

QUANTUMMOONLIGHT: A Low-Code Platform to Experiment with Quantum Machine Learning

Francesco Amato^a, Matteo Cicalese^a, Luca Contrasto^a, Giacomo Cubicciotti^a, Gerardo D'Ambola^a, Antonio La Marca^a, Giuseppe Pagano^a, Fiorentino Tomeo^a, Gennaro Alessio Robertazzi^a, Gabriele Vassallo^a, Gemma Catolino^b, Giammaria Giordano^a, Stefano Lambiase^a, Valeria Pontillo^a, Giulia Sellitto^a, Giovanni Acampora^c, Filomena Ferrucci^a, Fabio Palomba^a

^a*Software Engineering (SeSa) Lab — University of Salerno, Italy*

^b*Jheronimus Academy of Data Science & Tilburg University, The Netherlands*

^c*University of Naples, Italy*

Abstract

In the last decades, artificial intelligence (AI)—and machine learning in particular—has often been used to address multiple problems. Recent advances in the field revolve around the use of *quantum* machine learning, which promises to revolutionize program computation and boost software systems' problem-solving capabilities. However, using quantum computing technologies is not trivial and requires interdisciplinary skills and expertise. In this article, we propose QUANTUMMOONLIGHT, a community-based low-code platform that allows researchers and practitioners to configure and experiment with quantum machine learning pipelines, compare them with canonical machine learning algorithms, and share lessons learned and experience reports. We showcase the architecture and main features of QUANTUMMOONLIGHT, other than discussing its envisioned impact.

Tool site: <https://sesaquantummoonlight.ngrok.io/>

Tool video: <https://youtu.be/xhXj1uZ7P1M>

Keywords: Quantum ML, Low-Code Platforms, Artificial Intelligence.

1. Software Description

This section provides an overview on the tool's architecture and the main functionalities it offers to the target audience.

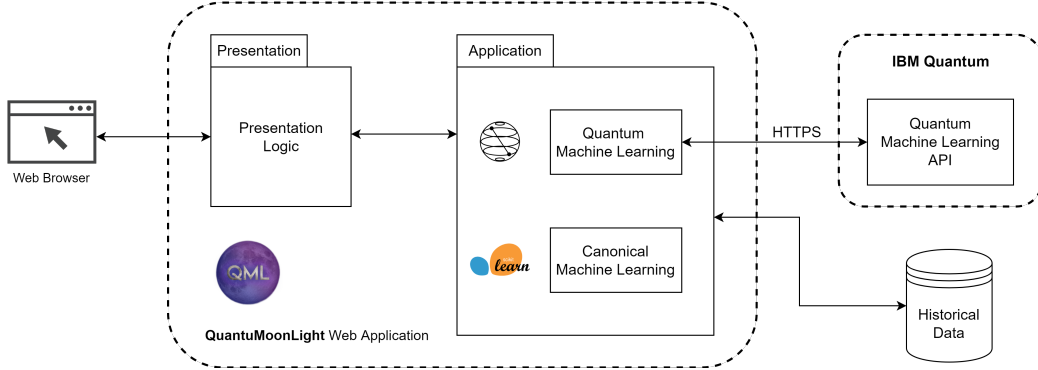


Figure 1: QUANTUMMOONLIGHT tool architecture.

1.1. Software architecture

Figure 1 shows the three-tier architecture of the QUANTUMMOONLIGHT tool, implemented as a web application that interacts with the IBM QUANTUM platform [1] to provide its functionalities. Specifically, the tool allows users to program quantum computers, via querying the API and web services of the IBM QUANTUM COMPUTING platform over HTTPS.

The tool enables the comparison between quantum and canonical machine learning solutions, implementing a number of state-of-the-art machine learning models provided by the SCIKIT-LEARN¹ library for PYTHON. QUANTUMMOONLIGHT relies on a relational database to maintain historical data about the performed experiments and to support the community-oriented blog. Furthermore, the tool requires users to be registered to acquire data about their permissions over the IBM QUANTUM platform [1]. Moreover, QUANTUMMOONLIGHT has been designed to be extensible: it implements design patterns that enable the quick addition of new quantum learners, configuration steps, and back-ends.

1.2. Software functionalities

QUANTUMMOONLIGHT enables users to (1) configure, experiment with, and evaluate quantum machine learning applications, (2) compare the performance of quantum and canonical machine learning solutions, and (3) share knowledge and results obtained in the performed experiments through a community-inspired blog on the web.

