PLOS ONE

RESEARCH ARTICLE

A new quantum approach to binary classification

Giuseppe Sergioli[1]*, **Roberto Giuntini**[1,2], **Hector Freytes**[1]

1 University of Cagliari, Cagliari, Italy, **2** Centro Linceo Interdisciplinare "B. Segre", Roma, Italy

* giuseppe.sergioli@gmail.com

Check for updates

OPEN ACCESS

Citation: Sergioli G, Giuntini R, Freytes H (2019) A new quantum approach to binary classification. PLoS ONE 14(5): e0216224. https://doi.org/10.1371/journal.pone.0216224

Editor: Austin Lund, University of Queensland, AUSTRALIA

Received: November 27, 2018

Accepted: April 16, 2019

Published: May 9, 2019

Copyright: © 2019 Sergioli et al. This is an open access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Data Availability Statement: The considered datasets are extracted from the PMLB repository (Penn Machine Learning Benchmark - https://github.com/EpistasisLab/penn-ml-benchmarks).

Funding: This work is supported by i) the Sardinia Region Project "Time-logical evolution of correlated microscopic systems", CRP 55, LR 7/8/2007 (G. Sergioli as principal investigator); ii) "Strategies and Technologies for Scientific Education and Dissemination" (number F71I17000330002) founded by Fondazione Sardegna (G. Sergioli as principal investigator); iii) by the Horizon 2020 program of the European

Abstract

This paper proposes a new quantum-like method for the binary classification applied to classical datasets. Inspired by the quantum Helstrom measurement, this innovative approach has enabled us to define a new classifier, called *Helstrom Quantum Centroid* (HQC). This binary classifier (inspired by the concept of distinguishability between quantum states) acts on density matrices—called *density patterns*—that are the quantum encoding of classical patterns of a dataset. In this paper we compare the performance of HQC with respect to twelve standard (linear and non-linear) classifiers over fourteen different datasets. The experimental results show that HQC outperforms the other classifiers when compared to the Balanced Accuracy and other statistical measures. Finally, we show that the performance of our classifier is positively correlated to the increase in the number of "quantum copies" of a pattern and the resulting tensor product thereof.

1 Introduction

In the past few decades, various methods based on quantum information theory have been used extensively to focus on a variety of problems concerning classification and clustering [1–4]. On the other hand, some classification methods developed in computer engineering have been employed to solve such problems as quantum-state discrimination [5–8], which are closely connected with certain recent developments in quantum cryptography. In view of these exchanges, quantum computation and machine learning are nowadays recognized as two closely-related connected research fields. A natural starting point for bridging these two different topics is to establish a common background. The initial idea was to represent classical patterns in terms of quantum objects, with an eye to increasing the computational efficiency of the classification algorithms. Following this intuition, in the past few years many attempts have been made to apply the quantum formalism to signal processing [9] and pattern recognition [10, 11].

A recourse to quantum states to represent classical patterns is naturally motivated by the possibility of exploiting the potential of quantum algorithms to boost the classification process. By way of example, it has been demonstrated that quantum algorithms can be used to reduce the time complexity of the standard k-nearest neighbor (kNN) classifier. Using the algorithms introduced in [3] (i.e., a quantum computational counterpart of the kNN classifier), it is

Commission: SYSMICS project, number: 689176, MSCA-RISE-2015 and iv) Fondazione Banco di Sardegna project "Science and its Logics", Cagliari, number: F72F16003220002 (H. Freytes and R. Giuntini as members).

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

possible to yield a polynomial reduction in query complexity compared to the corresponding classical algorithm. Extensive surveys concerning the applications of quantum computing in computational intelligence and machine learning can be found in [4, 12]. It is thought these approaches may lead to computational advantages from a quantum perspective [13, 14].

A different line of research, however, consists in using quantum formalism to obtain significant benefits in the classical computational contexts. We intend to explore this idea and provide a model aimed at processing binary classical datasets (in a supervised system), i.e., dataset containing only two classes of different objects (it should be remembered that there are several methods [15] to treat multiple class classification as a suitable combination of binary classifications).

The architecture of our model comprises the following three steps: *i*) *encoding*: each element (object, observation) of the dataset is encoded into a density operator, which is the standard mathematical tool to formally describe a quantum state; *ii*) *classification*: i.e., the application of a "quantum-inspired" version of a standard binary classifier on the encoded dataset is applied; *iii*) the result of the classification process is decoded in the initial classical space. The experimental setting is based on the application of our models to artificial and real datasets available from standard machine learning repositories.

A number of recent works [16–18] have introduced a "quantum-inspired" classifier—named *Quantum Nearest Mean Classifier* (QNMC)—based on the Nearest Mean Classifier (NMC). These works have highlighted the benefits of the QNMC on both artificial and real datasets, and in a biomedical context, in particular, QNMC has been employed to detect the pulmonary fibrosis in a dataset of suspected patients [19].

Here, we introduce a new binary classifier based on the Helstrom measurement [7] and provide a full comparison between this classifier and some other commonly used (linear and non linear) classifiers. The comparison is made by analyzing significant statistical quantities obtained by applying each algorithm to fourteen different datasets. We show that the new algorithm, on average, performs better than all the other competitors considered.

The paper is organized as follows: in Section 3 we describe the standard procedure to introduce a quantum-inspired classifier. In Section 4, we provide details of the so called Helstrom Quantum Centroid (HQC). Section 5 compares the performances of HCQ with other standard and commonly used classifiers by applying each of these on fourteen different datasets. Final remarks and further developments conclude the paper. All statistical data are listed in the Tables A-O in S1 File.

2 Quantum-inspired methods in the classification process

The general purpose of a classification process is to classify a set of objects, i.e., to assign to each object from the set, a label that represents real classes (for instance the class of the cats, the class of the cancer cells, the class of the human faces, etc). Following a standard classification procedure of supervised learning systems (i.e., learning from a training dataset of correctly labeled objects) we initially select the d features that characterize all the objects of a given dataset. Thus, each object is represented by a d-dimensional real vector $X = (x^1, \ldots, x^d) \in \mathbb{R}^d$. Formally, we say that a pattern is represented by a pair (X_i, l_i), where X_i is the d-dimensional vector associated with the object and l_i labels the class of which the objects is a member. We consider a class as being merely a set of objects and we confine ourselves to the very common case where each object belongs to one class of objects only. Let $L = \{l_1, \ldots, l_M\}$ be the set of labels corresponding to their respective classes. The goal of the classification process is to design a *classifier* that will attributes (in the most accurate way possible) a label (a class) to any unlabeled object. The strategy is divided into two stages; first, training the classifier and

second, performing the test proper. The dataset is therefore also divided into two parts: the first is used to train the classifier and the second is used to properly verify the accuracy of the classifier during a test [15, 20].

The aim of this work is to provide an original quantum-inspired version of the classical classification process and to show the extent to which this new model improves the accuracy (and also other significant statistical quantities) of the same process.

In order to provide a quantum approach to a standard classifier, the following three steps must be followed:

1. *Encoding*: a theoretical quantum object (i.e., a density operator in our case) is associated to each pattern, which we will call a *density pattern*;
2. *Classification*: we provide a quantum-inspired counterpart of standard classification procedure applied on a dataset of density operators (density patterns) instead of on a dataset of real vectors;
3. *Decoding*: we decode the results of the classification process in the domain of real vectors.

2.1 Encoding

It is well known that there are an infinite number of ways to encode a real vector into a density operator and the current state of the art suggests potential computational advantages of this sort. However, the relative advantages of each option may be strictly dataset-dependent: finding the "best" mode of encoding from real vectors to quantum states is still an open and complex problem. This is by no means surprising, given that (in accordance with the well known *No Free Lunch* Theorem [15]), in general no given classification method is superior in all aspects to all its competitors, because each dataset has its own unique and specific characteristics. Some recent works [16–18] have suggested different ways to encode real patterns into density operators and we have investigated how different encodings yield different classification performances. Throughout this paper we focus on an encoding that is defined by means of the inverse of the standard stereographic projection [21].

Let $X = (x^1, \ldots, x^d) \in \mathbb{R}^d$ be a d-dimensional vector. We map the vector $X \in \mathbb{R}^d$ into a vector $X' \in \mathbb{R}^{d+1}$ as follows: $X' = \alpha\,(2x^1, \ldots, 2x^d, \sum_{i=1}^{d} (x^i)^2 - 1)$ where α is a normalization factor given by $\alpha = \frac{1}{\sum_{i=1}^{d} (x^i)^2 + 1}$.

Now we define the encoded density pattern as: $\rho_X = (X')^\dagger \cdot (X')$ Hence, this encoding maps real d-dimensional vectors X into a $(d + 1)$-dimensional pure state ρ_X. This is the encoding that will be used in the experiment which is detailed in Section 4.

2.2 Classification

As stated, the main purpose of this work is to provide an original and convenient quantum-inspired classifier. By *quantum-inspired* we mean a classification process that is implemented by a classical computer on macroscopical objects, but which employs and exploits of quantum theory. Some variants of a quantum counterpart of the commonly-used standard *Nearest Mean Classifier* (NMC)—named *Quantum Nearest Mean Classifier* (QNMC)—have recently been analysed. Specifically, the QNMC has been applied to some real and some artificial datasets, obtaining a better performance (in terms of the accuracy of the process and other relevant statistical quantities) compared to the standard NMC. Interestingly, an experiment carried out on a real biomedical dataset provided surprising and highly promising results [19].

As indicated above, the classification process is clearly empirical and it is often hard (or impossible) to determine the superiority of one model over another. In the following sections, we introduce another quantum-inspired classifier and we compare its performances with respect to the QNCM and to some standard classical classifier.

2.3 Decoding

The encoding defined above is based on invertible functions that univocally map real vectors onto density operators. After the classification process, rather than applying inverse encoding to each density operator in order to retrieve the original real vector, it would seem more expedient to employ the corresponding label that has been attributed by the classification process. However, having assigned labels for each density pattern of the test set, the decoding turns out to be a non-essential step.

3 The Helstrom distance-inspired classifier

In this section we describe a quantum-style classification process for binary classification based on the Helstrom measurement (see [22, 23]), which was initially introduced by Helstrom in a seminal work that addressed the following question: "Suppose to deal with an unknown quantum state drawn from an ensemble of possible pure states where each state is labeled with respect to the class they came from. How well can we predict the class of this unknown quantum state?" [24]. This problem is generally known as the *quantum state discrimination problem* or *quantum classification problem* [7] when referred to machine learning. The answer clearly depends on a number of factors—e.g. crucially, on the amount of available information and on the way this quantity of information might be improved. Unlike in the classical case, in quantum information multiple copies of a quantum state provide more information about it than that is encoded in a single copy thereof. We shall exploit this property in order to improve the classification process inspired by Helstrom's construct.

Let $\rho_1, \ldots, \rho_k$ be density operators and let associate to each ρ_i operator an *a priori* probability p_i. The distribution of the a priori probabilities is determined empirically and in a contextual manner. Let us define a state as a pair $\{\rho, p\}$, where ρ is a density operator and p is its respective *a priori* probability and let us define a set of states as a set of pairs $\{(\rho_1, p_1), \ldots, (\rho_k, p_k)\}$ such that $\sum_{i=1}^{k} p_i = 1$.

Suppose to have a set consisting of two states $\{(\rho_1, p_1), (\rho_2, p_2)\}$ such that $p_1 + p_2 = 1$. It is possible to introduce the operator Λ as:

$$\Lambda = p_1\rho_1 - p_2\rho_2. \tag{1}$$

It is not difficult to realize that Λ (called *Helstrom observable*) is an Hermitian operator (with trace equal to 0 if $p_1 = p_2 = \frac{1}{2}$).

Let $\lambda^+ = \{\lambda_1^+, \ldots, \lambda_m^+\}$ be the set of all the eigenvectors of Λ associated with their respective non negative eigenvalues of Λ and let $\lambda^- = \{\lambda_1^-, \ldots, \lambda_l^-\}$ (with $m + l \leq n$) be the set of all the eigenvectors of Λ associated with their respective negative eigenvalues. Let $P^+ = \sum_{\lambda_i \in \lambda^+} P_{\lambda_i}$ and $P^- = \sum_{\lambda_i \in \lambda^-} P_{\lambda_i}$, where P_{λ_i} is the projector associated with the eigenvector λ_i that belongs to λ^+ or to λ^-. Notice that, since $P^+ + P^- = \mathbb{I}$—where $\mathbb{I}$ is the identity matrix—then the set $\{P^+, P^-\}$ is a von Neumann measurement with respect to the Helstrom observable [25]. The definitions of the projectors P^+ and P^- enable us to present the following result: suppose now we randomly pick a density operator ρ within a multiset S where each element of S can be only ρ_1 or ρ_2 [23, 26]. The probability that one can correctly discriminate whether ρ is ρ_1 or ρ_2 has

an upper bound given by:

$$P_{guess}^{(\rho_1,\rho_2)} = p_1 Tr(P^+\rho_1) + p_2 Tr(P^-\rho_2) \quad (2)$$

where the *a priori* probability can be interpreted as the respective frequencies of ρ_1 and ρ_2 over S, i.e., the number of occurrences of ρ_1 and, respectively, ρ_2 over the cardinality of S.

