Harunobu Ishii

hishii@bu.edu

Abstract

This research aims to introduce a potential application of Quantum Computing to a real-world dataset: binary classification of wine quality. The Quantum model used in this study is based on Bayesian theorem, and its circuit incorporates widely used quantum gates such as the X-Gate, CNOT Gate, RY Gate, and CRY Gate, as well as those unique to this research, the CCRY Gate and CCCRY Gate.

Similar to classical machine learning models, the addition of attributes enhanced the model's accuracy on a test dataset. The quantum machine learning model developed for this research achieved slightly better performance than a classical machine learning model, the random forest algorithm.

Applied quantum computing

Quantum Machine Learning Model for Wine Quality Classification

1. **Introduction**
   1. Motivation

Quantum computing has been gaining attentions from diverse fields. There has been significant amount of investment for Quantum Computing and significant technological advancement correspondingly. Quantum Computing, though it is still in early stage of development, is proven to overcome complexity problems that classical computing has faced for long time, such as Grover’s searching algorithm and Shor’s factorizing. Those algorithms are substantiated evidence that Quantum Computing will supersede in some areas, such as cryptography and optimization. As such, the author would like to explore areas where quantum computing can enhance computational performance over classical computation and establish applied quantum computations to innovate the real-world.

* 1. Project Overview

Quantum computing has been attracting attention from diverse fields. Significant investment and technological advancements in Quantum Computing have been made. Although still in the early stages of development, Quantum Computing has proven to overcome complex problems that classical computing has faced for a long time, such as with Grover's search algorithm and Shor's factoring. These algorithms are substantial evidence that Quantum Computing will excel in certain areas, such as cryptography and optimization. As such, the author aims to explore areas where quantum computing can enhance computational performance over classical computation and to establish applied quantum computations that innovate in the real world.

* 1. Dataset

This research will utilize the “Wine Quality Data Set” listed on Kaggle[[1]](#footnote-1) [1], which comprises 1599 instances with 11 features and 1 label:

1 - fixed acidity

2 - volatile acidity

3 - citric acid

4 - residual sugar

5 - chlorides

6 - free sulfur dioxide

7 - total sulfur dioxide

8 - density

9 - pH

10 - sulphates

11 - alcohol

12 - quality ('good' and 'bad' based on score >5 and <5)

1. **Experimentation Plan**

We aim to build models to classify a dataset of wines into “good” and “bad” quality categories. This will involve using a classical machine learning model, specifically a random forest[[2]](#footnote-2)[2] with a maximum depth of 2, and four quantum models (QMLs). The classical model will serve as a benchmark to identify the four most significant factors for the binary classification task, using the sklearn.ensemble.RandomForestClassifier().feature\_importances\_ method.

The QMLs will be developed with varying numbers of features: two, three, and four. These features will be re-coded as either 1 (if ≥ mean) or 0 (otherwise). From these encodings, the probability of each value will be calculated and mapped onto corresponding qubits. For instance, Qubit0 will represent the likelihood of a wine having high or low alcohol content, while Qubit1 will represent high or low sulphate content. Further details are elaborated on later in the paper.

The quantum models will adopt a hybrid approach, combining classical and quantum computing. This approach includes three main components: pre-processing, training, and post-processing. For data ingestion/pre-processing and post-processing, both classical and quantum models will utilize classical computing techniques, such as sklearn.model\_selection.train\_test\_split at a 20% ratio[[3]](#footnote-3)[3] and sklearn.metrics.confusion\_matrix[[4]](#footnote-4)[4].

1. **Quantum circuit and Gates**[[5]](#footnote-5)[[6]](#footnote-6)[5][6]

A quantum circuit is a sequence that assigns likelihoods to respective qubits and manipulates superpositions via gates, effectively transforming quantum states. Numerous types of gates are commonly available; our models will utilize four standard gates and two derived variants.

X-Gate

The X Gate is employed to invert the state of a targeted qubit. Initially, Qubit0 is set to the state |0>. When the X Gate is applied to Qubit0, it transitions to the state |1>, as illustrated in the subsequent diagram.