We design the tool’s features to make it suitable for empirical research aiming at evaluating the performance of quantum algorithms in various contexts, other than providing an instrument to facilitate the comparison with

¹SCIKIT-LEARN: <https://scikit-learn.org/stable/>


canonical algorithms. The blog feature can let researchers and practitioners discuss the conducted experiments, hence (1) increasing the awareness of the potential of quantum machine learning and (2) lowering the entry barriers faced by newcomers approaching such a complex research theme.

To enable experimentations, QUANTUMMOONLIGHT relies on the APIs provided by QISKIT, *i.e.*, an open-source SDK for working with quantum computers. Specifically, the tool allows users to exploit algorithms for data classification and quantum support vector regressors. For the sake of conciseness, this section provides an overview of the list of quantum machine learning algorithms used by the tool—we do not explain how such algorithms work.

To allow experiments involving classification tasks, QUANTUMMOONLIGHT provides the *quantum support vector classifier* and the *quantum neural network classifier*, other than the *Pegasos Quantum Support Vector Classifier* defined by Shalev-Shwartz *et al.* [2]. Regression tasks can be performed using *support network*, *neural network*, and *neural network regressor using TwoLayerQNN*. We refer to the documentation of QISKIT for extensive details describing the implementation of the algorithms supported by the tool.²

The IBM QUANTUM platform [1] provides a number of back-end systems—with different hardware potentiality—to perform the quantum operations.³ QUANTUMMOONLIGHT allows users to select the desired back-end to perform machine learning tasks, with some limitations based on the type of logged user—according to the IBM QUANTUM permissions policy.

The QuantuMoonLight Platform.

 QUANTUMMOONLIGHT is a web application designed to allow practitioners and researchers to experiment with quantum machine learning.

2. Evaluation of the Tool

We evaluated the tool on two different tasks concerned with code smells [3] and flaky tests [4] prediction, and we assessed its usability [5].

Code Smells and Flaky Tests Prediction. Code smells are symptoms of poor design and implementation choices applied by programmers during the development of a software system [3]. Flaky tests exhibit both a passing

²QISKIT Machine Learning API reference: https://qiskit.org/documentation/machine-learning/apidocs/qiskit_machine_learning.algorithms.html

³IBM QUANTUM back-end: <https://quantum-computing.ibm.com/services?services=systems>

and failing behavior when run against the same code, being unreliable and producing a non-deterministic outcome [4].

In the context of code smells, we considered part of the dataset provided by Palomba *et al.* [3], focusing on five projects and five different kinds of code smells, *i.e.*, *Complex Class*, *Long Method*, *Spaghetti Code*, *Large Class*, and *Feature Envy*. As for flaky tests, we relied on the dataset provided by Pontillo *et al.* [6], focusing on 18 open-source projects and 670 flaky tests.

For both tasks, we relied on features and oracles provided by the datasets, other than employing 10-fold cross-validation and the *Quantum Support Vector Classifier*. We measured the performance using *accuracy*, *precision*, *recall*, and *F-Measure*. When considering the code smells prediction task, the results achieved were in line with those reported in previous work [7], yet we observed a reduction in the training time. As for the task of flakiness prediction, we noticed that the performance achieved was *lower* compared to what shown by existing approaches [6, 8], while the computational cost drastically decreased. This suggests that the use of quantum technologies does not imply immediate improvements of the accuracy of state-of-the-art approaches and that further research is required—this hints at the potential impact of our tool, which researchers may use to experiment with quantum technologies to advance the current knowledge. Detailed results of the conducted experiments are available in the online appendix of this paper [9].

Usability Testing. We evaluated the usability of the tool by applying *iterative usability testing* [5], thus implementing an iterative process to get continuous feedback from users and keep improving the user experience of the tool. We conducted the test with 12 students of the course “*Introduction of Machine Learning*” at the Jheronimus Academy of Data Science (The Netherlands). In particular, students had to perform three tasks—described in Table 1—during each iteration, sharing feedback on the usability of the tool. After each iteration, we interviewed participants to assess the tool’s usability in terms of *learnability*, *efficiency*, and *satisfaction*. We improved the user interface of the tool [5] according to the feedback received. Overall, we conducted three iterations before reaching saturation. In the iterative process, the user interface changed in terms of size of graphical elements, website content, and feedback provided to users, *e.g.*, we included info boxes to guide users when interacting with the tool.

3. Impact

We envision that QUANTUMMOONLIGHT will have a notable impact both for researchers and practitioners.

Table 1: Iterative Usability Test.

#	Task Description
1	Perform a quantum regression task using the website.
2	Compare the quantum regression task with a canonical one.
3	Access the community blog and post the results of your tasks.

