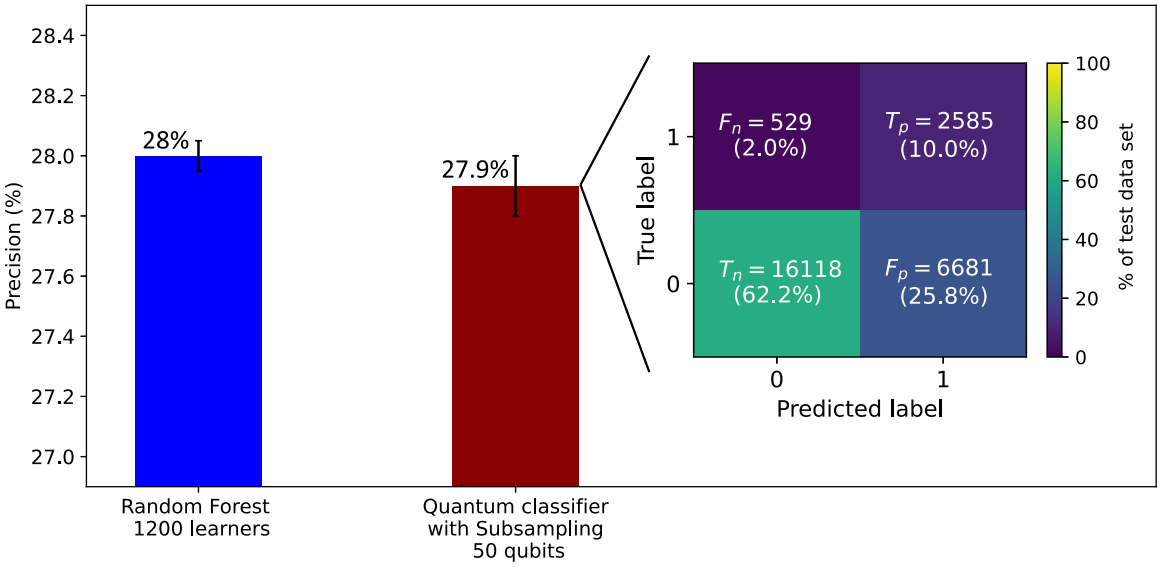

FIG. 7. QPU classification results obtained using the proposed quantum classifier based on subsampling (dark red) when optimizing 50-qubit-sized QUBOs. Its precision is compared with the precision obtained with the random forest approach (blue) using 1200 learners. On the right, the corresponding confusion matrix of the implemented model is displayed with the proportions of T_p, T_n, F_p, and F_n as percentages of the test data set.

the subsampling-based classifier exhibits similar behavior to the state-of-the-art SA algorithm.

B. QPU classification results

In this section, we present the classification results obtained using the quantum classifier based on subsampling (see Sec. II B), trained using the quantum optimizer implemented on a QPU of up to 60 qubits. This quantum classifier, leveraging the subsampling approach without boosting, is based on the optimization of QUBOs with positive off-diagonal values, amenable to efficient optimization with the current quantum hardware (see Sec. II D). We find the best results for 50 qubits (recalling that few statistics were available at 60 qubits), corresponding to an initial weak ensemble of 50 learners, whose percentages of kNNs and DTs have been optimally chosen through a hyperparameter optimization procedure. For this hyperparameter optimization (see Appendix B), the training set was split into 80% training and 20% cross-validation sets using stratified-shuffled splitting.

As depicted in Fig. 7, the proposed quantum classifier is able to achieve very similar performances to the classical random forest algorithm. Using bit strings with gap below 1%, our model reaches $P = 27.9 \pm 0.09\%$, closely approaching the benchmark threshold $P = 28.0 \pm 0.07\%$ for the same recall value of $R = 83\%$. Very interestingly, this result is obtained with only 50 initial learners compared with the random forest's ensemble of 1200 learners. The difference in the number of learners employed is of great relevance for the interpretability of the model. Indeed, the model-outputted decision for a new unseen point can be traced back more easily and better understood by the user. Furthermore, we report the total runtimes for this model up to 50 qubits in Table I of Appendix C. As seen, the best results for 50 qubits were obtained with a total runtime of around 50 min, against a total runtime of more than 3 h for the classical benchmark, representing a relevant practical speedup.

Next, we show in Fig. 8 precision values for $R = 83\%$ for two quantum classifiers (based on subsampling) with different compositions of kNNs and DTs respectively optimized to get the best possible performance up to hardware capabilities of 60 qubits (brown line; best results for 50 qubits are taken from here) and to get an optimal performance and a more favorable scaling trend at the same time (yellow line). For the sake of comparing scaling trends, linear extrapolations are applied, and the corresponding intersections between the interpolation lines and the benchmark threshold are marked. As seen, on the one hand, quantum classifier 1 (brown line) presents the best performance with a slowly increasing trend and is expected to beat the benchmark performance at around 150 qubits. On the other hand, quantum classifier 2 (yellow line) presents a lower level of performance in terms of available data points but an expected steeper increase, showing a predicted surpassing of the benchmark (blue line) and of quantum classifier 1 (brown line) at around 282 and 342 qubits, respectively.

