RESEARCH ARTICLE

Implementation and empirical evaluation of a quantum machine learning pipeline for local classification

Enrico Zardini[1]*, **Enrico Blanzieri**[1,2], **Davide Pastorello**[1,2¤]

1 Department of Information Engineering and Computer Science, University of Trento, Trento, Italy, **2** Trento Institute for Fundamental Physics and Applications, Trento, Italy

¤ Current address: Department of Mathematics, Alma Mater Studiorum, Università di Bologna, Bologna, Italy
* enrico.zardini@unitn.it

OPEN ACCESS

Citation: Zardini E, Blanzieri E, Pastorello D (2023) Implementation and empirical evaluation of a quantum machine learning pipeline for local classification. PLoS ONE 18(11): e0287869. https://doi.org/10.1371/journal.pone.0287869

Editor: Chun-An Chou, Northeastern University, UNITED STATES

Received: February 7, 2023

Accepted: June 14, 2023

Published: November 13, 2023

Data Availability Statement: The datasets used are available at https://doi.org/10.6084/m9.figshare.22333102.v1, while the raw results data are available at https://doi.org/10.6084/m9.figshare.22333147.v1. In addition, datasets and raw results data, together with configuration files and execution scripts, are reachable from the https://github.com/ZarHenry96/quantum-ml-pipeline GitHub repository.

Funding: This work was supported by Q@TN, the joint lab between University of Trento, FBK-Fondazione Bruno Kessler, INFN-National Institute

Abstract

In the current era, quantum resources are extremely limited, and this makes difficult the usage of quantum machine learning (QML) models. Concerning the supervised tasks, a viable approach is the introduction of a quantum locality technique, which allows the models to focus only on the neighborhood of the considered element. A well-known locality technique is the *k*-nearest neighbors (*k*-NN) algorithm, of which several quantum variants have been proposed; nevertheless, they have not been employed yet as a preliminary step of other QML models. Instead, for the classical counterpart, a performance enhancement with respect to the base models has already been proven. In this paper, we propose and evaluate the idea of exploiting a quantum locality technique to reduce the size and improve the performance of QML models. In detail, we provide (i) an implementation in Python of a QML pipeline for local classification and (ii) its extensive empirical evaluation. Regarding the quantum pipeline, it has been developed using Qiskit, and it consists of a quantum *k*-NN and a quantum binary classifier, both already available in the literature. The results have shown the quantum pipeline's equivalence (in terms of accuracy) to its classical counterpart in the ideal case, the validity of locality's application to the QML realm, but also the strong sensitivity of the chosen quantum *k*-NN to probability fluctuations and the better performance of classical baseline methods like the random forest.

1 Introduction

In the current era, known as noisy intermediate-scale quantum (NISQ) [1], the available quantum devices are limited in the number of qubits and the fidelity of gates. The qubit is the quantum analogue of a classical bit; in 2019, Google claimed to have reached quantum supremacy, i.e., an advantage with respect to classical computers, using a processor with 54 qubits [2], a very large number compared to the chips accessible for practical purposes nowadays. Nevertheless, the performance in the task considered by Google was still very low. Regarding the fidelity of gates, it is always undermined by the presence of intrinsic fabrication issues since these universal quantum architectures are based mainly on superconducting circuits.

for Nuclear Physics and CNR-National Research Council. In addition, this work was partially supported by project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU. Eventually, the authors gratefully acknowledge the Italian Ministry of University and Research (MUR), which, under the initiative "Dipartimenti di Eccellenza 2018-2022 (Legge 232/2016)", has provided the computational resources used in the experiments. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing interests: The authors have declared that no competing interests exist.

Specifically, the gate fidelity represents how much the operation performed by a gate is close to the ideal one [3]. This measure also worsens by increasing the number of qubits due to the lack of all-to-all connections between them.

In this scenario, the usage of quantum machine learning (QML) models turns out to be difficult: it is not possible to encode large quantities of data in the quantum devices due to the low number of qubits (typically, also a data index must be encoded); in addition, the presence of noise in the execution of quantum operations limits the quality of results, especially if a consistent amount of samples is considered. Several quantum machine learning models have already been proposed, either classical-quantum hybrid [4] or entirely quantum [5]. The latter type is, of course, the most interesting one since the models can fully exploit the quantum potentialities without the need to interact with classical procedures; at the same time, these models suffer more from the aforementioned issues. For instance, the quantum support-vector machine (SVM) proposed by Rebentrost et al. [5] has been implemented and tested by Z. Li et al. [6], but the task considered was very small.

In practice, the possibility of reducing the number of qubits required to solve a problem is extremely relevant in the NISQ era. In this way, it becomes feasible to address bigger, and thus more significant, problems and concretely exploit the potentialities of quantum machine learning. Focusing on the supervised tasks, the introduction of a quantum locality technique represents a valid direction in this sense. Indeed, by looking at only the neighbourhood of the considered element, it is possible to reduce the number of samples that the quantum models must process. An alternative approach could be reducing the number of qubits required to encode each sample by using an autoencoder or a dimensionality reduction technique such as the singular value decomposition (SVD). Actually, a classical-quantum hybrid version of these two techniques has already been proposed by Romero et al. [7] and X. Wang et al. [8], respectively. It is also worth highlighting that the locality addressed in this work is not the spatial locality of data features exploited by quantum convolutional neural networks in processing images [9, 10], but the locality of data samples in the features space. Indeed, (quantum) convolutional neural networks exploit the structure of neighboring features in an image to classify it. Instead, the approach described here consists in classifying the target instance based on the training data samples that are closer to it according to a chosen metric, and no specific structure in neighboring features is needed.

While the reduction of the number of input samples to a quantum machine learning model through a quantum locality technique has not been addressed yet in the literature, the classical counterpart has already been investigated and has proven successful, with performance improvements with respect to the base model. For instance, a local SVM trained on the samples selected by a k-nearest neighbors (k-NN) model has been proposed and empirically tested by Blanzieri and Melgani [11], with good results. In addition, local SVMs and their properties have been theoretically studied by, for example, Hable [12] and Meister and Steinwart [13]. Actually, the simplest and most effective locality technique is represented precisely by the k-NN, which picks out the elements closest to the target one according to a given metric. Different quantum variants of the k-NN algorithm have been proposed; moreover, an interesting application of a quantum k-NN version in combination with the Grover search algorithm [14] has also been presented by Sawerwain and Wróblewski [15], namely, a quantum recommendation system.

Eventually, the empirical evaluation has always been an essential part of the machine learning research paradigm, with the UCI Machine Learning Repository [16] playing a major role in setting de facto standard benchmarks for the experimental assessment of machine learning (ML) algorithms. Recently, massive benchmark datasets have been proposed and used for studying and advancing the state of the art of deep learning neural networks [17]. Instead, the area of quantum machine learning has not yet matured enough to produce established benchmarks; this is due to the mainly theoretical nature of the research in the current phase, as well as the limited dimension and reliability of the available machines. Nevertheless, a fair and systematic comparison between quantum and classical machine learning is necessary in order to progress. To this end, a platform that supports the execution of both kinds of algorithms, and provides access to the existing quantum devices and/or their simulators, is highly desirable.

In this work, we propose and assess the application of a quantum locality technique as a preliminary step of QML models, with the purpose of reducing their size and improving their performance. In particular, we provide the implementation and the empirical evaluation of a quantum pipeline consisting of a quantum k-nearest neighbors algorithm [18] and a quantum cosine-based binary classifier [19]. The code, which is publicly accessible at https://github.com/ZarHenry96/quantum-ml-pipeline, integrates access to available quantum computing resources, namely, IBM quantum computers, provides for testing on several UCI datasets whose dimension is compatible with the current quantum devices, and allows for comparisons with classical competitors. However, the quantum pipeline has not been tested on real quantum devices due to the retirement of the quantum computer of interest (the only one available with a free account that had enough qubits). Eventually, it is worth highlighting that the approach presented in this work is not limited to binary classification. In order to address a multiclass scenario as done in [20], it is sufficient to replace the quantum binary classifier with a quantum classification model able to manage multiple classes.

The remainder of the paper is structured as follows: Section 2 presents some background information, Section 3 describes the quantum pipeline, its implementation and complexity, Section 4 deals with the experimental evaluation and the results obtained, and Section 5 provides the conclusions.

2 Background

This section provides background information about quantum computing, quantum machine learning, the classical and the quantum versions of the k-nearest neighbors algorithm, and the quantum binary classifier proposed by Pastorello and Blanzieri [19].

2.1 Quantum machine learning

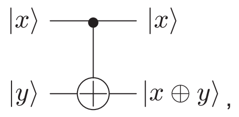
Quantum computing is a type of computation in which quantum phenomena, such as state superposition and entanglement, are exploited to perform calculations. It is the most prominent application of quantum information theory and delivers algorithms to solve efficiently problems that are hard for classical computers [21].

Quantum machine learning is an emerging research area related to quantum computing. Basically, in QML, quantum computing techniques are applied to machine learning tasks in order to pursue computational advantages given the present context of ever-growing amounts of data to manage. In detail, QML algorithms may present relevant benefits in time and space complexity with respect to classical ML algorithms. Remarkable examples are the embedding of quantum subroutines into ML schemes to efficiently calculate distances in the feature space [22], with advantages in classification and clustering, or Grover-based subroutines to find an item in an unsorted database [23], with a quadratic speedup with respect to an exhaustive search. The latter are employed, for instance, in pattern recognition. The first proposals of quantum versions of ML algorithms were presented about twenty years ago [24, 25], but the real interest in QML has sparked only in the last decade thanks to the development of the first available working prototypes of quantum machines like those manufactured by IBM [26], Rigetti [27], and D-Wave Systems [28], and the publication of many interesting results on quantum machine learning algorithms [4, 5, 29, 30].

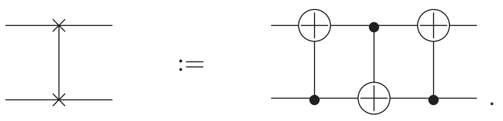
Within the quantum circuit model, one of the most popular, the basic concept of quantum computation is represented by the qubit, whose state is described by a unit vector $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ in a two-dimensional complex Hilbert space, of which $|0\rangle$ and $|1\rangle$ form an orthonormal basis. In particular, $|\rangle$ is a *ket* in the Dirac notation, which is used to denote quantum states, and $|0\rangle$ and $|1\rangle$ identify the vectors of the standard basis of $\mathbb{C}^2$. Instead, the absolute squares of the amplitudes $\alpha, \beta \in \mathbb{C}$ correspond to the probabilities of measuring the qubit in states 0 and 1, respectively; hence, $|\alpha|^2 + |\beta|^2 = 1$. After a measurement process, the state of a qubit collapses to the post-measurement state, either $|0\rangle$ or $|1\rangle$, according to the obtained outcome. In addition, the time evolution of isolated quantum systems (such as the qubits) is mathematically described by unitary operators, which are called quantum gates in the language of quantum computing. An example of a quantum gate acting on a single qubit is the Hadamard gate, whose action on the basis states is given by $H|0\rangle = |+\rangle$ and $H|1\rangle = |-\rangle$, where $|\pm\rangle = 1/\sqrt{2} \cdot (|0\rangle \pm |1\rangle)$. Basically, it creates a superposition state; the corresponding matrix representation and the circuital symbol are

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \qquad \text{and} \qquad -\boxed{H}- \quad .$$

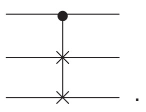
Another important quantum gate is the controlled NOT gate (or CNOT), which operates over two qubits and acts as follows with respect to the computational basis:



where $x, y \in \{0, 1\}$ and $\oplus$ is the sum modulo 2. In practice, it flips the state of the target qubit if the control qubit is in the state $|1\rangle$. In particular, by combining three CNOTs, it is possible to build the SWAP gate, a 2-qubit gate that swaps the qubits provided as input; its circuital definition is

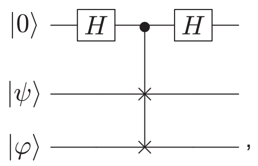


Instead, the controlled version of the SWAP gate (a 3-qubit gate) is called Fredkin gate as its classical version, which is universal for classical reversible computation, and the corresponding circuital symbol is



As an axiom of quantum mechanics, a system of n qubits is described in the space $(\mathbb{C}^2)^{\otimes n}$. As a consequence, the dimension of the space where we can represent the data grows exponentially in the number of qubits; this is one of the main advantages of quantum computations. In practice, quantum algorithms are developed by composing the available quantum gates in order to produce a quantum state that encodes the solution of a given problem, and the read-out is performed by measuring the quantum output state in the computational basis. Since the result of a quantum computation is probabilistic in general, a quantum algorithm must be repeated several times in order to provide a meaningful result. One of the main examples of the efficiency of quantum computation is the celebrated Shor's algorithm [31], which solves the integer factoring problem (that is generally suspected to be not in **P**) in polynomial time.

