A REPORT

### ON

**QUANTUM COMPUTING SOFTWARE**

BY

Name(s) of the ID.No.(s)

Student(s)

Abhinav Bandaru 2018A7PS0236H

Amulya Dwivedi 2018B5A30035P

Soumya Pochiraju 2018B5A70934H

Sarthak Kasat

Utkarsh Dwivedi 2018A7PS0131P

AT

TNSTC

A Practice School-I Station of

### BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

**(June, 2020)**

### A REPORT ON

**QUANTUM COMPUTING SOFTWARE**

BY

Name(s) of the ID.No.(s) Discipline(s) Student(s)

Abhinav Bandaru 2018A7PS0236H B.E. (Hons.) Computer Science

Amulya Dwivedi 2018B5A30035P M.Sc. Physics + B.E. (Hons.) EEE

Soumya Pochiraju 2018B5A70934H M.Sc. Physics + B.E. (Hons.) CS

Sarthak Kasat

Utkarsh Dwivedi 2018A7PS0131P B.E. (Hons.) Computer Science

Prepared in partial fulfillment of the Practice School-I Course Nos.

BITS C221/BITS C231/BITS C241 AT

TNSTC A Practice School-I Station of

### BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

**(June, 2020)**

**ACKNOWLEDGEMENTS**

We would also like to thank Birla Institute of Technology and Science for providing us with this opportunity to enhance our skills and work on this amazing project.

We would also like to thank Dr. Vinti Agarwal, who provided us with constant support during the entire project. We also Thank Pradheep Sir for sharing his expertise, guiding us and supporting us in this endeavour.

We would also like to thank our parents for supporting us throughout the project.

Finally, we thank IBM for providing students like us access to the IBM Quantum Computer and providing a platform where we could test our project. We also thank TNSTC for providing us this opportunity to work on such an amazing project.

### BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI (RAJASTHAN)

**Practice School Division**

**Station**: TNSTC **Centre:** Chennai

**Duration** 6 Weeks **Date of Start**: 18th May, 2020

**Date of Submission** 25th June, 2020

### Title of the Project: Quantum Computing Software

**ID No./Name(s)/ Discipline(s)/of the student(s)**

2018A7PS0236H/Abhinav Bandaru/B.E. (Hons.) Computer Science

2018B5A30035P/Amulya Dwivedi/M.Sc. Physics + B.E. EEE

2018B5A70934HSoumya Pochiraju/M.Sc. Physics + B.E. (Hons.) CS

**Name(s) and designation(s) of the expert(s):**

**Name(s) of the PS Faculty:**

Dr. Vinti Agarwal

**Key Words:** Quantum Computing, qiskit, pyquil, Entanglement, Superposition

**Project Areas:** Quantum Computing

Signature(s) of Student(s) Signature of PS Faculty

Date Date

**ABSTRACT**

This is a study of the speed and accuracy of current quantum supervised machine learning algorithms on a quantum computer and simulator, and the classical supervised machine learning algorithms. We compared many supervised ML algorithms and chose to study SVM (rbf kernel) because it had the best accuracy and we were limited by time. As the IBM Quantum Computer allows us to use only 5 qubits remotely, we reduced our features from 8 to 4. This reduced the overall accuracy by 0.2 % on the classical computer. The overall accuracy was 75.4% for 768 tuples.

The **Quantum SVM** **works not as an efficient or faster substitute** method to the classical SVM but as an addition to itin an effort to expand the domain where classical SVM is applicable. The QSVM method is appropriate in the cases where the quantum feature map that is required for classification is not easily simulatable classically. In those cases, running the algorithm on an actual quantum machine will provide a quantum advantage. The results on the simulator are as follows:

Feature map used: Raw Feature Map

Testing accuracy (614 samples, 0.2 split of total 768 samples for training and testing): 0.6364

Feature map used: Second Order Expansion

Testing accuracy (300 samples): 0.5212

We justify our low accuracies on three arguments: not all kernels are fit for classification of all datasets; the simulator is built on classical machines and hence, only takes in assumptions that we know about; we only used one shot as it was taking a long time to get back the result.

