Università Commerciale
Luigi Bocconi

Final work for the course BEMACS:
BSc in Economics, Management and Computer Science

Quantum Machine Learning: Python Implementation of a Hybrid Algorithm in Qiskit

Marco Palermo
October 2019

Abstract:
This research tackles the topic of *Quantum Machine Learning*, the application of quantum devices to the field of Computer Science that allows computers to learn from data, without being explicitly programmed. I have decided to investigate this sector because I believe quantum technology will be game-changing in the near future, so that keeping up with it is crucial for any data scientist. The goal of this work is to understand how Quantum Computing can help carry out Machine Learning techniques, by using a classical-quantum hybrid model to speed-up computations. This paper, divided in four section, starts by giving a general introduction on the theory behind the Quantum Computing technology. Then it describes the state of the art of the quantum hardware and software, providing examples of the latest improvements in both fields.
Finally, in the last section of the thesis, I deliver my own implementation of a quantum Support Vector Machine (SVM) Classifier. In order to learn how to implement algorithms that run on a quantum device, first I studied the literature on kernel methods and quantum-enhanced supervised learning techniques, then I implemented the algorithm, using the *Python* programming language and the *Qiskit* library, an open source software development kit for quantum research, provided by *IBM*.
The algorithm obtains an accuracy that is lower than modern classifiers used on mainstream computers. This is because the code provided by IBM only supports data with at most three features, so that I had to reduce the dimensionality of the datasets before feeding it to the QSVM. Taking into account this limitation, the classifier achieved very good result on the datasets, but hopefully with the introduction of more powerful quantum devices in the upcoming years, there will be a tremendous spike in the performance of the algorithm.

Acknowledgments:

I would like to express my gratitude to the three amazing mentors who helped me with this work, I dedicate my thesis to them, with great respect and appreciation.
First of all, I warmly thank Professor Emanuele Borgonovo, Director of the Bachelor's Degree in Economics, Management and Computer Science, for giving me the true honor of being my Supervisor and for teaching me the basic skills to undertake this work.
Special thanks go to Co-supervisor Professor Alessandro Rezzani, senior Business Intelligence consultant, specialized in predictive analytics and big data applications, who supported me with great availability and professionalism during all the phases of my thesis. Without his precious help and guidance this work wouldn’t exist.
Finally, I would like to express my deepest thanks to Professor Pierpaolo Marturano, CEO & Founder of *dotQuantum* and great expert on Quantum Computing, who gave me a lot of advice and practical suggestions.
All the people mentioned above played a really important role, but any error or inaccuracy is to be attributed to me only.

A classical computation is like a solo voice-one line
of pure tones succeeding each other.
A quantum computation is like a symphony-many lines
of tones interfering with one another.

Seth Lloyd

Index:

Keywords:

Quantum Computing, Quantum Machine Learning, Python, Qiskit, Kernel Methods, Quantum Support Vector Machine.

Premise:

My final work aims to explore the emerging field of *Quantum Machine Learning*, in order to get familiarity with a technological miracle that has the potential to completely reshape data analysis as we know it. The **purpose** of this work is twofold. On the one hand, this thesis aims at understanding the state of the art of the discipline, by studying the theory behind quantum machine learning and the literature surrounding the field. On the other hand, it wishes to analyse real-world applications, by trying out a quantum algorithm on a dataset and studying its performance.

In the **1st Section** of my thesis, in order to introduce the reader to the topic of Quantum Machine Learning, I give a general theoretical introduction to its two main components: the Quantum Computer and the field of Machine Learning. Then, in order to justify the choice of this topic for my thesis, I explain some of the incredible breakthroughs Quantum Machine Learning could bring in the near future, if full-fledged quantum devices became feasible. The amazing innovations Quantum Computers are likely to achieve are not limited to machine learning alone, so I indulge in a digression in order to describe the game-changing potential it has in other disciplines, namely Cryptography and Chemistry.

In the **2nd Section** of my work I talk about the importance of the quantum hardware, and I describe the two main methods used to build quantum devices: the Quantum Annealing Machine, mainly manufactured by the company D-Wave Systems, and the gate-based Quantum Devices, like the ones developed by IBM, Intel and Google. Finally, I explain what is a Topological Quantum Computer, and why if researchers figured out how to build it, this new hardware implementation might overtake today's superconducting technology altogether.

In the **3rd Section** I address the Quantum Software. First, I list the main requirements needed to program on a quantum computer: writing

quantum algorithms is especially challenging because it requires an understanding of Quantum Physics, Linear Algebra and Computer Science.
Then, I analyse the software technology available for quantum computers, most of which is freely available on the internet, so that anyone can program on an actual quantum computer in the Cloud.
In the **4th Section** I describe and implement my own quantum algorithm.
I decided to program the code for a quantum-enhanced classifier, the Quantum Support Vector Machine (QSVM). It's a classical-quantum hybrid algorithm that exploits the power of kernel methods to carry out the computations, processing the information on a quantum device and then extracting classical data from the quantum state.
I implemented the algorithm in *Python*, using the *Qiskit* (Quantum Information Science Kit) library, from *IBM*. Then, I tested my classifier on two famous datasets: the *Breast Cancer* and the *Iris* datasets from sklearn, in order to verify its performance.

Introduction:

In the last thirty years, quantum studies have gone from individual experiments on Quantum Physics performed by some bright minds, mostly Nobel Prize winners, to extensive interdisciplinary applied studies, with the USA, Russia and China rushing to conquer quantum supremacy.
Quantum mechanics is no longer a mere theoretical field, but rather the groundwork for many real-word applications, born from the collaboration of experts in different applied research fields, including computer scientists, mathematicians, statisticians, physicists and data scientists.
In particular there have been stunning improvements in *Quantum Machine Learning*, the subfield of *Quantum Computing* that processes quantum information to extract meaningful insights from a dataset.
In the near future, thanks to the progress achieved both in the quantum hardware and quantum software technologies, we could witness many new fascinating applications, which could revolutionize our daily lives, radically changing the rules of healthcare, manufacturing, social media, retail and finance.

1st Section – Introduction to Quantum Machine Learning

1.1 The Quantum Computer:

A **quantum computer** is a powerful machine that makes use of some peculiar phenomena of *Quantum Mechanics*, like entanglement and interference, in order to perform computations in an incredibly efficient and fast way. What distinguishes a quantum computer from a traditional digital device is that the basic units of information are **qubits** (quantum bits), rather than bits. These fundamental quantum building blocks have properties that transcend the binary architecture used in mainstream computers, providing an entirely new type of computation.