In QML, quantum algorithms are developed to solve machine learning tasks like classification, clustering, and pattern recognition [32]. To this end, a relevant question is the representation of classical data into quantum states. The standard encoding used in quantum computing is the so-called basis encoding: binary strings $(x_1, \ldots, x_n)$, with $x_i \in \{0, 1\}$ and $i = 1, \ldots, n$, are translated into states of n qubits $|x_1, \ldots, x_n\rangle$ belonging to the basis of the n-qubit Hilbert space. Alternatively, many QML algorithms exploit the amplitude encoding, in which a classical data instance $\mathbf{x} \in \mathbb{R}^d$ is encoded into the quantum superposition state $|\mathbf{x}\rangle = \|\mathbf{x}\|^{-1} \sum_{i=1}^{d} x_i|i\rangle$ of $\log_2 d$ qubits. Eventually, it is also worth presenting a simple example of quantum processing that is rather useful in QML (and is applied within the k-NN and the binary classifier considered in this work), i.e., the SWAP test [33]. The corresponding circuit is the following:



where $|\psi\rangle$ and $|\varphi\rangle$ are n-qubit states. In detail, the SWAP gate, which acts on $|\psi\rangle$ and $|\varphi\rangle$, is controlled by the qubit initially prepared in $|0\rangle$; this can be implemented through n Fredkin gates. A simple calculation shows that the probability of measuring the value 0 in the first qubit is $\mathbb{P}(0) = 1/2 \cdot (1 + |\psi|\varphi\rangle|^2)$, where $\psi|\varphi\rangle$ is the inner product between $|\psi\rangle$ and $|\varphi\rangle$ in the Dirac notation. In addition, the estimation of $\mathbb{P}(0)$ up to an error ϵ requires $O(\epsilon^{-2})$ repetitions as given by the binomial proportion confidence interval for a Bernoulli trial. In practice, the SWAP test allows the efficient computation of the fidelity of the quantum states $|\psi\rangle$ and $|\varphi\rangle$, with the fidelity being defined for two pure quantum states as

$$\mathcal{F}(|\psi\rangle, |\varphi\rangle) = |\psi|\varphi\rangle|^2 = (\cos(\psi, \varphi) \ \cdot \|\psi\| \cdot \|\varphi\|)^2 = \cos^2(\psi, \varphi) \ , \tag{1}$$

where $\cos(\psi, \varphi)$ is the cosine similarity of $|\psi\rangle$ and $|\varphi\rangle$, and the norms of $|\psi\rangle$ and $|\varphi\rangle$ are 1 by definition. Therefore, by encoding data vectors into the amplitudes of $|\psi\rangle$ and $|\varphi\rangle$, it is possible to compute their dot product and, thus, their cosine similarity (and distance) through the SWAP test.

2.2 K-NN and quantum k-NN(s)

The k-nearest neighbors algorithm [34] is a really simple classification algorithm, and consists of three steps:

- the computation of the chosen distance metric between the test element and all training data points;
- the extraction of the k elements closest to the test instance, namely, the k nearest neighbors;
- the assignment of the class label through a majority voting based on the labels of the k nearest neighbors (in the case of a classification task).

Instead, the quantum counterpart of the algorithm, of which several variants have been proposed, includes an additional step at the beginning, i.e., the preparation of data in a superposition state. This stage allows performing parallel operations, such as computing the distance of the test instance with respect to all training elements simultaneously (quantum parallelism).

Regarding the quantum *k*-NN variants, a conceptually simple one (but not so efficient) is described in the work of Fastovets et al. [35] and consists of two steps: the SWAP test algorithm is exploited to compute the distance (the Euclidean distance, in this case) between feature vectors, which are encoded in the amplitudes of a superposition state; a quantum minimization algorithm based on the Grover search [36], also known as Dürr's algorithm, is used to find the *k* nearest neighbors. In particular, each of the two steps requires multiple iterations with final measurements.

A more complex variant has been presented by both Dang et al. [37] and Y. Wang et al. [38], and applied to the image classification task. The workflow is the following: the features are encoded as amplitudes of a quantum superposition state; the distance between test and training instances is computed through a SWAP test, but without measurements; the amplitude estimation (AE) algorithm [39] is then used to transfer the distance values encoded in amplitudes to qubit states (measurements are not strictly necessary in this step [23]); finally, Dürr's algorithm is exploited to find the indices of the *k* elements with minimum distance with respect to the test instance. It is worth highlighting that both AE and Dürr's algorithms are quite complex and include an oracle, i.e., they are based on a black-box function. Moreover, a very similar workflow is present in the work of Wiebe et al. [23], although it is used for finding only the nearest neighbor.

Quantum *k*-NN versions employing a different metric, namely, the Hamming distance, have been proposed by Ruan et al. [40] and J. Li et al. [41]. Due to the metric chosen, the features must be expressed as bit strings; indeed, the Hamming distance represents the number of positions at which two strings differ. The advantage is a straightforward mapping to quantum states (basis encoding). In detail, the two considered *k*-NN variants share the initial steps: a superposition of the features quantum states is prepared; the difference between corresponding qubits, in training and test features, is computed through CNOT gates; the Hamming distance, which corresponds to the sum of the differences, is obtained by using the *incrementation* circuit presented by Kaye [42]. Then, in the first variant [40], the training instances with a distance lower than a given threshold value are selected by means of an OR gate and a projection operation (actually, to do this, the qubits differences are reversed before the summation). Thus, there is no *k* parameter and the number of nearest neighbors selected depends on the threshold value. Instead, in the second variant [41], a novel quantum search procedure inspired by a binary search is applied to the distances in order to find the minimum value. By iterating it and removing each time the current minimum, the *k* nearest neighbors are selected. The Hamming distance (computed with the procedure just described) is used also in the work of Zhou et al. [43] for image classification, but the search for the *k* minimum distance values is performed through Dürr's algorithm.

Another method for computing the Hamming distance is exploited in the quantum *k*-NN variants presented by Schuld et al. [44] and Wiśniewska and Sawerwain [45]. In detail, instead of summing up the qubits differences through the incrementation circuit, a unitary operation is applied to them in order to encode the values of the sums as quantum state amplitudes. This idea has been proposed first by Trugenberger [24]. After an ancillary qubit measurement, which is required to select the good amplitudes distribution (higher probabilities for lower distances), the classification is performed directly in both works without explicitly selecting the *k* nearest neighbors. However, it is possible to identify the neighbors by repeating the entire process multiple times and measuring the post-ancillary-measurement state (instead of executing the classification step).

The last interesting quantum k-NN variant has been presented by both Afham et al. [18] and Ma et al. [46]. In particular, after the encoding of the data features as amplitudes of a superposition state, a SWAP test is performed. Then, the state of an ancillary qubit and of a qubit register (array), which indexes the training data, is measured. By iterating the procedure just described, it is possible to estimate a quantity proportional to the fidelity [47] between the training data and the test instance states and, therefore, find the k nearest neighbors. Indeed, the fidelity corresponds to the squared scalar product for pure quantum states (see Eq 1). It is worth highlighting that, as shown by Ma et al. [46], multiple test instances can be processed in parallel by introducing an additional index register for the test data and putting the test instances in superposition (as the training ones). Actually, Afham et al. have recently proposed also another quantum k-NN variant [48]: it exploits a generalization of Dürr's algorithm to find the indices of the k nearest neighbors given a quite complex oracle as input. However, the resulting workflow is not so different from that of other previously described works. Basically, the oracle in question includes the SWAP test, a quantum analog-to-digital conversion algorithm [49] based on the phase estimation algorithm [50], and some quantum arithmetic.

2.2.1 A quantum k-NN variant in detail. Let us describe more in detail the quantum k-NN algorithm proposed by Afham et al. [18]. In order to do so, let us consider the dataset $\{\mathbf{x}_i\}_{i=0,\ldots,N-1}$, with $\mathbf{x}_i \in \mathbb{R}^d$, the test data instance $\mathbf{x} \in \mathbb{R}^d$, and the fidelity, i.e., the squared cosine similarity (see Eq 1), as a distance measure. Within the amplitude encoding, the cosine similarity between $\mathbf{x}_i$ and $\mathbf{x}$ is nothing but the inner product $\langle \mathbf{x}_i | \mathbf{x} \rangle$ between the corresponding quantum states. In addition, let us assume that N and d are powers of 2 without loss of generality. Then, let us consider an index register of $\log_2 N$ qubits, where the indexes of the training instances are stored within the basis encoding, two n-qubit registers (with $n = \log_2 d$), where data are encoded into the amplitudes of the quantum states, and an ancillary qubit. The four registers are initialized in the state

$$\frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle |\mathbf{x}_i\rangle \, |\mathbf{x}\rangle \, |0\rangle \in \mathsf{H}_{index} \otimes \mathsf{H}_n \otimes \mathsf{H}_n \otimes \mathsf{H}_a. \tag{2}$$

In the state (2), the superposition of the training data and the test instance are stored in two different registers. Now, let us perform the SWAP test on the two n-qubit registers controlled by the ancillary qubit, obtaining the state

$$|\psi\rangle = \frac{1}{2\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle [(|\mathbf{x}_i\rangle |\mathbf{x}\rangle + |\mathbf{x}\rangle |\mathbf{x}_i\rangle)|0\rangle + (|\mathbf{x}_i\rangle |\mathbf{x}\rangle - |\mathbf{x}\rangle |\mathbf{x}_i\rangle)|1\rangle].$$

The probability of getting the outcome $\alpha \in \{0, 1\}$ by a measurement process on the ancillary qubit is given by

$$\mathbb{P}(\alpha) = \frac{1}{2} + (-1)^{\alpha} \frac{1}{2N} \sum_{i=0}^{N-1} |\langle \mathbf{x}_i | \mathbf{x} \rangle|^2,$$

and the corresponding post-measurement state stored in the four registers is

$$|\Psi_{\alpha}\rangle = \frac{\sum_{i=0}^{N-1} |i\rangle (|\mathbf{x}_i\rangle |\mathbf{x}\rangle + (-1)^{\alpha} |\mathbf{x}\rangle |\mathbf{x}_i\rangle)}{\sqrt{2(N + (-1)^{\alpha} \sum_{i=0}^{N-1} |\langle \mathbf{x}_i | \mathbf{x} \rangle|^2)}} \, |\alpha\rangle.$$

After measuring the state of the ancillary qubit (α), the probability of obtaining the outcome i

by performing a subsequent measurement on the index register is given by

$$\mathbb{P}(i|\alpha) = \frac{1 + (-1)^{\alpha} |\langle \mathbf{x}|\mathbf{x}_i\rangle|^2}{N + (-1)^{\alpha} \sum_{i=0}^{N-1} |\langle \mathbf{x}|\mathbf{x}_i\rangle|^2}.$$

As a consequence,

$$\mathbb{Q}(i) := \mathbb{P}(i|0) - \mathbb{P}(i|1) = \frac{2(|\langle \mathbf{x}|\mathbf{x}_i\rangle|^2 - C)}{N(1 - C^2)}, \tag{3}$$

with $C = \frac{1}{N}\sum_i |\langle \mathbf{x}|\mathbf{x}_i\rangle|^2$ being a constant value. In practice, (3) is proportional to the squared cosine similarity $|\langle \mathbf{x}_i|\mathbf{x}\rangle|^2$ between $\mathbf{x}_i$ and $\mathbf{x}$. Therefore, by sampling from the index register, it is possible to identify the indexes with the highest $\mathbb{Q}$ values, i.e., those corresponding to the closest vectors to $\mathbf{x}$. Actually, since $\mathbb{Q}$ is proportional to the square of the cosine similarity, the values of each data feature must be concordant in sign in order to extract only the data instances most similar to $\mathbf{x}$ (and not also the most dissimilar ones).

2.3 Quantum binary classifier

Pastorello and Blanzieri [19] have recently presented a quantum binary classifier based on the cosine similarity metric. Its structure is simple: it iterates the preparation of a superposition state with training and test features encoded as amplitudes, a SWAP test involving states of one qubit, and a final measurement process. Specifically, the measurement outcomes allow estimating a probability value that is directly related to a weighted label assignment with the weights given by the cosine similarity.

