National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Stephen McMullan
Student ID: x19139497

School of Computing
National College of Ireland

Supervisor: Dr. Majid Latifi

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Stephen McMullan
Student ID:	x19139497
Programme:	Data Analytics
Year:	2021
Module:	MSc Research Project
Supervisor:	Dr. Majid Latifi
Submission Due Date:	16/08/2021
Project Title:	Configuration Manual
Word Count:	932
Page Count:	9

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard speci ed in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	19th September 2021

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator o ce must be placed into the assignment box located outside the o ce.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Stephen McMullan
x19139497

1 Juypter Notebooks

There are three Juypter notebooks delivered as part of this research project. They are written in Python and using scikit-learn Python machine learning library (Pedregosa et al.; 2011) and IBM Qiskit software development kit for quantum computing (ANIS et al.; 2021). The notebooks are as follows:

- QiskitIntro.ipynb : Introduction to Qiskit and quantum computing through examples
- ClassicalIris.ipynb : Exploration of the Iris dataset using conventional machine learning techniques from scikit-learn
- QuantumIris.ipynb : Classi cation of the Iris dataset observations using SVM kernels produced by quantum computing processors via Qiskit

1.1 Demonstration Videos

The following set of hyperlinked demonstration videos are a guided tour of the notebooks and the IBM Quantum environment

- QiskitIntro.ipynb Part I: Introduction to Qiskit and quantum computing through examples
- QiskitIntro.ipynb Part II: Introduction to Qiskit and quantum computing through examples
- ClassicalIris.ipynb: Exploration of the Iris dataset using conventional machine learning techniques from scikit-learn
- IBM Quantum: A guided tour of the IBM Quantum environment
- QuantumIris.ipynb: Classi cation of the Iris dataset observations using SVM kernels produced by quantum computing processors via Qiskit

2 Runtime Installation Prerequisites

The notebooks are designed to run on a local JuypterLab server installation or a cloud based notebook environment speci cally IBM Quantum Lab or Google Colaboratory.

2.1 Local JuypterLab Installation

It is recommended to install the Anaconda Python distribution platform as the simplest way to deploy JuypterLab. For instructions on deploying Anaconda please see `https://www.anaconda.com/products/individual`.

For instructions on installing JuypterLab individually as a stand alone application please see `https://jupyter.org/install`.

It is recommend that a PC with at least 4 CPUs, 8GB RAM and 100GB of free disk space is used. Otherwise the cloud hosted notebook servers are recommended.

Note that even if running a locally installed Juypter notebook server, if actual quantum processing backends are required then an IBM Quantum account and API token is required. Sign up for these at `https://quantum-computing.ibm.com/`

2.2 Cloud Hosted Juypter Notebook Servers

IBM Quantum Labs `https://quantum-computing.ibm.com/` is recommended especially with its additional IBM Quantum Composer and IBM Quantum Services forming a complete quantum computing development environment.

It is also possible to run the notebooks in Google Colaboratory `https://colab.research.google.com`. Note that even if running on Google Colaboratory, if actual IBM quantum processing backends are required then an IBM Quantum account and API token is required. Sign up for these at `https://quantum-computing.ibm.com/`

3 IBM Quantum Environment

Figure 1 shows the Juypter notebook environment in IBM Quantum Lab.