If we were to use classical algorithms on a quantum computer, its performance would be almost indistinguishable from the one of a regular computer. In order for a quantum computer to show its exceptional superiority it needs to use **quantum circuits**, algorithms that only run on quantum devices and can exploit the phenomena of *Quantum Mechanics*.

The unique and intriguing properties of quantum computers make them particularly effective at solving some **specific problems** and computational tasks that cannot be performed with conventional IT.

Two pioneering works in this field were Grover and Shor's algorithms[1], which represent an exceptional breakthrough, working in a quadratically and exponentially faster way than classical algorithms respectively.

Grover's algorithm (1996) is a true milestone for *Quantum Computing*, it is a search algorithm, namely an algorithm that is designed to find a specific element in a dataset. It employs an ingenious procedure that progressively increases the probability of observing the target item, achieving a computational time of order $O(\sqrt{n})$, a quadratic speed-up over all classical search methods. **Shor's algorithm[2]** (1994) on the other hand, uses the effects of quantum parallelism to get to the result of the prime factorization problem in just a few seconds, even when confronted with complex situations, where a classical computer would take years to output the result. The code has a time complexity that is $O((log\ n)^3)$.

[1] Kraker J., Valle C., *Shor's algorithm and Grover's algorithm in Quantum Computing*, Proquest, 2011.
[2] http://www-math.mit.edu/~shor/papers/Progress.pdf

Grover's and Shor's algorithms have been proven both mathematically and empirically, establishing undeniable evidence of their exceptional significance for the quantum research. Despite the great benefit of increased efficiency, quantum computation poses a major challenge as well: the problem of **decoherence**[3]. In fact, the wave-like nature of quantum particles makes them especially sensitive to the environment around them. If the particles are not perfectly isolated from the external environment, they tend to lose their quantum state and produce errors. Protecting quantum information from the loss of coherence is a challenging issue, because in order to measure the qubits, a quantum computer needs to interact with its surroundings, resulting in a dispersion of quantum information. The problem of decoherence still affects physicists today, so that any quantum measurement must be repeated several times, in order to extract a useful result, that can be interpreted using mainstream computing technology.

Despite this problem, it is undeniable that the processing of quantum information offers huge advantages, even though it is still in its infancy.

In the last years, Quantum Computing technology has improved at a very fast pace: there were significant advances in quantum error correction, breakthroughs in building quantum computers and the discovery of several new quantum algorithms and coding techniques. The **future** of *Quantum Computing*[4] is looking bright and we can only imagine how many more enhancements and metamorphoses there will be in this exciting and promising field.

Table 1: Strengths and Weaknesses of Quantum Computers

Strengths:	Weaknesses:
Extraordinary power	Sensitivity to interference with environment
High speed	Difficulties in state preparation
Fast method of factorization	Quantum statistical constraints
Multiple transactions simultaneously	Issues in error diagnosis, recovery and correction
Capacity to solve hitherto intractable tasks	Uncertainty and entropy of quantum gates

[3] Schlosshauer M.A., *Decoherence and the Quantum-to-Classical Transition,* Springer, Berlin, 2007.
[4] Nielsen M.A., Chuang I.L., *Quantum Computation and Quantum Information*, Cambridge University Press, 2010.

1.2 Machine Learning:

Machine Learning (ML) is the field of *Computer Science* that acquires and processes information to learn from stored data.

ML emulates biological architectures like the structure of the human brain, in order to create algorithms that learn to perform a specific task, without being explicitly programmed to do so, employing processes of inference and pattern recognition that resemble the way people acquire knowledge and experience. It is based on **mathematical models**, in fact the algorithms use optimization as their most essential element to train the computer to reach its objective, through an incentive system based on iterative punishment and reward.

The training data necessary for the development of the algorithms often requires the use of other coding practices for extraction and preprocessing, so that **Data Mining** and **Data Transformation techniques**[5], like *Scaling* and *Principal Component Analysis* (PCA), represent an initial fundamental step in many of these algorithms.

According to the nature of the data, we usually classify Machine Learning algorithms into two main classes:

- **Supervised Learning Algorithms**, which process labeled data, in order to assign new unseen data to existing previously known categories. Examples of real-world applications include speech recognition, DNA sequence classification and spam emails detection.
- **Unsupervised Learning Algorithms**, which use unlabeled data, processing them without any indication of the desired result nor prior training, in order to extract insights and patterns. These coding tools can be used to visualize important relationships in the house market, identify crime locations, understand handwritten digits or perform customer segmentation.

Despite its cutting-edge and effective applications, *Machine Learning* still has some **limitations**. It's mostly in the area of Unsupervised Learning that classical computers are still struggling to produce good results. On

[5] Witten I.H., Frank E., Hall M.A. , Pal C.J., *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufmann Publishers, 4th Edition, 2016.

the contrary, quantum computers have the power to solve this sort of problems in a short amount of time, going beyond the reach of any traditional computer. As a matter of fact, it's in the area of *Machine Learning* that quantum algorithms have proved the most outstanding results such as: **reduced computational complexity, higher generalization and good performance on previously unseen instances**. This is one of the main reasons why it's worth exploring the emerging field of *Quantum Computing* and all of its powerful and innovative coding techniques.

1.3 The Future of Quantum Machine Learning:

The exceptional computational potential of full-scale quantum computers is undeniable. But nowadays we're still far from building devices that are error-free, so that quantum computers are not precise enough to be commercialized for large-scale business and industrial applications.

In fact, despite the undoubtedly remarkable progress achieved both in the quantum hardware and software, there are still many open problems researchers are trying to solve. Even if full-fledged quantum computers become feasible, they probably won't replace classical computers. Instead, they will most likely be used together with traditional computers, as part of a **Quantum-Classical Hybrid** model that could speed up *Machine Learning* operations in high performance computing. So, in the next years we should expect more and more "quantum-enhanced" algorithms, namely codes which are improved by Quantum Information Processing resources. In fact, if quantum computers became mainstream technology, they would certainly be adopted by several companies and governments for *Machine Learning* applications, due to the **Big Data** phenomenon and the resulting need for greater velocity in processing business, civic and environmental data.

A quantum *Machine Learning* algorithm must address **three main issues**:

- the **encoding** of classical data into qubits,
- the **processing** of the quantum information,
- the **extraction** of classical data from the final quantum state.

Quantum computers perform at their best only when both the input and the output are quantum, as translation between classical and quantum information is a very time-consuming operation, but it's faster if performed on classical machines. So, the best hybrid configuration would consist in a **heterogeneous model** that combines classical and quantum computers in a synergistic way, making the best use of their distinctive characteristics. Such an implementation would benefit from performing the data processing on a quantum system, while relying on a classical computer for the data encoding and extraction. A form of collaboration between the two computational worlds has already been used in implementations of **feed-forward quantum neural networks**. In fact, the architecture of the algorithm mixes elements of classical and quantum computing, in a process of recursive embedding that enables higher generalization performance[6], so that the code works on a broad class of learning problems.