More in detail, let $X = \{\mathbf{x}_i, y_i\}_{i=0,\ldots,N-1}$, with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}\ \forall i \in \{0, \ldots, N-1\}$, be a training set of N data instances represented in a real feature space of dimension d and characterised by two-valued labels, and let $\mathbf{x} \in \mathbb{R}^d$ be a new (test) data instance to be classified as either −1 or 1. Let us take into account the following (classical) classification model:

$$y(\mathbf{x}) := \operatorname{sgn}\left(\sum_{i=0}^{N-1} y_i \cos(\mathbf{x}_i, \mathbf{x})\right), \tag{4}$$

where $\cos(\mathbf{x}_i, \mathbf{x}) := \frac{\mathbf{x}_i \cdot \mathbf{x}}{\|\mathbf{x}_i\|\|\mathbf{x}\|}$ is the cosine similarity between the training vector $\mathbf{x}_i$ and $\mathbf{x}$. In this model (4), any training vector contributes to the prediction of the new label, and such a contribution is weighted by the cosine similarity with respect to the new instance. Now, let us consider a $\log_2 N$-qubit register to encode the indexes of the training data vectors, a n-qubit register (with $n = \log_2 d$) to store the data instances within the amplitude encoding, and a single qubit to encode the values of the labels according to $b_i = \frac{1-y_i}{2} \in \{0, 1\}$. Then, let us construct the state

$$|x\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle|\mathbf{x}_i\rangle|b_i\rangle \in \mathsf{H}_{index} \otimes \mathsf{H}_n \otimes \mathsf{H}_l, \tag{5}$$

with H_l being the Hilbert space of the label qubit. The state in question (5) encodes the training set X as a quantum superposition of its elements and the respective labels; note that one qubit is sufficient for the encoding of all the labels. In addition, in the same registers, let us construct the state

$$|\psi_{\mathbf{x}}\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle|\mathbf{x}\rangle|-\rangle \in \mathsf{H}_{index} \otimes \mathsf{H}_n \otimes \mathsf{H}_l, \tag{6}$$

where the label qubit is in the state $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$; in this way, the test data vector $\mathbf{x}$ is represented in a quantum superposition of the two possible classes. To allow the coexistence of states (5) and (6) in the same registers, let us consider an ancillary qubit (a), and let us prepare the superposition state

$$\frac{1}{\sqrt{2}}(|x\rangle|0\rangle + |\psi_{\mathbf{x}}\rangle|1\rangle) \in \mathsf{H}_{index} \otimes \mathsf{H}_n \otimes \mathsf{H}_l \otimes \mathsf{H}_a. \tag{7}$$

The state (7) is entangled. Indeed, there is a quantum correlation between the state of the ancillary qubit a and the content of the registers. After the preparation of the initial state, let us perform a SWAP test between a second ancillary qubit (b), initialized in $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, and the qubit a. Let c be the control qubit (initialized in $|0\rangle$) in the SWAP test; a straightforward calculation shows that the probability of obtaining the outcome 1 by measuring the qubit c is

$$\mathbb{P}(1) = \frac{1}{4}(1 - \langle X|\psi_{\mathbf{x}}\rangle),$$

which is directly related to the considered classification model (4), since

$$\langle X|\psi_{\mathbf{x}}\rangle = \frac{1}{N\sqrt{2}}\sum_{i=0}^{N-1} y_i \cos(\mathbf{x}_i, \mathbf{x}).$$

Therefore, given the probability $\mathbb{P}(1)$ or an estimate thereof, it is possible to predict the label of $\mathbf{x}$ (according to model 4) by means of

$$y(\mathbf{x}) = \mathrm{sgn}[1 - 4\,\mathbb{P}(1)]. \tag{8}$$

3 Quantum pipeline

This section presents the quantum pipeline that has been implemented and tested, providing information about the components, the implementation, and the complexity. The code is available at https://github.com/ZarHenry96/quantum-ml-pipeline.

3.1 Components

The quantum pipeline evaluated in this work consists of a quantum k-NN followed by a quantum binary classification model, with the quantum k-NN providing the nearest neighbors as input to the subsequent model. The workflow is displayed in Fig 1.

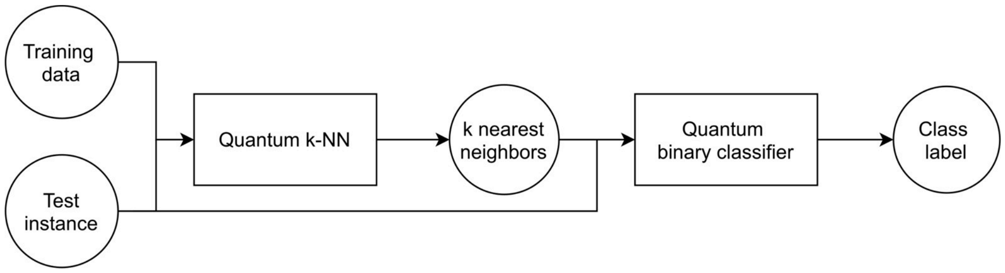


Fig 1. Quantum pipeline workflow overview.

https://doi.org/10.1371/journal.pone.0287869.g001

Concerning the quantum k-NN, several variants exist in the literature, as depicted in Section 2.2. However, some aspects must be considered. First, the variants based on the Hamming distance can not be applied directly to the most common problems, namely, the ones characterised by real-valued features, since the considered metric represents a distance on binary strings. Second, most variants include an oracle-based algorithm originating from Grover's, such as Dürr's or the amplitude estimation. As a consequence, they require a problem-dependent black-box function in order to be used; this negatively affects their ease of use and implementation. In addition, the usage of the amplitude estimation algorithm to transfer distance information to qubits states has, in turn, representation issues in the case of real-valued features: an approximation of the estimated distances is obliged. All these factors make the variant proposed by Afham et al. [18], described in detail in Section 2.2.1, the best candidate for experiments on real-valued datasets. Specifically, the parallel processing of multiple test instances suggested by Ma et al. [46] has not been taken into account here.

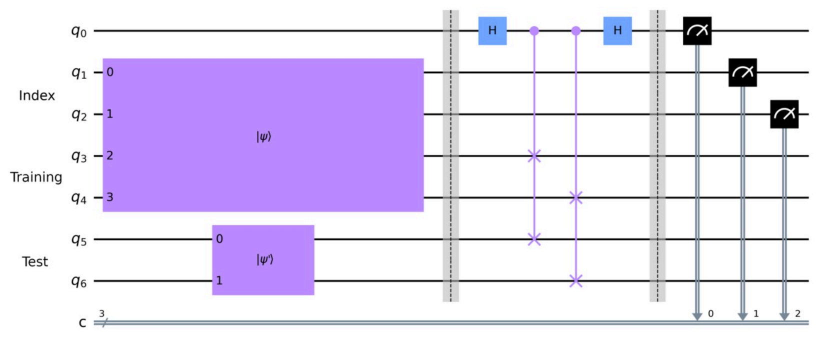
The quantum binary classification model selected for being combined with the quantum k-NN is the classifier described in Section 2.3 [19], whose structure is quite simple and very similar to that of the chosen quantum k-NN. Actually, the possibility of building a pipeline using these two quantum models was briefly anticipated in the pre-print version of the same article [51].

3.2 Implementation

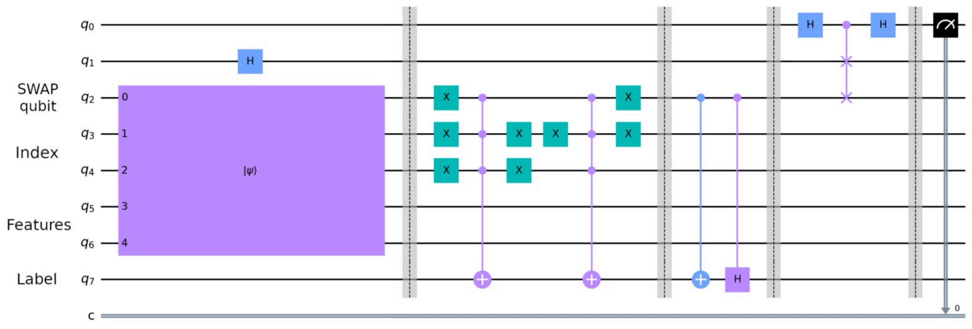
The quantum pipeline has been implemented using Qiskit, i.e., the open-source SDK provided by IBM [52]. Qiskit allows building quantum circuits using the Python programming language, and the circuit execution can be performed either on simulators or real quantum devices. Several simulation backends are provided by IBM, and it is possible not only to get measurement counts as in real quantum devices but also to retrieve the state vector of the circuit at any point of the execution, namely, the amplitude of each state.

In practice, exploiting Qiskit, code has been developed that automatically initializes and builds circuits for the quantum algorithms involved (quantum k-NN, quantum binary classifier) based on the dataset provided as input (i.e., the number of samples, the number of features, and features values). Moreover, the following execution modalities have been implemented for both algorithms: *classical*, which does not build any quantum circuit but runs the corresponding classical algorithm; *statevector*, which processes the final state vector of the circuit to provide the output, thus representing an ideal execution with an infinite number of runs; *simulation* (named *local simulation* in the code), which provides counts by sampling from the final probability distribution; *online simulation*, which is the same as *simulation* but on hardware provided by IBM; and *quantum*, which exploits real IBM quantum devices. Concerning the classical modality, the distance metric used for the k-NN is the cosine distance; the reason lies in the fact that the chosen quantum k-NN selects the k nearest neighbors based on the fidelity, which corresponds to the squared cosine similarity in the case of pure quantum states (see Eq 1). Instead, for the binary classifier, Eq (4) is used. It is also worth highlighting that no noise has been taken into account in the execution of gates in any simulated modality (including the *statevector*), and that the execution modality does not need to be the same for the pipeline components. Eventually, the possibility of retrieving the results of online executions at a later moment has been implemented due to the presence of long waiting times for the quantum devices.

The pseudocode of the quantum pipeline is shown in Algorithm 1. Actually, the pseudocode in question is valid for all execution modalities but the *classical*, which does not require circuits. If the *classical* modality is selected for one component, the corresponding block is



(A) Quantum k-NN.



(B) Quantum binary classifier.

Fig 2. Quantum pipeline circuits example. The first circuit (A) corresponds to the quantum k-NN, the second one (B) to the quantum binary classifier. In the case of the *statevector* modality, the final measurements are not present.

https://doi.org/10.1371/journal.pone.0287869.g002

replaced with the execution of the classical version of the algorithm. In any case, the first step is the unit-norm normalization of the training and test data (Line 1). This operation is essential for the amplitude encoding of data and consists in dividing the features of each instance by the instance norm. If a data instance has the norm equal to zero, all its attributes are replaced with an epsilon value (0.000001) to allow the normalization. The procedure in question is executed also in the case of a quantum pipeline with only classical components. Moreover, in the experiments, an additional normalization procedure has been applied to the data before the unit-norm one as part of the experimental setup common to all methods tested; the details are provided in Section 4.3. It is also worth mentioning that the values of each data feature must be concordant in sign, for the quantum k-NN to work properly (in the experiments, this has been ensured by the additional normalization procedure just mentioned); furthermore, the quantum binary classifier requires the training data class labels to be in {−1, 1}.