We also attempted to run the experiment on an actual quantum machine offered by IBM Q Experience. Unfortunately, long queue times and a limited number of parallel jobs by IBM restricted us from feeding the entire dataset, and hence, the results (accuracy lower than 50%) were not good.

We also designed an intricate **Hadamard Classifier**.

This is basically a simple interference circuit that implements a distance based binary classifier, based on the paper by Maria Schuld[7]. After the state preparation routine, the circuit consists only of a hadamard gate and 2 measurements. We implemented the circuit using the qiskit library, on the iris dataset, and we see that this classifier works very well with simple, normalized data.

**CONTENTS**

|  |  |
| --- | --- |
| TITLE | PAGE NUMBER |
| Introduction | 6 |
| Main Text | 7 |
| Conclusion | 12 |
| References | 14 |
| Glossary | 14 |

**INTRODUCTION**

In the medical field, there always exists a need to provide accurate and fast results. It takes several years to get a proper understanding of the effect of one drug in combination with others. Quantum computing can significantly shorten the period, as it has enough computational power to visualize all the possible outcomes. The potential of quantum computing in discovering drugs to target specific types of cancer prominently contributes to the growth of its use in the healthcare industry.

With our conventional computers fast approaching their size limit, they can’t be made any smaller. A general purpose transistor today is only 7 nm, manifolds smaller than the HIV virus!  
At sizes like these , the electrons start to show quantum effects like tunneling and teleportation, limiting our computational potential, however Quantum computing takes advantage of these very phenomena only to give exponentially faster results.

Its combination with the well-known techniques of ML gives us Quantum ML, which is understood to have a huge untapped potential.

Our Project is basically a comparative study that compares the speeds and accuracies of different ML and QML algorithms when applied on the same dataset, in order to evaluate the true potential of Quantum ML.

For the purposes of this project, we have chosen Support Vector Machine as it best fit our data and had the highest accuracy.

**Main Text**

**Data Set:**

We have chosen the **Pima Indian Diabetes dataset** for this purpose. This dataset gives a problem of ***binary classification***.

It contains **8 input parameters, namely: Pregnancies**(no. of times pregnant)**, Glucose**(Plasma glucose concentration a 2 hours in an oral glucose tolerance test**), Blood Pressure**(Diastolic blood pressure (mm Hg))**, Skin thickness**(Triceps skin fold thickness (mm))**, Insulin** (**2-Hour serum insulin (mu U/ml)), BMI (Body mass index (weight in kg/(height in m)^2) Diabetes Pedigree Function and Age.**

These are to be used to predict whether the given case is a **diabetic or a non diabetic.**

The dataset contains a total of 768 records of which 268 belonged to the class ’1’ i.e. **Diabetic** whereas the rest belonged to ‘0’, meaning **Non Diabetic**.

**Choosing an Appropriate Algorithm:**

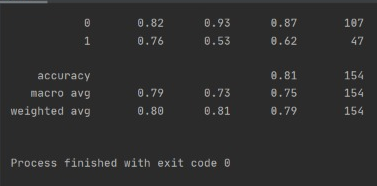
We considered the following 3 algorithms for the given classification problem:

***K Nearest Neighbours:***Since the dataset contains many complicated features, and since it involves calculation of Euclidean distance between each of the training examples, we did not expect it to be very fast. To speed up this computation if we would have reduced the number of neighbours being considered for the process, we would end up with **unreliable and underfit** results. Similarly, increasing them would decrease the speed and would **overfit** the training data.