TABLE I. Total training time including the time taken by training (ensemble training and optimization using the QPU) and hyperparameter tuning (10 min) of the proposed quantum-enhanced classifier based on subsampling.

Number of qubits	Total training time (min) Subsampling, QPU, heterogeneous
12	31.8
20	35.5
32	37.2
40	43.2
50	46.5

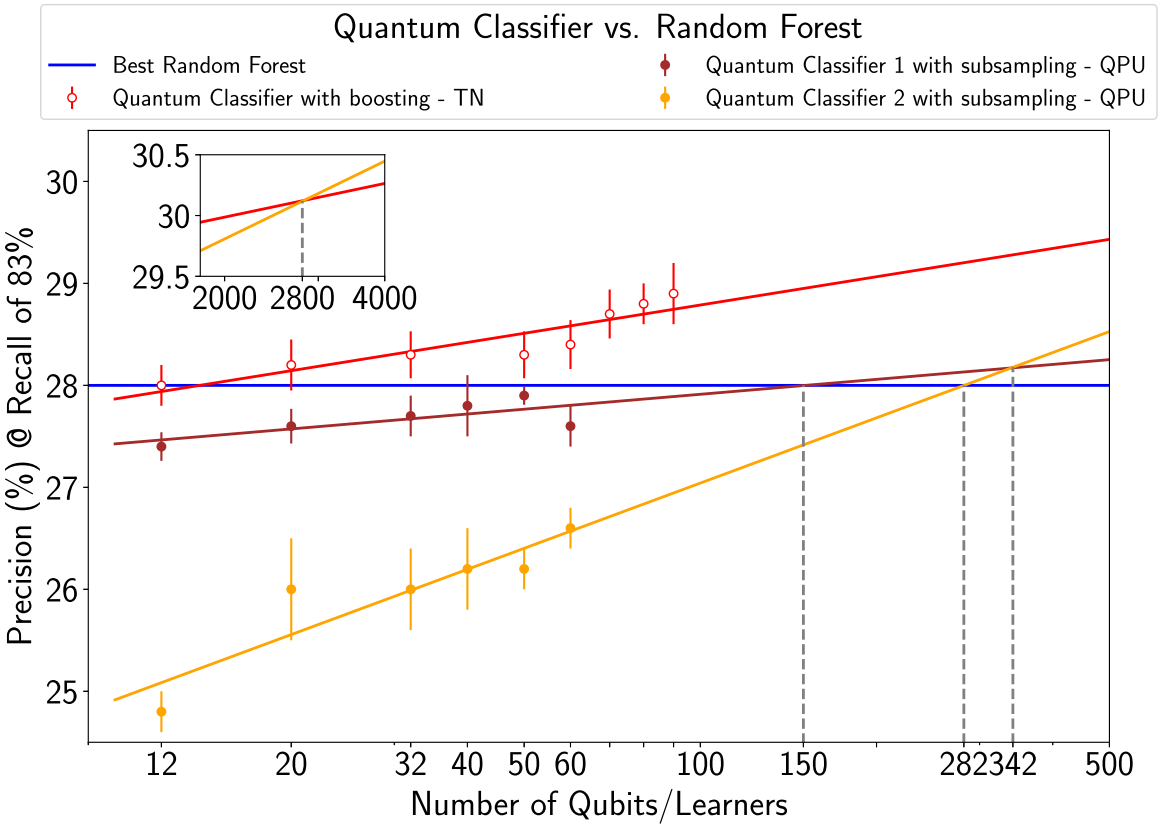

FIG. 8. Scaling of precision P of various proposed quantum classifiers with respect to the number of qubits, keeping $R = 83\%$. The two variations of the subsampling approach (yellow, brown) are implemented on a QPU (solid circles) with between 12 and 60 qubits. The boosting approach (red) is implemented using the TN optimizer (open circles) with between 12 and 90 qubits. The best performance of the random forest classifier acts as the threshold (blue). The error bars represent the variability in the corresponding performance across five iterations or QUBOs. Scaling projections are obtained by linear extrapolation (solid lines).

C. TN classification results

In Fig. 8 we also show the mean classification performance of the quantum classifier optimized with the TN and based on boosting (red line). This model being based on the boosting procedure leverages the optimization of QUBOs with negative off-diagonal values which cannot be currently directly optimized on a neutral atom QPU. It can be seen that even at low values of qubits or learners, the proposed model based on boosting already showed the same level of performance as the random forest with 1200 trees. With 90 learners, it shows a mean precision score of about $P = 29\%$ (reducing the false positives by 1%) corresponding to a recall of 83%. An outlook of the training and total runtimes for this heterogeneous model and the homogeneous variation with just DTs can be found in Table II of Appendix C. The best results for 90 qubits or learners present a total runtime of the order of 20 min against the total runtime of more than 3 h for the classical benchmark, attaining also in this case a relevant practical speedup.

Based on the scaling projections, it can be argued that this type of model is expected to remain the best-performing one. It can be seen in the inset of Fig. 8 that a crossing with quantum classifier 2 (yellow line) based on subsampling could occur for a large number of qubits, around 2800, although it is difficult to assess the reliability of the extrapolation for such high numbers.