1.4 Other Areas of Study – Security & Chemistry:

The potential applications of *Quantum Computing* seem truly endless.
Quantum computers could outperform traditional devices in many tasks, such as search and optimization, scheduling and modelling, planning and logistics, simulation and forecasting, as well as the resolution of large systems of linear equations.
Quantum Machine Learning is certainly one of the most interesting and game-changing fields of application of quantum computers, but there are also many other areas in which quantum devices has been achieving considerable results, including Cybersecurity, Cryptography, Chemistry, Genetic, Genomics, Energy, Mobility, Aerospace Physics and Social Network. In the field of **Cybersecurity**, for instance, effects such as superposition and entanglement offer information-processing benefits, that can be applied to Cryptography in order to improve random number generation. A fundamental step forward in the field of **Quantum Cryptography** was the discovery of the *Quantum Key Distribution (QKD)*, an algorithm which allows extremely secure forms of encryption. The state

[6]Narayanan A., Meneer T., *Quantum Artificial Neural Network Architectures and Components*, in Information Sciences, October 2000.

of Geneva in Switzerland has already adopted forms of quantum cryptography: it has been using the *Quantum Key Distribution* and the *BB84 algorithms* in its elections for ten years. Furthermore, the Chinese government has recently invested $10bn to finance the quantum technology research carried out by Prof. Jian-Wei Pan's team[7], who recently made many important Cryptography achievements focused on the implementation of satellite-based Quantum Key Distribution. Future applications of *Quantum Cryptography* may include: e-commerce, e-government, e-health and transmission of biometric data.

Another promising application of quantum computation is **Quantum Chemistry**, where researchers carry out computations to uncover the structure, properties and motion of individual atoms and molecules.

One of the latest, most game-changing breakthroughs in the field came in 2018, from the *University of Cambridge*, where the quantum computing team, in collaboration with *JSR Corporation*, successfully implemented state-of-the-art quantum algorithms to calculate the excited states of molecules, taking into account multi-reference characteristics[8]. This spectacular achievement establishes the relevance of *Quantum Computing* as a terrific tool to study the **properties of molecules and their reactions**, in ways that can't be addressed by classical computers, because the computational time needed to analyze the molecules increases as a power of the number of atoms, so that the complexity skyrockets way too fast for any regular CPU. The constant implementation of more powerful quantum computers, with better stability and fault tolerance, opens up new opportunities for the field of *Quantum Chemistry*, increasing the accuracy of the results obtained on small molecular systems and allowing researchers to process molecules of larger and larger sizes. Quantum computers might even allow researchers to solve the ultimate open question in chemistry: *Schrödinger equation*. The wave functions obtained as a solution to the equation would uncover all the hidden mysteries of the electrons within atoms and molecules, giving us the power to predict their physico-chemical properties and chemical reactions[9].

[7] http://quantum.ustc.edu.cn/web/index.php/en/node/1
[8] https://cambridgequantum.com/cqc-and-jsr-corp-issue-statement-on-their-quantum-computing-project/
[9] https://phys.org/news/2019-01-quantum-chemistry.html

In the future, Quantum Chemistry will find many applications in a lot of different fields such as: Thermodynamics, Spectroscopy, chemical reactions, molecular structure, solvent effects and excited state Chemistry.

2nd Section – The State of the Art of the Hardware

2.1 The Physical Device:

All computation ultimately reduces to a merely **physical process.**
The materials and the building techniques used in the construction of the device determine the overall performance, from the amount of data available to storage, to the accuracy of the measurement and the computational time needed to solve a problem[10]. A quantum computer is a really huge machine that is extremely expensive to build, because it uses complex physical processes in order to control the spin of the qubits. There are several methods that can be used to build a quantum hardware, such as: Nuclear Magnetic Resonance (NMR), ion traps, all optical, super conducting, and quantum dots[11]. But all these quantum devices are really unstable as they are particularly sensitive to vibration, magnetic fields, light, heat and radio waves. In order to **limit the error size** of the computations within acceptable boundaries, they require a high level of **isolation** from the external environment: the quantum computers must be built within cryogenic chambers and kept at extremely low temperatures in order to reduce the decoherence errors.
Nowadays quantum computers are still not precise nor reliable enough for any substantial business use. Despite this major drawback, building machines with a high number of qubits could be a very lucrative investment for the future. In fact, developing more powerful machines allows for an **exponential increase in the space available** for storing quantum states (n qubits can store a number of states equal to 2^n), which, in turn, enables to code longer and longer algorithms, paving the way for a possible quantum revolution, that may reshape the whole

[10] Piccinini G., *Computation in Physical Systems*, in: Stanford Encyclopedia of Philosophy, Edward N. Zalta, Edition, 2017. https://plato.stanford.edu/entries/computation-physicalsystems/
[11] http://khaledelleithy.org/Conferences/Quantum%20Computing%20Hardware%20Implementation%20Methods

technology industry. This is why several multinational firms are spending stellar amounts of money on developing quantum technology, undeterred by all the uncertainties that await this new field.

There are two main kind of quantum hardware devices:

- **quantum annealing machines**, powerful implementations that can have more than two thousand superconducting qubits but cannot execute hill-climbing operations, like Shor's algorithm. This kind of quantum technology is fabricated using niobium wiring and has a really complex building process. A company leader in developing quantum annealing machines is *D-Wave Systems*[12].
- **gate-based devices**, supporting a universal model of quantum computation. This class of quantum hardware is somewhat simpler to manufacture, since it is most commonly fabricated from aluminum wiring, which is much easier to find and handle.

Some examples of companies developing gate-based quantum computers are *IBM*, *Google*, *Rigetti Computing* and *Intel*[13].

2.2 Quantum Annealing Machines:

The market for quantum annealing computers is dominated by North American companies. One of the main players is ***D-Wave Systems***, a Canadian firm, which has been building quantum computers since its foundation in 1999. *D-Wave Systems* is at the forefront of quantum research and collaborates with many important technology institutions, including *Lockheed Martin Corporation*, *Jet Propulsion Laboratory*, *University of Toronto*, *University of Southern California*, *Los Alamos National Laboratory* and *Virginia Tech*. In 2013, *D-Wave Systems* sold a quantum computer to *Google/NASA*[14], at a price of about 10 million dollars, becoming the **first company** to ever commercialize quantum technology.

D-Wave's highly specialized type of quantum computer, which can be considered a **Quantum Annealing machine**, because it's based on

[12] https://www.dwavesys.com/our-company/meet-d-wave

[13] National Academies of Sciences, Engineering and Medicine, *Quantum Computing: Progress and Prospects*, Washington, DC,The National Academies Press, 2019.