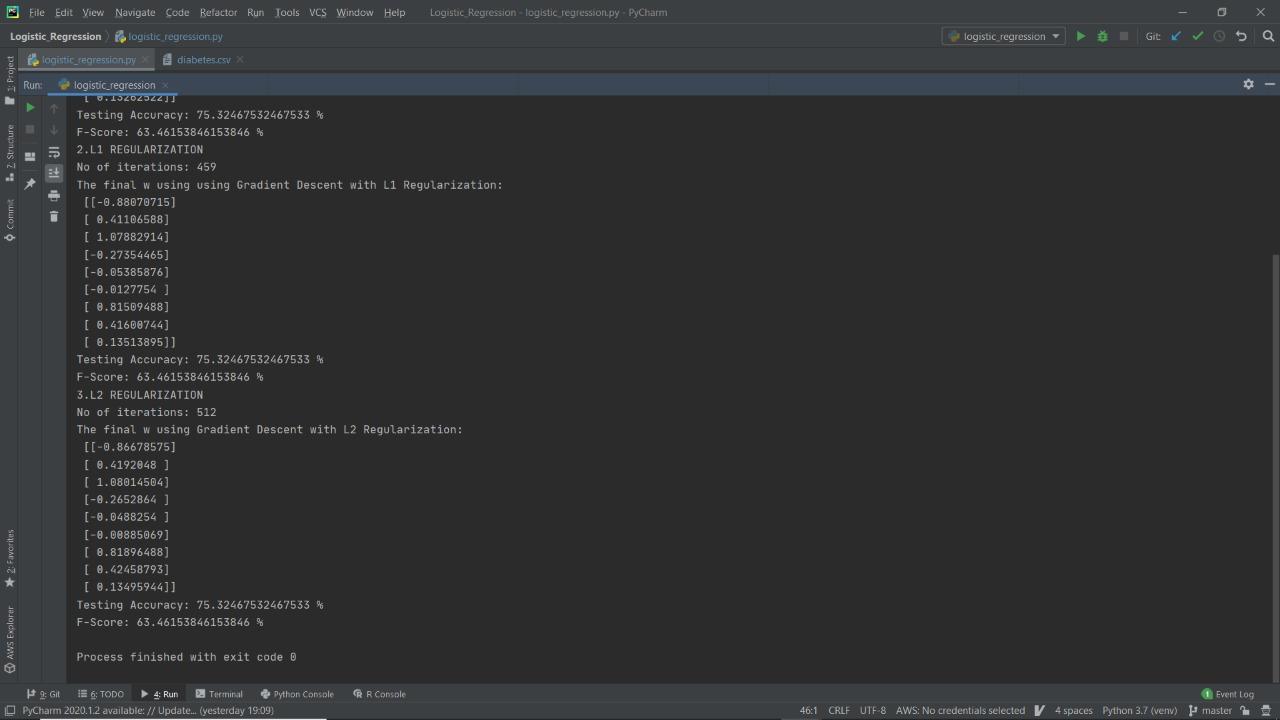
***Logistic Regression/Support Vector Machines:***

Both of the algorithms have similar pros and cons and we expected their performance to be comparable. We only had the option of running both the algorithms and seeing which one fits our particular dataset better.

Below are the results found when the two were run multiple times on our dataset:



**Results found from SVM (Using rbf kernel)**



**Results found from Logistic Regression (Using L1, L2 regularization)**

As can be clearly seen, both the algorithms have a comparable accuracy, however the Support Vector Machine algorithm fits the dataset better.

Also SVM being quite feasible and implementable (because of the **presence of a special library in Qiskit** for it) in quantum systems, made it even more preferable.

**Procedure/Method:**

We utilized the famous Python library **Qiskit** for quantum machine learning. We used both a local quantum computer simulator and a real quantum chip as our backend. We applied **two versions of SVM**, a **simple, classical, kernel based SVM** on a classical computer that utilized a regular CPU **and its quantum based implementation** for the required binary classification.