D. Current limitations and future upgrades

The training of the proposed model involves optimization of a heterogeneous ensemble comprising DTs and kNNs. Due to the slow execution speed and large memory requirements of kNNs, the training time of this model was found to be relatively higher than the homogeneous models comprising

TABLE II. Total training time including the time taken by training (ensemble training and optimization using tensor network optimization) and hyperparameter tuning (10 min) of the proposed homogeneous and heterogeneous quantum-enhanced classifiers based on boosting.

	Total training time (min)	
Number of qubits	Boosting, TN, homogeneous	Boosting, TN, heterogeneous
12	10.1	11.3
20	10.3	12.1
32	10.6	13.6
50	11.4	15.5
60	12.1	16.7
70	13.9	17.8
80	14.8	19.4
90	15.2	20

just DTs. In the near future, this could be overcome by using a faster implementation of kNNs [53], leveraging GPU architectures, for instance.

Implementing the boosting variation of the classifier directly on neutral atom hardware is at the moment hindered by several limitations. As presented in Sec. II D, perfectly embedding a QUBO into atomic positions requires one to satisfy some constraints of the form $\{Q_{ij} = U_{ij}\}_{i \neq j}$. Choosing a specific Rydberg state to implement the Ising model implies that the interaction U will be positive between all atomic pairs. Choosing another Rydberg state such as $|1\rangle$ can enable one to have negative interactions, but only globally. Thus QUBOs with both positive and negative off-diagonal values, such as those produced in the boosting variation, cannot be natively implemented, restricting the type of classification models accessible on a current neutral atom QPU.

However, as shown in Fig. 8, the subsampling variation, which produces QUBOs with all positive off-diagonal values, could be able to beat the threshold set by the benchmark at around 150 qubits. In order to optimize a heterogeneous ensemble of this size with RGS, a pattern with around 290 traps is needed. In a recent work [38], a new prototype successfully produced atomic arrays of size $N = 324$ with patterns of 625 traps. As more capabilities become available for this hardware technology, future implementations may offer an opportunity to achieve an industry-relevant quantum value by beating state-of-the-art methods at larger numbers of qubits or learners.

V. CONCLUSIONS AND OUTLOOK

In this paper, we propose a quantum-enhanced machine learning solution for the prediction of credit rating downgrades, also known as fallen-angels forecasting. Our algorithm comprises a hybrid classical-quantum classification model based on QBoost, tested on a neutral atom quantum platform and benchmarked against random forest, one of the state-of-the-art classical machine learning techniques used in the finance industry. We report that the proposed classifier trained on a QPU achieved competitive performance with 27.9% precision against the benchmarked 28% precision for the same recall of approximately 83%. However, the proposed approach outperformed its classical counterpart with respect to interpretability with only 50 learners employed versus 1200 for the random forest and comparable runtimes. These results were obtained leveraging the hardware-tailored random graph sampling method to optimize QUBOs up to size 60. The RGS method showed similar performance to the simulated annealing approach and was able to provide solutions to QUBO within an acceptable repetition budget.

We also report a classification model based on the proposed heterogeneous structure and leveraging the boosting procedure that, although it is not amenable to being trained on current hardware, was trained using a quantum-inspired optimizer based on tensor networks. This model showed the capability to already perform better across all the relevant metrics, achieving a precision of 29%, which is 1% above the benchmark, with just 90 learners (against 1200) and runtimes of around 20 min compared with more than 3 h for the benchmarked random forest.

Going forward, hardware upgrades in terms of qubit numbers will lead to performance improvements. This behavior is illustrated in Fig. 8, where we show how the precision of the quantum classifiers evolves with system size. Extrapolating from these results and keeping other factors constant, the proposed quantum classifiers could outperform the benchmarked model within a few hundred addressable qubits. In addition, hardware improvements enabling the resolution of QUBOs with negative off-diagonal values could offer additional advantages to the quantum solution and improve performance over the classical benchmark.

These results open up the way for quantum-enhanced machine learning solutions to a variety of similar problems that can be found in the finance industry. Interpretability and performance improvements for real cases with complex and highly imbalanced data sets are pressing issues. As such, we foresee a large number of applications for quantum-enhanced machine learning, especially implemented on neutral atom platforms, in solving computationally challenging problems of the financial sector.

ACKNOWLEDGMENTS

L.L., L.O-G., S.G., B.A., J.R.K.C., V.E.E., A.S., and L.H. thank Mathieu Moreau, Anne-Claire Le Hénaff, Mourad Beji, Henrique Silvério, Louis-Paul Henry, Thierry Lahaye, Antoine Browaeys, Christophe Jurczak, and Georges-Olivier Reymond for fruitful discussions. G.D.B., U.A.S., M. S., L. A., F.I., A.D., S.M., I.C., M.K. and R.O. acknowledge fruitful discussions with the rest of the technical team as well as Creative Destruction Lab, BIC-Gipuzkoa, DIPC, Ikerbasque, Diputacion de Gipuzkoa, and the Basque Government for constant support. All the authors thank Loïc Chauvet and Ali El Hamidi.