[14] https://www.repubblica.it/tecnologia/2016/01/08/news/google_il_nostro_computer_quantistico_funziona_-130785803/

something similar to the chemical event, except it employs magnetic energy instead of heat. "Annealing" is a process used in metallurgy in which a material, such as steel, is first heated and then cooled down slowly, in order to change its physical properties and produce a desirable effect, usually greater toughness and ductility. In the *D-Wave* computer, researchers encode a problem in the form of **magnetic pulses**, which take the qubits, made of superconducting loops, to higher-energy states. Then, the qubits shift towards a lower-energy state, giving a solution to the problem. As of today it's still not clear whether quantum annealing machines constitute a speed-up over classical computers, and researchers are still looking for situations where this could be the case.

Nonetheless, *D-Wave* is constantly upgrading its quantum technology, producing machines that support a **higher number of qubits** and **reduced noise** (unwanted electrical or electromagnetic energy that degrades the quality of signals and data). In 2015 *D-Wave Systems* sold to a cyber security firm, named Temporal Defense Systems Inc., its innovative model *2000Q*[15], a computer with stunning performances and security, specifically built to run quantum computing problems.

This machine has 2048 qubits available, and the Canadian company has already announced their upcoming, next-generation Pegasus quantum processor chip, which will have more than 5000 qubits.

D-Wave has launched an innovative online service called "Leap"[16] that gives internet users free access to a *D-Wave 2000Q* quantum computer **in the cloud**.

2.3 Gate-based Devices:

A gate-based device is a quantum computer characterized by the use of **Quantum Gates**, powerful commands that employ the properties of Quantum Mechanics, such as superposition, sensitivity to contexts, entanglement or linearity of evolution. The gates can manipulate the state of a single qubit, or make two or more qubits interact in unique ways. Classical computers can't use this kind of operations, because the space

[15] https://www.dwavesys.com/press-releases/d-wave%C2%A0announces%C2%A0d-wave-2000q-quantum-computer-and-first-system-order

[16] https://www.dwavesys.com/take-leap

needed to store the quantum information grows exponentially with the number of qubits. In fact, the world's most powerful supercomputer to date wouldn't be able to store the quantum states of just 50 qubits[17].

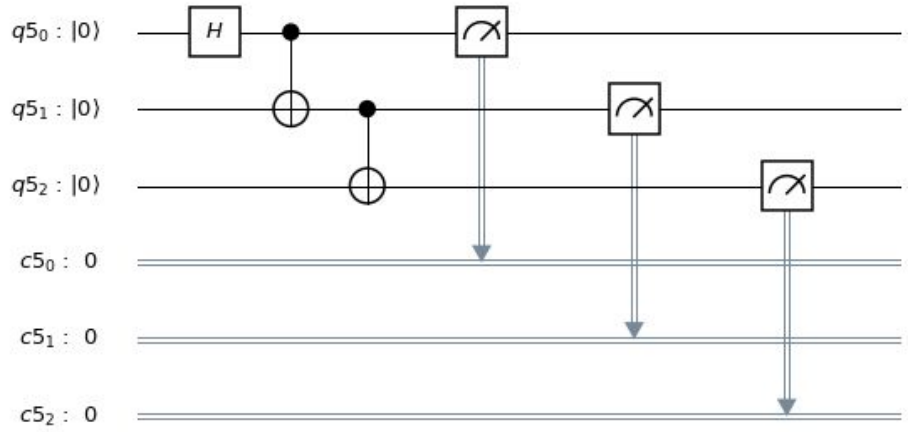


Image 1: Quantum Circuit.

The graphic below shows a simple sequence of gates, with a Hadamard gate, two CNOT gates and three measurements, carried out on python, using the *Qiskit library*.
The graphic representation on the Quantum Circuit resembles a musical score, in which each line represents a qubit, while gates are represented by square boxes.

The main unsolved issue with gate-based quantum computers is that running a quantum algorithm requires to keep the quantum information for a long time. This is extremely hard because the data is stored in qubits, usually made of **electrons or photons**, which are extremely fluctuating and unstable particles, and lose their quantum state if they are exposed to any heat. This is why, to reduce imprecisions and errors in the calculations, the quantum computers are provided with a refrigerator that keeps the qubits at a temperature near absolute zero.
The most common implementation of gate-based quantum computers uses a **Superconducting Architecture**, that employs special materials that show zero resistance to currents, when cooled below their critical temperature (Superconductors). *IBM* is one of the main companies building this kind of gate-based quantum computers and making them available for public use in the cloud. Some of its operative quantum devices are: the 20-qubit *IBM Q Tokyo*, the 14-qubits *IBM Q Melbourne*, the 5-qubits *IBM Q Tenerife*, and the 5-qubit *IBM Q Yorktown*.

[17] https://qt.eu/understand/underlying-principles/gate-based-qc/

Lately there have been several major milestones in the development of more powerful gate-based superconducting Quantum Computing technologies. For instance, on the 10th of November 2017, ***IBM*** Research announced a 50-qubit quantum computer system, the most advanced they have built so far.

Then, at the 2018 *Consumer Electronics Show* in Las Vegas, ***Intel*** announced the successful fabrication of a 49-qubit superconducting quantum test chip, improving on the corporation's previous 7-qubit and 14-qubit chips. The new chip code named *Tangle Lake*[18], comes with several gold connections, used in order to control and operate each qubit.

Finally, at the *American Physical Society* March 2018 conference, ***Google*** announced a 72-qubit gate-based superconducting system, called *Bristlecone*[19], that puts the company at the first place in the race for the world's biggest quantum processor based on quantum gates. In addition to increasing the qubit count, the device also improves precision, reducing the error rate from *Google*'s former chip.

2.4 Topological Quantum Computers:

All the quantum computers described above use **physical qubits**, particles or atoms that behave like a two-states quantum system[20].

But these qubit implementations are just approximations of the flawless theoretical quantum unit: the logical qubit, and as such, they are prone to errors and do not fully express the computational power that a full-fledged quantum device would have in theory.

Contemporary qubits are subject to issues of stability, decoherence, fault tolerance and scalability, making it really hard to exploit quantum computers for any business application. Researchers are developing algorithms for error correction, like the *VQE* (Variational Quantum Eigensolver), *QAOA* (Quantum Approximate Optimization Algorithm) or the *QVECTOR*, bur these can only mitigate the noise without giving a final solution to the problem. Instead, the doorway to an improved generation

[18] https://newsroom.intel.com/press-kits/quantum-computing/#gs.w57km0
[19] https://cloudblogs.microsoft.com/quantum/2018/09/06/developing-a-topological-qubit/
[20] https://quantumcomputingreport.com/our-take/better-qubits-versus-more-efficient-error-correction/

of quantum devices could lie in the successful implementation of a completely different, more robust kind of qubit.