Implementing the classical version was no trouble at all, and was implemented easily. However the Quantum version first had to be tested on a simulator and only when all resultant bugs were fixed, was sent to IBM-Q servers for implementation on a real Quantum chip. We had to face long queues for this and sometimes uncontrollable noise tended to corrupt our findings.

Apart from this our team also designed an intricate **Hadamard Classifier** to bring out more clarity on how quantum circuits are designed and work, inside out. This classifier has elucidated the **Reversible Computing** that goes on inside a Quantum system as opposed to the classical system where almost all operations are irreversible in nature. What we mean by this is that, **in quantum computing the output bit value can be processed to give back the input bit values.**

**Implementation of the Classical SVM:**

We implemented two kinds of codes for our classification system. The first one to test the accuracy of the classifier and the second one which was deployed to a website for users to interact with. Both the codes use the library function **SVC** which is a part of the scikit-learn library. They’re both trained and tested using the **‘rbf’ kernel**[[11](https://en.wikipedia.org/wiki/Radial_basis_function_kernel)].

For calculating the accuracy of the classifier, we split the data in a **2:8 train-test ratio**, i.e 80% for training the data and 20% for testing the data. **The highest accuracy achieved was 80% and the average accuracy was 75.6%.**

Since, the maximum number of features allowed on the IBM Quantum Computer are 5, we used the **chi squared method**[[10](https://towardsdatascience.com/chi-square-test-for-feature-selection-in-machine-learning-206b1f0b8223#:~:text=In%20feature%20selection%2C%20we%20aim,hypothesis%20of%20independence%20is%20incorrect.)] to reduce the number of features from **8 to 4**. **This increased the highest accuracy achieved of the classical classifier to 82% and lowered the average accuracy to 75.4%**. The features that this data is based upon are, **Glucose levels, Insulin Levels, BMI and the Diabetes Pedigree function**. The same features were used for the quantum implementation whose details are given below.

**Implementation of the Quantum SVM:**

We used the method of **supervised learning** using **quantum-enhanced feature space method** [[6](https://arxiv.org/abs/1804.11326)] for classification of the chosen dataset. Qiskit already provides a library for the implementation of this method.

The **Quantum SVM** **works not as an efficient or faster substitute** method to the classical SVM **but as an addition to it** in an effort to expand the domain where classical SVM is applicable. The level of accuracy of classical SVM depends on the kernel chosen amongst the numerous kernels available (RBF, Polynomial etc.), most of the popular ones are efficiently implementable classically, but the paper makes the point that in some cases when the dataset’s complexity extends beyond that of the classical kernel’s ability then a quantum feature map might provide the solution. A **quantum feature map** is a mapping of classical data to different dimensions with the help of quantum circuits. **This QSVM method is appropriate in the cases where the quantum feature map that is required for classification is not easily simulatable classically.** In those cases, running the algorithm on an actual quantum machine will provide a quantum advantage.

Hence, because of the difference of this novel approach from the classical SVM, the **classification task was done only in the capacity of a study and not as actual comparison to the classical SVM implemented above, either in terms of speed or accuracy.**

**Details of the experiment (on ibm\_qasm\_simulator, 1 shot):**

Feature map used: **Raw Feature Map**

Testing accuracy (614 samples, 0.2 split of total 768 samples for training and testing): **0.6364**

Feature map used: **Second Order Expansion**

Testing accuracy (300 samples): **0.5212**

We can justify our low accuracy in the **Raw Feature Map** and even lower accuracy in the **Second Order Expansion** by three arguments. Firstly, we may assume that just like not all kernels are fit for classification of all datasets (each one is appropriate for some class of distribution of data), similarly quantum feature maps are not universal in that for a certain data distribution they may serve better and if they are not easily simulatable classically, then they may even provide a quantum advantage in terms of efficiency. Secondly, while IBM built a powerful simulator, it is finally built on a classical machine and can at most try to simulate a quantum environment and it must also make approximations regarding the same, all of which may contribute to low quality implementation of the QSVM algorithm, hence the accuracy. Thirdly, and most importantly, due to enormous time taken in executing only one shot of the algorithm, we contended with it, laying this study as experimental. We’re sure that if time permitted, increased shots would have led to better accuracy.