APPENDIX A: RGS DETAILS

1. Optimized relabeling

We describe here the relabeling process used in random graph sampling (see Sec. II F). For a given cycle where N traps out of N_t are filled with atoms, a first measure of the system before the quantum processing part enables us to locate the atoms, as shown in Fig. 9(a). The latter are randomly labeled, and this memorized labeling usually orders the bit string measured after the quantum processing. However, we can choose another labeling more specific to the QUBO we want to solve. This postprocessing step determines a labeling σ_{opt} of the atoms such that the resulting interaction matrix better reproduces the QUBO matrix than the one obtained from the randomly generated graph. For each way of labeling the atoms from 1 to N, i.e., each permutation of length N, we compute the separation

$$s_Q(\sigma) = \sum_{i<j} \|U_{\sigma(i)\sigma(j)} - Q_{ij}\|, \tag{A1}$$

where Q is the QUBO matrix, σ is a permutation of length N, and $U_{\sigma(i)\sigma(j)}$ is the interaction term between atoms originally named i and j. The two matrices are normalized to allow a proper comparison. We perform a random search with fixed budget n_{iter} over the $N!$ possible labeling permu-

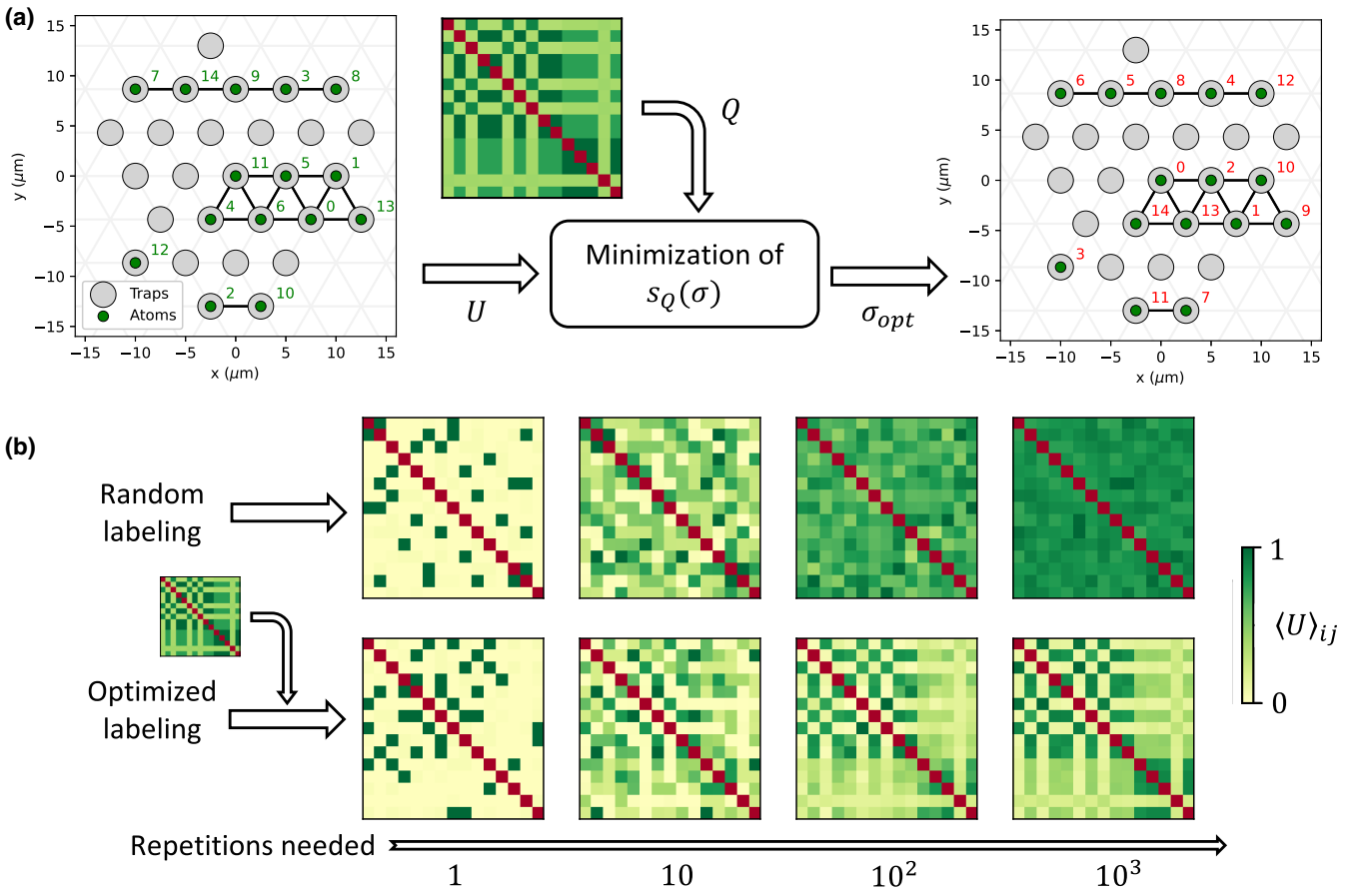

FIG. 9. (a) Fifteen atoms (green dots) are filling a fraction of a triangular trap layout (gray circles). Each atom is randomly labeled (green), and from their positions an interaction matrix U is derived. Using the QUBO to solve, Q, and Eq. (A1), a permutation of the labels σ_{opt} is found. The atoms are relabeled (red) such that the resulting interaction matrix better replicates Q. (b) Normalized interaction matrices obtained when averaging over many repetitions of traps loading. While the random labeling (top row) produces a uniform matrix, the optimized labeling (bottom row) enables us to access some features of the QUBO at each cycle, producing an average matrix resembling Q.