Impact on Research. The platform allows researchers to easily access the quantum hardware provided by IBM, enabling them to experiment with quantum technologies in a more “usable” manner. Perhaps more importantly, the community-inspired blog allows researchers to share lessons learned, experiences, and reflections that may increase awareness about quantum computing, driving the community to grow.

Impact on Practice. QUANTUMMOONLIGHT opens a “window” on the quantum world that can allow managers and team leaders—as well as data scientists and software developers—to exploit such a technology to make decisions faster. Practitioners can indeed use the platform to verify the suitability of quantum computing solutions for specific software engineering tasks and make informed decisions on their adoption in practice. At the same time, practitioners can read about successful experiences through the blog, learning how to configure quantum machine learning solutions.

Undoubtedly, quantum technology is still far from being exploitable for everyday tasks. However, QUANTUMMOONLIGHT can contribute in this regard, facilitating the adoption of such technology by a growing audience.

4. Concluding Remarks

QUANTUMMOONLIGHT is a web application to experiment with quantum machine learning. The tool allows users to (1) configure and experiment with quantum regression and classification tasks, (2) compare the results of quantum and traditional machine learning solutions, and (3) share findings and insights about quantum machine learning via a community-oriented blog. We evaluated the tool on two prediction tasks, *i.e.*, flaky tests and code smells prediction, other than its usability through iterative usability testing.

We aim at extending the set of implemented quantum algorithms with more advanced ones—*e.g.*, the *Variational Quantum Eigensolver (VQE)* and the *Quantum Boltzmann Machine (QBM)*. Furthermore, we plan to extend our application from an integration point of view, including solutions from

other providers—*e.g.*, MICROSOFT AZURE QUANTUM and XANADU—to broaden the possibilities for the target audience.

Acknowledgements

We acknowledge the use of IBM QUANTUM services for this work. The points of views expressed are those of the authors, and do not reflect the official policy or position of IBM or the IBM QUANTUM team. The work is partially supported by the Swiss National Science Foundation (SNF Project No. PZ00P2_186090) and by the EMELIOT national research project (MUR Project No. 2020W3A5FY).

References

- [1] IBM Quantum, <https://quantum-computing.ibm.com/> (2021).
- [2] S. Shalev-Shwartz, Y. Singer, N. Srebro, A. Cotter, Pegasos: Primal estimated sub-gradient solver for svm, *Mathematical programming* 127 (1) (2011) 3–30.
- [3] F. Palomba, G. Bavota, M. D. Penta, F. Fasano, R. Oliveto, A. D. Lucia, On the diffuseness and the impact on maintainability of code smells: a large scale empirical investigation, *Empirical Software Engineering* 23 (3) (2018) 1188–1221.
- [4] Q. Luo, F. Hariri, L. Eloussi, D. Marinov, An empirical analysis of flaky tests, in: *Proceedings of the 22nd ACM SIGSOFT international symposium on foundations of software engineering*, 2014, pp. 643–653.
- [5] A. Genov, Iterative usability testing as continuous feedback: A control systems perspective, *Journal of Usability Studies* 1 (1) (2005) 18–27.
- [6] V. Pontillo, F. Palomba, F. Ferrucci, Static test flakiness prediction: How far can we go?, *Empirical Software Engineering* 27 (7) (2022) 1–44.
- [7] D. Di Nucci, F. Palomba, D. A. Tamburri, A. Serebrenik, A. De Lucia, Detecting code smells using machine learning techniques: are we there yet?, in: *2018 IEEE 25th international conference on software analysis, evolution and reengineering (saner)*, IEEE, 2018, pp. 612–621.
- [8] A. Alshammari, C. Morris, M. Hilton, J. Bell, Flakeflagger: Predicting flakiness without rerunning tests, in: *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, IEEE, 2021, pp. 1572–1584.

- [9] F. Amato, M. Cicalese, L. Contrasto, G. Cubicciotti, G. D’Ámbola, A. La Marca, G. Pagano, F. Tomeo, G. Catolino, G. Giordano, S. Lambiase, V. Pontillo, G. Sellitto, F. Ferrucci, F. Palomba, QUANTUMMOONLIGHT: A low-code platform to experiment with quantum machine learning — online appendix (2022). doi:10.6084/m9.figshare.20108336.
- [10] H. Abdi, L. J. Williams, Principal component analysis, Wiley interdisciplinary reviews: computational statistics 2 (4) (2010) 433–459.
- [11] E. R. Dougherty, J. Hua, C. Sima, Performance of feature selection methods, Current genomics 10 (6) (2009) 365–374.