The *Microsoft* quantum team, composed of mathematicians, computer scientists, theoretical physicists and engineers, is currently researching a **topological qubit**[21], that would use Electron fractionalization and Ground state degeneracy to be much more resistant to environmental noise than a regular superconducting qubit, improving gate fidelities and coherence times. The successful development of this type of qubit would be game-changing for the quantum research, allowing for machines with far greater scalability and precision.

A **logical qubit** is the abstract concept of a qubit that is inherently error-free because it has enough coherence time to perform quantum algorithms of any depth, without noise. In order to implement a logical qubit today, on a contemporary quantum computer, one may need to use a 1000:1 Physical-to-Logical ratio. Microsoft Topological quantum computers may only require a 10:1 Physical-to-Logical ratio, allowing them to eventually overtake today's superconducting technology.

3rd Section – Using a Quantum Computer

3.1 Quantum Programming V Classical Programming

As challenging as is it to implement physical quantum devices, the main restraint that is slowing down the advance of Quantum Computing is the **software**, rather than the hardware. Quantum computation is radically different from classical computation: information is stored and processed in a peculiar way, so that traditional algorithms cannot be simply adapted and reworked to become functional on quantum machines. Instead, in order to develop *quantum algorithms* (also called *quantum circuits*) it's necessary to approach the problem from the beginning and come up with a **unique** software **solution** that uses quantum functions and properties. The huge difference between the coding techniques comes from the fact that currently, there is very little classical knowledge that can be transferred into the quantum environment.

[21] https://en.wikipedia.org/wiki/Topological_quantum_computer
https://medium.com/swlh/topological-quantum-computing-5b7bdc93d93f

A programming language for classical computers, like *Python*, *C#* or *Java*, is fairly easy to pick up. Knowing the basics of programming is enough to start experimenting with it, by writing short algorithms and solving simple mathematical problems. **Coding on a quantum computer**, on the contrary, requires you to master complex skills, such as visualizing quantum states as vectors on a tridimensional space, conceptualizing the vectors as approximations of classical probability distributions and using the properties of the quantum world to perform operations on data. Thus, in order to use a quantum computer, you need a good grasp of **advanced topics from diverse academic fields**.

In particular, *Quantum Computing* requires:

- **Physics** for Quantum Mechanics
- **Mathematics** for Linear Algebra and Statistics
- **Computer Science** for subjects like algorithms, information theory, cryptography, optimization, complexity and Machine Learning.[22]

Having already talked about Computer Science matters in the previous paragraphs, here I'll focus on laying out the main concepts of Linear Algebra, Statistics and Quantum Mechanics.

3.2 Concepts of Quantum Mechanics:

Quantum Mechanics (QM) is the branch of *Physics* that explains the behavior of **atoms and subatomic particles** at the smallest scales of energy levels. Understanding the basics of this subject is really crucial, because a quantum computer is nothing more than a machine that is able to store and manipulate tiny particles (qubits) in order to perform computations.

A first fundamental property of quantum mechanics is the **Superposition of States**, according to which any two qubit states can be added together to create a new one and conversely, each quantum state can be interpreted as a superposition of two others.

In other words, quantum objects can exist in multiple states at the same time . This happens because, qubits occupy a **composite state** that is a combination of the two basis states: |0> and |1>. Each qubit contains

[22]Mermin N. D., *Quantum Computer Science An Introduction*, Cornell University, New York, 2007.

statistical information rather than digital, but in the measurement process, the rich amount of quantum information stored in qubits is irretrievably lost, and the result of the measurement collapses to a classical bit, assuming value 0 or 1. Moreover, once the qubits interact with their large classical environment, they tend to lose their quantum properties and produce noise (**quantum decoherence**)[23]. Thus, in order to perform calculations with a quantum machine you also have to predict the errors it could make because of interference effects.

Entanglement is a major phenomenon used in quantum computation, that can really show the power of quantum algorithms. It's a strong form of correlation among pairs or groups of qubits, that makes their states fully dependent on each other, so that you cannot describe entangled particles independently, but only as part of a quantum system.

Entangled states are connected regardless of the physical distance that separates them, thus a measurement or operation on one component provides instant information on the other qubits in the system, even if they are far away from the qubit being measured.

Entanglement is at the basis of the solution of several tasks that cannot be solved on classical computers, and sometimes allows to obtain an exponential increase in computing capacity.

3.3 Notions of Linear Algebra & Statistics:

From a mathematical point of view, a qubit can be described as a unit vector in a complex two-dimensional Hilbert vector space (C^2).

This means that a quantum state is the generalization of a stochastic vector, because it's not restricted to real numbers and nonnegative real numbers, but allows the use of complex values as well.

To represent the elements of a complex vector space the column vector notation is impractical because the vectors lie in an exponentially large space. Instead, it is convenient to utilize the **Dirac notation** or "bra-ket notation"[24], composed of angle brackets and vertical bars, so called because the inner product (or dot product) of two states is denoted by a bracket, $\langle\Phi|\psi\rangle$, consisting of a left part $\langle\Phi|$, called "bra", and a right part |

[23] Schlosshauer M., *Decoherence And the Quantum-To-Classical Transition,* Springer, 2007.
[24] https://docs.microsoft.com/en-us/quantum/concepts/dirac-notation?view=qsharp-preview

$\psi\rangle$, called "ket". Dirac notation is a standard notation of quantum mechanics, used to visualize quantum states as an expansion of complex vectors in the canonical basis, composed of the two fundamental states $|0\rangle$ and $|1\rangle$, in such a way: $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$.

The simplest possible quantum state is a qubit, a vector in a superposition of these two basis states, where the squares of the absolute values of the entries adds up to 1, so that the normalization of the vector happens in the two norm, rather than the one norm: $|\alpha|^2 + |\beta|^2 = 1$.

To obtain a geometric representation of the qubit, we can imagine that all of its possible states can be positioned on the surface of a sphere of unitary radius called ***Bloch Sphere***, where the two poles represent the two fundamental states: $|0\rangle$ and $|1\rangle$. All the other points on the sphere represent a possible superposition. This intuitive representation is very useful for interpreting the operations performed on a single qubit.

The *Bloch Sphere* allows us to establish a one-to-one correspondence between the representation of a generic state of the qubit: $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ and its representation on the surface of the sphere in R^3:

$|\psi\rangle = e^{i\gamma}cos(\theta/2)|0\rangle + e^{i\varphi}sin(\theta/2)|1\rangle$.