Algorithm 1: Quantum pipeline.

```
 Input: training data D, test instance x, number of nearest neighbors
k, execution modalities (not classical) for the two components
exec_mods
 Result: class label label ∈ {-1, 1}
1 D, x ← normalization(D, x);
  /* Quantum k-NN */
2 circ_qknn ← buildQKNNCircuit(D, x, exec_mods[0]);     // See Fig 2A
3 res_qknn ← execute(circ_qknn, exec_mods[0]);
```

```
4 k_nn ← getKNearestNeighbors(D, k, $res_{qknn}$, exec_mods[0]);
  /* Quantum binary classifier */
5 $circ_{qbc}$ ← buildQBCCircuit(k_nn, x, exec_mods[1]);     // See Fig 2B
6 $res_{qbc}$ ← execute($circ_{qbc}$, exec_mods[1]);
7 label ← getLabel($res_{qbc}$, exec_mods[1]);
8 return label;
```

Focusing on the non-classical modalities, the step following the data normalization is the construction of the quantum *k*-NN circuit based on the normalized data (Line 2). A sample circuit is shown in Fig 2A; actually, the vertical barriers in the figure have been added for illustrative purposes, but, in the implementation, one barrier is placed between the ancillary qubit measurement and the index register measurement to ensure sampling from the desired probability distributions. In detail, a quantum *k*-NN circuit consists of three main slices: registers initialization, SWAP test without measurement, and final measurements. In the first slice, the training data index is set up, simultaneously to the encoding of the training and the test features. Specifically, the index (q_1-q_2) and the training features (q_3-q_4) registers are jointly initialized for simplicity; indeed, they are entangled according to the quantum *k*-NN algorithm. After that, the SWAP test without measurement is performed by means of two Hadamard gates and the controlled SWAP gates, whose number increases linearly with the number of feature qubits. In particular, the SWAP gates act on the training (q_3-q_4) and the test features (q_5-q_6) registers. Eventually, the state of the first ancillary qubit (the SWAP test measurement qubit q_0) and of the index register is measured; obviously, in the case of the *statevector* modality, the measurements are not present. Regarding the number of qubits required ($qubits_{qknn}$), which is dataset dependent, it is given by

$$qubits_{qknn} = 1 + qubits_{qknn\_index} + 2 * qubits_{features}\,, \tag{9}$$

where the value 1 corresponds to the SWAP test measurement qubit (q_0), $qubits_{qknn\_index}$ represents the number of index qubits (q_3-q_4), and $qubits_{features}$ is the number of feature qubits (either q_3-q_4 or q_5-q_6).

Once the quantum *k*-NN circuit has been built, it is executed according to the specified modality (Line 3). Actually, if the circuit includes measurements, it is executed multiple times to provide the desired number of counts (*simulation_shots* parameter described in Section 4.3). Finally, the execution output (res_{qknn}), which corresponds to either a state vector or state counts (the number of times each state has been observed), is processed in order to extract the *k* nearest neighbors (Line 4). In detail, the amplitudes/counts are exploited to estimate the quantity $P(i|0) - P(i|1)$, with i being a training data index and the 0/1 value being the state of q_0 (see Eq 3). Then, the training data are sorted according to this quantity, which is proportional to the similarity with respect to the test instance, allowing the identification of the nearest neighbors.

The next step is the construction of the quantum binary classifier circuit based on the selected *k* nearest neighbors and the normalized test instance (Line 5). An example circuit is displayed in Fig 2B; here, the vertical barriers have been added only for illustrative purposes. In particular, it is possible to distinguish five main slices in the circuit: registers initialization, training data labels configuration, test label set up, SWAP test without measurement, and final measurement. Concerning the initialization, the first qubit subjected to the swap (q_1) is set to the uniform superposition of 0 and 1 ($|+\rangle$). Instead, the second one (q_2) is entangled with the register (q_3-q_4) representing the training data index and the register (q_5-q_6) in which both the training and the test features are encoded in superposition. Hence, it is jointly initialized with all them (from q_3 to q_6) for simplicity. Regarding the qubit representing the label (q_7), it takes only well defined values (0, 1, and the uniform superposition of them), thus it is configured separately (second and third slices of the circuit) to limit the complexity of the joint initialization operation. More in detail, the training data labels (second slice) are encoded in the last qubit of the circuit by selecting the desired index register states through NOT gates (denoted as *X*s in the image) and setting the corresponding label states through multi-controlled NOT gates (this procedure is required only for one label value, the one associated with the 1 state, namely, −1). The NOT gate applied to the last SWAP qubit (q_2) at the beginning and at the end of this step is necessary in order to work on the states associated with the training data. Instead, for the test instance (third slice of the circuit), the label qubit is set to the $|-\rangle$ state according to the algorithm by means of a controlled NOT and a controlled Hadamard gates. Then, the SWAP test without measurement is performed (fourth slice), and the state of the first qubit (q_0) is measured (last slice). Also in this case, there is no final measurement for the *statevector* modality. The number of qubits required by the quantum binary classifier for a generic dataset ($qubits_{qbc}$) is given by

$$qubits_{qbc} = 3 + qubits_{qbc\_index} + qubits_{features} + 1\,. \tag{10}$$

In detail, the value 3 corresponds to the qubits needed by the SWAP test (q_0-q_1-q_2), $qubits_{qbc\_index}$ represents the number of index qubits (q_3-q_4), $qubits_{features}$ is the number of feature qubits (q_5-q_6), and the 1 represents the qubit used for the label encoding (q_7).

After the construction of the quantum binary classifier circuit, there is the execution step (Line 6); the considerations about the presence of measurements made for the quantum *k*-NN also hold for the classifier. Eventually, the execution output is processed (Line 7) in order to predict the test instance label. In particular, the amplitudes/counts are used to estimate the probability $P(1)$ of obtaining 1 by measuring the state of the qubit q_0. The probability in question allows predicting the class label (−1 if $P(1) > 0.25$, 1 otherwise, according to Eq 8), which is then returned as the last operation of the pipeline (Line 8).

3.3 Complexity observations

Afham et al. [18] define the gate complexity of their quantum *k*-NN algorithm, which is equal to $O(log_2\ d)$, with *d* being the number of data features. However, the complexity in question is expressed in terms of controlled SWAP (Fredkin) gates, which are not elementary gates. Moreover, the registers initialization is not included because they assume the presence of an initialization quantum oracle. Instead, for their quantum binary classifier, Pastorello and Blanzieri [19] provide the overall time complexity, which is equal to $O(\epsilon^{-2}\ log_2(Nd))$, with ϵ being the desired upper bound to the prediction error, and *N* being the training dataset size. Nevertheless, the authors assume the presence of a QRAM, i.e., a quantum random access memory [53], from which to retrieve the state given as input to the SWAP test (fourth and fifth slices of the circuit shown in Fig 2B).

Concerning the pipeline implementation presented in this work, some observations can be made for execution modalities different from *classical*. First of all, the initial unit-norm normalization step has $O(Nd)$ complexity, since it is necessary to scan all the data features. The construction of the quantum *k*-NN circuit has $O(2^{\lceil log_2 N\rceil + \lceil log_2 d\rceil}) + \lceil log_2 d\rceil)$ complexity, with the first term given by the joint index-training initialization and the second one given by the SWAP test. Instead, the complexity of the circuit execution depends on the execution modality and its implementation inside Qiskit; if it is not a *statevector* execution, the complexity is influenced also by the number of measurements. Regarding the number of gates, it is worth highlighting that the registers initialization is an expensive operation. Indeed, the generation of an arbitrary target state requires to find the correct sequence of elementary gates. The controlled SWAP gates also have a significant impact since, as mentioned previously, their number depends on the number of data features and they are not elementary operations. The last quantum k-NN step is the extraction of the nearest neighbors, which includes the processing of the execution output. In detail, for a *statevector* execution, it is necessary to process the final state vector of the circuit, with a $O(2^{1+\lceil log_2 N\rceil+2\lceil log_2 d\rceil})$ complexity. Instead, for all the other execution modalities, the state counts must be processed, and the corresponding complexity is $O(2^{1+\lceil log_2 N\rceil})$. In any case, after the execution output processing, the k nearest neighbors are extracted by sorting the index values with a $O(Nlog_2 N)$ complexity.

Looking at the second half of the pipeline, namely, the quantum binary classifier, its complexity is related to the number of nearest neighbors k. In particular, the construction of the classifier circuit has $O(2^{1+\lceil log_2 k\rceil+\lceil log_2 d\rceil} + k2\lceil log_2 k\rceil)$ complexity, with the second term being a worst case estimate for the training labels set up. Regarding the circuit execution and the number of gates, the observations made for the quantum k-NN hold also for the classifier. Actually, there is only one controlled SWAP gate in this case. Nevertheless, there could be several (maximum k) multi-controlled CNOT gates, which in turn have a significant impact on the performance. Eventually, there is the execution output processing, which has $O(2^{4+\lceil log_2 k\rceil+\lceil log_2 d\rceil})$ complexity if the execution modality is *statevector*, $O(1)$ complexity otherwise. The final label prediction has constant complexity.

To give an idea of the runtimes, for $N = 168$, $d = 12$, and $k = 9$, the pipeline execution time on the machine used in the experiments (whose specifications are provided at the beginning of Section 4) is in the order of 2-3 seconds. This holds for both the *statevector* and the *simulation* modalities, with both components having the same execution modality. Actually, among the two, the *simulation* modality turns out to be a little more time consuming. In the situation just described, the circuit size is 17 qubits for the quantum k-NN and 12 qubits for the quantum binary classifier.

4 Empirical evaluation

This section deals with the quantum and classical algorithms taken into account, the datasets selection and preparation, the setup of the experiments, and the results obtained. In particular, the experiments have been executed on a shared machine with an Intel Xeon Gold 6238R processor running at 2.20GHz and 125 GB of RAM.

4.1 Methods

The "quantum k-NN" [18]—"quantum binary classifier" [19] pipeline described in Section 3 has been tested under different execution modality combinations, which are reported in Table 1A. In detail, these experiments aimed at understanding the performance of the classical

Table 1. Quantum pipeline modalities (A), quantum binary classifier modalities (B), and baseline methods (C) considered.

(A)	(B)	(C)
Quantum pipeline	**Quantum classifier**	**Baseline method**
classical—classical	statevector	random forest
statevector—classical	simulation	SVM
classical—statevector		k-NN
statevector—statevector		k-NN + classifier
simulation—classical		k-NN + SVM
classical—simulation		
simulation—simulation		

https://doi.org/10.1371/journal.pone.0287869.t001

pipeline, verifying the equivalence between *classical* and *statevector* (the latter represents an ideal execution), and analysing the impact of simulating the pipeline components. Initially, pipelines including quantum executions were planned too. Nevertheless, the quantum device of interest was dismissed before the experiments could have been executed and no equivalent machine (in the number of qubits) has been made available yet (at time of writing, the maximum number of qubits for a free account is five). As a consequence, the reliability of the results presented here is strictly related to that of the simulator provided by Qiskit, which is widely used nowadays, given the size of the available universal quantum devices.

The quantum binary classifier alone has been evaluated using the modalities reported in Table 1B. In particular, the *classical* modality has not been taken into account due to the effective equivalence with respect to *statevector*. The purpose of these experiments was to collect data to confirm a potential improvement in the performance of the model with the introduction of locality in the form of the quantum *k*-NN.

Eventually, the pipeline performance have been compared with some classical baseline methods, which are reported in Table 1C. Most of these algorithms have been taken directly from scikit-learn [54], but others have been composed starting from scikit-learn procedures. In detail, the default parameters provided by scikit-learn have been used for the random forest (e.g., the number of trees has been set to 100), whereas, for the SVM, two different kernels have been tested, i.e., Gaussian and linear. Two distance metrics have also been evaluated for the *k*-NN: the cosine and the Euclidean distance. Finally, two model pipelines have been considered. The *k-NN* + *classifier* represents the classical analogue of the implemented quantum pipeline. Indeed, the classifier in question is the binary classifier based on the cosine similarity defined in Eq (4). Instead, the *k-NN* + *SVM* has been taken into account in order to evaluate the benefits of replacing the binary classifier with a more complex model like the SVM. Actually, these baseline pipelines have been tested using both metrics (cosine and Euclidean) for the *k*-NN and both kernels (Gaussian and linear) for the SVM. It is also worth mentioning that the *k-NN* + *classifier* with cosine distance metric differs from the *classical—classical* execution of the quantum pipeline for the absence of the input data unit-norm normalization.

4.2 Datasets

All the datasets used in the experiments are from the UCI Machine Learning Repository [16]. In detail, they have been selected according to the following criteria: the associated machine learning task is classification (all the considered models are classifiers); the attributes are real-valued, preferably, but some integer features are accepted (actually, the *02_transfusion* dataset has only integer attributes but is marked as real-valued on the UCI site); the number of features is less than or equal to 16; the number of instances is reasonable, namely, less than one thousand.

The reason for the filter on the type of attributes lies in the amplitude encoding exploited by the considered quantum models. In particular, numerical data are required, and integer values are acceptable because of the unit-norm normalization. Regarding the limit on the number of features, it is related to the experiments originally planned on the real quantum device. The number of qubits of that machine was 15. Hence, if the number of qubits required to encode the features ($qubits_{features}$) had exceeded four (more than 16 attributes), the number of embeddable training instances in the quantum *k*-NN circuit would have been less than 17 ($qubits_{qknn\_index} \leq 4$), a too-small quantity (see Eq 9). Eventually, the constraint on the number of instances was aimed at allowing the encoding of the dataset in the circuit without the need for a too drastic subsampling.