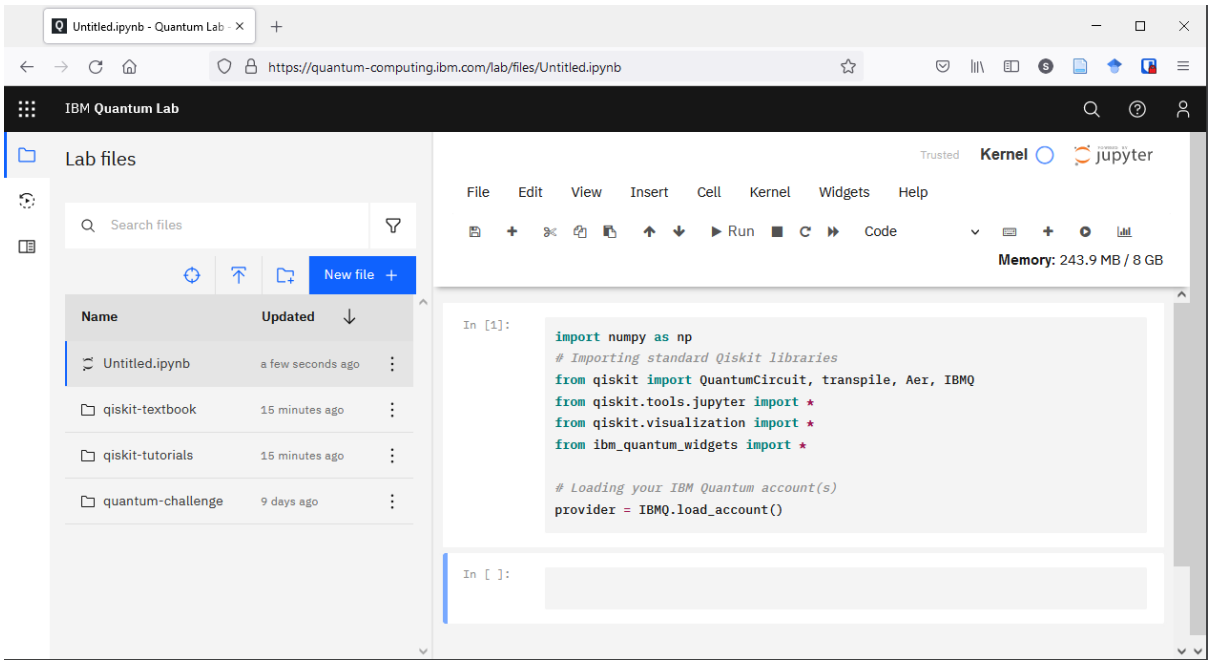

Figure 1: Juypter notebook environment in IBM Quantum Lab

Figure 2 shows the graphical construction of a SWAP Test circuit in IBM Quantum Composer with Qiskit Python code generation along with quantum state representation as a probability distribution histogram and in Bloch Sphere representation.

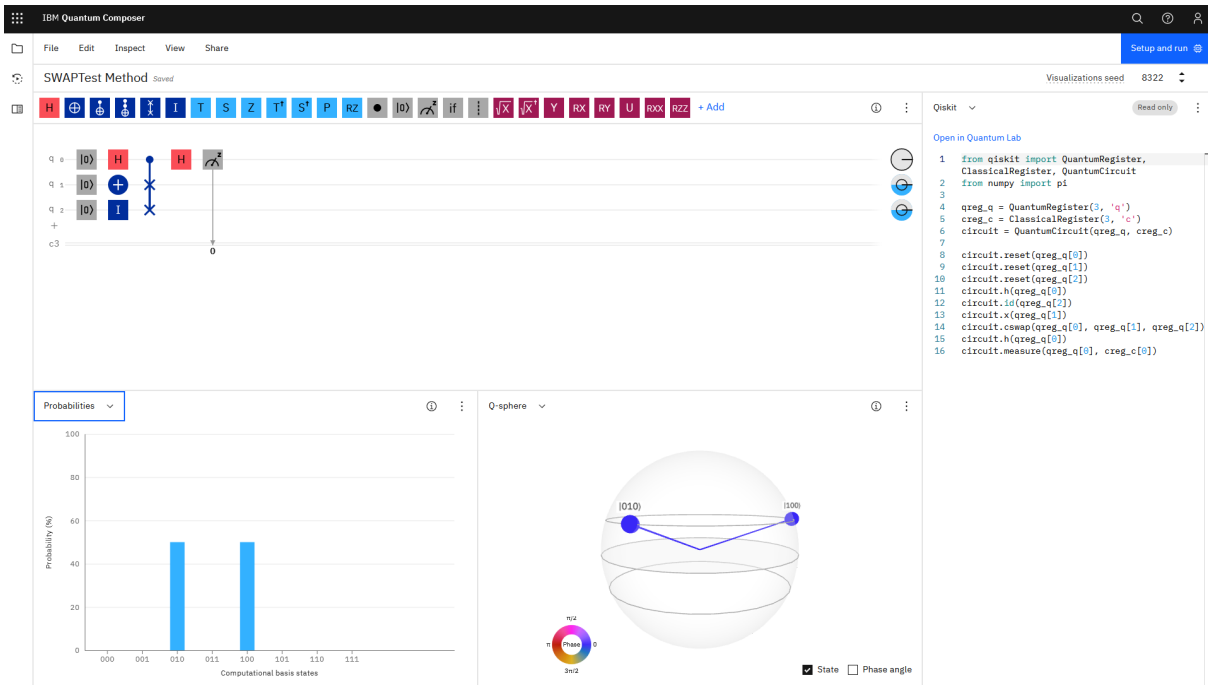

Figure 2: SWAP Test circuit for measuring delity between quantum states in IBM Quantum Composer

Figure 3 shows a sample of the publicly accessible physical quantum computing systems from IBM Quantum Services. The number of physical qubits determines the maximum size of the dataset that can be encoded.