The quantity $P_{guess}^{(\rho_1,\rho_2)}$ is generally called the *Helstrom bound* of the error in the discrimination between the two density operators ρ_1 and ρ_2 and it can be seen as a measurement of distinguishability between ρ_1 and ρ_2 [7, 27]. Intuition seems to tell us that $p_1 Tr(P^+\rho_1)$ represents the conditional probability that, if we pick $\rho = \rho_1$, then ρ is correctly identified as ρ_1; the same applies for $p_2 Tr(P^-\rho_2)$. Clearly, if there is an equal probability that, randomly picking a density operator ρ, this ρ is ρ_1 or ρ_2, then $p_1 = p_2 = \frac{1}{2}$. Given the fact that the accuracy of a classification process is an empirical value (it can change, in principle, for any different run of the process) it is not easy to find a formal correlation between accuracy and $P_{guess}^{(\rho_1,\rho_2)}$. However, let us consider a binary classification process based on the distinguishability between two centroids ρ_1 and ρ_2 of the two different classes. In this case it is robustly reasonable to assume that a high value of $P_{guess}^{(\rho_1,\rho_2)}$ (i.e., a high probability of distinguishing between the two centroids) should be related to a "good" performance of the classifier. Further, in order to optimize the value of $P_{guess}^{(\rho_1,\rho_2)}$, it will be useful to recall the fact that providing additional copies of quantum states makes it possible to obtain a lowered error in the discrimination probability [28].

To obtain a significantly lower error in the state discrimination, let us now describe the new quantum-style classification procedure, inspired by the Helstrom model referred to above. Hereafter, we will consider binary classification (i.e., with two classes only) and we will assume that the labels $l_m \in \{+, -\}$, for all $m \in \{1, \ldots, M\}$.

Given a training dataset $S_{tr} = \{(X_1, l_1), \ldots, (X_M, l_M)\}$ we can define the *positive class* S_{tr}^+ and the *negative class* S_{tr}^- of S_{tr} as follows:

$$S_{tr}^+ = \{(X_i, l_i) \in S_{tr} : l_i = +\} \text{ and } S_{tr}^- = \{(X_i, l_i) \in S_{tr} : l_i = -\}. \quad (3)$$

S_{tr}^+ (S_{tr}^-, respectively) is the set of all patterns of the training dataset belonging to the class labeled by + (−, respectively).

By M^+ (M^-, respectively) we will denote the cardinality of S_{tr}^+ (of S_{tr}^-, respectively). Clearly, $M^+ + M^- = M$. Following standard procedure, in order to introduce a quantum version of the classification process, we first need to encode any real vector X_i in terms of a density operator ρ_{X_i}. After using one encoding from real vectors to density operators (by means of the stereographic encoding outlined in Section 2.1 or the informative encoding introduced in [17]), it is possible to establish the definitions for the quantum centroids in the positive and the negative classes.

Let $QS_{tr} = \{\{\rho_{X_1}, l_1\}, \ldots, \{\rho_{X_M}, l_M\}\}$ be a quantum training dataset of density patterns. The *Quantum Centroids* for the positive and the negative class are given, respectively, by:

$$\rho^+ = \frac{1}{M^+} \sum_{i \in \{m : l_m = +\}} \rho_{X_i} \text{ and } \rho^- = \frac{1}{M^-} \sum_{i \in \{m : l_m = -\}} \rho_{X_i}. \quad (4)$$

Notice that the quantum centroids are generally mixed states and clearly do not correspond to the density patterns obtained by encoding the centroids of the original dataset. Interestingly, the expressions of the quantum centroids do not remain invariant if the coordinates of the dataset are rescaled. If we then rescale all the features of a given dataset by a real parameter t, the new centroid C_t of the rescaled dataset is obtained by rescaling the coordinates of the original centroid C. However, it seems quite clear that in general, the encoding process does not

preserve the centroid, a fact that significantly affects the classification process. In [21] we have shown that the failure of invariance under rescaling plays the role of an asset: indeed, the rescaling factor t can be used as a free parameter to optimize the accuracy of the classification process.

Given a training dataset, it is possible to define the *quantum Helstrom observable* associated to it as:

$$\Lambda_Q = \frac{M^+}{M}\rho^+ - \frac{M^-}{M}\rho^-. \tag{5}$$

This expression is analogous to Eq 1, where the *a priori* probability is obtained as the frequencies of the elements belonging to the positive and negative classes, respectively, over the entire training dataset.

Let $\lambda_Q^+ = \{\lambda_{Q1}^+, ..., \lambda_{Qm}^+\}$ be the set of all eigenvectors of Λ_Q associated with the non negative eigenvalues and let $\lambda_Q^- = \{\lambda_{Q1}^-, ..., \lambda_{Ql}^-\}$ (with $m + l \leq n$) be the set of all eigenvectors of Λ_Q associated with the negative eigenvalues. Let $P_Q^+ = \sum_{\lambda_{Qi} \in \lambda_Q^+} P_{\lambda_{Qi}}$ and $P_Q^- = \sum_{\lambda_{Qi} \in \lambda_Q^-} P_{\lambda_{Qi}}$, where $P_{\lambda_{Qi}}$ is the projector associated with the eigenvector λ_{Qi} that belongs to λ_Q^+ or to λ_Q^-, respectively. The *positivity* and *negativity* of the eigenvalues of the Helstrom observables should not be confused with the *positivity* and *negativity* of the classes. More precisely, in P_Q^+ and P_Q^-, the signs + and − are referred to the sign of the eigenvalues of the Helstrom observable; on the other hand, in the expression of the centroids ρ^+ and ρ^-, the signs + and − are referred to the positive and negative class, respectively.

Given an arbitrary pattern X that belongs to the *test dataset*, the classification of X as belonging to the positive or to the negative class is dictated by the following classification function.

Let ρ_X be the density pattern associated with the d-dimensional real vector X (obtained by means of encoding). It is possible to define a *Helstrom Quantum Centroid* classifier (HQC) as follows:

$$HQC(\rho_X) = \begin{cases} + & \text{if} \quad Tr(\rho_X P_Q^+) \geq Tr(\rho_X P_Q^-); \\ - & \text{otherwise.} \end{cases} \tag{6}$$

Finally, in this case and referring to Eq 2 the Helstrom bound will be reasonably defined as:

$$P_{guess}^{(\rho^+, \rho^-)} = \frac{M^+}{M} Tr(\rho^+ P_Q^+) + \frac{M^-}{M} Tr(\rho^-) P_Q^-. \tag{7}$$

3.1 The HQC with copies

We shall now describe a variant of HQC that essentially consists in making n copies of each density pattern of the dataset. This procedure is consistent due to the fact that, unlike in the classical case, in quantum information the state $\rho \otimes \ldots \otimes \rho$ (the n-fold tensor product of ρ) is generally more informative than the single state ρ.

Accordingly, we proceed as follows both for the training and for the test datasets.

Given a training dataset S_{tr} and its respective quantum training dataset $QS_{tr} = \{\{\rho_{X_1}, l_1\}, \ldots, \{\rho_{X_M}, l_M\}\}$ let us define the n-quantum training dataset $(\otimes^n)QS_{tr}$ as follows:

$$(\otimes^n)QS_{tr} = \{\{\otimes^n \rho_{X_1}, l_1\}, \ldots, \{\otimes^n \rho_{X_M}, l_M\}\}, \tag{8}$$

where by $\otimes^n \rho_{X_i}$ we denote the n-fold tensor product of ρ_{X_i}.

Thus, according to Eqs 1 and 4 we can define the n-quantum centroids $(\otimes^n)\rho^+$ and $(\otimes^n)\rho^-$ as well as the n-quantum Helstrom observable $(\otimes^n)\Lambda_Q$ with its associated von Neumann measurement $\{(\otimes^n)P_Q^+, (\otimes^n)P_Q^-\}$. Let us remark that in general $(\otimes^n)\rho^+ \neq \otimes^n \rho^+$, i.e., the centroid of the density patterns of $(\otimes^n)QS_{tr}$ labeled by + is not the n-fold tensor product of the centroid of the density patterns of QS_{tr} labeled by +. Similarly holds for ρ^-, Λ_Q, P_Q^+ and P_Q^-.

We can define the n-Helstrom Quantum Centroid classifier ($(\otimes^n)HQC$) in a similar way to Eq 6.

Let ρ_X be the density pattern associated with the d-dimensional real vector X. Then,

$$(\otimes^n)HQC(\rho_X) = \begin{cases} + & \text{if} \quad Tr(\otimes^n \rho_X((\otimes^n)P_Q^+)) \geq Tr(\otimes^n \rho_X((\otimes^n)P_Q^-); \\ - & \text{otherwise.} \end{cases} \tag{9}$$

and the *n-Helstrom bound* can be generalized as follows:

$$P_{guess}^{((\otimes^n)\rho^+,(\otimes^n)\rho^-)} = \frac{M^+}{M} Tr((\otimes^n)\rho^+(\otimes^n)P_Q^+) + \frac{M^-}{M} Tr((\otimes^n)\rho^-(\otimes^n)P_Q^-). \tag{10}$$

It can be proved (see Appendix A) that, in the simple special case where all the density patterns (belonging to both training and test datasets) are diagonal, with dimension two and conditions are such that $M^+ = M^-$, then the value of the n-Helstrom bound decreases by making a copy for each density pattern. A systematic and complete theoretical analysis is yet to be undertaken and shall be left to a future study.

In the next section we show some results based on the application of this classifier to different two-classes datasets. The outcomes of our experiments show that on average the $(\otimes^n)$HQC classifier outperforms the large number of commonly used classifiers.

4 The experiment

In this section we show some significant improvement to the performances of HQC and $(\otimes^n)$HQC with respect to a large set of classifiers that generally perform well for different kinds of datasets. Given the nature of HQC, we restrict the experimental setup to binary datasets, i.e., datasets with only two classes.

Depending on the particular distribution of the dataset, it is possible that a pattern belonging to a given class is incorrectly classified. For an arbitrary pattern (X_i, λ_i), four cases are possible:

- X_i is a *true positive* (TP) object: the pattern (X_i, λ_i) belongs to the positive class $\mathcal{S}_{\text{tr}}^+$, and it is correctly classified, i.e., $Cl(X_i) = +$
- X_i is a *true negative* (TN) object: the pattern (X_i, λ_i) belongs to the negative class $\mathcal{S}_{\text{tr}}^-$, and it is correctly classified, i.e., $Cl(X_i) = -$
- X_i is a *false positive* (FP) object: the pattern (X_i, λ_i) belongs to negative class $\mathcal{S}_{\text{tr}}^-$, but it is incorrectly classified, i.e., $Cl(X_i) = +$
- X_i is a *false negative* (FN) object: the pattern(X_i, λ_i) belongs to positive class $\mathcal{S}_{\text{tr}}^+$, but it is incorrectly classified, i.e., $Cl(X_i) = -$

Then, by applying the classifier to the test set, it is possible to assess its performance by considering the following significant statistical quantities:

- *Accuracy*: $Ac = \frac{\#TP+\#TN}{\#TP+\#TN+\#FP+\#FN}$
- *Sensitivity*: $Se = \frac{\#TP}{\#TP+\#FN}$

- *Specificity*: $Sp = \frac{\#TN}{\#TN+\#FP}$
- *Balanced Accuracy*: $Ba = \frac{1}{2}\left(\frac{\#TP}{\#TP+\#FN} + \frac{\#TN}{\#FP+\#TN}\right)$
- *Precision*: $Pr = \frac{\#TP}{\#TP+\#FP}$
- *F–measure*: $F - m = \frac{2Pr \cdot Se}{Pr+Se}$
- *Cohen's k parameter*: $k = \frac{Pr(a)-Pr(e)}{1-Pr(e)}$ where $Pr(a) = \frac{\#TP+\#TN}{M'_+ + M'_-}$ and $Pr(e) = \frac{(\#TP+\#FP)(\#TP+\#FN)+(\#FP+\#TN)(\#TN+\#FN)}{(M'_+ + M'_-)^2}$.

  Notice that $-1 \leq k \leq 1$; intuitively, the case $k = 1$ corresponds to a perfect classification (error $E = 0$); on the other hand, the case $k = -1$ results in an entirely wrong classification (error $E = 1$).

In the following section we apply HQC to several datasets and we compare its performances with other commonly used classifiers, showing the marked average superiority of HCQ compared to all the other classifiers.

4.1 Experimental setup and methodology

In order to assess the performances of HQC and $(\otimes^n)$HQC, we first apply these classifiers to fourteen different datasets and we evaluate all the statistical quantities described above. Then, having applied other commonly used (and generally well-performing) classifiers to the same datasets, we then compare the resulting outcomes.