Table 2. Datasets properties (the dataset names are links that lead to the corresponding UCI pages). Note: "qb." stands for qubits.

Name	Original size	Original classes #	Features #	Size (15 qb.)	Size (32 qb.)
01_iris_setosa_versicolor	150	3	4	100	-
01_iris_setosa_virginica				100	-
01_iris_versicolor_virginic				100	-
02_transfusion [55]	748	2	4	748	-
03_vertebral_column_2C	310	2	6	310	-
04_seeds_1_2	210	3	7	140	-
05_ecoli_cp_im	336	8	7	220	-
06_glasses_1_2	214	6	9	80	146
07_breast_tissue_adi_fadmasgla	106	6	9	71	-
08_breast_cancer [56]	116	2	9	80	116
09_accent_recognition_uk_us	329	6	12	80	210
10_leaf_11_9 [57]	340	30	14	30	-

https://doi.org/10.1371/journal.pone.0287869.t002

The well-formed datasets matching the just presented criteria were 10; indeed, some have been discarded due to missing fields or unclear structure. However, most of these datasets were characterised by more than two classes, whereas Pastorello and Blanzieri's classifier (and also the classical SVM) works on binary labels. Therefore, only the two most represented classes have been retained, mapping them to $\{-1, 1\}$ and discarding the other instances (in the case of a tie, the first two classes have been chosen). Moreover, if the consequent dataset size was still exceeding the number of instances encodable in the quantum k-NN circuit using 15 qubits, a random subsampling maintaining the ratio between classes has been applied. Concerning this last step, the size reduction due to the split in training and test set, which is accurately described in the next section, has also been taken into account. The resulting datasets and their properties are reported in Table 2.

It is worth highlighting some aspects about the data shown in the table: the dataset name includes a suffix representing the names of the selected classes if they originally were more than two; the Iris dataset (01) represents an exception since all the possible combinations between classes have been taken into account, leading the total number of datasets to 12; the suggested three classes merging has been applied to *07_breast_tissue_adi_fadmasgla*. Regarding the sizes of the datasets used in the experiments, they are reported in the last two columns: the former (*Size, 15 qb.*) contains the sizes resulting after performing the entire process described in the previous paragraph, whereas the latter (*Size, 32 qb.*) shows the sizes without the final subsampling step for the datasets for which was needed. In particular, the three datasets that required the subsampling (i.e., *06_glasses_1_2*, *08_breast_cancer*, and *09_accent_recognition_uk_us*) have been used to analyse the effect of a larger training set, and the value 32 represents the limit number of qubits for an online simulation (far fewer qubits are required in practice).

Another thing that is worth mentioning is the fact that, before reducing its number of classes, the Iris dataset (01) has been modified by correcting the two wrong instances as reported in the Iris UCI page. All the datasets used in the experiments are available together with the code at https://github.com/ZarHenry96/quantum-ml-pipeline.

4.3 Experimental setup

Each method presented in Section 4.1 has been applied to each dataset reported in Table 2 (both 15 and 32 qubits limit), with the exception of the "quantum binary classifier"—"*02_transfusion* dataset" pair, since an additional qubit would have been necessary (see Eq 10).

Table 3. Parameters of the experiments.

Parameter	Value(s)
k_folds	5
k	3, 5, 7, 9
simulation_shots	1024
simulation_runs	5

https://doi.org/10.1371/journal.pone.0287869.t003

The model evaluation technique chosen is the k-fold cross-validation, which works as follows: the dataset is split into k subsets, also called folds; then, $k - 1$ folds form the training set, whereas the remainder becomes the test set; the last step is iterated k times so that each fold is used once as the test set. In particular, the stratified k-fold cross-validation has been exploited. This means that the folds have been generated in such a way that the ratio between classes in the test set remained as close as possible to that of the original dataset. Eventually, it is worth mentioning that, in all experiments, the same seed has been used for the generation of folds. In this way, all methods have processed exactly the same folds.

The parameters of the experiments are reported in Table 3. In detail, *k_folds* represents the number of folds, which has been set to 5, a value commonly used in ML. Instead, k corresponds to the number of nearest neighbors selected, a fundamental parameter for both the quantum pipelines, the classical k-NN, and the baseline pipelines (*k-NN + classifier* and *k-NN + SVM*). Therefore, several odd (small) values in arithmetic progression have been evaluated. Concerning *simulation_shots*, it represents the number of measurements (hence, circuit executions) performed in the *simulation* modality, and its value is the same for both the quantum k-NN and the quantum binary classifier; specifically, 1024 is the default value provided by Qiskit. Finally, *simulation_runs* is the number of runs that have been executed for the experiments including either a quantum model (k-NN/classifier) in the *simulation* modality or the random forest. Indeed, the methods in question are stochastic, and no seed has been set for them. In particular, the value that has been chosen (5) represents a trade-off between the possibility of evaluating the statistical significance of the results and the amount of computational resources required to obtain them.

In each cross-validation step of each experiment, a min-max data normalization procedure has been applied before executing the model/pipeline. In particular, each attribute in the training set has been rescaled to the interval [0, 1] by subtracting the minimum value and dividing by the range. In the case of a zero range (constant attribute), the attribute has been set to zero. As usual, the test instances have been normalized using the training set parameters (minimum and range values). If a test instance attribute exceeded the interval edges after the normalization (since it was larger/smaller than the maximum/minimum in the training set), it has been clipped to the exceeded edge value. In addition, after the feature normalization, the input data to the quantum models (including the quantum classifier alone) have undergone a unit-norm normalization procedure, as described in Section 3.2. Actually, the min-max normalization has also avoided any sign-related issue for the quantum k-NN, which requires the values of each attribute to be concordant in sign.

4.4 Results

The results are presented by means of accuracy-based scatter plots, with the accuracy being defined for a fold as

$$accuracy = \frac{number\ of\ correctly\ classified\ instances\ in\ the\ fold}{total\ number\ of\ instances\ in\ the\ fold}.$$

In the case of multiple runs, the fold average accuracy is reported. Moreover, the statistical significance of the results is verified through the Wilcoxon signed-rank test [58], since the data considered are paired.

In detail, the different execution modalities of both the quantum pipeline and the quantum binary classifier are analysed first. Then, the two quantum models are compared with each other, and the effect of the dataset size on their performance is shown. Finally, the baseline methods are taken into account: an evaluation of the considered distance metrics (cosine/Euclidean) is presented, followed by a comparison of the quantum pipeline and the baseline methods.

4.4.1 Execution modalities comparison (quantum pipeline). Some comparisons between execution modalities for the quantum pipeline on the *15 qubits* datasets are shown in Fig 3. In these plots, each point represents the accuracy obtained in a fold (or its average across runs). In particular, the *statevector—statevector* modality turns out to be equivalent to the *classical—classical* (Fig 3A), as expected. Indeed, it is an ideal execution (with an infinite number of runs and without noise) of the quantum circuits implementing the quantum *k*-NN and the

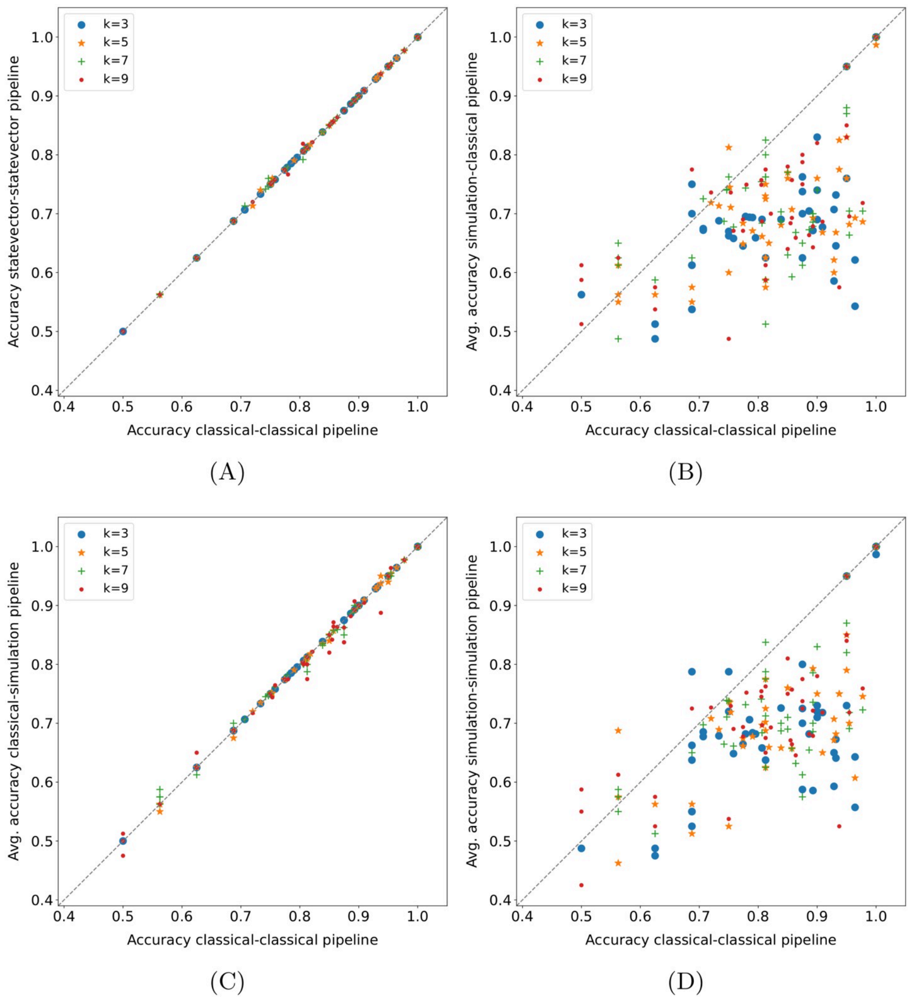


Fig 3. Execution modalities comparison on *15 qubits* datasets for the quantum pipeline. Each point represents the accuracy obtained in a fold (or its average across runs).

https://doi.org/10.1371/journal.pone.0287869.g003

Table 4. Wilcoxon signed-rank test ($\alpha = 0.05$) applied to the fold accuracy distributions shown in Fig 3. The values reported in the table are the p-values obtained.

	k = 3	k = 5	k = 7	k = 9
Fig 3A	1.000	0.276	1.000	0.655
Fig 3B	8.80E-08	1.18E-07	2.46E-06	4.01E-06
Fig 3C	0.414	0.052	0.486	0.092
Fig 3D	1.04E-07	2.03E-07	2.27E-07	4.60E-07

https://doi.org/10.1371/journal.pone.0287869.t004

quantum binary classifier, thus, it should be equivalent in accuracy to its classical counterpart. The few deviations (from the main diagonal) in Fig 3A are due to two aspects: the different policies used by the two modalities to select the nearest neighbors in the case of a distance tie; the presence of instances with identical features and different class labels in the *02_transfusion* dataset. Despite that, the Wilcoxon signed-rank test has confirmed that the difference between the two modalities is not statistically significant (Table 4). It is also worth recalling that the advantage of the considered quantum models/pipelines with respect to their classical counterparts lies in the execution time and not in the accuracy. The scatter plots comparing the *classical—classical* modality with the *statevector—classical* and the *classical—statevector* (not reported) are identical or almost identical (absence of deviant points) to Fig 3A.

Although the quantum pipeline is equivalent (in accuracy) to the classical one in the ideal case, this is not true when simulated (even more so when executed on a quantum device, which is also noisy). By looking at Fig 3B and 3D, it turns out that simulating the quantum *k*-NN with 1024 shots (i.e., measurements) adversely affects the performance of the pipeline, independently from the *k* value. Indeed, almost all points lie below the main diagonal, and the difference is statistically significant for all *k* values, as reported in Table 4. In practice, the implemented quantum *k*-NN is really sensitive to fluctuations in the estimated probabilities, and a higher number of shots is needed to obtain better results. Instead, it seems that simulating only the quantum classifier with 1024 shots (Fig 3C) does not have a significant impact on the pipeline performance (the difference is not statistically significant). Nevertheless, the effective usage of the model following the classical (and, thus, the *statevector*) *k*-NN is very low in the datasets used. Most times, all the elements selected by the classical *k*-NN belong to the same class, and therefore it is not possible to draw definitive conclusions about the subsequent classifier. The average *usage on dataset* of the second model for the cosine distance metric and each *k* value is reported in Table 5A.