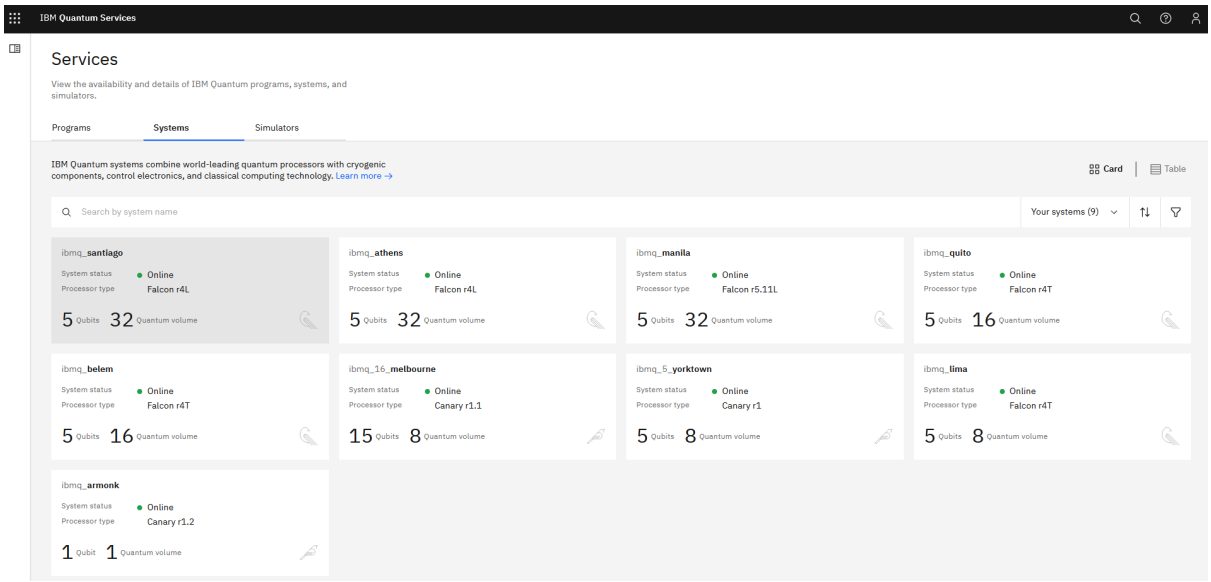

Figure 3: A sample of the publicly accessible physical quantum computing systems from IBM Quantum Services

4 Python Dependencies

scikit-learn Pedregosa et al. (2011) is a standard Python machine learning library dependency so it is assumed that this is present in your Python environment. If not then install it using 'pip install scikit-learn'. It comes with the Anaconda Python distribution as standard.

Qiskit must be installed as a one-time operation. It has quite a long dependency list. Please uncomment the following section in the notebooks that require Qiskit specifically. It does not need to be repeated once the packages are installed in your Python environment.

```
# !pip install qiskit
# !pip install qiskit[visualization]
# !pip install qiskit_machine_learning
```

Figure 4: Qiskit python dependencies

5 IBM Quantum Tokens and Quantum Processor Backend Access

Note that in order to access the actual IBM quantum processor backends then you need an IBM Quantum API token. This is available upon setting up an account with IBM Quantum at `https://quantum-computing.ibm.com/`. Your Token API will be displayed on the dashboard as shown in Figure 5