Chart

Description automatically generated

CNOT-Gate

The CNOT Gate, denoted as CX in the Qiskit API [[7]](#footnote-7)[7], tands for Controlled NOT gate. It is used to flip the state of a target qubit (Qubit1 in this case) if the state of a control qubit (Qubit0 in this case) is |1>. As illustrated, Qubit1, which was initially in the state |0>, is flipped to the state |1> by the CNOT gate.Chart

Description automatically generated

RY-Gate

The RY-Gate rotates the state of a qubit by an angle derived from a probability value using a helper function, prob\_to\_angle[[8]](#footnote-8)[8]. In the example provided, qubit0 undergoes a rotation corresponding to 50% probability by the RY Gate. As a result, the superposition of qubit0 shifts from |0> to an equal superposition of 0.5|0> + 0.5|1>.

A picture containing chart

Description automatically generated

CRY-Gate

The CRY-Gate, indicated by the "C" for Controlled, rotates the state of a qubit by a specific angle when its controlling qubit is in the |1> state. Initially, Qubit1 is set to |1> after being flipped by the X gate. Subsequently, Qubit1 undergoes a 50% rotation by the RY Gate. Consequently, the superposition of Qubit1 becomes 0.5|0> + 0.5|1>.Chart

Description automatically generated

Derivative gates of the CRY gate, such as the CCRY gate and the CCCRY gate, are explored in depth in the subsequent section.

1. **Model**

The flow chart provided below presents a structured blueprint of the quantum machine learning model designed for binary classification of wine quality.

Diagram

Description automatically generated

The first step is data ingestion, conducted in the classical process involving data validation, cleaning, conversion, and splitting into train and test sets. Since later steps of the model require probabilities of the occurrence of certain variables and conditional probabilities of combinations of variables, such as alcohol content, sulphates content, volatile acidity, and sulfur dioxide content, these values are converted into binary at this stage. As mentioned earlier, a value of 1 is assigned when a measurement exceeds the dataset average, and 0 otherwise. The dataset is then split into a training dataset and a test dataset at an 80:20 ratio.

Step 2 involves calculating the probabilities of each of the four variables that contribute the most. To build a Bayesian model with two variables, for example, alcohol and sulphates content, the model needs the marginal probability of each being above or below average and the joint probabilities of their combined states. For two variables, there are four joint probabilities.

In Step 3, a "Norm" value is calculated, which represents an estimated conditional probability—specifically, the likelihood of 'good' wine quality given the higher or lower alcohol and sulphates content. This value is refined through a recursive training process in the "train\_qbn\_wine" function, primarily influenced by the "to\_params" and "calculate\_qual\_params" functions, which are discussed in Step 5.

The fourth step is where the quantum computing begins.

Step 4 marks the beginning of the quantum computing process. It starts by assigning the likelihoods of alcohol and sulphates content to the first two qubits, the third qubit is for norm values, and the fourth qubit represents quality. After this operation, the superpositions of qubit0 (alcohol) and qubit1 (sulphates) become 0.57|0> + 0.43|1> and 0.63|0> + 0.37|1>, respectively.

Text

Description automatically generatedText

Description automatically generated

It then entangles these qubits to apply norm parameters to the third qubit.

Text

Description automatically generated

Lastly, it applies the quality parameter to the fourth qubit.

Text

Description automatically generated

To apply the norm parameter consistent with qubit0 and qubit1, we devised a supportive function named CCRY [[9]](#footnote-9)[9]. Its operational sequence is as follows:

1. The parameter is applied to qubit2 only if both qubit0 and qubit1 are in the state 1. Initially, qubit2 is rotated by half of the desired probability when qubit1 is in state 1.
2. Qubit1 is then flipped if qubit0 is in state 1.
3. Subsequently, qubit2 is rotated back by half of the probability when qubit1 is in state 1. Since qubit1's state was flipped in the preceding step, this action restores qubit2 to its original state if qubit0 is in state 1 but qubit1 is in state 0. We aim to avoid applying any probability in this scenario because the intention is to apply it exclusively when both qubit0 and qubit1 are in state 1.
4. Qubit1 is flipped again if qubit0 is in state 1.
5. Finally, the remaining half of the probability is applied to qubit2 if qubit0 is in state 1.