4.4.2 Execution modalities comparison (quantum binary classifier alone). The comparison between *statevector* and *simulation* modalities for the quantum binary classifier alone on the *15 qubits* datasets is displayed in Fig 4. The *classical* modality has not been taken into account due to the effective equivalence with respect to *statevector*; in addition, the *02_transfusion* dataset has not been included since an additional qubit would have been required. Also in

Table 5. Average *usage on dataset* of the second model for the pipelines including the classical (or statevector) *k*-NN with cosine distance (A) and Euclidean distance (B). The *usage on dataset* is 1 when the second model is always employed.

(A) Cosine.			(B) Euclidean.		
k	15 qubits	32 qubits	k	15 qubits	32 qubits
3	0.245 ± 0.204	0.375 ± 0.128	3	0.218 ± 0.229	0.402 ± 0.133
5	0.357 ± 0.290	0.590 ± 0.160	5	0.298 ± 0.305	0.617 ± 0.187
7	0.406 ± 0.323	0.690 ± 0.158	7	0.346 ± 0.346	0.686 ± 0.188
9	0.461 ± 0.356	0.758 ± 0.137	9	0.381 ± 0.371	0.755 ± 0.166

https://doi.org/10.1371/journal.pone.0287869.t005

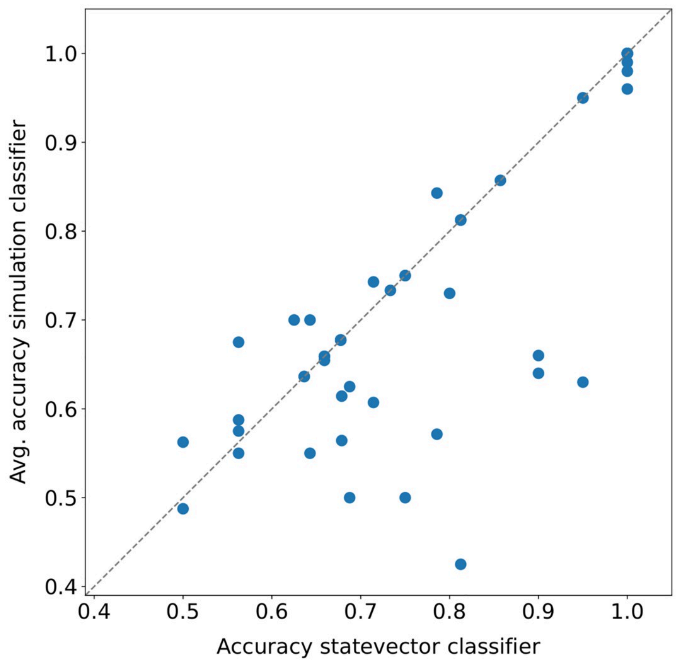


Fig 4. Execution modalities comparison on *15 qubits* datasets for the quantum binary classifier. The *02_transfusion* dataset is not present, and each point represents the accuracy obtained in a fold (or its average across runs). The p-value obtained by applying the Wilcoxon signed-rank test ($\alpha = 0.05$) to the fold accuracy distributions is 0.016.

https://doi.org/10.1371/journal.pone.0287869.g004

this case, each dot represents the accuracy obtained in a fold (or its average across runs). Here, it is possible to observe that the quantum binary classifier as well is affected by the probability fluctuations and, more practically, by the number of shots. Indeed, the *simulation* performance is worse than the *statevector*'s overall, and the difference is confirmed by the Wilcoxon signed-rank test (p-value = 0.016).

4.4.3 Quantum pipeline—Quantum binary classifier comparison. The comparison between the quantum pipeline and the quantum binary classifier on the *15 qubits* datasets is illustrated in Fig 5; the structure of the scatter plots is the same as that of the charts in Fig 3, although here the *02_transfusion* dataset is not present to allow the comparison and the *k* values in the legend refer only to the pipeline. As shown in Fig 5A, the *statevector—statevector* pipeline performs better than the *statevector* classifier for all *k* values. In detail, it statistically outperforms the classifier alone (Table 6). Actually, there are some folds in which the classifier alone achieves a higher accuracy, but they represent a clear minority. This confirms the effectiveness of applying a quantum locality technique such as the quantum *k*-NN as a preliminary step of a quantum classifier, and more in general, the worth of locality. Instead, the pipeline superiority is far less marked when the simulated versions of the pipeline and the quantum binary classifier are considered (Fig 5B). Indeed, it turns out that the quantum pipeline, mainly due to the implemented quantum *k*-NN, is far more negatively affected by probability fluctuations than the classifier alone. Nevertheless, it still manages to perform better overall, independently from the number of nearest neighbors selected (the difference is statistically significant also in this case). Eventually, looking at the different *k* values in these two plots, there is no dominant one; indeed, the best *k* value is typically dataset-dependent. To select the optimal value for a given dataset, it would be necessary to evaluate the performance of several *k* values

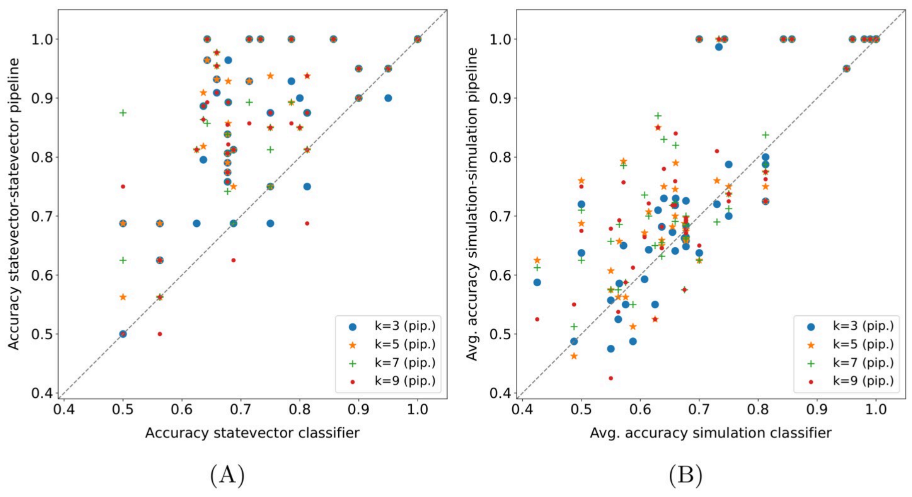


Fig 5. Quantum pipeline—Quantum binary classifier comparison on common *15 qubits* datasets. Each point represents the accuracy obtained in a fold (or its average across runs); the *k* values refer only to the pipeline.

https://doi.org/10.1371/journal.pone.0287869.g005

using part of the dataset as a validation set, since it is not possible to determine it a priori. However, this is beyond the scope of this paper.

4.4.4 Dataset sizes comparison. The effect of the dataset size on the performance of the quantum pipeline and the quantum binary classifier is analysed in Fig 6A. In detail, only the three datasets in Table 2 having both *15 qubits* and *32 qubits* size have been considered for this plot. Moreover, since a fold by fold comparison would not make sense in this case, each point in the chart represents the mean fold accuracy on a dataset (or its average across runs). Finally, the results for all *k* values have been included for the pipelines; as a consequence, the number of pipeline points in the plot is four times higher with respect to the classifier alone. In practice, a larger dataset tends to have a beneficial effect on the performance of the pipeline in the ideal case (*statevector—statevector*) and an overall neutral effect on its simulation: the occurrences of improvement and worsening are the same in the latter. Instead, overall, the performance of the quantum classifier alone worsens in both cases. This represents another point in favor of the quantum pipeline, which turns out to be able to take advantage of a larger number of samples. Actually, the difference is statistically significant in the case of the *statevector—statevector* pipeline, whereas it is not in the others, as shown in Table 7A. However, in this case, the low number of points must also be taken into account.

4.4.5 Distance metrics comparison. As mentioned in Section 4.1, two distance metrics, namely, the cosine and the Euclidean distances, have been evaluated for the baseline methods based on the *k*-NN algorithm, i.e., the *k*-NN, the *k-NN + classifier*, and the *k-NN + SVM* with both Gaussian and linear kernels. The comparison between these two metrics on the *15 qubits* datasets is shown in Fig 6B, with each point representing the accuracy obtained in a fold by one of the four just cited methods. Basically, the Euclidean distance statistically outperforms

Table 6. Wilcoxon signed-rank test ($\alpha = 0.05$) applied to the fold accuracy distributions shown in Fig 5. The values reported in the table are the p-values obtained.

	k = 3	k = 5	k = 7	k = 9
Fig 5A	5.39E-07	7.88E-07	2.55E-06	2.98E-06
Fig 5B	0.042	0.001	3.35E-04	5.84E-04

https://doi.org/10.1371/journal.pone.0287869.t006

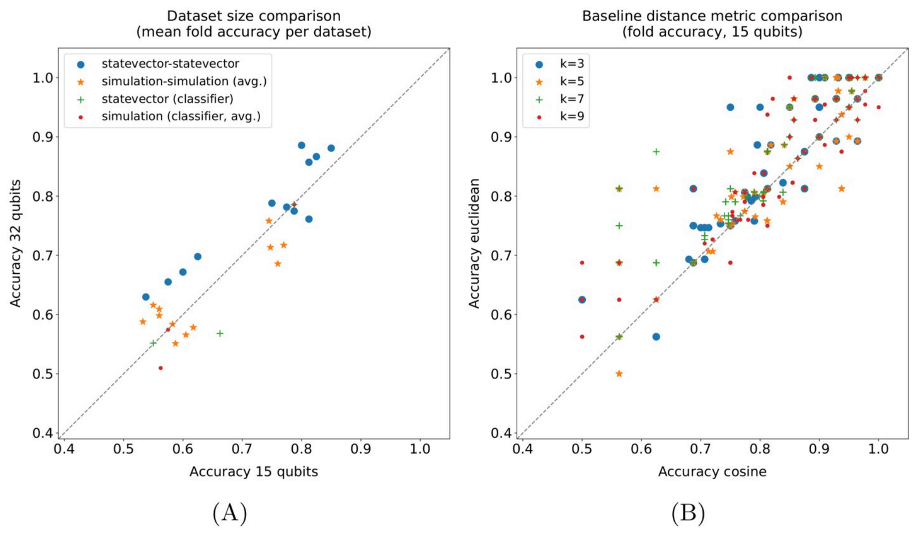


Fig 6. Dataset sizes (A) and distance metrics (B) comparisons. In the dataset sizes comparison (A), each point represents the mean fold accuracy obtained on a dataset (or its average across runs); the pipeline comparisons include all *k* values. In the distance metrics comparison (B), the results obtained by the *k*-NN-based baseline methods (*k-NN*, *k-NN + classifier*, *k-NN + SVM Gaussian*, *k-NN + SVM linear*) on the *15 qubits* datasets are taken into account; each point represents the accuracy obtained in a fold.

https://doi.org/10.1371/journal.pone.0287869.g006

(Table 7B) the cosine one on the datasets used for all *k* values. Therefore, it would be advantageous to have a quantum *k*-NN version based on that metric.

4.4.6 Quantum pipeline—Baseline methods comparison. Some comparisons between the *statevector—statevector* pipeline and baseline methods on the *15 qubits* datasets are displayed in Figs 7 and 8. As usual, each point represents the accuracy obtained in a fold (or its average across runs); moreover, the *k* values in the legends of Fig 7A and 7B refer only to the pipeline. In practice, the random forest achieves better results than the quantum pipeline for all *k*-values (Fig 7A), and the same applies to the best SVM, i.e., the SVM with the Gaussian kernel (Fig 7B). The difference turns out to be statistically significant in both cases, as shown in Table 8, with the exception of $k = 5$ for the SVM—pipeline comparison. Instead, the SVM with the linear kernel (not shown here) performs just slightly better than the pipeline, and the difference is not statistically significant.

Regarding the *k*-NN, the version with cosine distance metric turns out to be equivalent to the *statevector—statevector* pipeline (Fig 8A, the few deviations are the same as the ones in Fig 3A); as shown in Table 9, the difference is not statistically significant. Therefore, it is also equivalent to the *classical—classical* pipeline (see Section 4.4.1). This means that, on the considered datasets, a label assignment based on the *k* nearest neighbors extracted using the cosine

Table 7. Wilcoxon signed-rank test ($\alpha = 0.05$) applied to the mean fold accuracy distributions shown in Fig 6A (A). Same test applied to the fold accuracy distributions shown in Fig 6B (B).