tations. The permutation minimizing s_Q is then used to read out the measured bit string. Searching for such a permutation is reasonably fast for the sizes that can be loaded in the QPU. We have checked that for $N \leqslant 100$, this takes less than $n_{\text{iter}} \times 2$ ms. In the following, we set $n_{\text{iter}} = 10N$ so as to scale only linearly with the number of qubits and not as $N!$. This may not be sufficient to identify the best permutation, but it remains enough to reproduce some of the features of the QUBO at each cycle as shown in Fig. 9(b). Furthermore, on average, the whole QUBO is much better represented with this optimized relabeling step than simply using a random permutation. It is worth pointing out that this optimization step can be done retrospectively, after the quantum data have been acquired, as long as we have access to the initial traps' filling. Thus its execution time does not limit the duration of a cycle, and this can become effectively a postprocessing step performed on a classical computer.

We benchmark this approach on a set of randomly generated QUBOs of size 15 and compare it with a classical uniform sampling of $\mathbb{B}^N$ and with a numerical simulation of a QAOA [54], all using a similar budget of 1000 cycles, or measurements. Getting into the details, the QAOA is allowed ten iterations with 100 cycles each in order to optimize the duration of three pulses. The cost function evaluated at each iteration is $\langle \mathcal{H}_Q \rangle$ averaged over the 100 measurements. The atoms are located at the same positions for each iteration, meaning that an experimental implementation would use the rearrangement algorithm, lengthening the duration of each cycle. In contrast, for RGS, the positions are random at each cycle, while the pulse sequence remains the same: three pulses with nonoptimized durations. We show the results of these three methods in Fig. 10 with both the convergence of each one with respect to the number of cycles performed and the aggregated bit-string distributions sorted by increasing values of $\mathcal{H}_Q$. Not only does RGS converge faster, achieving a gap of less than 1% with three times fewer cycles than the QAOA, but also it produces, on average, sampled distributions with a greater concentration on bit strings with low value. For this set, a bit string sampled using RGS+relabeling is on average in around the best 11% of $\mathbb{B}^N$, while one sampled with the QAOA is in around the best 17%.

2. Configuration clustering

In this section, we elaborate on how to extract usable bit strings of size n from ones of size $N > n$ measured in the quantum setup. Those smaller bit strings can in specific cases be used to solve QUBOs of size n.

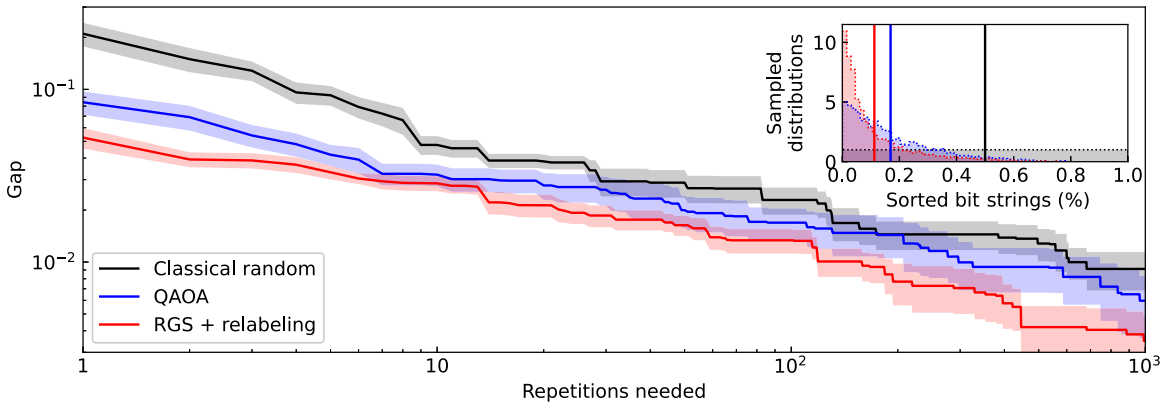

FIG. 10. Results obtained from a classical random search, a numerically simulated QAOA, and numerically simulated RGS with QUBO-dependent labeling (averaged over ten random QUBO instances of size $N = 15$). The inset shows the frequency and the mean of the proposed solutions with each method, ranked by their cost function value.