The datasets we are dealing with are extracted from the PMLB repository (Penn Machine Learning Benchmark) [29]. This repository includes datasets that typically take one of three forms. The first is accessible, well-studied *real-world data*, taken from different real-world problem domains of interest. The second is *simulated data*, or data that have been artificially generated, often to 'look' like real-world data, but with known, underlying patterns. The third form is *toy data*, which we define here as data that is also artificially generated with a known embedded pattern but without an emphasis on representing real-world data. We consider all of these kinds of datasets. More precisely, we have run on these 14 datasets: *Banana*, *Prnn synth*, *Analcatdata aids*, *Haberman*, *Moon*, *Lupus*, *Gaussian*, *Titanic*, *Analcatdata boxing1*, *Analcatdata asbestos*, *Appendicitis*, *Analcatdata boxing2*, *Hill Valley (with noise)*, *Hill Valley (without noise)*.

We have applied HQC on these datasets by making copies of each element of a given dataset as described in 3.1. Next, using *HelstromQuantumCentroid#* we indicate the application of HQC with # number of copies. In particular, HQC has been run for # $\in \{1, \ldots, 4\}$. The other classifiers that we applied to the datasets listed above, are: *BernoullyNB*, *Logistic Regression*, *Gaussian NB*, *K-Neighbors Classifier*, *Random Forest Classifier*, *Ada Boost Classifier*, *Nearest Centroid*, *Linear Discriminant Analysis*, *Extra Trees Classifier*, *Gradient Boosting Classifier*, and *QNMC*.

The experimental procedure was repeated identically for each dataset and was implemented by a standard version of Python. The experiment essentially involves following the four steps described here:

1. Tuning: we run the HQC algorithm by considering the full dataset as a training set. However, before each encoding, we rescale the dataset factor t, as discussed in Section 4. We range the rescaling factor t along consecutive subintervals of length 0.5 within the interval (0.1, 10). For each dataset D_i, we obtain the *optimal* value of t_i, i.e., the value of t_i that produces the best Balanced Accuracy of the HQC classifier (with respect to the given interval).

We rescale each dataset D_i by t_i and we encode the dataset using the encoding described in Section 2.1. Note that the action of the rescaling factor t does not affect the performances of all the other (classical) standard classifiers.

2. Cross validation: for each dataset we randomly pick 80% of the dataset as the training set and we effect the classification on the rest of the dataset. Once the same procedure has been repeated 10 times, we present all statistical values introduced at the beginning of Section 5.
3. Local Comparison: we repeat the same steps described above for all the 14 datasets and for all the classifiers. In particular, we apply HQC with 1,2,3 and 4 copies. In so doing, we obtain a comparison table for each dataset. See Tables A—O in S1 File.
4. Global Comparison: by extracting data related to the Balanced Accuracy from Tables A-O in S1 File, we can provide a comparison between and among all the classifiers. In particular, in Fig 1, we resume the value of the mean Balanced Accuracy of each classifier and for any dataset. In Fig 2, we indicate the respective standard deviations. In Fig 3, we depict a two-by-two comparison between all the classifiers. In particular, for each classifier in the left-hand column we count the success rate achieved by comparing the Balance Accuracy obtained by applying it on each dataset with the Balance accuracy obtained by applying the other classifiers named in the line below the picture. Finally, in Table O in S1 File we summarize the Average Success Rate of each of the classifiers over all datasets.

4.2 Discussion

The crucial result is shown in Fig 3 and Table O in S1 File: over all the 14 datasets HQC is, on average, the classifier that consistently exhibits the best performance. In particular, HQC with 4 copies has an Average Success Rate equal to 72.8%, while the first standard classifier below

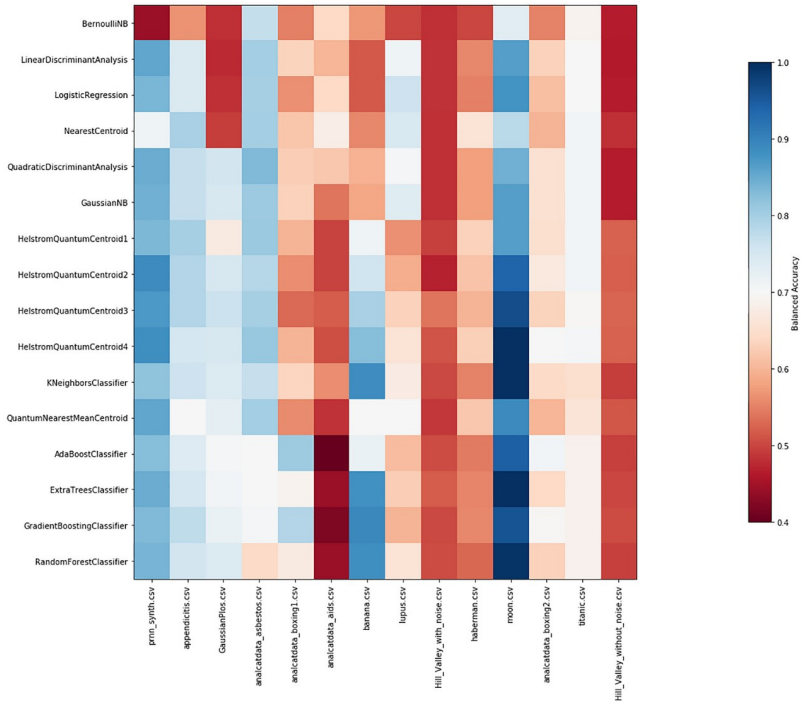

Fig 1. Biclustering of the 16 classifiers and 14 datasets according to Balanced Accuracy.

https://doi.org/10.1371/journal.pone.0216224.g001

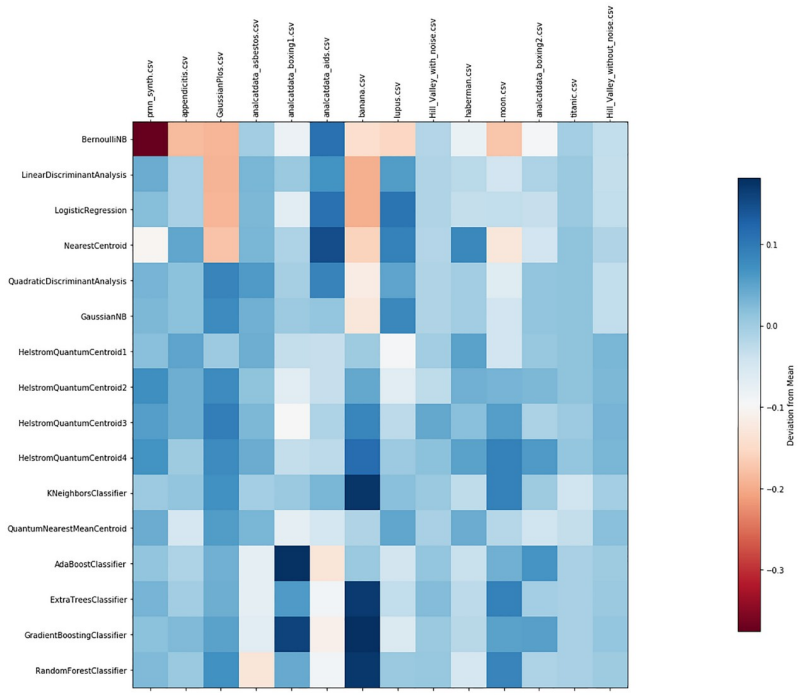

Fig 2. Deviation from the mean Balanced Accuracy across all 16 classifiers.

https://doi.org/10.1371/journal.pone.0216224.g002

HQC is the Gaussian NB (Gaussian Naive Bayes) whose Average Success Rate is -only- equal to 58%. Further, by looking at Fig 1, it is possible to see that HQC with 4 copies is one of the best-performing classifiers (in terms of Balanced Accuracy) for almost all the datasets (except for the *analcatdata asbestos* and *lupus* datasets). We can also see that, in general, an increase in the number of copies produces an improvement in all the statistical quantities: Tables A-O in S1 File demonstrate the high efficiency of the HQC classifier not only with respect to the Balanced Accuracy but, proportionally, also with respect to all the other statistical quantities. Also, the values of these quantities—on average—improve more and more as the number of the copies increases.

5 Concluding remarks and further developments

In this paper we have introduced an innovative technique—inspired by the formalism of quantum theory—to design a new kind of supervised classifier (HQC) and we have provided a full comparison of the performances of this new classifier with respect to many others (linear and non linear) frequently used classifiers. The HQC proves to be superior, on average, compared to all the statistical parameters we considered. As a result, we believe that the potential of the quantum formalism as an application for classification processes in the classical context is extremely promising.

However, further research of both a theoretical and applied nature, is needed. From a theoretical viewpoint, special attention will be devoted to investigating the encoding step in order to find, for any dataset (or for specific classes of datasets) the *most suitable* encoding procedure, i.e., the encoding that provides the best performance of the HQC classifier. Another theoretical investigation should be devoted to generalizing HQC in such a way as to also use the classifier for n-ary classification (not only binary classification). Finally, as we have seen, an

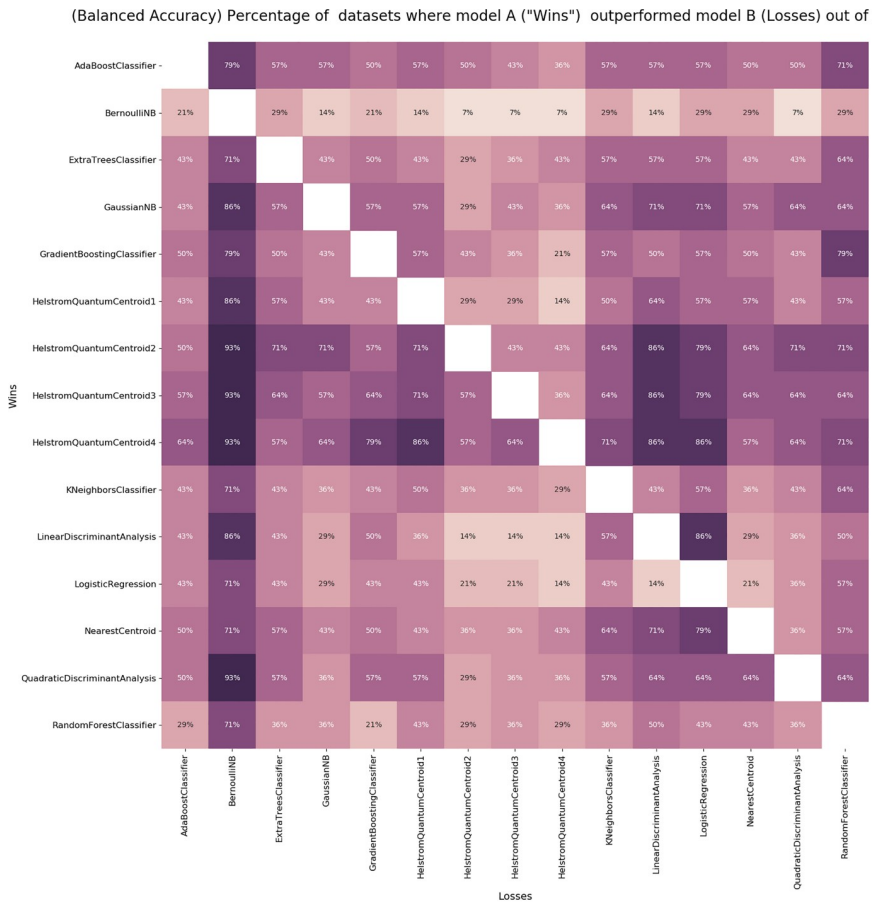

Fig 3. When a classifier A outperforms a classifier B according to the Balanced Accuracy.

https://doi.org/10.1371/journal.pone.0216224.g003

increase in the number of copies of the HQC classifier, frequently benefits the classification process. However, this increase provides a non-negligible computation cost that prevents us from running the algorithm with more than four copies in cases of large datasets or in cases of many-features vectors. Hence, this technical problem also deserves further attention.

On the other hand, with regard to practical implementation, previous works discussed above [19] have pointed to possible applications of these kinds of approaches in the biomedical field. Anyway, these non-standard applications are, though, still at an initial stage. An in-depth interdisciplinary investigation will be carried on in this direction: these early achievements might well be extended to different contexts of application, including pattern recognition, fingerprint recognition and, in short, all the contexts where the classification procedure plays a crucial role.

Appendix A

Let us consider two classes of the same cardinality $C_1 = \{\rho_i\}_{i=1,\cdots,n}$ and $C_2 = \{\sigma_i\}_{i=1,\cdots,n}$ of 2×2- diagonal density matrices, where $diag[\rho_i] = [1 - r_i, r_i]$ and $diag[\sigma_i] = [1 - s_i, s_i]$ and where the non trivial cases $0 < r_i, s_i < 1$ are considered. Let us indicate by ρ and σ the centroids of the classes C_1 and C_2, respectively, and let us suppose that $\rho \neq \sigma$. By referring to Eq 10, we show that $P_{guess}^{((\otimes^2)\rho,(\otimes^2)\sigma)} \geq P_{guess}^{(\rho,\sigma)}$.