We also attempted to run the experiment on an actual quantum machine offered by IBM Q Experience. Unfortunately, long queue times and limit on number of parallel jobs by IBM restricted us from feeding the entire dataset, yet observing the kernels given out by the quantum machine on few samples on the same algorithm we ran on the simulator, and comparing it with the kernels calculated by simulator on the same samples, we observed correlations between the results of the actual quantum backend and the simulator. Hence, we could positively hope that the algorithm would run similarly on real backends, and would perhaps give better results if the new accompanying problems that come with operating on real quantum computers (like noise levels) are handled properly.

**Implementation of the Hadamard Classifier:**

We implemented a Hadamard gate distance based binary classifier based on a paper by **Maria Schuld** [[7](https://arxiv.org/pdf/1703.10793.pdf)]. This is a simple interference circuit, where the classical data is encoded using **amplitude encoding**. It can be viewed as a linear kernel. Depending on the way the data encoding is done, we can also generate more complex polynomial kernels. Thus, this circuit is a simple way to understand how quantum kernels can be computed. The main advantage of computing kernels through quantum circuits is that since quantum states already exist in a high dimensional **Hilbert space**, computation of high dimensional vectors’ dot products requires much less number of operations than the classical method.

In this circuit, amplitude encoding is done in such a way that the attributes of the classical vectors are stored in the amplitudes of the quantum state vector. So for n attributes, we will need **log2(n) qubits** for the attributes. And for m number of data points, we will need **log2(m)** number of qubits. So the total number of qubits required for m training vectors with n features each would be **log2(m) + log2(n) +2**. The encoding is done by performing rotation operations that rotate a zero state ket vector to the desired state vector.

Once the state preparation routine is done, a hadamard gate is applied on the ancilla qubit which interferes with the vectors and computes the distance. Then, we perform a conditional measurement on the class label qubit the ancilla collapses to the zero state.This conditional measurement where we only select cases where the qubit has collapsed to one state is called **post selection**. Here, post selection succeeds with a probability of around 0.5 when the data has been scaled and normalized. The class label qubit is a sum of the labels(0 or 1) weighted by the distances of the vectors to the unclassified data point. Since this is a probabilistic measurement, we need to repeat the measurement multiple times and then choose the label that occurs with maximum probability.

The circuit we created **takes in 4 training vectors and classifies one unclassified test vector.** The circuit for 4 training points requires **one ancillary qubit, 2 index qubits, one data encoding qubit and one qubit** for the class label. Each vector has 2 attributes. It works on the **Iris dataset.** When the data points are scaled and normalized, it has an accuracy of almost 100% for the first to classes of Irises.

The problems with implementing this classifier on near-term quantum computers is that as the number of datapoints and features increases, the number of gates required increases polynomially. This is because the rotations have to be **multiple controlled rotation** operations, and each of these operations is decomposed into simple single qubit gates, which do not grow linearly as the number of control qubits increases. As the number of gates increases, the noise increases, and our calculations get destroyed by the noise. Another problem might be that many qubits are required to implement this for a large number of data points.

**CONCLUSION**

By our study, we attempted different approaches to solve the same problem and tried to explore how the new and developing field of Quantum Computation could help us in that. We reserved the implementation of classical SVM as a base method that is used in classification tasks, tried a novel approach QSVM to perform the same classification and finally touched on another quantum approach called the Hadamard Classifier.

The most important realisation that we arrived on was that the advances in Quantum Computing can be incorporated into the domain of Machine Learning, mostly to fill holes in the areas where classically implemented ML algorithms are not that successful and it, in no way, is in any foreseeable future, is going to replace the classical computation.