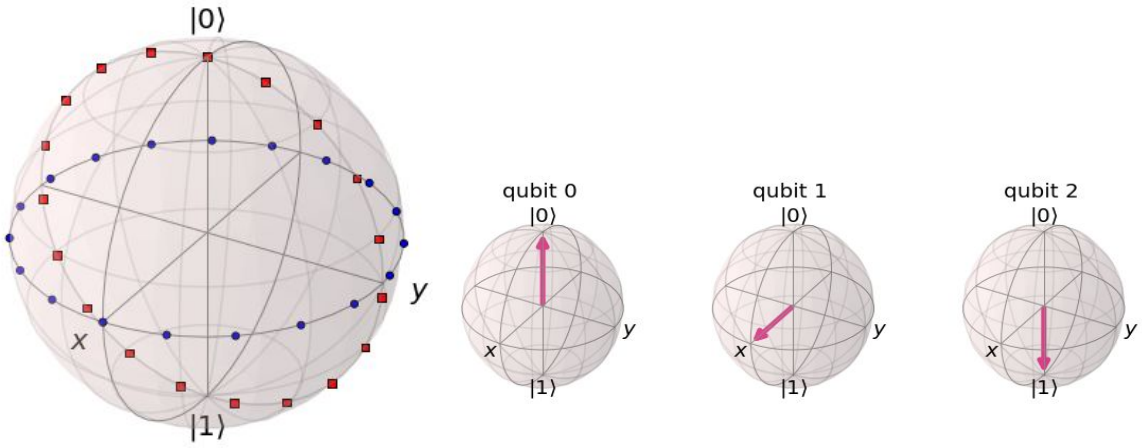


Image 2: The Bloch Sphere.

To visualize the *Bloch Sphere*, I used the *QuTip* package and related Bloch method. Then I created three other graphs by running the Qiskit function: plot_bloch_multivector(job.result().get_statevector(circuit)) on a quantum circuit where qubit 0 is in the basis state corresponding to the |up⟩ state:, qubit 1 is in the state obtained after applying a Hadamard gate on the basis state (50-50 chance of measuring either $|0\rangle$ or $|1\rangle$) and qubit 2 is in the state obtained after applying a NOT gate on the initial state.

Quantum logic gates are the functions used in order to perform operations with qubits. They are represented by unitary matrices, thus every gate or sequence of gates is **reversible**, meaning that any operation can be undone by applying the opposite sequence of gates.
The most famous gates act on one or two qubits, transforming the quantum state, in order to reach the desired configuration.
Some examples of important **single-qubit gates** are:
- the *Hadamard or H gate*, which sets a qubit from the basis state to the state a state where it's equally likely to observe either of the two basis states. Thus, after applying the gate on a qubit in base state there is a 50-50 probability of observing either 0 or 1.
- the *Pauli Gates*: Pauli-X, Pauli-Y and Pauli-Z, each of which performs a rotation around the respective axis of the Bloch sphere by π radians.
The Pauli-X is the equivalent of the NOT logic operator with respect to the canonical basis because it transforms $|0\rangle$ into $|1\rangle$ and $|1\rangle$ into $|0\rangle$.

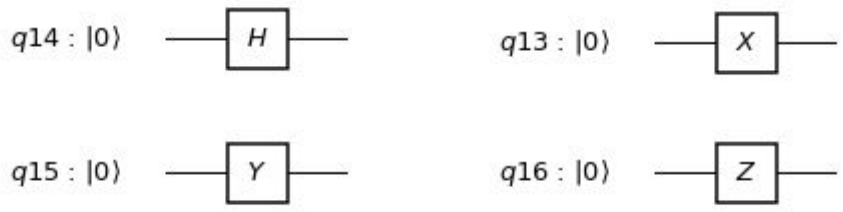


Image 3: Single-qubit Gates.

From left to right: quantum circuits showing a Hadamard gate, a Pauli-X gate, a Pauli-Y gate and a Pauli-Z gate.
Images obtained using the *Qiskit* function circuit.draw(output='mpl') on the related circuits.

Among the most relevant **two-qubit gates** are:

- the *Controlled NOT gate* (also C-NOT or CNOT) which takes as input one qubit as the target of the operation, and one as the control. In a singly controlled gate, denoted $\Lambda(G)$, the value of the control qubit determines whether the gate is applied or not. In particular, the CNOT gate flips the target qubit if and only if the control qubit assumes value 1.
- the *SWAP gate*, that exchanges the states of two qubits.
  It is obtained by applying three iterations of the CNOT gate, to achieve a cyclical permutation of the two qubits.

Image 4: Two-qubit Gates.

From left to right: quantum circuits showing a *Controlled NOT gate* and a *SWAP gate*. Images obtained using the *Qiskit* function **circuit.draw(output='mpl')**[25] on the related circuits.

3.4 Quantum Error Correction (QEC)

The correction of quantum errors is a crucial open problem, although the discovery of many different error correction codes have brought remarkable improvements. As I mentioned in the first section of this work, a quantum computer is really sensitive to the interactions with the **surrounding environment**, which determines its **decoherence,** namely the dispersion of the quantum information kept in the qubits. The innate fragility of the physical components represents one of the main obstacles to the development of large scale quantum computers. To cope with the sensitivity to vibrations, temperature fluctuations and electromagnetic waves, quantum computers are isolated from any interaction with the environment by building large glass containers around them with a cooling system that constantly keeps the temperature near absolute zero.

Alongside the debilitating effects of decoherence, due to external interference, the other main cause of quantum errors consists in a series of small **unitary errors** in the quantum circuits, due to imperfect **implementation** of the quantum gates. Unlike classical gates, quantum gates cannot be implemented with the same precision, because they are unitary transformations chosen from a continuum of possible values, therefore the effects of small imperfections in the doors accumulate and then they can cause serious failures in the calculation.

To be protected against errors and recovered without damage, quantum information must be cleverly encoded and then suitably deposited in a quantum memory.

[25] https://Qiskit.org/documentation/terra/visualizing_a_quantum_circuit.html

Quantum error correction codes can protect quantum information from both decoherence and "unit errors" due to faulty quantum gates. The code consists in a set of orthogonal states named quantum codewords.

The first quantum error correcting codes were discovered in 1995 by Peter Shor, a mathematician at *AT&T Research* and in 1996 by Andrew Steane a Professor of Physics at the *University of Oxford*. Shor's strategy is based on the brilliant idea of storing quantum information not in a single qubit but in an entanglement of nine qubits, so that if a qubit suffers errors, the transmitted quantum information is not lost, but is recovered correctly. A year later Steane, working independently, achieved a similar result, but his code uses only seven bits instead of nine.