By Eq 4, we have that $diag[\rho] = \frac{1}{n}[n - \sum_{i=1}^{n} r_i, \sum_{i=1}^{n} r_i]$ and $diag[\sigma] = \frac{1}{n}[n - \sum_{i=1}^{n} s_i, \sum_{i=1}^{n} s_i]$. By Eq 5 we obtain: $diag[\Lambda_Q] = [\alpha, -\alpha]$, where $\alpha = \frac{\sum_{i=1}^{n}(s_i - r_i)}{2n}$. Let us notice that α can not be zero because $\rho \neq \sigma$. The values $\{\alpha, -\alpha\}$ are the eigenvalues of Λ_Q and the respective eigenvectors are: $\{(1, 0)^\dagger, (0, 1)^\dagger\}$. Without loss of generality, let us suppose that $\alpha > 0$. In this case, $P_Q^+ = (1, 0)^\dagger(1, 0)$ and analogously $P_Q^- = (0, 1)^\dagger(0, 1)$; hence, by a straightforward calculation we obtain that $P_{guess}^{(\rho,\sigma)} = \frac{1}{2}Tr(P_Q^+\rho) + \frac{1}{2}Tr(P_Q^-\sigma) = \frac{1}{2} + \alpha$. Now we proceed in order to evaluate the quantity $P_{guess}^{((\otimes^2)\rho,(\otimes^2)\sigma)}$.

Let us consider the two classes $C_1^{(2)} = \{\rho_i \otimes \rho_i\}_{i=1,\ldots,n}$ and $C_2^{(2)} = \{\sigma_i \otimes \sigma_i\}_{i=1,\ldots,n}$. Let $(\otimes^2)\rho$ and $(\otimes^2)\sigma$ be the centroids of $C_1^{(2)}$ and $C_2^{(2)}$, respectively. We have that $diag[(\otimes^2)\rho] = \frac{1}{n}[\sum_{i=1}^{n}(1 - r_i)^2, \sum_{i=1}^{n} r_i(1 - r_i), \sum_{i=1}^{n} r_i(1 - r_i), \sum_{i=1}^{n} r_i^2]$. Analogously for $diag[(\otimes^2)\sigma]$. By defining $(\otimes^2)\Lambda_Q = \frac{1}{2}((\otimes^2)\rho - (\otimes^2)\sigma)$, it is easy to see that the eigenvalues of $(\otimes^2)\Lambda_Q$ are $\{2\alpha - \beta, \beta - \alpha, \beta - \alpha, -\beta\}$ (where α has been previously defined and $\beta = \frac{\sum_{i=1}^{n}(s_i^2 - r_i^2)}{2n}$) and the respective eigenvectors are $\{(1, 0, 0, 0)^\dagger, (0, 1, 0, 0)^\dagger, (0, 0, 1, 0)^\dagger, (0, 0, 0, 1)^\dagger\}$. For the sake of simplicity, we also call the eigenvalues of $(\otimes^2)\Lambda_Q$ as $\{\mu_1, \mu_2, \mu_3, \mu_4\}$. Now, reminding that we have assumed $\alpha > 0$, from the assumption that $0 < r_i, s_i < 1$, it trivially follows that $0 < \beta < 2\alpha$. Hence, only the following cases are possible:

1. $0 < \beta < \alpha \Rightarrow \mu_1 > 0, \mu_2 = \mu_3 < 0, \mu_4 < 0$;
2. $\alpha < \beta < 2\alpha \Rightarrow \mu_1 > 0, \mu_2 = \mu_3 > 0, \mu_4 < 0$;
3. $\beta = \alpha \Rightarrow \mu_1 > 0, \mu_2 = \mu_3 = 0, \mu_4 < 0$.

Case 1) In this case is $(\otimes^2)P_Q^+ = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$ and $(\otimes^2)P_Q^- = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$.

Hence, $P_{guess}^{((\otimes^2)\rho,(\otimes^2)\sigma)} = \frac{1}{2n}\sum_{i=1}^{n}(1 + r_i^2 - 2r_i + 2s_i - 2s_i^2 + s_i^2) = \frac{1}{2} + 2\alpha - \beta$ and $P_{guess}^{((\otimes^2)\rho,(\otimes^2)\sigma)} - P_{guess}^{(\rho,\sigma)} = \alpha - \beta$ that is always positive, because $\alpha > 0$ and $0 < \beta < \alpha$.

Case 2) In this case is $(\otimes^2)P_Q^+ = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$ and $(\otimes^2)P_Q^- = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$.

Hence, $P_{guess}^{((\otimes^2)\rho,(\otimes^2)\sigma)} = \frac{1}{2} + \beta$ and $P_{guess}^{((\otimes^2)\rho,(\otimes^2)\sigma)} - P_{guess}^{(\rho,\sigma)} = \beta - \alpha$ that is always positive because $\alpha < \beta < 2\alpha$.

Finally, Case 3) follows the same scheme of Case 2) and it allows us to obtain that $P_{guess}^{((\otimes^2)\rho,(\otimes^2)\sigma)} = P_{guess}^{(\rho,\sigma)}$.

Supporting information

S1 File. Table A. Banana dataset (with #*n* number of copies). Table B. Prnn-synth dataset (with #*n* number of copies). Table C. Analcatdata aids dataset (with #*n* number of copies). Table D. Haberman dataset (with #*n* number of copies). Table E. Moon dataset (with #*n* number of copies). Table F. Lupus dataset (with #*n* number of copies). Table G. GaussianPlos dataset (with #*n* number of copies). Table H. Titanic dataset (with #*n* number of copies). Table I. Analcatdata-boxing1 dataset (with #*n* number of copies). Table J. Analcatdata-asbestos dataset

(with #*n* number of copies). Table K. Appendicitis dataset (with #*n* number of copies). Table L. Analcatdata-boxing2 dataset (with #*n* number of copies). Table M. Hill-Valley-with-noise dataset (with #*n* number of copies). Table N. Hill-Valley-without-noise dataset (with #*n* number of copies). Table O. Average Success Rate.
(PDF)

Acknowledgments

We are grateful to Francesco Paoli for the accurate linguistic review of the manuscript.

Author Contributions

Conceptualization: Giuseppe Sergioli, Roberto Giuntini, Hector Freytes.

Data curation: Giuseppe Sergioli, Roberto Giuntini.

Formal analysis: Giuseppe Sergioli, Roberto Giuntini, Hector Freytes.

Funding acquisition: Giuseppe Sergioli.

Investigation: Giuseppe Sergioli, Roberto Giuntini, Hector Freytes.

Methodology: Giuseppe Sergioli, Roberto Giuntini, Hector Freytes.

Project administration: Giuseppe Sergioli.

Software: Roberto Giuntini.

Supervision: Hector Freytes.

Writing – original draft: Hector Freytes.

Writing – review & editing: Giuseppe Sergioli.

References

1. Caraiman S, Manta V. Image processing using quantum computing. In IEEE 16th international conference on system theory, control and computing (ICSTCC), pp. 1-6, 2012.
2. Trugenberg CA. Quantum pattern recognition. Quantum Information Processing, Vol.1, N.6, pp. 471–493, 2002. https://doi.org/10.1023/A:1024022632303
3. Wiebe N, Kapoor A, Svore KM. Quantum nearest-neighbor algorithms for machine learning. Quantum Information and Computation, Vol. 15, N. 34, pp.318–358, 2015.
4. Wittek P. Quantum Machine Learning: What Quantum Computing Means to Data Mining. Academic Press, 2014.
5. Chefles A. Quantum state discriminator. Contemp Phys Vol. 41, N. 6, pp.401–424, 2000. https://doi.org/10.1080/00107510010002599
6. Hayashi A, Horibe M, Hashimoto T. Quantum pure-state identification. Phyys Rev A Vol.72, N. 5:052306, 2005. https://doi.org/10.1103/PhysRevA.72.052306
7. Helstrom CW. *Quantum Detection and Estimation Theory*, Academic Press, 1976.
8. Lu S, Braunstein SL. Quantum decision tree classifier. Quantum Information Processing Vol. 13, N. 3, pp. 757–770, 2014. https://doi.org/10.1007/s11128-013-0687-5
9. Eldar YC, Oppenheim AV. Quantum signal processing. IEEE Signal Processing Magazine, 19(6):12–32, 2002. https://doi.org/10.1109/MSP.2002.1043298
10. Schuld M, Sinayskiy I, Petruccione F. An introduction to quantum machine learning. Contemporary Physics, 56(2):172–185, 2014. https://doi.org/10.1080/00107514.2014.964942
11. Schuld M, Sinayskiy I., Petruccione F. The quest for a Quantum Neural Network. Quantum Information Processing, 13(11):2567–2586, 2014. https://doi.org/10.1007/s11128-014-0809-8
12. Manju A, Nigam MJ. Applications of quantum inspired computational intelligence: a survey. Artificial Intelligence Review, 42(1):79–156, 2014. https://doi.org/10.1007/s10462-012-9330-6

13. Lloyd S, Mohseni M, Rebentrost P. Quantum algorithms for supervised and unsupervised machine learning. arXiv:1307.0411, 2013.
14. Lloyd S, Mohseni M, Rebentrost P. Quantum principal component analysis. Nature Physics, 10 (9):631–633, 2014. https://doi.org/10.1038/nphys3029
15. Duda RO, Hart PE, Stork DG. Pattern Classification. Wiley Interscience, 2nd edition, 2000.
16. Santucci E. Quantum Minimum Distance Classifier. Entropy, 19(12):659, 2017. https://doi.org/10.3390/e19120659
17. Santucci E, Sergioli G. Classification problem in a quantum framework. In Quantum Foundations, Probability and Information, Proceedings of the Quantum and Beyond Conference, Vaxjo, Sweden, 13-16 June 2016; Khrennikov A., Bourama T.; Springer: Berlin, Germany 2018, in press, 2018.
18. Sergioli G, Santucci E, Didaci L, Miszczak JA, Giuntini R. A quantum inspired version of the NMC classifier. Soft Computing, 22(3), 691–705, 2018. https://doi.org/10.1007/s00500-016-2478-2
19. Sergioli G, Russo G, Santucci E, Stefano A, Torrisi SE, Palmucci S, et al. Quantum-inspired minimum distance classification in biomedical context, International Journal of Quantum Information (to appear).
20. Webb A.R., Copsey KD. Statistical Pattern Recognition, J. Wiley & Sons, 2011.
21. Sergioli G, Bosyk GM, Santucci E, Giuntini R. A quantum-inspired version of the classification problem. International Journal of Theoretical Physics, Vol 56, N. 12, pp. 3880–3888, 2017. https://doi.org/10.1007/s10773-017-3371-1
22. Aïmeur E, Brassard G, Gambs S. Machine learning in a quantum world. Conference of the Canadian Society for Computational Studies of Intelligence, Springer Berlin Heidelberg, 2006.
23. Gambs S. Quantum classification, arXiv:0809.0444v2 [quant-ph], 2008.
24. Bergou J, Herzog U, Hillery M. Discrimination of quantum states. Lectures Notes in Physics, Vol 649, pp. 417–465, Springer-Berlin, 2004. https://doi.org/10.1007/978-3-540-44481-7_11
25. Nielsen MA, Chuang IL. Quantum Computation and Quantum Information—10th Anniversary Edition. Cambridge University Press, Cambridge, 2010.
26. Qiu D. Minimum-error discrimination between mixed states. arXiv:0707.3970[quant-phis], 2007.
27. Guta M, Kotlowski W. Quantum learning: asymptotically optimal classification of qubit states. New Journal of Physics, 12-123032, 2010.
28. Audenaert KMR, Calsamiglia J, Munoz-Tapia R, Bagan E, Masanes LI, Acin A, et al. Discriminating States. The quantum Chernof Bound, Phys. Rev. Lett. 98, 160501, 2007.
29. Olson RS, La Cava W, Orzechowski P, Urbanowicz RJ, Moore JH. PMLB: a large benchmark suite machine learning evaluation and comparison. BioData Mining, 10: 1–13, 2017.