At each computation cycle, a pattern of N_t traps is filled by $N \sim \mathcal{B}(N_t, p)$ atoms. Many cycles would then produce bit strings whose sizes follow a Gaussian distribution centered around $N_t\, p$ as shown in Fig. 11. For each cycle, knowing which traps were filled (as shown in the inset), we can isolate a cluster of atoms with the following rule: Two atoms belong to different clusters if their pair distance exceeds the pattern spacing. Therefore, due to the rapid decay of the interaction with the distance, i.e., $U(r_{ij}) \propto r_{ij}^{-6}$, we can consider that clusters do not interact between them. Indeed, here, two non-neighboring atoms are interacting at least 27 times more weakly than a pair of neighboring ones. Segmenting an N-sized bit string leads to the extraction of s smaller bit strings with sizes n_i such that $\sum_i^s n_i = N$. Applying this method to the original distribution of ∼65 000 measurements, ranging in size from 34 to 66 atoms, outputs a wider distribution of ∼334 000 bit strings ranging in size from 1 to 66. This method produces bit strings from fully interacting systems, as no atom remains isolated. However, it reduces the number of measurements made at a large size N.

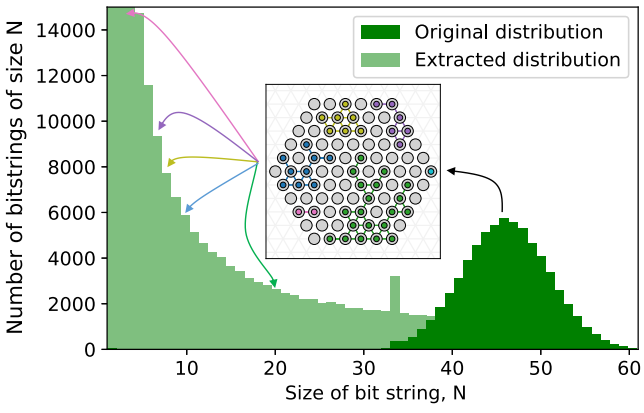

FIG. 11. Clustering of atomic configurations to extract n-sized bit strings from N-sized ones with $N > n$. From an original distribution of 65 000 bit strings (dark green), we construct a larger distribution of 334 000 bit strings (light green). A bit string of size 45 has been measured with the atomic configuration displayed in the inset. Atoms are sorted between clusters (various colors) of sizes 2, 6, 7, 9, and 20, and the initial bit string is cut into five smaller bit strings, usable to solve QUBOs of corresponding sizes.

The resulting bit strings can only be used to solve QUBOs of corresponding sizes and which would have produced the same trap pattern as the one used to acquire the original distribution. In this implementation, since we only consider the QUBOs' output by the subsampling approach detailed in Sec. II B, all of them are similar in structure, being produced by the same data set and with the same hyperparameters for weak learner ensemble generation. We apply the relabeling step to the extracted distribution in order to solve the considered sets of QUBOs (see Sec. III A).

APPENDIX B: HYPERPARAMETER TUNING

For hyperparameter tuning of the proposed quantum classifier, grid-search cross-validation-based optimization over a list of possible values was used. An important hyperparameter of QBoost is regularization (or λ), which serves to penalize complex models with more learners in order to achieve a better generalization on unseen data [see Eq. (6)].

As the number of base learners N was increased, the time required to train the classifier increased. Since tuning of the regularization parameter involved retraining the classifier with many different values of λ, it was a computationally expensive procedure that needed to be sped up. Consequently, taking advantage of an insight into the cost function, it was proposed to run the hyperparameter-tuning procedure using a smaller and simpler variation of the classifier, comprising $N = 10$ learners, and use the corresponding optimal λ_{10} for any $N > 10$ by multiplying it by a factor of $10/N$, inspired by the scaling of the λ's boundary discussed in Ref. [25]. In other words, for all the variations of the proposed quantum classifiers, the hyperparameter-tuning procedure took a fixed time of about 10 min (see Tables I and II).

APPENDIX C: RUNTIMES

Table I presents QPU runtimes of the proposed quantum classifier based on subsampling, i.e., the time taken by training which includes ensemble training on the CPU, optimization on the QPU, and hyperparameter tuning (see Appendix B). In

this case, the training set was oversampled. The QPU runtimes are obtained by multiplying the number of cycles needed to output a bit string with gap [see Eq. (12)] smaller than 1% by the current repetition rate of the device. Table II, on the other hand, presents runtimes of the two different variations of the proposed quantum classifier based on boosting, with an undersampled training set. While the homogeneous variation is based on an ensemble of only decision trees, the heterogeneous variation comprises a mix of decision trees (DTs) and k-nearest neighbors (kNNs). It can be seen that the training of a heterogeneous classifier generally takes longer than the training of a homogeneous classifier.

[1] S. Lessmann, B. Baesens, H.-V. Seow, and L. C. Thomas, Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research, Eur. J. Oper. Res. **247**, 124 (2015).

[2] X. Xie, S. Pang, and J. Chen, Hybrid recommendation model based on deep learning and stacking integration strategy, Intell. Data Anal. **24**, 1329 (2020).

[3] S. Chen, Z. Guo, and X. Zhao, Predicting mortgage early delinquency with machine learning methods, Eur. J. Oper. Res. **290**, 358 (2021).

[4] O. Husain and E. Kambal, Credit scoring based on ensemble learning: The case of sudanese banks, in *Advances in Data Mining* (Springer, Cham, Switzerland, 2019), p. 60.