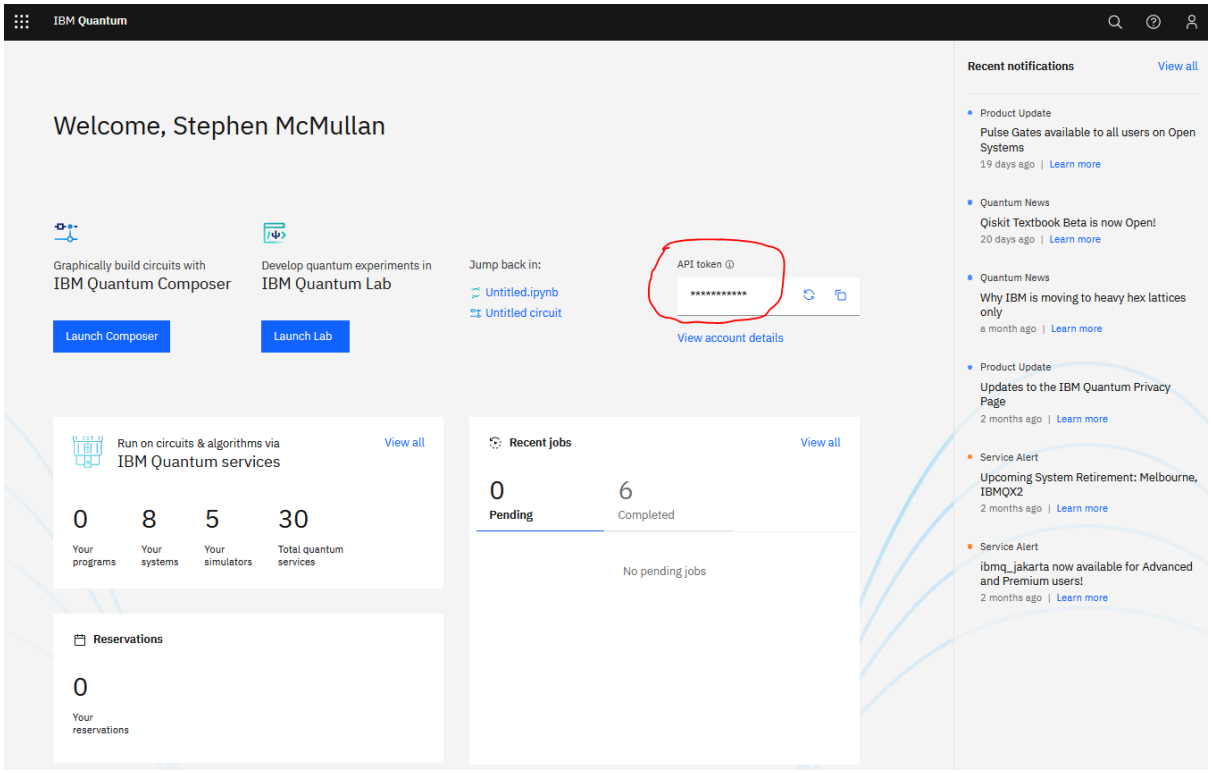

Figure 5: IBM Token on IBM Quantum dashboard

This token should then be inserted into the notebooks at the position shown in Figure 6

```
from qiskit import IBMQ

# Uncomment these lines and run if you haven't accessed your IBM Quantum account before from JuypterLab
# There is no need to uncomment them if running directly in IBM Quantum Lab

#TOKEN = "API token from IBM Quantum dashboard"
#IBMQ.save_account(TOKEN)
provider = IBMQ.load_account()
```

Figure 6: IBM Token storage in Juypter notebook

6 IBM Quantum Backends and Runtimes

In the main QuantumIris.ipynb notebook, the least busy IBM Quantum backend is chosen by the notebook. However access to these devices is done on a first come, first served basis and a fairshare method of processor execution during the job runtime. This can lead to substantial delays in running the notebook.

For example, Figure 7 shows an elapsed time of over 50 minutes for the least busy public access 5-qubit IBM quantum processor to produce a precomputed kernel for the Iris dataset for an actual CPU time of just over 1 minute. The notebook calculates 4 different precomputed kernels using the quantum processor so there is a substantial runtime for the entire notebook as delivered.

```
CPU times: user 1min, sys: 2.15 s, total: 1min 2s
Wall time: 50min 1s
```

Figure 7: IBM Quantum backend runtime for a single precomputed quantum kernel

An alternative to running on the actual IBM quantum processor backend is to use a local Qiskit simulator. There are two alternative quantum simulators provided as shown in Figure 8. The statevector simulator simulates an ideal error-corrected quantum processing backend and represents the best theoretically achievable result. Perhaps of more interest is the qasm simulator which has a built in noise model for simulating current era NISQ devices which are non-error corrected and susceptible to environmental noise. This simulator will be more representative of the physical quantum processing backends but run a lot faster. Uncomment the simulator of choice from the code block shown in Figure 8

```
from qiskit.providers.ibmq import least_busy

small_devices = provider.backends(filters=lambda x: x.configuration().n_qubits == 5 and not x.configuration().simulator)
backend = least_busy(small_devices)

# Alternatively for comparison, an ideal simulator (statevector_simulator) or a noisy
# simulator (qasm_simulator) may be chosen in preference to an actual quantum processor

#from qiskit import Aer
#backend = Aer.get_backend('statevector_simulator')
#backend = Aer.get_backend('qasm_simulator')

print(backend)
```