Tables

Classifier	Accuracy	Sensitivity	Specificity	Balanced Accuracy	Precision	F-measure	Cohen's k	Helstrom Error
BernoulliNB	0.57 ± 0.02	0.59 ± 0.02	0.55 ± 0.03	0.57 ± 0.02	0.55 ± 0.02	0.52 ± 0.02	0.14 ± 0.04	
LogisticRegression	0.56 ± 0.02	0.13 ± 0.02	0.91 ± 0.04	0.52 ± 0.02	0.2 ± 0.02	0.55 ± 0.13	0.04 ± 0.05	
GaussianNB	0.61 ± 0.01	0.31 ± 0.03	0.86 ± 0.02	0.59 ± 0.01	0.42 ± 0.03	0.64 ± 0.03	0.18 ± 0.03	
KNeighborsClassifier	0.89 ± 0.01	0.86 ± 0.02	0.91 ± 0.02	0.88 ± 0.01	0.87 ± 0.02	0.88 ± 0.02	0.77 ± 0.03	
RandomForestClassifier	0.88 ± 0.01	0.88 ± 0.02	0.89 ± 0.02	0.88 ± 0.01	0.87 ± 0.01	0.86 ± 0.02	0.76 ± 0.02	
AdaBoostClassifier	0.72 ± 0.01	0.64 ± 0.03	0.79 ± 0.03	0.72 ± 0.01	0.67 ± 0.02	0.72 ± 0.02	0.44 ± 0.03	
NearestCentroid	0.55 ± 0.02	0.58 ± 0.03	0.53 ± 0.03	0.55 ± 0.02	0.54 ± 0.02	0.5 ± 0.02	0.11 ± 0.04	
LinearDiscriminantAnalysis	0.56 ± 0.02	0.13 ± 0.02	0.91 ± 0.04	0.52 ± 0.02	0.2 ± 0.02	0.55 ± 0.13	0.04 ± 0.05	
QuadraticDiscriminantAnalysis	0.62 ± 0.02	0.35 ± 0.03	0.84 ± 0.03	0.59 ± 0.02	0.45 ± 0.03	0.64 ± 0.05	0.2 ± 0.05	
ExtraTreesClassifier	0.88 ± 0.01	0.87 ± 0.02	0.89 ± 0.01	0.88 ± 0.01	0.87 ± 0.01	0.87 ± 0.01	0.77 ± 0.02	
GradientBoostingClassifier	0.9 ± 0.01	0.85 ± 0.02	0.94 ± 0.01	0.89 ± 0.01	0.88 ± 0.01	0.92 ± 0.01	0.79 ± 0.02	
QuantumNearestMeanCentroid	0.7 ± 0.01	0.7 ± 0.03	0.7 ± 0.02	0.7 ± 0.02	0.65 ± 0.02	0.68 ± 0.02	0.4 ± 0.03	
HelstromQuantumCentroid1	0.7 ± 0.02	0.81 ± 0.03	0.61 ± 0.02	0.71 ± 0.02	0.63 ± 0.02	0.71 ± 0.02	0.42 ± 0.04	0.36 ± 0.0
HelstromQuantumCentroid2	0.78 ± 0.01	0.56 ± 0.03	0.96 ± 0.02	0.76 ± 0.02	0.92 ± 0.03	0.69 ± 0.02	0.54 ± 0.03	0.33 ± 0.0
HelstromQuantumCentroid3	0.8 ± 0.01	0.76 ± 0.02	0.84 ± 0.02	0.8 ± 0.01	0.79 ± 0.02	0.78 ± 0.01	0.6 ± 0.01	0.3 ± 0.0
HelstromQuantumCentroid4	0.83 ± 0.01	0.81 ± 0.02	0.85 ± 0.02	0.83 ± 0.01	0.81 ± 0.03	0.81 ± 0.02	0.66 ± 0.02	0.27 ± 0.0

Table 1. Banana dataset (with $\#n$ number of copies)

Classifier	Accuracy	Sensitivity	Specificity	Balanced Accuracy	Precision	F-measure	Cohen's k	Helstrom Error
BernoulliNB	0.48 ± 0.1	0.57 ± 0.24	0.4 ± 0.23	0.48 ± 0.1	0.51 ± 0.13	0.49 ± 0.11	-0.03 ± 0.2	
LogisticRegression	0.85 ± 0.06	0.9 ± 0.07	0.79 ± 0.14	0.85 ± 0.06	0.86 ± 0.05	0.83 ± 0.1	0.69 ± 0.12	
GaussianNB	0.83 ± 0.07	0.86 ± 0.1	0.81 ± 0.14	0.83 ± 0.07	0.84 ± 0.06	0.83 ± 0.1	0.67 ± 0.13	
KNeighborsClassifier	0.82 ± 0.08	0.81 ± 0.12	0.83 ± 0.12	0.82 ± 0.08	0.82 ± 0.08	0.84 ± 0.11	0.64 ± 0.16	
RandomForestClassifier	0.82 ± 0.04	0.81 ± 0.1	0.83 ± 0.12	0.82 ± 0.04	0.81 ± 0.04	0.84 ± 0.1	0.63 ± 0.09	
AdaBoostClassifier	0.84 ± 0.07	0.88 ± 0.08	0.81 ± 0.13	0.84 ± 0.07	0.85 ± 0.06	0.83 ± 0.1	0.69 ± 0.13	
NearestCentroid	0.71 ± 0.11	0.78 ± 0.1	0.63 ± 0.19	0.71 ± 0.11	0.73 ± 0.09	0.7 ± 0.12	0.42 ± 0.21	
LinearDiscriminantAnalysis	0.84 ± 0.06	0.86 ± 0.1	0.82 ± 0.12	0.84 ± 0.06	0.84 ± 0.06	0.84 ± 0.09	0.68 ± 0.12	
QuadraticDiscriminantAnalysis	0.85 ± 0.06	0.88 ± 0.09	0.82 ± 0.12	0.85 ± 0.06	0.85 ± 0.06	0.84 ± 0.09	0.69 ± 0.12	
ExtraTreesClassifier	0.84 ± 0.04	0.86 ± 0.08	0.82 ± 0.12	0.84 ± 0.04	0.84 ± 0.04	0.84 ± 0.09	0.67 ± 0.08	
GradientBoostingClassifier	0.83 ± 0.06	0.83 ± 0.06	0.83 ± 0.12	0.83 ± 0.06	0.83 ± 0.05	0.84 ± 0.1	0.66 ± 0.11	
QuantumNearestMeanCentroid	0.85 ± 0.08	0.9 ± 0.06	0.8 ± 0.15	0.85 ± 0.08	0.84 ± 0.12	0.86 ± 0.07	0.7 ± 0.16	
HelstromQuantumCentroid1	0.84 ± 0.07	0.84 ± 0.1	0.85 ± 0.11	0.84 ± 0.08	0.86 ± 0.08	0.84 ± 0.07	0.68 ± 0.15	0.28 ± 0.01
HelstromQuantumCentroid2	0.88 ± 0.06	0.91 ± 0.07	0.85 ± 0.07	0.88 ± 0.06	0.86 ± 0.07	0.89 ± 0.06	0.76 ± 0.11	0.22 ± 0.01
HelstromQuantumCentroid3	0.89 ± 0.06	0.92 ± 0.07	0.86 ± 0.07	0.89 ± 0.06	0.87 ± 0.07	0.89 ± 0.05	0.77 ± 0.12	0.2 ± 0.0
HelstromQuantumCentroid4	0.88 ± 0.06	0.9 ± 0.07	0.87 ± 0.07	0.89 ± 0.06	0.87 ± 0.09	0.88 ± 0.07	0.77 ± 0.13	0.18 ± 0.0

Table 2. Prnn_synth dataset (with $\#n$ number of copies)

Classifier	Accuracy	Sensitivity	Specificity	Balanced Accuracy	Precision	F-measure	Cohen's k	Helstrom Error
BernoulliNB	0.56 ± 0.06	0.48 ± 0.1	0.64 ± 0.03	0.56 ± 0.06	0.52 ± 0.09	0.57 ± 0.07	0.12 ± 0.12	
LogisticRegression	0.52 ± 0.06	0.4 ± 0.02	0.64 ± 0.11	0.52 ± 0.06	0.46 ± 0.04	0.54 ± 0.09	0.04 ± 0.12	
GaussianNB	0.5 ± 0.04	0.32 ± 0.01	0.68 ± 0.07	0.5 ± 0.04	0.39 ± 0.03	0.51 ± 0.06	0.0 ± 0.08	
KNeighborsClassifier	0.54 ± 0.0	0.4 ± 0.02	0.68 ± 0.01	0.54 ± 0.0	0.47 ± 0.01	0.56 ± 0.0	0.08 ± 0.0	
RandomForestClassifier	0.46 ± 0.0	0.56 ± 0.06	0.36 ± 0.05	0.46 ± 0.0	0.51 ± 0.03	0.47 ± 0.0	-0.08 ± 0.0	
AdaBoostClassifier	0.52 ± 0.02	0.4 ± 0.02	0.64 ± 0.03	0.52 ± 0.02	0.46 ± 0.02	0.53 ± 0.03	0.04 ± 0.04	
NearestCentroid	0.6 ± 0.02	0.4 ± 0.02	0.8 ± 0.05	0.6 ± 0.02	0.5 ± 0.0	0.67 ± 0.04	0.2 ± 0.03	
LinearDiscriminantAnalysis	0.52 ± 0.06	0.36 ± 0.03	0.69 ± 0.15	0.52 ± 0.06	0.43 ± 0.01	0.56 ± 0.11	0.04 ± 0.12	
QuadraticDiscriminantAnalysis	0.56 ± 0.02	0.36 ± 0.03	0.76 ± 0.07	0.56 ± 0.02	0.45 ± 0.01	0.61 ± 0.06	0.12 ± 0.04	
ExtraTreesClassifier	0.44 ± 0.02	0.44 ± 0.06	0.44 ± 0.02	0.44 ± 0.02	0.44 ± 0.04	0.44 ± 0.02	-0.12 ± 0.04	
GradientBoostingClassifier	0.46 ± 0.04	0.32 ± 0.01	0.6 ± 0.1	0.46 ± 0.04	0.37 ± 0.01	0.45 ± 0.05	-0.08 ± 0.08	
QuantumNearestMeanCentroid	0.5 ± 0.02	0.52 ± 0.06	0.48 ± 0.02	0.5 ± 0.02	0.5 ± 0.0	0.51 ± 0.03	0.0 ± 0.04	
HelstromQuantumCentroid1	0.68 ± 0.0	0.46 ± 0.35	0.72 ± 0.28	0.59 ± 0.04	0.86 ± 0.14	0.48 ± 0.28	0.2 ± 0.07	0.38 ± 0.02
HelstromQuantumCentroid2	0.44 ± 0.0	0.44 ± 0.1	0.44 ± 0.1	0.44 ± 0.0	0.43 ± 0.03	0.43 ± 0.07	-0.13 ± 0.0	0.34 ± 0.01
HelstromQuantumCentroid3	0.52 ± 0.04	0.53 ± 0.1	0.52 ± 0.02	0.53 ± 0.04	0.52 ± 0.02	0.52 ± 0.04	0.05 ± 0.08	0.22 ± 0.0
HelstromQuantumCentroid4	0.6 ± 0.04	0.57 ± 0.07	0.66 ± 0.16	0.61 ± 0.05	0.64 ± 0.14	0.58 ± 0.02	0.22 ± 0.09	0.24 ± 0.03

Table 3. Analcatdata_aids dataset (with $\#n$ number of copies)

Classifier	Accuracy	Sensitivity	Specificity	Balanced Accuracy	Precision	F-measure	Cohen's k	Helstrom Error
BernoulliNB	0.74 ± 0.0	1.0 ± 0.0	0.0 ± 0.0	0.5 ± 0.0	0.85 ± 0.0	0.74 ± 0.0	0.0 ± 0.0	
LogisticRegression	0.74 ± 0.01	0.95 ± 0.03	0.15 ± 0.05	0.55 ± 0.01	0.84 ± 0.01	0.76 ± 0.01	0.13 ± 0.02	
GaussianNB	0.74 ± 0.01	0.94 ± 0.02	0.16 ± 0.01	0.55 ± 0.0	0.84 ± 0.01	0.76 ± 0.0	0.13 ± 0.01	
KNeighborsClassifier	0.7 ± 0.03	0.88 ± 0.01	0.21 ± 0.14	0.54 ± 0.06	0.81 ± 0.01	0.76 ± 0.03	0.09 ± 0.14	
RandomForestClassifier	0.7 ± 0.02	0.89 ± 0.04	0.17 ± 0.05	0.53 ± 0.01	0.81 ± 0.01	0.75 ± 0.0	0.07 ± 0.01	
AdaBoostClassifier	0.73 ± 0.02	0.88 ± 0.01	0.3 ± 0.1	0.59 ± 0.04	0.83 ± 0.01	0.78 ± 0.02	0.2 ± 0.09	
NearestCentroid	0.71 ± 0.01	0.79 ± 0.03	0.48 ± 0.03	0.63 ± 0.0	0.8 ± 0.01	0.81 ± 0.0	0.26 ± 0.0	
LinearDiscriminantAnalysis	0.74 ± 0.01	0.95 ± 0.03	0.16 ± 0.04	0.56 ± 0.0	0.84 ± 0.01	0.76 ± 0.0	0.14 ± 0.0	
QuadraticDiscriminantAnalysis	0.75 ± 0.01	0.92 ± 0.01	0.26 ± 0.09	0.59 ± 0.04	0.84 ± 0.01	0.78 ± 0.02	0.22 ± 0.08	
ExtraTreesClassifier	0.71 ± 0.01	0.89 ± 0.04	0.2 ± 0.08	0.55 ± 0.02	0.82 ± 0.01	0.76 ± 0.01	0.11 ± 0.04	
GradientBoostingClassifier	0.7 ± 0.01	0.84 ± 0.04	0.33 ± 0.07	0.58 ± 0.01	0.8 ± 0.01	0.78 ± 0.01	0.18 ± 0.02	
QuantumNearestMeanCentroid	0.61 ± 0.07	0.65 ± 0.13	0.51 ± 0.12	0.58 ± 0.01	0.79 ± 0.01	0.71 ± 0.08	0.14 ± 0.03	
HelstromQuantumCentroid1	0.63 ± 0.06	0.61 ± 0.1	0.69 ± 0.02	0.65 ± 0.04	0.84 ± 0.03	0.7 ± 0.08	0.24 ± 0.06	0.42 ± 0.01
HelstromQuantumCentroid2	0.66 ± 0.02	0.71 ± 0.03	0.53 ± 0.01	0.62 ± 0.02	0.81 ± 0.02	0.76 ± 0.02	0.22 ± 0.03	0.36 ± 0.01
HelstromQuantumCentroid3	0.69 ± 0.01	0.76 ± 0.0	0.48 ± 0.04	0.62 ± 0.02	0.8 ± 0.05	0.78 ± 0.03	0.22 ± 0.05	0.31 ± 0.01
HelstromQuantumCentroid4	0.57 ± 0.0	0.6 ± 0.03	0.46 ± 0.07	0.53 ± 0.02	0.76 ± 0.0	0.67 ± 0.02	0.05 ± 0.04	0.27 ± 0.01