(A) Dataset size.		**(B) Distance metric.**	
	p-value		**p-value**
statev.-statev.	0.016	**k = 3**	9.85E-10
sim.-sim.	0.910	**k = 5**	8.33E-08
statev.	0.750	**k = 7**	7.65E-15
sim.	0.250	**k = 9**	4.86E-08

https://doi.org/10.1371/journal.pone.0287869.t007

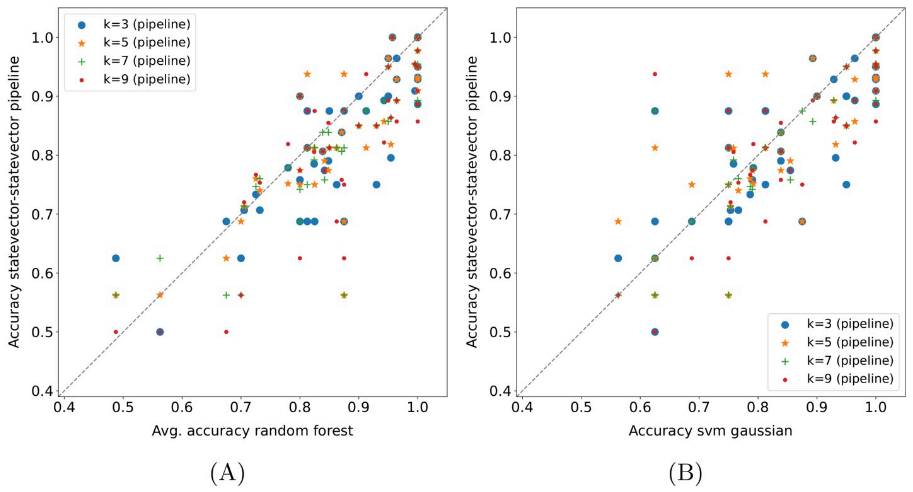


Fig 7. Quantum pipeline—Baseline methods comparison on *15 qubits* datasets. The pipeline modality is *statevector—statevector*, each point represents the accuracy obtained in a fold (or its average across runs), and the *k*-values refer only to the pipeline.

https://doi.org/10.1371/journal.pone.0287869.g007

distance produces the same outcome by applying either a majority voting or the binary classifier based on the cosine similarity. Indeed, the lack of unit-norm normalization of the input data in the baseline methods does not affect the cosine distance (or the cosine similarity), which intrinsically normalizes them. However, it is also worth remarking on the low usage of the models following the nearest neighbors extraction (Table 5A). Instead, the *k*-NN with Euclidean distance statistically outperforms the quantum pipeline for all *k* values, and the resulting scatter plot is identical to Fig 8B (see Table 9 for the statistical test results). In fact, the observation made on the label assignment criteria for the cosine distance also holds for the Euclidean distance. Basically, with the same distance metric, the *k*-NN and the *k-NN* + *classifier* perform equally on the datasets taken into account. As a consequence, the *statevector—statevector* pipeline achieves the same results as the *k-NN* + *classifier* with cosine distance metric (the corresponding scatter plot is identical to Fig 8A) and is outperformed by the same model with Euclidean distance for all *k* values (Fig 8B). Actually, in the end, the *k-NN* + *classifier* with cosine distance and the *classical—classical* pipeline represent exactly the same model due to the observation on the unit-norm normalization just made. Finally, concerning the baseline pipelines including the SVM, the best among them is the *k-NN* + *SVM* with Euclidean distance and Gaussian kernel, which statistically outperforms the *statevector—statevector* pipeline independently from the *k* value, as shown in Fig 8C (the statistical test results are reported in Table 9). The other versions of this baseline pipeline still win the comparison, although by a less margin. In particular, the *k-NN* + *SVM* with cosine distance and linear kernel is almost equivalent to the quantum pipeline, whereas the others are clearly superior. Moreover, the *k-NN* + *SVM* with Euclidean distance and linear kernel turns out to be equivalent to the *k-NN* + *classifier* with the same distance metric, which does not hold for the cosine distance (the SVM performs slightly better than the binary classifier in that case).

Although it is not possible to draw definitive conclusions due to the low effective usage of the model following the *k* nearest neighbors extraction, the SVM seems to perform better than the cosine similarity classifier in the baseline pipelines (especially with Gaussian kernel); as regards the Euclidean distance metric, the effective usage situation is slightly worse than that of the cosine distance, as reported in Table 5B. Hence, it could be beneficial to try to combine

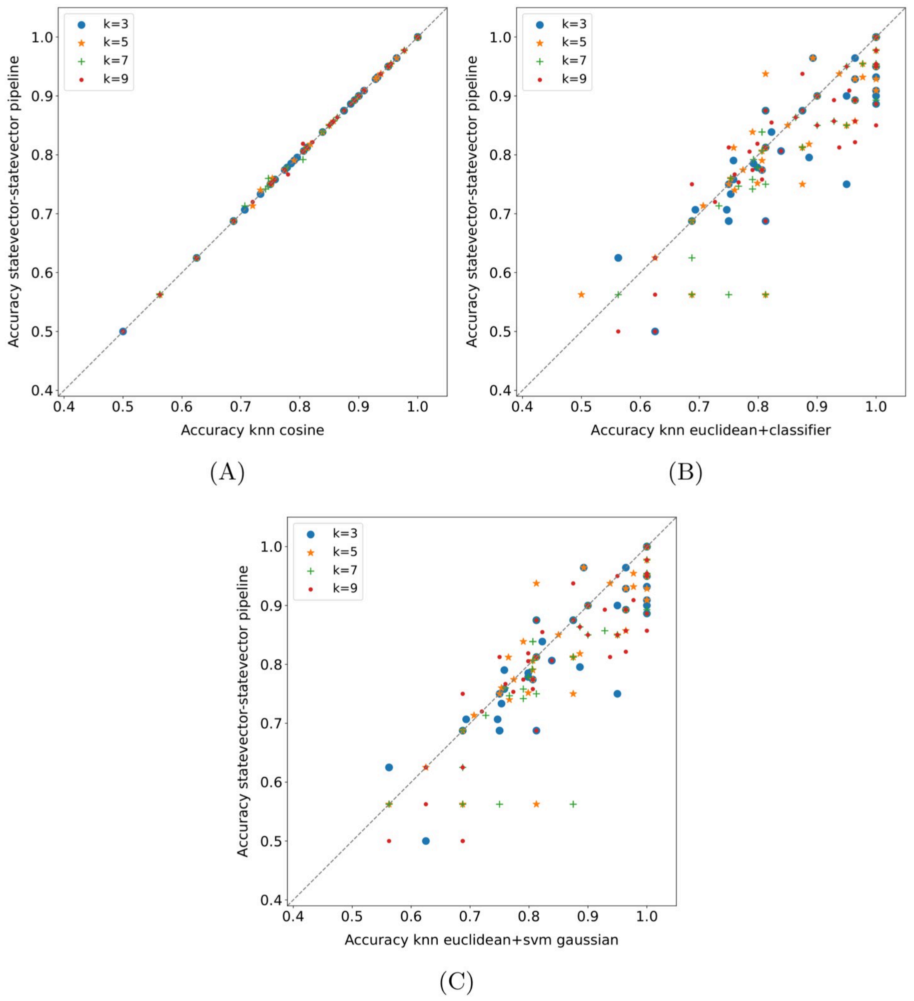


Fig 8. Quantum pipeline—(*k*-NN-based) baseline methods comparison on *15 qubits* datasets. Each point in these plots represents the accuracy obtained in a fold.

https://doi.org/10.1371/journal.pone.0287869.g008

Table 8. Wilcoxon signed-rank test ($\alpha = 0.05$) applied to the fold accuracy distributions shown in Fig 7. The values reported in the table are the p-values obtained.

	k = 3	k = 5	k = 7	k = 9
Fig 7A	1.54E-04	0.001	1.24E-04	1.88E-04
Fig 7B	0.020	0.106	0.003	0.004

https://doi.org/10.1371/journal.pone.0287869.t008

Table 9. Wilcoxon signed-rank test ($\alpha = 0.05$) applied to the fold accuracy distributions shown in Fig 8. The values reported in the table are the p-values obtained.

	k = 3	k = 5	k = 7	k = 9
Fig 8A	1.000	0.276	1.000	0.655
Fig 8B	0.003	0.006	5.22E-05	0.001
Fig 8C	0.003	0.002	2.15E-05	4.36E-04

https://doi.org/10.1371/journal.pone.0287869.t009

a quantum k-NN version with a quantum SVM. Eventually, it is worth making a last observation: the classical SVMs tend to outperform the corresponding pipelines (in terms of kernel) with cosine distance and be outperformed by/be equivalent to the corresponding ones with Euclidean distance (but, again, the low usage of the second model in the pipelines must be taken into account).

5 Conclusion

In this paper, we have proposed and evaluated the usage of a quantum locality technique as a preliminary step to enhance the performance and reduce the size of quantum machine learning models. Specifically, we have provided an implementation in Python of a quantum pipeline consisting of a quantum k-NN [18] and a quantum binary classifier [19] and its extensive empirical evaluation. Details about the implementation (based on Qiskit) and the complexity (both theoretical and practical) of the pipeline have been supplied, as well as information on the experiments performed, i.e., methods, datasets, and the setup used. First of all, the results have demonstrated the quantum pipeline's equivalence to its classical counterpart in terms of accuracy and its capability of taking advantage of larger datasets (both of them in the ideal case). Moreover, the validity of locality's application to the quantum realm has been confirmed, with the quantum pipeline outperforming the quantum binary classifier in both the ideal and simulated cases. However, the strong sensitivity of the implemented quantum k-NN to probability fluctuations has also emerged; indeed, the quantum binary classifier is affected by the same issue to a lesser extent. In addition, the quantum pipeline has been outperformed by baseline methods like the random forest and the SVMs even in the ideal case. Nevertheless, the potentialities of the implemented pipeline have not been fully exploited in these experiments; in fact, for the considered datasets, the effective usage of the second model in the pipeline is very low (the application of a majority voting or a cosine-based classifier has the same effect).

Regarding the possibility of reducing the number of qubits required to solve a problem by introducing the quantum k-NN as a preliminary step, the following relationship holds for the chosen quantum models (it is obtained from Eqs 9 and 10):

$$qubits_{qknn} \leq qubits_{qbc} \Leftrightarrow qubits_{features} \leq 3\,. \tag{11}$$

Basically, the application of the implemented quantum k-NN as a preliminary step of the quantum binary classifier is advantageous (not disadvantageous) in the number of qubits if and only if the number of data features is less than 5 (less than 9). However, it is worth remarking that Eq (11) is related to the specific quantum k-NN variant and subsequent quantum model that have been employed in this work. Furthermore, the introduction of the quantum k-NN has turned out to be advantageous in terms of performance, as shown in Section 4.4.3.

Another thing worth mentioning regards the unit-norm normalization and, hence, all quantum models exploiting the amplitude encoding of data. In detail, two instances characterized by the same ratio between feature values but different norms (e.g., all features have the same value, but this value is different for the two instances) are normalized to the same data vector. Obviously, this is a significant information loss issue. A possible workaround (not applied in this work) consists in the introduction, before the normalization, of an additional feature related to the norm.

Given the results obtained, the next step will be the development of a version of the quantum k-nearest neighbors algorithm more robust to probability fluctuations. Ideally, the quantum k-NN in question should produce the nearest neighbors quantum states as output without repeated executions of the same circuit due to the need of estimating some probability value through measurement operations. In this way, it would be possible to realize unified pipelines with other QML models without the need to break the pipeline circuit into distinct segments (the quantum k-NN output would be directly provided as input to the subsequent model). Other features of interest are the exploitation of the Euclidean distance as distance metric (it performs better than the cosine distance, as shown in Section 4.4.5), the amplitude encoding of the distance values, the absence of oracles, and a low qubit usage. The new quantum k-NN variant will also be integrated with more complex QML models, such as the quantum SVM [5], since its classical counterpart has seemed to perform better than the binary classifier as second model in the pipeline. Eventually, more complex datasets will be taken into account, as, most times, extracting the nearest neighbors already identifies the class labels in the datasets used here.

Author Contributions

Conceptualization: Enrico Zardini, Enrico Blanzieri, Davide Pastorello.

Data curation: Enrico Zardini.

Formal analysis: Enrico Zardini, Enrico Blanzieri, Davide Pastorello.

Investigation: Enrico Zardini.