[5] B. R. Gunnarsson, S. V. Broucke, B. Baesens, M. Óskarsdóttir, and W. Lemahieu, Deep learning for credit scoring: Do or don't?, Eur. J. Oper. Res. **295**, 292 (2021).

[6] X. Dastile and T. Celik, Making deep learning-based predictions for credit scoring explainable, IEEE Access **9**, 50426 (2021).

[7] R. Orús, S. Mugel, and E. Lizaso, Quantum computing for finance: Overview and prospects, Rev. Phys. **4**, 100028 (2019).

[8] R. Patel, C.-W. Hsing, S. Sahin, S. S. Jahromi, S. Palmer, S. Sharma, C. Michel, V. Porte, M. Abid, S. Aubert, P. Castellani, C.-G. Lee, S. Mugel, and R. Orus, Quantum-inspired tensor neural networks for partial differential equations, arXiv:2208.02235.

[9] S. Mugel, C. Kuchkovsky, E. Sanchez, S. Fernandez-Lorenzo, J. Luis-Hita, E. Lizaso, and R. Orus, Dynamic portfolio optimization with real datasets using quantum processors and quantum-inspired tensor networks, Phys. Rev. Res. **4**, 013006 (2022).

[10] P. Bermejo and R. Orus, Variational quantum and quantum-inspired clustering, arXiv:2206.09893.

[11] A. Borle, V. E. Elfving, and S. J. Lomonaco, Quantum approximate optimization for hard problems in linear algebra, arXiv:2006.15438.

[12] O. Kyriienko, A. E. Paine, and V. E. Elfving, Solving nonlinear differential equations with differentiable quantum circuits, Phys. Rev. A **103**, 052416 (2021).

[13] A. E. Paine, V. E. Elfving, and O. Kyriienko, Quantum quantile mechanics: Solving stochastic differential equations for generating time-series, Adv. Quantum Technol. **6**, 2300065 (2023).

[14] O. Kyriienko, A. E. Paine, and V. E. Elfving, Protocols for trainable and differentiable quantum generative modelling, arXiv:2202.08253.

[15] O. Kyriienko and E. B. Magnusson, Unsupervised quantum machine learning for fraud detection, arXiv:2208.01203.

[16] M. Schuld, I. Sinayskiy, and F. Petruccione, An introduction to quantum machine learning, Contemp. Phys. **56**, 172 (2015).

[17] J. Biamonte, P. Wittek, N. Pancotti, and P. Rebentrost, Quantum machine learning, Nature (London) **549**, 195 (2017).

[18] M. Schuld and N. Killoran, Quantum machine learning in feature Hilbert spaces, Phys. Rev. Lett. **122**, 040504 (2019).

[19] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, Quantum circuit learning, Phys. Rev. A **98**, 032309 (2018).

[20] L.-P. Henry, S. Thabet, C. Dalyac, and L. Henriet, Quantum evolution kernel: Machine learning on graphs with programmable arrays of qubits, Phys. Rev. A **104**, 032416 (2021).

[21] A. E. Paine, V. E. Elfving, and O. Kyriienko, Quantum kernel methods for solving regression problems and differential equations, Phys. Rev. A **107**, 032428 (2023).

[22] B. Albrecht, C. Dalyac, L. Leclerc, L. Ortiz-Gutiérrez, S. Thabet, M. D'Arcangelo, V. E. Elfving, L. Lassablière, H. Silvério, B. Ximenez, L.-P. Henry, A. Signoles, and L. Henriet, Quantum feature maps for graph machine learning on a neutral atom quantum processor, Phys. Rev. A **107**, 042615 (2023).

[23] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, and P. J. Coles, Variational quantum algorithms, Nat. Rev. Phys. **3**, 625 (2021).

[24] H. Neven, V. S. Denchev, G. Rose, and W. G. Macready, Training a binary classifier with the quantum adiabatic algorithm, arXiv:0811.0416.

[25] H. Neven, V. S. Denchev, G. Rose, and W. G. Macready, Training a large scale classifier with the quantum adiabatic algorithm, arXiv:0912.0779.

[26] V. S. Denchev, N. Ding, S. V. N. Vishwanathan, and H. Neven, Robust classification with adiabatic quantum optimization, in *Proceedings of the 29th International Conference on International Conference on Machine Learning*, ICML'12 (Omnipress, Madison, WI, 2012), pp. 1003–1010.

[27] A. Mott, J. Job, J.-R. Vlimant, D. Lidar, and M. Spiropulu, Solving a Higgs optimization problem with quantum annealing for machine learning, Nature (London) **550**, 375 (2017).

[28] E. Boyda, S. Basu, S. Ganguly, A. Michaelis, S. Mukhopadhyay, and R. R. Nemani, Deploying a quantum annealing processor to detect tree cover in aerial imagery of California, PLoS ONE **12**, e0172505 (2017).

[29] J. Dulny III and M. Kim, Developing quantum annealer driven data discovery, arXiv:1603.07980.

[30] L. Breiman, Random forests, Mach. Learn. **45**, 5 (2001).

[31] Y. Freund and R. E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, J. Comput. Syst. Sci. **55**, 119 (1997).