Table 4. Haberman dataset (with $\#n$ number of copies)

Classifier	Accuracy	Sensitivity	Specificity	Balanced Accuracy	Precision	F-measure	Cohen's k	Helstrom Error
BernoulliNB	0.82 ± 0.05	0.66 ± 0.12	0.99 ± 0.01	0.82 ± 0.05	0.78 ± 0.08	0.99 ± 0.01	0.65 ± 0.11	
LogisticRegression	0.86 ± 0.02	0.85 ± 0.09	0.87 ± 0.05	0.86 ± 0.02	0.86 ± 0.03	0.87 ± 0.03	0.72 ± 0.04	
GaussianNB	0.88 ± 0.02	0.88 ± 0.08	0.88 ± 0.04	0.88 ± 0.02	0.88 ± 0.03	0.88 ± 0.03	0.76 ± 0.04	
KNeighborsClassifier	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	
RandomForestClassifier	0.96 ± 0.0	0.97 ± 0.03	0.95 ± 0.03	0.96 ± 0.0	0.96 ± 0.0	0.95 ± 0.03	0.92 ± 0.0	
AdaBoostClassifier	0.97 ± 0.03	0.97 ± 0.03	0.97 ± 0.03	0.97 ± 0.03	0.97 ± 0.03	0.97 ± 0.03	0.94 ± 0.06	
NearestCentroid	0.78 ± 0.03	0.79 ± 0.11	0.78 ± 0.06	0.78 ± 0.03	0.78 ± 0.04	0.79 ± 0.02	0.57 ± 0.05	
LinearDiscriminantAnalysis	0.88 ± 0.02	0.88 ± 0.08	0.88 ± 0.04	0.88 ± 0.02	0.88 ± 0.03	0.88 ± 0.03	0.76 ± 0.04	
QuadraticDiscriminantAnalysis	0.88 ± 0.03	0.87 ± 0.09	0.88 ± 0.04	0.88 ± 0.02	0.87 ± 0.03	0.88 ± 0.02	0.75 ± 0.05	
ExtraTreesClassifier	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	
GradientBoostingClassifier	0.96 ± 0.02	0.97 ± 0.03	0.95 ± 0.01	0.96 ± 0.02	0.96 ± 0.02	0.95 ± 0.01	0.92 ± 0.04	
QuantumNearestMeanCentroid	0.89 ± 0.0	0.88 ± 0.1	0.9 ± 0.1	0.89 ± 0.0	0.92 ± 0.08	0.89 ± 0.01	0.78 ± 0.0	
HelstromQuantumCentroid1	0.86 ± 0.04	0.74 ± 0.04	0.99 ± 0.01	0.87 ± 0.03	0.99 ± 0.01	0.85 ± 0.03	0.73 ± 0.07	0.24 ± 0.01
HelstromQuantumCentroid2	0.94 ± 0.0	0.97 ± 0.01	0.9 ± 0.01	0.94 ± 0.0	0.9 ± 0.02	0.94 ± 0.01	0.87 ± 0.01	0.2 ± 0.01
HelstromQuantumCentroid3	0.98 ± 0.02	1.0 ± 0.0	0.96 ± 0.04	0.98 ± 0.02	0.96 ± 0.04	0.98 ± 0.02	0.96 ± 0.04	0.09 ± 0.01
HelstromQuantumCentroid4	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	0.06 ± 0.0

Table 5. Moon dataset (with $\#n$ number of copies)

Classifier	Accuracy	Sensitivity	Specificity	Balanced Accuracy	Precision	F-measure	Cohen's k	Helstrom Error
BernoulliNB	0.6 ± 0.01	0.0 ± 0.0	1.0 ± 0.0	0.5 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	
LogisticRegression	0.7 ± 0.07	0.58 ± 0.13	0.79 ± 0.02	0.68 ± 0.07	0.6 ± 0.1	0.64 ± 0.07	0.37 ± 0.15	
GaussianNB	0.67 ± 0.08	0.55 ± 0.27	0.75 ± 0.06	0.65 ± 0.11	0.54 ± 0.18	0.57 ± 0.07	0.29 ± 0.2	
KNeighborsClassifier	0.72 ± 0.05	0.54 ± 0.01	0.85 ± 0.08	0.69 ± 0.04	0.61 ± 0.05	0.72 ± 0.12	0.4 ± 0.1	
RandomForestClassifier	0.68 ± 0.0	0.66 ± 0.05	0.69 ± 0.04	0.68 ± 0.0	0.62 ± 0.01	0.59 ± 0.02	0.34 ± 0.0	
AdaBoostClassifier	0.7 ± 0.0	0.57 ± 0.07	0.79 ± 0.06	0.68 ± 0.01	0.6 ± 0.02	0.65 ± 0.04	0.37 ± 0.01	
NearestCentroid	0.72 ± 0.03	0.75 ± 0.14	0.71 ± 0.13	0.73 ± 0.0	0.68 ± 0.02	0.66 ± 0.08	0.45 ± 0.02	
LinearDiscriminantAnalysis	0.75 ± 0.03	0.66 ± 0.1	0.81 ± 0.12	0.73 ± 0.01	0.68 ± 0.01	0.73 ± 0.11	0.47 ± 0.03	
QuadraticDiscriminantAnalysis	0.64 ± 0.1	0.47 ± 0.3	0.77 ± 0.04	0.62 ± 0.13	0.47 ± 0.24	0.51 ± 0.14	0.23 ± 0.25	
ExtraTreesClassifier	0.66 ± 0.0	0.66 ± 0.1	0.65 ± 0.08	0.66 ± 0.01	0.6 ± 0.03	0.56 ± 0.02	0.3 ± 0.02	
GradientBoostingClassifier	0.71 ± 0.04	0.54 ± 0.04	0.83 ± 0.1	0.69 ± 0.03	0.6 ± 0.02	0.7 ± 0.11	0.38 ± 0.07	
QuantumNearestMeanCentroid	0.72 ± 0.02	0.4 ± 0.01	0.94 ± 0.06	0.67 ± 0.02	0.85 ± 0.15	0.54 ± 0.02	0.38 ± 0.05	
HelstromQuantumCentroid1	0.69 ± 0.03	0.23 ± 0.17	0.98 ± 0.02	0.61 ± 0.09	0.75 ± 0.25	0.34 ± 0.23	0.23 ± 0.19	0.42 ± 0.03
HelstromQuantumCentroid2	0.7 ± 0.12	0.32 ± 0.22	0.98 ± 0.02	0.65 ± 0.1	0.94 ± 0.06	0.42 ± 0.24	0.33 ± 0.22	0.38 ± 0.04
HelstromQuantumCentroid3	0.71 ± 0.06	0.32 ± 0.09	0.98 ± 0.02	0.65 ± 0.06	0.9 ± 0.1	0.47 ± 0.12	0.33 ± 0.13	0.37 ± 0.05
HelstromQuantumCentroid4	0.71 ± 0.01	0.3 ± 0.1	0.98 ± 0.02	0.64 ± 0.06	0.88 ± 0.12	0.44 ± 0.13	0.31 ± 0.11	0.37 ± 0.03

Table 6. Lupus dataset (with $\#n$ number of copies)

Classifier	Accuracy	Sensitivity	Specificity	Balanced Accuracy	Precision	F-measure	Cohen's k	Helstrom Error
BernoulliNB	0.52 ± 0.01	0.34 ± 0.3	0.7 ± 0.3	0.52 ± 0.01	0.33 ± 0.24	0.76 ± 0.24	0.04 ± 0.01	
LogisticRegression	0.53 ± 0.01	0.53 ± 0.01	0.54 ± 0.03	0.53 ± 0.01	0.53 ± 0.0	0.53 ± 0.01	0.06 ± 0.01	
GaussianNB	0.77 ± 0.02	0.66 ± 0.07	0.88 ± 0.03	0.77 ± 0.02	0.74 ± 0.04	0.84 ± 0.01	0.54 ± 0.05	
KNeighborsClassifier	0.72 ± 0.04	0.61 ± 0.03	0.84 ± 0.04	0.72 ± 0.04	0.69 ± 0.04	0.79 ± 0.05	0.44 ± 0.08	
RandomForestClassifier	0.69 ± 0.01	0.68 ± 0.01	0.7 ± 0.02	0.69 ± 0.01	0.69 ± 0.02	0.7 ± 0.02	0.38 ± 0.03	
AdaBoostClassifier	0.7 ± 0.04	0.65 ± 0.07	0.76 ± 0.02	0.7 ± 0.04	0.68 ± 0.05	0.72 ± 0.03	0.4 ± 0.09	
NearestCentroid	0.52 ± 0.04	0.5 ± 0.03	0.55 ± 0.05	0.52 ± 0.04	0.51 ± 0.03	0.53 ± 0.04	0.05 ± 0.08	
LinearDiscriminantAnalysis	0.54 ± 0.01	0.52 ± 0.01	0.56 ± 0.03	0.54 ± 0.01	0.53 ± 0.0	0.54 ± 0.01	0.09 ± 0.02	
QuadraticDiscriminantAnalysis	0.78 ± 0.02	0.66 ± 0.06	0.89 ± 0.02	0.77 ± 0.02	0.74 ± 0.03	0.86 ± 0.01	0.55 ± 0.04	
ExtraTreesClassifier	0.68 ± 0.02	0.69 ± 0.04	0.68 ± 0.0	0.68 ± 0.02	0.69 ± 0.03	0.68 ± 0.01	0.37 ± 0.04	
GradientBoostingClassifier	0.7 ± 0.02	0.66 ± 0.03	0.74 ± 0.01	0.7 ± 0.02	0.69 ± 0.03	0.72 ± 0.01	0.4 ± 0.04	
QuantumNearestMeanCentroid	0.74 ± 0.01	0.66 ± 0.02	0.81 ± 0.03	0.74 ± 0.01	0.78 ± 0.02	0.72 ± 0.0	0.47 ± 0.01	
HelstromQuantumCentroid1	0.67 ± 0.03	0.46 ± 0.13	0.89 ± 0.08	0.68 ± 0.02	0.84 ± 0.1	0.57 ± 0.09	0.35 ± 0.06	0.41 ± 0.01
HelstromQuantumCentroid2	0.76 ± 0.02	0.6 ± 0.0	0.91 ± 0.03	0.76 ± 0.02	0.87 ± 0.04	0.71 ± 0.01	0.51 ± 0.03	0.31 ± 0.01
HelstromQuantumCentroid3	0.76 ± 0.03	0.62 ± 0.03	0.89 ± 0.02	0.76 ± 0.03	0.86 ± 0.02	0.72 ± 0.03	0.52 ± 0.06	0.31 ± 0.01
HelstromQuantumCentroid4	0.76 ± 0.02	0.63 ± 0.06	0.88 ± 0.08	0.76 ± 0.01	0.86 ± 0.07	0.72 ± 0.01	0.52 ± 0.03	0.29 ± 0.0

Table 7. GaussianPlos dataset (with $\#n$ number of copies)