Figure 8: IBM Quantum Backends

The circuit executions may be monitored in IBM Quantum Jobs view as shown in Figure 9

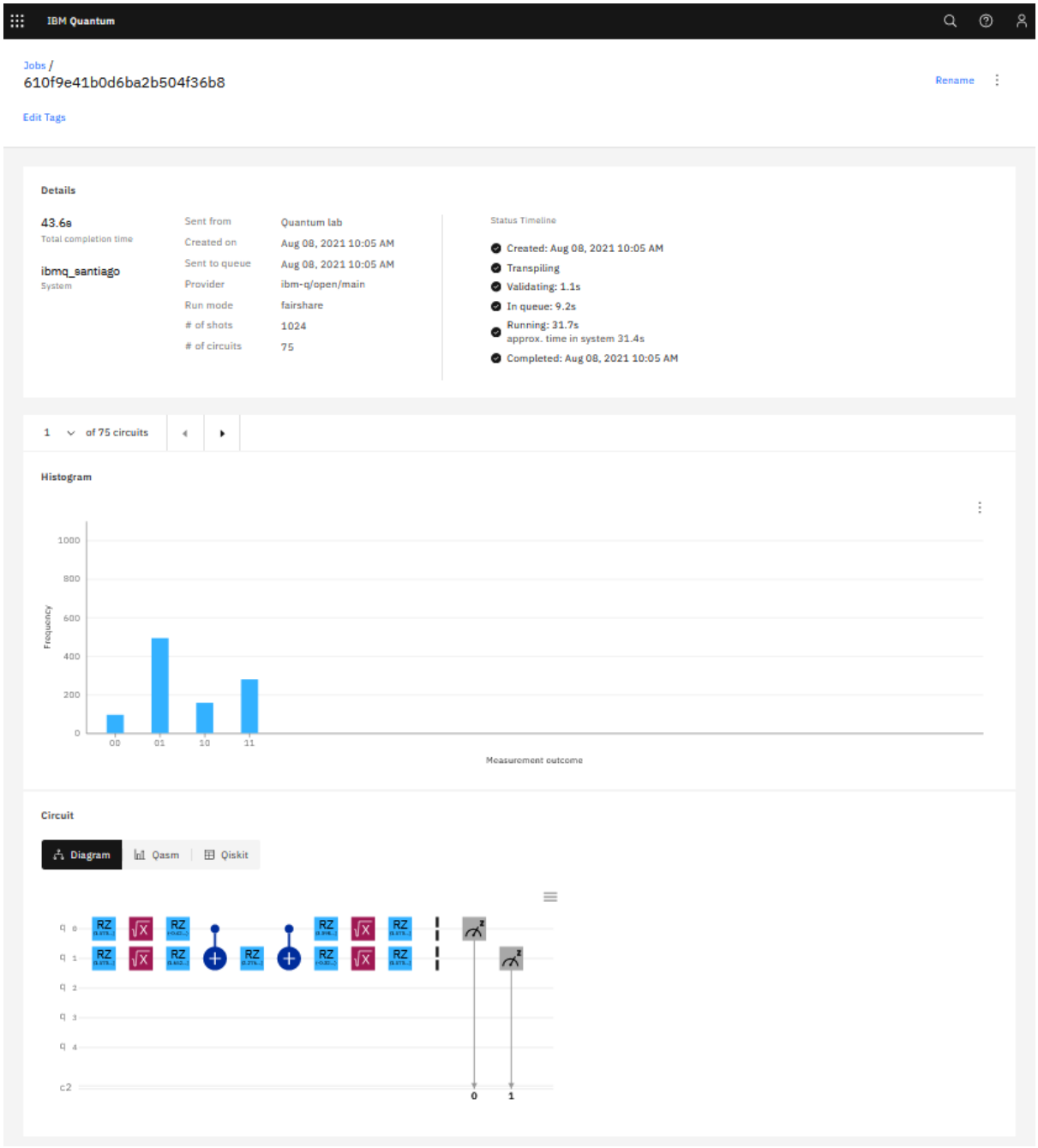

Figure 9: IBM Quantum Job statistics

References

ANIS, M. S., Abraham, H., AduO ei, Agarwal, R., Agliardi, G., Aharoni, M., Akhalwaya, I. Y., Aleksandrowicz, G., Alexander, T., Amy, M., Anagolum, S., Arbel, E., Asfaw, A., Athalye, A., Avkhadiev, A., Azaustre, C., Banerjee, A., Banerjee, S., Bang, W., Bansal, A., Barkoutsos, P., Barnawal, A., Barron, G., Barron, G. S., Bello, L., Ben-Haim, Y., Bevenius, D., Bhatnagar, D., Bhobe, A., Bianchini, P., Bishop, L. S., Blank, C., Bolos, S., Bopardikar, S., Bosch, S., Brandhofer, S., Brandon, Bravyi, S., Bronn, N., Bryce-Fuller, Bucher, D., Burov, A., Cabrera, F., Calpin, P., Capelluto, L., Carballo, J., Carrascal, G., Carvalho, I., Chen, A., Chen, C.-F., Chen, E., Chen, J. C., Chen, R., Chevallier, F., Cholarajan, R., Chow, J. M., Churchill, S., Claus, C., Clauss, C., Clothier, C., Cocking, R., Cocuzzo, R., Connor, J., Correa, F., Cross, A. J., Cross, A. W., Cross, S., Cruz-Benito, J., Culver, C., Corcoles-Gonzales, A. D., D, N., Dague, S., Dandachi, T. E., Dangwal, A. N., Daniel, J., Daniels, M., Dartiailh, M., Davila, A. R., Debouni, F., Dekusar, A., Deshmukh, A., Deshpande, M., Ding, D., Doi, J., Dow, E. M., Drechsler, E., Dumitrescu, E., Dumon, K., Duran, I., EL-Safty, K., Eastman, E., Eberle, G., Ebrahimi, A., Eendebak, P., Egger, D., Espiricueta, A., Everitt, M., Facoetti, D., Fernandez, P. M., Ferracin, S., Ferrari, D., Ferrera, A. H., Fouilland, R., Frisch, A., Fuhrer, A., Fuller, B., GEORGE, M., Gacon, J., Gago, B. G., Gambella, C., Gambetta, J. M., Gammanpila, A., Garcia, L., Garg, T., Garion, S., Gates, T., Gil, L., Gilliam, A., Giridharan, A., Gomez-Mosquera, J., Gonzalo, Gonzalez, S. d. l. P., Gorzinski, J., Gould, I., Greenberg, D., Grinko, D., Guan, W., Gunnels, J. A., Gupta, N., Günther, J. M., Haglund, M., Haide, I., Hamamura, I., Hamido, O. C., Harkins, F., Hasan, A., Havlicek, V., Hellmers, J., Herok, *n*., Hillmich, S., Horii, H., Howington, C., Hu, S., Hu, W., Huang, J., Huisman, R., Imai, H., Imamichi, T., Ishizaki, K., Ishwor, Iten, R., Itoko, T., Javadi, A., Javadi-Abhari, A., Javed, W., Jivrajani, M., Johns, K., Johnstun, S., Jonathan-Shoemaker, JosDenmark, JoshDumo, Kachmann, T., Kale, A., Kanazawa, N., Kane, J., Kang-Bae, Kapila, A., Karazeev, A., Kassebaum, P., Kelso, J., Kelso, S., Khanderao, V., King, S., Kobayashi, Y., Kovyrshin, A., Krishnakumar, R., Krishnan, V., Krsulich, K., Kumkar, P., Kus, G., LaRose, R., Lacal, E., Lambert, R., Lapeyre, J., Latone, J., Lawrence, S., Lee, C., Li, G., Liu, D., Liu, P., Maeng, Y., Maheshkar, S., Majmudar, K., Malyshev, A., Mandouh, M. E., Manela, J., Manjula, Marecek, J., Marques, M., Marwaha, K., Maslov, D., Maszota, P., Mathews, D., Matsuo, A., Mazhandu, F., McClure, D., McElaney, M., McGarry, C., McKay, D., McPherson, D., Meesala, S., Meirom, D., Mendell, C., Metcalfe, T., Mevissen, M., Meyer, A., Mezzacapo, A., Midha, R., Minev, Z., Mitchell, A., Moll, N., Montanez, A., Monteiro, G., Mooring, M. D., Morales, R., Moran, N., Morcuende, D., Mostafa, S., Motta, M., Moyard, R., Murali, P., Müggenburg, J., Nadlinger, D., Nakanishi, K., Nannicini, G., Nation, P., Navarro, E., Naveh, Y., Neagle, S. W., Neuweiler, P., Ngoueya, A., Nicander, J., Nick-Singstock, Niroula, P., Norlen, H., NuoWenLei, O'Riordan, L. J., Ogunbayo, O., Ollitrault, P., Onodera, T., Otaolea, R., Oud, S., Padilha, D., Paik, H., Pal, S., Pang, Y., Panigrahi, A., Pascuzzi, V. R., Perriello, S., Peterson, E., Phan, A., Piro, F., Pistoia, M., Piveteau, C., Plewa, J., Pocreau, P., Pozas-Kerstjens, A., Pracht, R., Prokop, M., Prutyanov, V., Puri, S., Puzzuoli, D., Perez, J., Quintiii, Rahman, R. I., Raja, A., Rajeev, R., Ramagiri, N., Rao, A., Raymond, R., Reardon-Smith, O., Redondo, R. M.-C., Reuter, M., Rice, J., Riedemann, M., Risinger, D., Rocca, M. L., Rodr guez, D. M., RohithKarur, Rosand, B., Rossmannek, M., Ryu, M., SAPV, T., Saha, A., Saki, A. A., Sandberg, M., Sandesara, H.,