After them, many other researchers developed different methods to correct quantum errors, the best results were achieved by Knill & Laflamme who detailed the requirements for quantum error-correcting codes; by Eckert & Macchiavello, who derived the quantum Hamming bound; by Bennett, who studied the close relationship between entanglement purification and quantum error correction; by Gottesman and Calderbank, whose work allowed error correction on large and complex quantum algorithms. The **choice** of which error correction code to use has a fundamental role since it influences the whole quantum calculation, from the arrangement of the qubits at the physical level to the compilation strategies at the software level. Despite the remarkable progress that's already been achieved, many issues are still open, the main ones being: the definition of the capacity of a quantum channel, the practical implementation of an error correcting code, fault tolerance and thresholds, self-correcting systems, operator quantum error correction and error avoidance via energy gaps. So quantum error correction, remain one of the **main challenges** in the quantum computing field.

4th Section – Empirical Part

4.1 Developing Frameworks:

Quantum computing is still in its infantile phase and the technologies of Quantum computing appear futuristic still. We do not know how long it will take to realize the full potential of quantum computing. Nowadays we

have working quantum computers that are able to execute basic algorithms and solve simple problems[26], even if only for a short time. But as more and more quantum computers are built, the availability and diversity of the software technology is increasing.

Giants like *Google*, *IBM*, *Microsoft*, *Intel*, but also research centers like the *MIT* and *Harvard University* in the United States are investing heavily in **Quantum Research**, in an attempt to achieve quantum supremacy. Russia and China are trying to compete with the USA, in fact the Chinese government has invested about 10 billion dollars in quantum research. The European Union has devoted more than one billion in the ***Quantum Flagship***[27]. In particular, Europe is focusing on the studies from the Italian physicist Tommaso Calarco and his team, working in the *Center for Integrated Quantum Sciences and Technology (IQST)* at the *University of Ulm* in Germany. Currently, around the world, there are about a thousand people dealing with quantum computing at a theoretical level; and about 300 developers dealing with software and trying to develop algorithms designed specifically for quantum computers. In the near future, there will probably be a great increase in the volume of quantitative research.

Most of the leading quantum computing companies and quantum service providers are encouraging **open learning and research**, by providing free, online, open platforms for performing quantum computing on the actual hardware. Anyone can use a computer to experiment with the quantum technology and implement algorithms that run on the real devices, thanks to the use of **cloud services**.

There are already several programming languages that can be used to write and run quantum programs. In fact, in recent times, there has been a **proliferation of software projects** for the world of quantum computing, created using different classical programming languages as a starting point. On the web it's possible to find a list of **open-source frameworks** for quantum programming[28] and a list of the different **simulators** to execute the related quantum programs[29].

---

[26] Gribbin J., *Computing with Quantum Cats: From Colossus to Qubits,* Prometheus Books, 2014.

[27] http://www.qtflagship.cnr.it/

[28] https://qosf.org/project_list/

[29] https://quantiki.org/wiki/list-qc-simulators

The most renowned and widely used projects are:

- **Qiskit** (Quantum Information Science Kit) from *IBM*
- **Q#** (pronounced "Q-sharp") from *Microsoft*
- **Forest** from *Rigetti Computing*
- **Project Q** from *ETH Zürich.*

All of these allow the user to connect to a true quantum computer in the cloud, using the *Hardware as a Service (HaaS)* model. They are also available in the *Python* programming language as libraries, which allow the user to import all of the quantum classes and functions into an intuitive code editor.

After experimenting with all four, I selected the *Qiskit* package to implement my quantum machine learning algorithm.

4.2 Kernel Methods:

Kernel Methods[30] are quantum functions that can be used in supervised quantum Machine Learning in order to solve especially hard classification problems. **Classification** is the task of categorizing new and unseen objects into the desired number of classes. It is achieved by training the learning algorithm on a given dataset of labeled objects, that acts as a target, in order for the algorithm to assign the correct category to each object. The most basic classification problems deal with linearly separable data in a two-dimensional data space, that can be separated even by the simplest classifier. But when we're dealing with a dataset in a high dimensional space, it might be very difficult to classify the data into groups. In particular, the most complex classification problems regard data structures where one class is embedded inside the other.

In order to solve the classification in these situations, we can use **Nonlinear Embedding** to separate the points so that different classes occupy **distinct high dimensional spaces**. In the new embedding space, it's much easier to separate the classes with a hyperplane, such that every point situated above the hyperplane belongs to the first class, while the points located below it belong to the other.

[30] https://arxiv.org/pdf/math/0701907.pdf

The Kernel method achieves the same effect used in Nonlinear Embedding without having to calculate the actual embedding: it introduces a new notion of distance (or similarity) between points in a high dimensional space by replacing the the inner product with a ***Kernel* function**.

$$K(x_i, x_j) = < f(x_i), f(x_j) >$$

Where, K is the Kernel function, x_i, x_j are n dimensional inputs, $< x_i, x_j >$ is the dot product and f is a map from n-dimension to m-dimension space.

The *Kernel* method performs less computations than the typical Nonlinear Embedding, improving the overall speed of the process and allowing for an **exponential speed-up** in many algorithms. In fact, in many cases where a learning algorithm would use an inner product, a Kernel function can be used instead, to reduce the computational time.

Table 2: Examples of different Kernel functions

Linear Kernel	(x_i, x_j)
Polynomial Kernel	$((x_i, x_j) + c)^d$
Radial Basis Function Kernel	$exp(-\gamma \parallel x_i - x_j \parallel^2)$

The key to the improved performance of quantum kernel methods over regular supervised learning techniques lies in the fact that the mathematical theory of kernel methods and quantum mechanics are very similar to each other. Quantum computers can use the quantum state space as feature space, solving Kernels that are inefficient or even impossible to compute with classical hardware[31].

A recent **research by *IBM* and *MIT***, published in *Nature*[32] (*Supervised learning with quantum-enhanced feature spaces*) gave experimental proof of the computational speed-ups enabled by quantum algorithms, as a result of exponentially large quantum state spaces in kernel methods[33]. Some examples of algorithms based on a quantum-enhanced feature space include: kernelized k-means clustering, where the kernel achieves an exponential decay over the Euclidean distance, support vector

[31] https://syncedreview.com/2019/04/03/ibm-proposes-quantum-enhanced-feature-space-for-ml/
[32] https://www.nature.com/articles/s41586-019-0980-2
[33] Havlíček V, Córcoles A., Temme K., Harrow A.W., Kandala A., Chow J. M. & Gambetta J. M., *Supervised learning with quantum-enhanced feature spaces*, Nature International Journal of Science No. 576, March 2019.

machines, which ensure certain sparsity structure in the model and generalize well, and variational quantum circuits, which classify the data explicitly in the Hilbert space.