Classifier	Accuracy	Sensitivity	Specificity	Balanced Accuracy	Precision	F-measure	Cohen's k	Helstrom Error
BernoulliNB	0.77 ± 0.01	0.46 ± 0.0	0.92 ± 0.02	0.69 ± 0.01	0.57 ± 0.01	0.74 ± 0.05	0.43 ± 0.02	
LogisticRegression	0.78 ± 0.02	0.49 ± 0.02	0.92 ± 0.01	0.7 ± 0.02	0.59 ± 0.03	0.73 ± 0.04	0.44 ± 0.04	
GaussianNB	0.77 ± 0.01	0.52 ± 0.0	0.89 ± 0.01	0.71 ± 0.01	0.6 ± 0.01	0.7 ± 0.02	0.45 ± 0.02	
KNeighborsClassifier	0.78 ± 0.0	0.44 ± 0.06	0.95 ± 0.03	0.69 ± 0.01	0.57 ± 0.03	0.81 ± 0.07	0.44 ± 0.02	
RandomForestClassifier	0.78 ± 0.01	0.44 ± 0.06	0.95 ± 0.04	0.69 ± 0.01	0.56 ± 0.02	0.83 ± 0.12	0.44 ± 0.01	
AdaBoostClassifier	0.78 ± 0.01	0.44 ± 0.06	0.95 ± 0.04	0.69 ± 0.01	0.56 ± 0.02	0.83 ± 0.12	0.44 ± 0.01	
NearestCentroid	0.75 ± 0.01	0.59 ± 0.01	0.82 ± 0.0	0.71 ± 0.01	0.6 ± 0.01	0.62 ± 0.01	0.42 ± 0.01	
LinearDiscriminantAnalysis	0.78 ± 0.02	0.49 ± 0.02	0.92 ± 0.01	0.7 ± 0.02	0.59 ± 0.03	0.73 ± 0.04	0.44 ± 0.04	
QuadraticDiscriminantAnalysis	0.77 ± 0.01	0.52 ± 0.0	0.89 ± 0.01	0.71 ± 0.01	0.6 ± 0.01	0.7 ± 0.02	0.45 ± 0.02	
ExtraTreesClassifier	0.78 ± 0.01	0.44 ± 0.06	0.95 ± 0.04	0.69 ± 0.01	0.56 ± 0.02	0.83 ± 0.12	0.44 ± 0.01	
GradientBoostingClassifier	0.78 ± 0.01	0.44 ± 0.06	0.95 ± 0.04	0.69 ± 0.01	0.56 ± 0.02	0.83 ± 0.12	0.44 ± 0.01	
QuantumNearestMeanCentroid	0.68 ± 0.08	0.66 ± 0.04	0.68 ± 0.13	0.67 ± 0.04	0.52 ± 0.08	0.57 ± 0.03	0.32 ± 0.1	
HelstromQuantumCentroid1	0.75 ± 0.0	0.6 ± 0.01	0.81 ± 0.0	0.71 ± 0.0	0.61 ± 0.0	0.61 ± 0.0	0.42 ± 0.0	0.35 ± 0.0
HelstromQuantumCentroid2	0.75 ± 0.01	0.6 ± 0.03	0.81 ± 0.01	0.71 ± 0.01	0.61 ± 0.0	0.61 ± 0.02	0.42 ± 0.02	0.3 ± 0.01
HelstromQuantumCentroid3	0.76 ± 0.03	0.56 ± 0.03	0.85 ± 0.06	0.71 ± 0.01	0.66 ± 0.07	0.6 ± 0.01	0.43 ± 0.04	0.34 ± 0.01
HelstromQuantumCentroid4	0.76 ± 0.02	0.56 ± 0.03	0.85 ± 0.04	0.71 ± 0.0	0.65 ± 0.05	0.6 ± 0.0	0.43 ± 0.02	0.33 ± 0.0

Table 8. Titanic dataset (with $\#n$ number of copies)

Classifier	Accuracy	Sensitivity	Specificity	Balanced Accuracy	Precision	F-measure	Cohen's k	Helstrom Error
BernoulliNB	0.63 ± 0.02	0.9 ± 0.1	0.14 ± 0.14	0.52 ± 0.02	0.76 ± 0.03	0.66 ± 0.01	0.04 ± 0.04	
LogisticRegression	0.63 ± 0.0	0.9 ± 0.03	0.14 ± 0.05	0.52 ± 0.01	0.76 ± 0.01	0.66 ± 0.01	0.05 ± 0.02	
GaussianNB	0.69 ± 0.06	0.87 ± 0.05	0.36 ± 0.07	0.61 ± 0.06	0.79 ± 0.04	0.72 ± 0.03	0.25 ± 0.14	
KNeighborsClassifier	0.66 ± 0.06	0.78 ± 0.09	0.43 ± 0.0	0.61 ± 0.04	0.75 ± 0.05	0.72 ± 0.02	0.22 ± 0.1	
RandomForestClassifier	0.66 ± 0.01	0.78 ± 0.01	0.43 ± 0.0	0.61 ± 0.01	0.75 ± 0.01	0.72 ± 0.0	0.22 ± 0.01	
AdaBoostClassifier	0.77 ± 0.02	0.85 ± 0.05	0.62 ± 0.05	0.73 ± 0.0	0.82 ± 0.02	0.81 ± 0.01	0.48 ± 0.02	
NearestCentroid	0.54 ± 0.01	0.54 ± 0.0	0.55 ± 0.02	0.54 ± 0.01	0.6 ± 0.0	0.69 ± 0.01	0.08 ± 0.02	
LinearDiscriminantAnalysis	0.63 ± 0.0	0.82 ± 0.03	0.29 ± 0.05	0.55 ± 0.01	0.74 ± 0.01	0.68 ± 0.01	0.12 ± 0.02	
QuadraticDiscriminantAnalysis	0.69 ± 0.04	0.88 ± 0.01	0.33 ± 0.1	0.61 ± 0.05	0.79 ± 0.02	0.71 ± 0.03	0.24 ± 0.12	
ExtraTreesClassifier	0.65 ± 0.03	0.73 ± 0.04	0.5 ± 0.02	0.62 ± 0.03	0.73 ± 0.03	0.73 ± 0.02	0.23 ± 0.06	
GradientBoostingClassifier	0.79 ± 0.02	0.86 ± 0.01	0.67 ± 0.05	0.76 ± 0.03	0.84 ± 0.02	0.83 ± 0.02	0.53 ± 0.06	
QuantumNearestMeanCentroid	0.54 ± 0.02	0.46 ± 0.02	0.68 ± 0.07	0.57 ± 0.03	0.73 ± 0.01	0.57 ± 0.01	0.12 ± 0.05	
HelstromQuantumCentroid1	0.48 ± 0.0	0.27 ± 0.05	0.88 ± 0.04	0.57 ± 0.01	0.81 ± 0.01	0.4 ± 0.06	0.11 ± 0.01	0.45 ± 0.01
HelstromQuantumCentroid2	0.54 ± 0.02	0.43 ± 0.07	0.74 ± 0.04	0.59 ± 0.01	0.75 ± 0.02	0.55 ± 0.06	0.14 ± 0.03	0.4 ± 0.0
HelstromQuantumCentroid3	0.48 ± 0.03	0.42 ± 0.09	0.6 ± 0.07	0.51 ± 0.01	0.66 ± 0.01	0.51 ± 0.07	0.02 ± 0.02	0.36 ± 0.03
HelstromQuantumCentroid4	0.54 ± 0.04	0.42 ± 0.01	0.76 ± 0.1	0.59 ± 0.05	0.77 ± 0.08	0.55 ± 0.03	0.15 ± 0.09	0.36 ± 0.02

Table 9. Analcatdata_boxing1 dataset (with $\#n$ number of copies)

Classifier	Accuracy	Sensitivity	Specificity	Balanced Accuracy	Precision	F-measure	Cohen's k	Helstrom Error
BernoulliNB	0.82 ± 0.04	0.89 ± 0.06	0.76 ± 0.02	0.83 ± 0.04	0.81 ± 0.04	0.75 ± 0.03	0.64 ± 0.08	
LogisticRegression	0.82 ± 0.04	0.84 ± 0.0	0.8 ± 0.07	0.82 ± 0.03	0.81 ± 0.04	0.78 ± 0.06	0.64 ± 0.07	
GaussianNB	0.81 ± 0.0	0.81 ± 0.03	0.8 ± 0.02	0.81 ± 0.01	0.79 ± 0.01	0.77 ± 0.01	0.61 ± 0.01	
KNeighborsClassifier	0.77 ± 0.06	0.76 ± 0.03	0.78 ± 0.09	0.77 ± 0.06	0.75 ± 0.06	0.74 ± 0.09	0.54 ± 0.12	
RandomForestClassifier	0.84 ± 0.01	0.76 ± 0.03	0.91 ± 0.0	0.83 ± 0.02	0.81 ± 0.02	0.87 ± 0.01	0.68 ± 0.03	
AdaBoostClassifier	0.8 ± 0.03	0.73 ± 0.01	0.85 ± 0.07	0.79 ± 0.03	0.76 ± 0.03	0.8 ± 0.06	0.58 ± 0.06	
NearestCentroid	0.81 ± 0.05	0.89 ± 0.06	0.74 ± 0.04	0.81 ± 0.05	0.8 ± 0.05	0.73 ± 0.05	0.62 ± 0.1	
LinearDiscriminantAnalysis	0.82 ± 0.06	0.89 ± 0.0	0.76 ± 0.11	0.83 ± 0.06	0.82 ± 0.05	0.76 ± 0.09	0.64 ± 0.12	
QuadraticDiscriminantAnalysis	0.81 ± 0.0	0.81 ± 0.03	0.8 ± 0.02	0.81 ± 0.01	0.79 ± 0.01	0.77 ± 0.01	0.61 ± 0.01	
ExtraTreesClassifier	0.86 ± 0.03	0.87 ± 0.02	0.85 ± 0.07	0.86 ± 0.02	0.84 ± 0.02	0.83 ± 0.06	0.71 ± 0.05	
GradientBoostingClassifier	0.8 ± 0.03	0.78 ± 0.01	0.8 ± 0.07	0.79 ± 0.03	0.78 ± 0.02	0.77 ± 0.05	0.59 ± 0.06	
QuantumNearestMeanCentroid	0.81 ± 0.03	0.84 ± 0.05	0.78 ± 0.01	0.81 ± 0.03	0.76 ± 0.01	0.79 ± 0.03	0.61 ± 0.05	
HelstromQuantumCentroid1	0.8 ± 0.01	0.81 ± 0.03	0.78 ± 0.05	0.8 ± 0.01	0.75 ± 0.03	0.78 ± 0.0	0.59 ± 0.02	0.29 ± 0.0
HelstromQuantumCentroid2	0.78 ± 0.07	0.71 ± 0.19	0.82 ± 0.02	0.77 ± 0.09	0.76 ± 0.06	0.73 ± 0.13	0.55 ± 0.16	0.19 ± 0.04
HelstromQuantumCentroid3	0.8 ± 0.03	0.82 ± 0.05	0.8 ± 0.1	0.81 ± 0.02	0.76 ± 0.14	0.78 ± 0.05	0.59 ± 0.07	0.18 ± 0.02
HelstromQuantumCentroid4	0.78 ± 0.05	0.75 ± 0.21	0.74 ± 0.09	0.74 ± 0.06	0.69 ± 0.1	0.71 ± 0.15	0.5 ± 0.13	0.16 ± 0.04

Table 10. Analcatdata_asbestos dataset (with $\#n$ number of copies)

Classifier	Accuracy	Sensitivity	Specificity	Balanced Accuracy	Precision	F-measure	Cohen's k	Helstrom Error
BernoulliNB	0.8 ± 0.01	0.15 ± 0.15	0.96 ± 0.04	0.56 ± 0.06	0.19 ± 0.19	0.25 ± 0.25	0.13 ± 0.13	
LogisticRegression	0.87 ± 0.04	0.47 ± 0.17	0.96 ± 0.01	0.72 ± 0.09	0.57 ± 0.17	0.74 ± 0.14	0.5 ± 0.19	
GaussianNB	0.88 ± 0.03	0.52 ± 0.12	0.96 ± 0.01	0.74 ± 0.07	0.62 ± 0.12	0.77 ± 0.1	0.55 ± 0.13	
KNeighborsClassifier	0.86 ± 0.01	0.52 ± 0.12	0.94 ± 0.01	0.73 ± 0.05	0.58 ± 0.08	0.68 ± 0.02	0.5 ± 0.09	
RandomForestClassifier	0.88 ± 0.03	0.52 ± 0.12	0.96 ± 0.01	0.74 ± 0.07	0.62 ± 0.12	0.77 ± 0.1	0.55 ± 0.13	
AdaBoostClassifier	0.88 ± 0.01	0.48 ± 0.02	0.98 ± 0.0	0.73 ± 0.01	0.61 ± 0.02	0.83 ± 0.0	0.54 ± 0.02	
NearestCentroid	0.84 ± 0.01	0.71 ± 0.01	0.87 ± 0.01	0.79 ± 0.01	0.64 ± 0.03	0.58 ± 0.04	0.54 ± 0.04	
LinearDiscriminantAnalysis	0.87 ± 0.04	0.47 ± 0.17	0.96 ± 0.01	0.72 ± 0.09	0.57 ± 0.17	0.74 ± 0.14	0.5 ± 0.19	
QuadraticDiscriminantAnalysis	0.86 ± 0.05	0.47 ± 0.17	0.95 ± 0.02	0.71 ± 0.1	0.56 ± 0.18	0.69 ± 0.19	0.48 ± 0.21	
ExtraTreesClassifier	0.85 ± 0.0	0.34 ± 0.06	0.98 ± 0.02	0.66 ± 0.02	0.46 ± 0.04	0.83 ± 0.17	0.39 ± 0.02	
GradientBoostingClassifier	0.85 ± 0.03	0.53 ± 0.07	0.93 ± 0.02	0.73 ± 0.05	0.58 ± 0.08	0.65 ± 0.1	0.49 ± 0.11	
QuantumNearestMeanCentroid	0.81 ± 0.02	0.82 ± 0.07	0.81 ± 0.04	0.82 ± 0.01	0.52 ± 0.08	0.63 ± 0.04	0.51 ± 0.04	
HelstromQuantumCentroid1	0.85 ± 0.02	0.64 ± 0.07	0.89 ± 0.02	0.77 ± 0.03	0.58 ± 0.08	0.61 ± 0.08	0.51 ± 0.06	0.25 ± 0.01
HelstromQuantumCentroid2	0.79 ± 0.02	0.76 ± 0.06	0.8 ± 0.04	0.78 ± 0.01	0.49 ± 0.01	0.59 ± 0.01	0.46 ± 0.0	0.17 ± 0.02
HelstromQuantumCentroid3	0.84 ± 0.01	0.71 ± 0.11	0.87 ± 0.01	0.79 ± 0.05	0.57 ± 0.03	0.63 ± 0.06	0.53 ± 0.06	0.14 ± 0.03
HelstromQuantumCentroid4	0.89 ± 0.04	0.81 ± 0.03	0.91 ± 0.04	0.86 ± 0.04	0.69 ± 0.15	0.73 ± 0.1	0.66 ± 0.12	0.11 ± 0.01