Text

Description automatically generated **A picture containing text, clock

Description automatically generated**

We also created the CCCRY function to control three qubits for three variables.

**A screenshot of a computer

Description automatically generated with medium confidence**

**A picture containing clock, clipart

Description automatically generated**

The previous explanation used the terms "apply probability" and "rotate" interchangeably because these gates accept only angles, not probabilities. Therefore, an additional function has been established to convert probabilities into angles[[10]](#footnote-10)[10].

****

The complete model, incorporating two variables, is depicted in the diagram below.

Chart

Description automatically generated

The model is trained during the fifth step. As mentioned in the third step, the 'train\_qbn\_wine' function executes the model training. This function's role is to generate recursions, producing 'results' (which are the quantum states at each recursive training iteration).

The 'results' are then input into the 'to\_params' function. The 'to\_params' function calculates 2^(n+1) parameters (Norm parameters). These parameters are computed for each category (such as higher/lower alcohol, higher/lower sulphates, and high/low quality) by dividing the sum of the previously determined parameters of the data favored by Norm (equal to 1) by the total sum of all data in that category. This new set of parameters is then applied to the model in the subsequent iteration.

Finally, the 'calculate\_qual\_params' function returns a set of two parameters. These are obtained by dividing the sum of the Norm parameters for good quality wine by the sum of the Norm parameters for all data. Conceptually, these parameters can be interpreted as an indication of the previous accuracy of the estimated Norm parameters.

1. **Result**

We experimented with a total of five models: one classical machine learning model, the random forest model, and four quantum models. The following sections show the results and specifications for each model.

* 1. Random Forest model

The random forest model utilized the Scikit-learn API. With a depth of two layers, it achieved an accuracy of 72.5%. While this is a basic implementation with considerable room for improvement, we will use the 72.5% as a benchmark and proceed, as the primary purpose of this research is to explore the potential of quantum computing.

This implementation identified four major contributing variables in the dataset: alcohol, sulphates, volatile acidity, and sulfur dioxide contents. Future quantum computing models will be constructed based on these variables.

Diagram

Description automatically generatedChart, bar chart

Description automatically generated

* 1. Quantum Model with Two Variables

The first quantum model incorporated two variables, Alcohol and Sulphates, to derive Norm parameters for predicting Quality. This model reached a peak accuracy of 71% after 20 iterations of training.

Diagram

Description automatically generated Chart, line chart

Description automatically generated

* 1. Quantum Model with Three Variables

We developed two quantum models incorporating three variables: Alcohol, Sulphates, and Volatile Acidity. One model calculates Norm parameters using all three variables, while the other uses two—Alcohol and Sulphates—and applies a conditional probability to the Quality qubit based on the Norm parameter and Volatile Acidity.

5.3-1 Bayesian Inference with Three Variables This model, which calculates the Norm parameter using three variables, attained an accuracy of 52% after 40 iterations.

Diagram

Description automatically generated Chart, line chart

Description automatically generated

5.3-2 Bayesian Inference with Two Variables and One Independent Variable Here, the Norm parameter is derived from two variables, and the probability of Volatile Acidity is applied directly to the Quality qubit. This model achieved an accuracy of 74% after 30 iterations.

Diagram

Description automatically generated Chart, line chart

Description automatically generated

* 1. Quantum Model with Four Variables

Considering the performance of the three-variable models, it seemed more promising to explore a model deriving Norm parameters from two variables. This model calculates the Norm parameter using Alcohol and Sulphates and applies probabilities of Volatile Acidity and Sulfur Dioxide directly to the Quality qubits as conditional probabilities. It achieved a maximum accuracy of 74% after 30 iterations.

Diagram

Description automatically generated Chart, line chart

Description automatically generated

1. **Conclusion and Future Directions**

The quantum machine learning models, with both three and four variables, achieved 74% accuracy, slightly outperforming the classical random forest model. Among the four quantum models, the second set with three variables showed more volatile performance and even deteriorated after 20 and 30 iterations. The other three quantum models showed that increasing the number of training iterations had an impact on the results up to 10 iterations, but only minimal changes were observed afterward. The model with four variables showed a 3% improvement over the two-variable model. The most notable difference was in the initial accuracy after a single iteration: the two-variable model started at 53%, while the four-variable model began at 61%.