Illustrative Examples

QUANTUMMOONLIGHT provides its features through a four-page website. In the following section, we describe the page representing the core functionalities of the developed tool. A more detailed description of all the pages is available in the online appendix of this paper [9].

Figure 2 shows the GUI of the web page enabling to configure and run experiments with QUANTUMMOONLIGHT. The first step consists in loading the desired dataset into the tool (①). QUANTUMMOONLIGHT allows uploading the training, test, and prediction sets, depending on the specific experiment the user would like to run. The tool also provides the possibility to quickly setup experiments; the user can indeed check whether they would like to apply default configurations in terms of feature extraction, feature selection, and validation techniques. In particular, the latter is implemented using a percentage split that automatically assigns to the training set 80% of the instances in the uploaded dataset and 20% in the test set. The user can then select the quantum machine learner to use among the supported ones (②).

A user can already execute quantum experiments with the first two configuration steps. However, further customization is allowed through the “*Advanced Options*” menu, showed in sub-figures ③–⑦. In terms of validation, the tool provides an additional setting, *i.e.*, the k -fold cross-validation. The platform also enables the automation of a number of data pre-processing steps, such as data balancing or standardization. The user can manage features by employing Principal Component Analysis [10] for feature extraction and choosing to rely on the k -best features [11] for prediction. In both cases, the user can select the number of features to work with. Finally, the user can select the back-end to use, namely the characteristics that the quantum

The image shows a web-based configuration interface for quantum classification, divided into two main panels. The left panel, titled '1 Load your Dataset for Quantum Classification', contains three file upload sections: 'Training Set to upload', 'Test Set to Upload', and 'Prediction Set to Upload', each with a 'Choose File' button. Below these are four checkboxes: 'Automatically Split the Training Set' (checked), 'Reduce columns with Feature Extraction(PCA)' (checked), 'Reduce columns with Feature Selection(K Best)' (checked), and 'Reduce rows with Prototype Selection' (unchecked). A dropdown menu for 'Select the algorithm:' is set to 'QSVR'. An 'Advanced Options' button is at the bottom of this panel. The right panel contains seven numbered sections: '3 Validation' with radio buttons for 'Simple Train-Test Split' (selected) and 'K-fold Cross Validation'; '4 Preprocessing' with checkboxes for 'Data Imputation', 'Data Balancing', 'MinMax Scaling', and 'Standard Scaling'; '5 Feature Extraction' with a text input 'Insert nr of columns to reduce Feature Extraction:' set to '2'; '6 Feature Selection' with a text input 'Insert nr of columns to reduce Feature Selection:' set to '2'; and '7 Select quantum system:' with a dropdown menu set to 'Least Busy IBM Backend'. An 'Upload' button is located at the bottom center of the interface.

Figure 2: The GUI provided by QUANTUMOONLIGHT to run experiments.

machine should have when executing the experiment—in Figure 2, the “*Least busy IBM back-end*” was selected.

The user can confirm the selected settings at the end of the configuration process. QUANTUMOONLIGHT will connect to the IBM services, start the execution, and inform the user via e-mail once the experiment is concluded. Once the results are available, the user can request the comparison with the performance of canonical machine learners through the dedicated *analysis* page—described in the online appendix of this paper [9]

Current code version

Nr.	Code metadata description	Please fill in this column
C1	Current code version	v1
C2	Permanent link to repository	https://github.com/Robertales/QuantuMoonLight
C3	Permanent link to Reproducible Capsule	NA
C4	Legal Code License	Common Development and Distribution License 1.0
C5	Code versioning system used	git
C6	Software code languages, tools, and services used	python, IBM Quantum
C7	Compilation requirements, operating environments	PYTHON \geq 3.7, ANACONDA \geq 2021.11, MYSQL \geq 7.0
C8	Link to developer documentation/manual	https://github.com/Robertales/QuantuMoonLight
C9	Support email for questions	sesalab@unisa.it

Table 2: Code metadata.

Current executable software version

Nr.	(Executable) software meta-data description	Please fill in this column
S1	Current software version	v1
S2	Permanent link to executables of this version	https://sesaquantumoonlight.ngrok.io/
S3	Permanent link to Reproducible Capsule	NA
S4	Legal Software License	Common Development and Distribution License 1.0
S5	Computing platforms/Operating Systems	web-based application
S6	Installation requirements & dependencies	PYTHON \geq 3.7, ANACONDA \geq 2021.11, MYSQL \geq 7.0
S7	Link to user manual	https://github.com/Robertales/QuantuMoonLight
S8	Support email for questions	sesalab@unisa.it

Table 3: Software metadata.