Table 11. Appendicitis dataset (with $\#n$ number of copies)

Classifier	Accuracy	Sensitivity	Specificity	Balanced Accuracy	Precision	F-measure	Cohen's k	Helstrom Error
BernoulliNB	0.53 ± 0.01	0.87 ± 0.13	0.13 ± 0.13	0.5 ± 0.0	0.66 ± 0.04	0.54 ± 0.0	0.0 ± 0.0	
LogisticRegression	0.65 ± 0.01	0.7 ± 0.01	0.59 ± 0.01	0.65 ± 0.01	0.69 ± 0.01	0.67 ± 0.01	0.3 ± 0.02	
GaussianNB	0.7 ± 0.03	0.77 ± 0.14	0.61 ± 0.09	0.69 ± 0.03	0.73 ± 0.06	0.7 ± 0.01	0.39 ± 0.06	
KNeighborsClassifier	0.6 ± 0.03	0.66 ± 0.12	0.53 ± 0.07	0.59 ± 0.02	0.63 ± 0.06	0.62 ± 0.0	0.19 ± 0.05	
RandomForestClassifier	0.62 ± 0.01	0.69 ± 0.0	0.54 ± 0.03	0.62 ± 0.01	0.66 ± 0.0	0.64 ± 0.01	0.23 ± 0.02	
AdaBoostClassifier	0.74 ± 0.03	0.7 ± 0.04	0.77 ± 0.03	0.74 ± 0.03	0.74 ± 0.04	0.78 ± 0.03	0.47 ± 0.07	
NearestCentroid	0.63 ± 0.0	0.65 ± 0.02	0.61 ± 0.03	0.63 ± 0.0	0.65 ± 0.01	0.66 ± 0.01	0.25 ± 0.01	
LinearDiscriminantAnalysis	0.66 ± 0.03	0.72 ± 0.06	0.59 ± 0.01	0.65 ± 0.03	0.69 ± 0.04	0.67 ± 0.01	0.31 ± 0.05	
QuadraticDiscriminantAnalysis	0.64 ± 0.05	0.72 ± 0.14	0.54 ± 0.06	0.63 ± 0.04	0.67 ± 0.07	0.64 ± 0.02	0.26 ± 0.09	
ExtraTreesClassifier	0.64 ± 0.01	0.66 ± 0.12	0.62 ± 0.11	0.64 ± 0.0	0.66 ± 0.05	0.68 ± 0.03	0.28 ± 0.01	
GradientBoostingClassifier	0.7 ± 0.04	0.68 ± 0.01	0.72 ± 0.08	0.7 ± 0.04	0.71 ± 0.03	0.74 ± 0.06	0.4 ± 0.08	
QuantumNearestMeanCentroid	0.63 ± 0.02	0.66 ± 0.09	0.61 ± 0.05	0.63 ± 0.02	0.66 ± 0.05	0.65 ± 0.02	0.26 ± 0.04	
HelstromQuantumCentroid1	0.62 ± 0.03	0.64 ± 0.06	0.61 ± 0.0	0.62 ± 0.03	0.65 ± 0.01	0.64 ± 0.02	0.24 ± 0.06	0.4 ± 0.0
HelstromQuantumCentroid2	0.61 ± 0.08	0.72 ± 0.16	0.51 ± 0.02	0.61 ± 0.09	0.62 ± 0.01	0.66 ± 0.08	0.22 ± 0.18	0.35 ± 0.04
HelstromQuantumCentroid3	0.64 ± 0.02	0.69 ± 0.19	0.6 ± 0.18	0.65 ± 0.0	0.68 ± 0.08	0.66 ± 0.06	0.29 ± 0.02	0.34 ± 0.02
HelstromQuantumCentroid4	0.61 ± 0.05	0.62 ± 0.02	0.62 ± 0.09	0.62 ± 0.06	0.65 ± 0.11	0.63 ± 0.06	0.23 ± 0.11	0.31 ± 0.03

Table 12. Analcatdata_boxing2 dataset (with $\#n$ number of copies)

Classifier	Accuracy	Sensitivity	Specificity	Balanced Accuracy	Precision	F-measure	Cohen's k	Helstrom Error
BernoulliNB	0.5 ± 0.0	0.62 ± 0.38	0.38 ± 0.38	0.5 ± 0.0	0.49 ± 0.17	0.5 ± 0.0	0.0 ± 0.0	
LogisticRegression	0.51 ± 0.01	0.23 ± 0.02	0.79 ± 0.03	0.51 ± 0.01	0.32 ± 0.01	0.52 ± 0.02	0.02 ± 0.01	
GaussianNB	0.52 ± 0.01	0.17 ± 0.03	0.86 ± 0.05	0.52 ± 0.01	0.26 ± 0.03	0.57 ± 0.06	0.03 ± 0.03	
KNeighborsClassifier	0.51 ± 0.01	0.51 ± 0.05	0.51 ± 0.03	0.51 ± 0.01	0.51 ± 0.03	0.51 ± 0.01	0.01 ± 0.02	
RandomForestClassifier	0.48 ± 0.01	0.5 ± 0.02	0.46 ± 0.0	0.48 ± 0.01	0.49 ± 0.02	0.48 ± 0.01	-0.04 ± 0.03	
AdaBoostClassifier	0.5 ± 0.0	0.4 ± 0.0	0.61 ± 0.0	0.5 ± 0.0	0.44 ± 0.0	0.5 ± 0.0	0.01 ± 0.01	
NearestCentroid	0.51 ± 0.01	0.23 ± 0.02	0.79 ± 0.03	0.51 ± 0.01	0.32 ± 0.01	0.52 ± 0.02	0.02 ± 0.01	
LinearDiscriminantAnalysis	0.51 ± 0.01	0.23 ± 0.02	0.79 ± 0.03	0.51 ± 0.01	0.32 ± 0.01	0.52 ± 0.02	0.02 ± 0.01	
QuadraticDiscriminantAnalysis	0.52 ± 0.01	0.17 ± 0.03	0.86 ± 0.05	0.52 ± 0.01	0.26 ± 0.03	0.57 ± 0.06	0.03 ± 0.03	
ExtraTreesClassifier	0.48 ± 0.01	0.48 ± 0.01	0.48 ± 0.0	0.48 ± 0.01	0.48 ± 0.01	0.48 ± 0.01	-0.04 ± 0.01	
GradientBoostingClassifier	0.51 ± 0.01	0.53 ± 0.04	0.48 ± 0.02	0.51 ± 0.01	0.52 ± 0.02	0.51 ± 0.01	0.01 ± 0.02	
QuantumNearestMeanCentroid	0.5 ± 0.01	0.51 ± 0.27	0.49 ± 0.29	0.5 ± 0.01	0.51 ± 0.01	0.47 ± 0.14	-0.0 ± 0.02	
HelstromQuantumCentroid1	0.5 ± 0.01	0.5 ± 0.4	0.5 ± 0.38	0.5 ± 0.01	0.48 ± 0.03	0.4 ± 0.24	-0.01 ± 0.01	0.5 ± 0.0
HelstromQuantumCentroid2	0.52 ± 0.0	0.17 ± 0.01	0.87 ± 0.03	0.52 ± 0.02	0.58 ± 0.09	0.26 ± 0.02	0.04 ± 0.04	0.48 ± 0.01
HelstromQuantumCentroid3	0.54 ± 0.01	0.31 ± 0.0	0.78 ± 0.04	0.54 ± 0.02	0.58 ± 0.05	0.4 ± 0.01	0.08 ± 0.03	0.48 ± 0.01
HelstromQuantumCentroid4	0.5 ± 0.01	0.28 ± 0.19	0.72 ± 0.17	0.5 ± 0.01	0.49 ± 0.03	0.32 ± 0.17	0.0 ± 0.02	0.47 ± 0.0

Table 13. Hill_Valley_with_noise dataset (with $\#n$ number of copies)

Classifier	Accuracy	Sensitivity	Specificity	Balanced Accuracy	Precision	F-measure	Cohen's k	Helstrom Error
BernoulliNB	0.5 ± 0.0	0.91 ± 0.09	0.09 ± 0.09	0.5 ± 0.0	0.65 ± 0.02	0.5 ± 0.0	-0.0 ± 0.0	
LogisticRegression	0.52 ± 0.01	0.87 ± 0.01	0.16 ± 0.02	0.51 ± 0.01	0.64 ± 0.0	0.51 ± 0.0	0.03 ± 0.01	
GaussianNB	0.51 ± 0.0	0.88 ± 0.0	0.13 ± 0.01	0.51 ± 0.0	0.65 ± 0.0	0.51 ± 0.0	0.02 ± 0.01	
KNeighborsClassifier	0.5 ± 0.03	0.5 ± 0.05	0.5 ± 0.0	0.5 ± 0.03	0.5 ± 0.04	0.5 ± 0.03	-0.01 ± 0.05	
RandomForestClassifier	0.49 ± 0.01	0.53 ± 0.0	0.45 ± 0.02	0.49 ± 0.01	0.51 ± 0.0	0.49 ± 0.01	-0.02 ± 0.02	
AdaBoostClassifier	0.49 ± 0.01	0.49 ± 0.06	0.49 ± 0.04	0.49 ± 0.01	0.49 ± 0.04	0.5 ± 0.01	-0.01 ± 0.02	
NearestCentroid	0.51 ± 0.01	0.82 ± 0.01	0.2 ± 0.03	0.51 ± 0.01	0.63 ± 0.0	0.51 ± 0.01	0.02 ± 0.02	
LinearDiscriminantAnalysis	0.52 ± 0.01	0.87 ± 0.01	0.16 ± 0.02	0.52 ± 0.01	0.65 ± 0.0	0.51 ± 0.0	0.03 ± 0.02	
QuadraticDiscriminantAnalysis	0.51 ± 0.0	0.88 ± 0.0	0.13 ± 0.01	0.51 ± 0.0	0.65 ± 0.0	0.51 ± 0.0	0.02 ± 0.01	
ExtraTreesClassifier	0.51 ± 0.01	0.53 ± 0.01	0.48 ± 0.01	0.51 ± 0.01	0.52 ± 0.01	0.51 ± 0.01	0.01 ± 0.01	
GradientBoostingClassifier	0.51 ± 0.01	0.52 ± 0.0	0.5 ± 0.02	0.51 ± 0.01	0.52 ± 0.0	0.52 ± 0.01	0.03 ± 0.01	
QuantumNearestMeanCentroid	0.53 ± 0.0	0.78 ± 0.03	0.26 ± 0.03	0.52 ± 0.0	0.52 ± 0.0	0.63 ± 0.01	0.05 ± 0.0	
HelstromQuantumCentroid1	0.52 ± 0.01	0.81 ± 0.01	0.23 ± 0.0	0.52 ± 0.0	0.52 ± 0.02	0.63 ± 0.02	0.04 ± 0.01	0.48 ± 0.0
HelstromQuantumCentroid2	0.53 ± 0.01	0.77 ± 0.02	0.27 ± 0.03	0.52 ± 0.01	0.52 ± 0.02	0.62 ± 0.02	0.04 ± 0.01	0.47 ± 0.01
HelstromQuantumCentroid3	0.53 ± 0.0	0.82 ± 0.0	0.24 ± 0.0	0.53 ± 0.0	0.52 ± 0.01	0.64 ± 0.01	0.06 ± 0.01	0.48 ± 0.0
HelstromQuantumCentroid4	0.53 ± 0.01	0.81 ± 0.01	0.24 ± 0.02	0.53 ± 0.0	0.52 ± 0.02	0.64 ± 0.01	0.05 ± 0.01	0.48 ± 0.0

Table 14. Hill_Valley_without_noise dataset (with $\#n$ number of copies)

Position	Classifier	Average Success Rate
#1	HelstromQuantumCentroid4	72.8%
#2	HelstromQuantumCentroid3	65.13%
#3	GaussianNB	58%
#4	HelstromQuantumCentroid2	57.07%
#5	HelstromQuantumCentroid1	56.6%
#5	QuadraticDiscriminantAnalysis	56.6%
#6	GradientBoostingClassifier	52.73%
#7	ExtreTreesClassifier	51.93%
#8	KNeighborsClassifier	51.47%
#9	NearestCentroid	49.13%
#10	RandomForestClassifier	45.87%
#11	QuantumNearestMeanCentroid	43.93%
#12	AdaBoostClassifier	42.93%
#13	LinearDiscriminantAnalysis	42%
#14	LogisticRegression	36.4%
#15	BernoulliNB	17.4%

Table 15. Average Success Rate