Methodology: Enrico Zardini, Enrico Blanzieri, Davide Pastorello.

Software: Enrico Zardini.

Supervision: Enrico Blanzieri, Davide Pastorello.

Validation: Enrico Zardini.

Visualization: Enrico Zardini.

Writing – original draft: Enrico Zardini.

Writing – review & editing: Enrico Zardini, Enrico Blanzieri, Davide Pastorello.

References

1. Preskill J. Quantum Computing in the NISQ era and beyond. Quantum. 2018; 2:79. https://doi.org/10.22331/q-2018-08-06-79
2. Arute F, Arya K, Babbush R, Bacon D, Bardin JC, Barends R, et al. Quantum supremacy using a programmable superconducting processor. Nature. 2019; 574(7779):505–510. https://doi.org/10.1038/s41586-019-1666-5 PMID: 31645734
3. Magesan E, Blume-Kohout R, Emerson J. Gate fidelity fluctuations and quantum process invariants. Phys Rev A. 2011; 84:012309. https://doi.org/10.1103/PhysRevA.84.012309
4. Havlíček V, Córcoles AD, Temme K, Harrow AW, Kandala A, Chow JM, et al. Supervised learning with quantum-enhanced feature spaces. Nature. 2019; 567(7747):209–212. https://doi.org/10.1038/s41586-019-0980-2 PMID: 30867609
5. Rebentrost P, Mohseni M, Lloyd S. Quantum Support Vector Machine for Big Data Classification. Phys Rev Lett. 2014; 113:130503. https://doi.org/10.1103/PhysRevLett.113.130503 PMID: 25302877
6. Li Z, Liu X, Xu N, Du J. Experimental Realization of a Quantum Support Vector Machine. Phys Rev Lett. 2015; 114:140504. https://doi.org/10.1103/PhysRevLett.114.140504 PMID: 25910101
7. Romero J, Olson JP, Aspuru-Guzik A. Quantum autoencoders for efficient compression of quantum data. Quantum Science and Technology. 2017; 2(4):045001. https://doi.org/10.1088/2058-9565/aa8072
8. Wang X, Song Z, Wang Y. Variational Quantum Singular Value Decomposition. Quantum. 2021; 5:483. https://doi.org/10.22331/q-2021-06-29-483

9. Bokhan D, Mastiukova AS, Boev AS, Trubnikov DN, Fedorov AK. Multiclass classification using quantum convolutional neural networks with hybrid quantum-classical learning. Frontiers in Physics. 2022; 10. https://doi.org/10.3389/fphy.2022.1069985

10. Wu J, Tao Z, Li Q. wpScalable Quantum Neural Networks for Classification. In: 2022 IEEE International Conference on Quantum Computing and Engineering (QCE); 2022. p. 38–48.

11. Blanzieri E, Melgani F. An Adaptive SVM Nearest Neighbor Classifier for Remotely Sensed Imagery. In: 2006 IEEE International Symposium on Geoscience and Remote Sensing. Denver, CO, USA: IEEE; 2006. p. 3931–3934.

12. Hable R. Universal Consistency of Localized Versions of Regularized Kernel Methods. Journal of Machine Learning Research. 2013; 14(5):153–186.

13. Meister M, Steinwart I. Optimal Learning Rates for Localized SVMs. Journal of Machine Learning Research. 2016; 17(194):1–44.

14. Grover LK. A fast quantum mechanical algorithm for database search. In: Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing. STOC'96. New York, NY, USA: Association for Computing Machinery; 1996. p. 212–219. Available from: https://doi.org/10.1145/237814.237866.

15. Sawerwain M, Wróblewski M. Recommendation systems with the quantum k–NN and Grover algorithms for data processing. International Journal of Applied Mathematics and Computer Science. 2019; 29(1):139–150. https://doi.org/10.2478/amcs-2019-0011

16. Dua D, Graff C. UCI Machine Learning Repository; 2017. Available from: http://archive.ics.uci.edu/ml.

17. Alom MZ, Taha TM, Yakopcic C, Westberg S, Sidike P, Nasrin MS, et al. A State-of-the-Art Survey on Deep Learning Theory and Architectures. Electronics. 2019; 8(3). https://doi.org/10.3390/electronics8030292

18. Afham A, Basheer A, Goyal SK. Quantum k-nearest neighbor machine learning algorithm; 2020. Available from: https://arxiv.org/abs/2003.09187v1.

19. Pastorello D, Blanzieri E. A Quantum Binary Classifier based on Cosine Similarity. In: 2021 IEEE International Conference on Quantum Computing and Engineering (QCE); 2021. p. 477–478.

20. Chalumuri A, Kune R, Manoj BS. A hybrid classical-quantum approach for multi-class classification. Quantum Information Processing. 2021; 20(3):119. https://doi.org/10.1007/s11128-021-03029-9

21. Nielsen MA, Chuang IL. Quantum Computation and Quantum Information. Cambridge University Press; 2000.

22. Schuld M, Fingerhuth M, Petruccione F. Implementing a distance-based classifier with a quantum interference circuit. EPL (Europhysics Letters). 2017; 119(6):60002. https://doi.org/10.1209/0295-5075/119/60002

23. Wiebe N, Kapoor A, Svore KM. Quantum Algorithms for Nearest-Neighbor Methods for Supervised and Unsupervised Learning. Quantum Info Comput. 2015; 15(3–4):316–356.

24. Trugenberger CA. Quantum Pattern Recognition. Quantum Information Processing. 2002; 1(6):471–493. https://doi.org/10.1023/A:1024022632303 PMID: 12513243

25. Schützhold R. Pattern recognition on a quantum computer. Phys Rev A. 2003; 67:062311. https://doi.org/10.1103/PhysRevA.67.062311

26. IBM. International Business Machines Corporation; 2022. Available from: https://www.ibm.com/quantum-computing.

27. Rigetti. Rigetti Computing; 2022. Available from: https://www.rigetti.com/what-we-build.

28. D-Wave. D-Wave Systems; 2022. Available from: https://www.dwavesys.com.

29. Dunjko V, Taylor JM, Briegel HJ. Quantum-Enhanced Machine Learning. Phys Rev Lett. 2016; 117:130501. https://doi.org/10.1103/PhysRevLett.117.130501 PMID: 27715099

30. Biamonte J, Wittek P, Pancotti N, Rebentrost P, Wiebe N, Lloyd S. Quantum machine learning. Nature. 2017; 549:195–202. https://doi.org/10.1038/nature23474 PMID: 28905917

31. Shor PW. Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings 35th Annual Symposium on Foundations of Computer Science; 1994. p. 124–134.

32. Schuld M, Sinayskiy I, Petruccione F. An introduction to quantum machine learning. Contemporary Physics. 2015; 56(2):172–185. https://doi.org/10.1080/00107514.2014.964942

33. Buhrman H, Cleve R, Watrous J, de Wolf R. Quantum Fingerprinting. Phys Rev Lett. 2001; 87:167902. https://doi.org/10.1103/PhysRevLett.87.167902 PMID: 11690244

34. Fix E, Hodges JL. Discriminatory Analysis, Nonparametric Discrimination: Consistency Properties. USAF School of Aviation Medicine, Randolph Field; 1951. 4.

35. Fastovets DV, Bogdanov YI, Bantysh BI, Lukichev VF. Machine learning methods in quantum computing theory. In: International Conference on Micro- and Nano-Electronics 2018. vol. 11022. International

Society for Optics and Photonics. Zvenigorod, Russia: SPIE; 2019. p. 752–761. Available from: https://doi.org/10.1117/12.2522427.

36. Dürr C, Høyer P. A Quantum Algorithm for Finding the Minimum; 1999. Available from: https://arxiv.org/abs/quant-ph/9607014.

37. Dang Y, Jiang N, Hu H, Ji Z, Zhang W. Image classification based on quantum K-Nearest-Neighbor algorithm. Quantum Information Processing. 2018; 17(9):239. https://doi.org/10.1007/s11128-018-2004-9

38. Wang Y, Wang R, Li D, Adu-Gyamfi D, Tian K, Zhu Y. Improved Handwritten Digit Recognition using Quantum K-Nearest Neighbor Algorithm. International Journal of Theoretical Physics. 2019; 58(7):2331–2340. https://doi.org/10.1007/s10773-019-04124-5

39. Brassard G, Hoyer P, Mosca M, Tapp A. Quantum amplitude amplification and estimation. Contemporary Mathematics. 2002; 305:53–74. https://doi.org/10.1090/conm/305/05215

40. Ruan Y, Xue X, Liu H, Tan J, Li X. Quantum Algorithm for K-Nearest Neighbors Classification Based on the Metric of Hamming Distance. International Journal of Theoretical Physics. 2017; 56(11):3496–3507. https://doi.org/10.1007/s10773-017-3514-4

41. Li J, Lin S, Yu K, Guo G. Quantum K-nearest neighbor classification algorithm based on Hamming distance. Quantum Information Processing. 2021; 21(1):18. https://doi.org/10.1007/s11128-021-03361-0

42. Kaye P. Reversible addition circuit using one ancillary bit with application to quantum computing; 2004. Available from: https://arxiv.org/abs/quant-ph/0408173.

43. Zhou NR, Liu XX, Chen YL, Du NS. Quantum K-Nearest-Neighbor Image Classification Algorithm Based on K-L Transform. International Journal of Theoretical Physics. 2021; 60(3):1209–1224. https://doi.org/10.1007/s10773-021-04747-7

44. Schuld M, Sinayskiy I, Petruccione F. Quantum Computing for Pattern Classification. In: Pham DN, Park SB, editors. PRICAI 2014: Trends in Artificial Intelligence. Cham: Springer International Publishing; 2014. p. 208–220.

45. Wiśniewska J, Sawerwain M. Recognizing the pattern of binary Hermitian matrices by quantum kNN and SVM methods. Vietnam Journal of Computer Science. 2018; 5(3):197–204.

46. Ma Yz, Song Hf, Zhang J. Quantum Algorithm for K-Nearest Neighbors Classification Based on the Categorical Tensor Network States. International Journal of Theoretical Physics. 2021; 60(3):1164–1174. https://doi.org/10.1007/s10773-021-04742-y

47. Jozsa R. Fidelity for Mixed Quantum States. Journal of Modern Optics. 1994; 41(12):2315–2323. https://doi.org/10.1080/09500349414552171

48. Basheer A, Afham A, Goyal SK. Quantum *k*-nearest neighbors algorithm; 2021. Available from: https://arxiv.org/abs/2003.09187.

49. Mitarai K, Kitagawa M, Fujii K. Quantum analog-digital conversion. Phys Rev A. 2019; 99:012301. https://doi.org/10.1103/PhysRevA.99.012301

50. Cleve R, Ekert A, Macchiavello C, Mosca M. Quantum algorithms revisited. Proceedings of the Royal Society of London Series A: Mathematical, Physical and Engineering Sciences. 1998; 454(1969):339–354. https://doi.org/10.1098/rspa.1998.0164

51. Pastorello D, Blanzieri E. A quantum binary classifier based on cosine similarity; 2021. Available from: https://arxiv.org/abs/2104.02975.

52. Anis MS, Abraham H, AduOffei, Agarwal R, Agliardi G, Aharoni M, et al. Qiskit: An Open-source Framework for Quantum Computing; 2021.

53. Giovannetti V, Lloyd S, Maccone L. Quantum Random Access Memory. Phys Rev Lett. 2008; 100:160501. https://doi.org/10.1103/PhysRevLett.100.160501 PMID: 18518173

54. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. 2011; 12:2825–2830.

55. Yeh IC, Yang KJ, Ting TM. Knowledge Discovery on RFM Model Using Bernoulli Sequence. Expert Syst Appl. 2009; 36(3):5866–5871. https://doi.org/10.1016/j.eswa.2008.07.018

56. Patrício M, Pereira J, Crisóstomo J, Matafome P, Gomes M, Seiça R, et al. Using Resistin, glucose, age and BMI to predict the presence of breast cancer. BMC Cancer. 2018; 18(1):29. https://doi.org/10.1186/s12885-017-3877-1 PMID: 29301500

57. Silva PFB, Marçal ARS, da Silva RMA. Evaluation of Features for Leaf Discrimination. Springer Lecture Notes in Computer Science. 2013; 7950:197–204. https://doi.org/10.1007/978-3-642-39094-4_23

58. Wilcoxon F. Individual Comparisons by Ranking Methods. Biometrics Bulletin. 1945; 1(6):80–83. https://doi.org/10.2307/3001968