This research utilized classical computation to identify significant contributing variables. However, in future work, I aim to explore methods for identifying key variables using quantum computing, potentially employing techniques like Grover's Search Algorithm [[11]](#footnote-11)[11], which could theoretically find a desired objective in square root time complexity (sqrt(n)). It would also be intriguing to investigate the application of Shor's algorithm[[12]](#footnote-12)[12], as there is a belief that it can enhance the time complexity of optimization problems.

1. **References**

[1] Naresh Bhat, Wine Quality Classification - Data set for Binary Classification [https://www.kaggle.com/nareshbhat/wine-quality-binary-classification?select=wine.csv]

[2] Scikit-learn, sklearn.ensemble.RandomForestClassifier [https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html]

[3] Scikit-learn, sklearn, sklearn.model\_selection.train\_test\_split [https://scikit-learn.org/stable/modules/generated/sklearn.model\_selection.train\_test\_split.html]

[4] Scikit-learn, sklearn, sklearn.metrics.confusion\_matrix¶ [https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion\_matrix.html]

[5] Chris Bernhardt, QUANTUM COMPUTING FOR EVERYONE (P.118)

[6] Dr. Frank Zickert - Hands-On Quantum Machine Learning With Python Volume 1\_ Get Started-PYQML (2021) (P. 147)

[7] Qiskit [https://qiskit.org/textbook/ch-gates/multiple-qubits-entangled-states.html#cnot] (3. Multi-Qubit Gates 3.1 The CNOT-Gate)

[8] Dr. Frank Zickert - Hands-On Quantum Machine Learning With Python Volume 1\_ Get Started-PYQML (2021) (P. 141)

[9] Dr. Frank Zickert - Hands-On Quantum Machine Learning With Python Volume 1\_ Get Started-PYQML (2021) (P.276)

[10] Dr. Frank Zickert - Hands-On Quantum Machine Learning With Python Volume 1\_ Get Started-PYQML (2021) (P.260)

[11][12] Chris Bernhardt, QUANTUM COMPUTING FOR EVERYONE (P.176)

1. Naresh Bhat, Wine Quality Classification - Data set for Binary Classification [https://www.kaggle.com/nareshbhat/wine-quality-binary-classification?select=wine.csv] [↑](#footnote-ref-1)
2. Scikit-learn, sklearn.ensemble.RandomForestClassifier [https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html] [↑](#footnote-ref-2)
3. Scikit-learn, sklearn, sklearn.model\_selection.train\_test\_split [https://scikit-learn.org/stable/modules/generated/sklearn.model\_selection.train\_test\_split.html] [↑](#footnote-ref-3)
4. Scikit-learn, sklearn, sklearn.metrics.confusion\_matrix¶ [https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion\_matrix.html] [↑](#footnote-ref-4)
5. Chris Bernhardt, QUANTUM COMPUTING FOR EVERYONE (P.118) [↑](#footnote-ref-5)
6. Dr. Frank Zickert - Hands-On Quantum Machine Learning With Python Volume 1\_ Get Started-PYQML (2021) (P. 147) [↑](#footnote-ref-6)
7. Qiskit [https://qiskit.org/textbook/ch-gates/multiple-qubits-entangled-states.html#cnot] (3. Multi-Qubit Gates 3.1 The CNOT-Gate) [↑](#footnote-ref-7)
8. Dr. Frank Zickert - Hands-On Quantum Machine Learning With Python Volume 1\_ Get Started-PYQML (2021) (P. 141) [↑](#footnote-ref-8)
9. Dr. Frank Zickert - Hands-On Quantum Machine Learning With Python Volume 1\_ Get Started-PYQML (2021) (P.276) [↑](#footnote-ref-9)
10. Dr. Frank Zickert - Hands-On Quantum Machine Learning With Python Volume 1\_ Get Started-PYQML (2021) (P.260) [↑](#footnote-ref-10)
11. Chris Bernhardt, QUANTUM COMPUTING FOR EVERYONE (P.176) [↑](#footnote-ref-11)
12. Chris Bernhardt, QUANTUM COMPUTING FOR EVERYONE (P.176) [↑](#footnote-ref-12)