Sapra, R., Sargsyan, H., Sarkar, A., Sathaye, N., Schmitt, B., Schnabel, C., Schoenfeld, Z., Scholten, T. L., Schoute, E., Schulterbrandt, M., Schwarm, J., Seaward, J., Sertage, I. F., Setia, K., Shah, F., Shammah, N., Sharma, R., Shi, Y., Shoemaker, J., Silva, A., Simonetto, A., Singh, D., Singh, P., Singkanipa, P., Siraichi, Y., Siri, Sistos, J., Sitdikov, I., Sivarajah, S., Sletfjerding, M. B., Smolin, J. A., Soeken, M., Sokolov, I. O., Sokolov, I., SooluThomas, Star sh, Steenken, D., Stypulkoski, M., Suau, A., Sun, S., Sung, K. J., Suwama, M., S*n*lowik, O., Takahashi, H., Takawale, T., Tavernelli, I., Taylor, C., Taylour, P., Thomas, S., Tillet, M., Tod, M., Tomasik, M., Torre, E. d. l., Toural, J. L. S., Trabing, K., Treinish, M., Trenev, D., TrishaPe, Truger, F., Tsilimigkounakis, G., Tulsi, D., Turner, W., Vaknin, Y., Valcarce, C. R., Varchon, F., Vartak, A., Vazquez, A. C., Vijaywargiya, P., Villar, V., Vishnu, B., Vogt-Lee, D., Vuillot, C., Weaver, J., Weidenfeller, J., Wieczorek, R., Wildstrom, J. A., Wilson, J., Winston, E., WinterSoldier, Woehr, J. J., Woerner, S., Woo, R., Wood, C. J., Wood, R., Wood, S., Wootton, J., Yang, B., Yeralin, D., Yonekura, R., Yonge-Mallo, D., Young, R., Yu, J., Yu, L., Zachow, C., Zdanski, L., Zhang, H., Zoufal, C., aeddins-ibm, b63, bartek-bartlomiej, bcamorrison, brandhsn, catornow, charmerDark, deeplokhande, dekel.meirom, dime10, ehchen, fanizzamarco, fs1132429, gadial, galeinston, georgios-ts, gruu, hhorii, hykavitha, jliu45, jscott2, klinvill, krutik2966, ma5x, michelle4654, msuwama, ntgiwsvp, ordmoj, pahwa, s., pritamsinha2304, saswati-qiskit, sethmerkel, shaashwat, sternparky, strickroman, tigerjack, tsura-crisaldo, welien, willhbang, yang.luh and Cepulkovskis, M. (2021). Qiskit: An Open-source Framework for Quantum Computing.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python, *Journal of Machine Learning Research* **12**: 2825{2830.