As we saw, our QSVM implementation was not able to perform at par to the classical counterpart in diabetes classification, it illustrates that quantum algorithms (QSVM in this case) musn’t be employed in tasks with low level of complexity or low dimensional data, for the classical implementations are best those tasks. **Quantum implementations of these tasks turn out to be less efficient because of the high maintenance required of quantum systems.** Hence, quantum computing solutions shouldn’t be viewed as universal instead applicable to a subset of problems.

At the same time, we clearly observed the difficulty of simulating quantum environments classically (by noting the enormous time it takes for the simulator to execute quantum circuits), hence it is safe to say that for the data whose processing can be easily done by quantum methods, and there must be data such as this, for that classical solutions will prove to be sub-par, as they are inefficient at simulating quantum environments, because who can do it better than a quantum system itself! In relation to our study, namely for the QSVM, for the data which is sufficiently complex such that its classification requires quantum feature maps that are not easily simulatable classically (as complicated quantum feature maps often are), then running them on quantum systems will prove to be efficient. This is where quantum computing will be advantageous

We conclude our study by laying down our vision of quantum computing, especially in tasks of machine learning. In the domain of machine learning, due to the enormous data that sometimes needs processing, quantum computing might prove out to be useful. But, due to the extreme conditions required for operation of quantum computers, it is highly unlikely that they will ever become a household device, yet at the same time seeing huge investments in this area brings hope that they will become more accessible through the cloud. Even then, quantum computing excels at a fixed domain of problem solving, while is, bluntly put, inefficient at best, to the other tasks. So, classical computing will continue to dominate in areas of **simple computation** (simple, as in which has no or negligible part in quantum effects like entanglement, interference etc) but the most notable solutions will be achieved when both quantum and classical computing work in collaboration to the best of their abilities.

**REFERENCES**

1. <https://qiskit.org/documentation/apidoc/aqua.html>
2. <https://scikit-learn.org/stable/modules/svm.html>
3. <https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html>
4. <http://docs.rigetti.com/en/stable/>
5. <https://ai.googleblog.com/2020/03/announcing-tensorflow-quantum-open.html>
6. <https://arxiv.org/abs/1804.11326>
7. <https://arxiv.org/pdf/1703.10793.pdf>
8. <https://www.nature.com/articles/s41534-020-0272-6>
9. [Comparisons of Performance between Quantum and Classical ML](https://scholar.smu.edu/cgi/viewcontent.cgi?article=1047&context=datasciencereview)
10. [Chi-square test for feature selection](https://towardsdatascience.com/chi-square-test-for-feature-selection-in-machine-learning-206b1f0b8223#:~:text=In%20feature%20selection%2C%20we%20aim,hypothesis%20of%20independence%20is%20incorrect.)
11. <https://en.wikipedia.org/wiki/Radial_basis_function_kernel>
12. <https://scholar.smu.edu/cgi/viewcontent.cgi?article=1047&context=datasciencereview>

**GLOSSARY**

Euclidean distance: the ordinary straight-line distance between two points in Euclidean space.

K Nearest Neighbours: the k-nearest neighbours algorithm is a non-parametric method used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space.

Logistic Regression: is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary)

Support Vector Machines: are supervised learning models with associated learning algorithms that analyse data used for classification and regression analysis.

Irreversible Operations: When output bit cannot give back the corresponding input bits, it is called Irreversible Computing, for example, if 1 & 0 get passed through an AND gate and give us ‘0’ at the output bit, we will never be able to decipher the exact combination of input that resulted in this as this 3 possible combinations are there to get 0 output in this case.

Normalizing the data: Fit the data to a normal distribution

Scaling the data: Scale down the norm of each vector to 1. This does lead to loss of information of one dimension, but if used in the appropriate cases, it does not affect the performance of the classifier

Feature Map: A Feature Map maps the original vectors to a higher dimensional feature space.