[32] R. Wang, AdaBoost for feature selection, classification and its relation with SVM, a review, Phys. Proc. **25**, 800 (2012).

[33] J. N. van Rijn, G. Holmes, B. Pfahringer, and J. Vanschoren, The online performance estimation framework: heterogeneous

ensemble learning for data streams, Mach. Learn. **107**, 149 (2018).

[34] C. M. Bishop, *Pattern Recognition and Machine Learning* (Springer, New York, 2006), Vol. 4.

[35] F. Glover, G. Kochenberger, and Y. Du, A tutorial on formulating and using QUBO models, arXiv:1811.11538.

[36] E. Farhi, J. Goldstone, and S. Gutmann, A quantum approximate optimization algorithm, arXiv:1411.4028.

[37] E. Crosson and A. W. Harrow, Simulated quantum annealing can be exponentially faster than classical simulated annealing, in *Proceedings of the 2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)* (IEEE, Piscataway, NJ, 2016), pp. 714–723.

[38] K.-N. Schymik, B. Ximenez, E. Bloch, D. Dreon, A. Signoles, F. Nogrette, D. Barredo, A. Browaeys, and T. Lahaye, *In situ* equalization of single-atom loading in large-scale optical tweezer arrays, Phys. Rev. A **106**, 022611 (2022).

[39] L. Henriet, L. Beguin, A. Signoles, T. Lahaye, A. Browaeys, G.-O. Reymond, and C. Jurczak, Quantum computing with neutral atoms, Quantum **4**, 327 (2020).

[40] N. Šibalić and C. S. Adams, *Rydberg Physics* (IOP, London, 2018).

[41] M. D. Lukin, M. Fleischhauer, R. Cote, L. M. Duan, D. Jaksch, J. I. Cirac, and P. Zoller, Dipole blockade and quantum information processing in mesoscopic atomic ensembles, Phys. Rev. Lett. **87**, 037901 (2001).

[42] The Pauli matrices are

$$\hat{\sigma}^x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix},$$

$$\hat{\sigma}^y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix},$$

$$\hat{\sigma}^z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

The application of a Pauli matrix on the ith site of the quantum system is represented by a tensor (or Kronecker) product of matrices: $\sigma_i^\alpha = \mathbb{I} \otimes \cdots \otimes \sigma^\alpha \otimes \cdots \mathbb{I}$, for $\alpha = x, y, z$. For instance, $\hat{\sigma}_1^x|00\rangle = |10\rangle$. A similar construction is used for composite operators such as $\hat{n}_i\hat{n}_j$.

[43] S. Ebadi, A. Keesling, M. Cain, T. T. Wang, H. Levine, D. Bluvstein, G. Semeghini, A. Omran, J.-G. Liu, R. Samajdar, X.-Z. Luo, B. Nash, X. Gao, B. Barak, E. Farhi, S. Sachdev, N. Gemelke, L. Zhou, S. Choi, H. Pichler *et al.*, Quantum optimization of maximum independent set using Rydberg atom arrays, Science **376**, 1209 (2022).

[44] H. Pichler, S.-T. Wang, L. Zhou, S. Choi, and M. D. Lukin, Quantum optimization for maximum independent set using Rydberg atom arrays, arXiv:1808.10816.

[45] M. Lanthaler, C. Dlaska, K. Ender, and W. Lechner, Rydberg-blockade-based parity quantum optimization, Phys. Rev. Lett. **130**, 220601 (2023).

[46] E. Farhi, J. Goldstone, S. Gutmann, J. Lapan, A. Lundgren, and D. Preda, A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem, Science **292**, 472 (2001).

[47] S. Zbinden, A. Bärtschi, H. Djidjev, and S. Eidenbenz, Embedding algorithms for quantum annealers with Chimera and Pegasus connection topologies, in *High Performance Computing*, Lecture Notes in Computer Science Vol. 12151 (Springer, Cham, Switzerland, 2020), pp. 187–206.

[48] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, Optimization by simulated annealing, Science **220**, 671 (1983).

[49] A. Das and B. K. Chakrabarti, Colloquium: Quantum annealing and analog quantum computation, Rev. Mod. Phys. **80**, 1061 (2008).

[50] R. Orús, A practical introduction to tensor networks: Matrix product states and projected entangled pair states, Ann. Phys. (Amsterdam) **349**, 117 (2014).

[51] R. Orús, Tensor networks for complex quantum systems, Nat. Rev. Phys. **1**, 538 (2019).

[52] G. Vidal, Efficient classical simulation of slightly entangled quantum computations, Phys. Rev. Lett. **91**, 147902 (2003).

[53] Y. Song, J. Liang, J. Lu, and X. Zhao, An efficient instance selection algorithm for k nearest neighbor regression, Neurocomputing **251**, 26 (2017).

[54] H. Silvério, S. Grijalva, C. Dalyac, L. Leclerc, P. J. Karalekas, N. Shammah, M. Beji, L.-P. Henry, and L. Henriet, Pulser: An open-source package for the design of pulse sequences in programmable neutral-atom arrays, Quantum **6**, 629 (2022).