4.3 Quantum Support Vector Machine:

The *Quantum Support Vector Machine (QSVM)* is a quantum supervised learning model that solves the problem of data classification and regression. It is the most famous example of hybrid classical-quantum algorithm based on a quantum-enhanced feature space. The algorithm uses a *Kernel* method on a quantum computer in order to obtain a speedup over traditional classification methods. In the future, it could be extremely useful in scientific applications that need a huge amount of computational power, like the discovery of new drugs, the design of new chemical compound, the analysis of our genetic makeup, or the mapping of our brain circuitry.

The algorithm works by maximizing the distance between the two classes of different data points, which are then separated by some linear or non-linear function[34]. It achieves a complexity that is logarithmic in the size of the vectors and the number of training examples: in the least squares QSVM $O(log(MN))$ states are required, where N is the number of features that represent a single data instance, and M is the number of training instances. This means that in cases where classical sampling algorithms require polynomial time, an exponential speed-up is obtained through the efficient solution of the kernel equations on the quantum hardware. But there are some drawbacks, as the logarithmic complexity also implies that the model is dense rather than sparse, making the linear and polynomial kernels easy to implement, while the Radial basis function kernels are really challenging[35].

Classical Support Vector Machines mostly use sparse data structures for their computations, so that only few of these techniques can be translated on quantum devices. As the quantum hardware improves, though, it becomes easier to tackle more and more complex classification problems,

[34] https://hackernoon.com/quantum-machine-learning-d0037f59f31a
[35] https://peterwittek.com/understanding-quantum-svms.html

finding hidden patterns within massive loads of data: eventually, the sheer power of quantum computers will probably overcome all current mainstream classification techniques.

4.4 Algorithm Implementation:

● **Description:**

After having studied the theory behind quantum computers, I experimented with the empirical part, which is the most difficult but also the most fascinating. Then I implemented a classical-quantum hybrid classifier using the code QSVM provided in the quantum library *Qiskit-Aqua* version 0.11[36] from *IBM.*

I have used the quantum-enhanced supervised learning model to categorize two famous datasets from the *Sklearn* library: the *Breast Cancer Dataset*[37] and the *Iris Dataset*[38].

Here, I present my results, analyzing the classification accuracy obtained in each of the two cases.

● **Goal:**

The goal of this work is to design the code for a hybrid classical-quantum classifier: the Quantum Support Vector Machine (QSVM), supervised learning code that uses a *Kernel method* on a quantum enhanced feature space. The algorithm is able to connect to the *IBM Q* network and run on any of the real-world quantum devices provided by the company, exploiting the power of Quantum Computing to perform computations.

● **Methodology:**

First of all, I load all the packages I need for my algorithm.

Then I import the dataset I want to classify from *sklearn.datasets.*

I divide the dataset into a *training set* (70%) and *test set* (30%).

I do some preliminary data analysis in order to understand the structure of the data I'm working with.

Then I use the function *seaborn.pairplot*, in order to highlight the relationships between different pairs of features in the dataset.

---

[36] https://Qiskit.org/documentation/aqua/library.html#aqua-library
https://Qiskit.org/documentation/aqua/index.html

[37] **The Breast cancer dataset** was obtained in 1995 from the University of Wisconsin Hospitals, Clinical Sciences Center Madison, from Dr. William H. Wolberg.

[38] **The Iris flower dataset** was published by Ronald Fisher in 1936 in "The use of multiple measurements in taxonomic problems as an example of linear discriminant analysis".

In the preprocessing phase, I decide what are the best features scaling methods to apply to the dataset in order to standardize the range of values across all the features.
I use *Principal Component Analysis (PCA)* in order to reduce the dimensionality of my dataset without losing too much information.
Then I visualize the structure of the data on a graph by plotting the two-dimensional Principal Component Analysis.
At this point, I am ready to set up my quantum classifier, by calling the *Qiskit* method QSVM with the appropriate parameters.
I run the Quantum Support Vector machine on the dataset to classify the target data and I store the outcome in a variable called result.
Finally, I analyze the overall classification accuracy, and then the precision obtained on each single class, plotting it using a Heatmap graph.

- **Selected Parts of Code:**

```
In [1]: # I import the necessary packages
        %matplotlib inline
        import numpy as np
        import pandas as pd
        import seaborn as sns
        import matplotlib.pyplot as plt
        from sklearn import datasets
        from sklearn.model_selection import train_test_split
        from sklearn.preprocessing import StandardScaler, MinMaxScaler
        from sklearn.decomposition import PCA
        from sklearn.metrics import classification_report
        from qiskit_aqua.utils import split_dataset_to_data_and_labels
        from qiskit_aqua.input import SVMInput
        from qiskit_aqua import run_algorithm
        from sklearn.metrics import confusion_matrix
        from qiskit import Aer
        import qiskit
```

3.1 Exploratory Data Analysis

```
In [3]: # I import the breast cancer dataset
        # classified based on whether the tumor is malignant or not (1 = Yes, 0 = No)
        # 0 = Benign
        # 1 = Malignant
        cancer = datasets.load_breast_cancer()

        class_labels = ['Benign', 'Malignant']
```

```
        # I divide the dataset into a training set and a test set (70% training, 30% testing)
        X_train, X_test, Y_train, Y_test = train_test_split(cancer.data,
                                                            cancer.target, test_size=0.3)

        # I visualize the data using pandas
        df_cancer = pd.DataFrame( np.c_[cancer['data'], cancer['target']],
                                 columns = np.append(cancer['feature_names'], ['target']))
        df_cancer.head()
```

```
In [4]: df_cancer.describe().T
```

4 Features Scaling

Scaling is a fundamental step in data preprocessing, in order to standardize the range of values across all the features. This prevents a single feature with a wider range than the others to account for most of the distance.

```
In [6]: # I scale the data to fit it to a normal distribution
        scaler = StandardScaler().fit(X_train)
        X_train = scaler.transform(X_train)
        X_test = scaler.transform(X_test)
```

4.1 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a simple but really powerful linear transformation technique. PCA is often used to emphasize variation and bring out strong trends in a data set.

```
In [7]: n_dim = 2

        # I use PCA to reduce the number of dimensions of the dataset down to n_dim = 2
        pca = PCA(n_components=n_dim).fit(X_train)
        X_train = pca.transform(X_train)
        X_test = pca.transform(X_test)
```

3.3 Quantum Machine Learning model

I perform a Classification on the breast cancer dataset, using the Quantum Support Vector Machine (QSVM) algorithm from the quantum computing package qiskit, version 0.8.

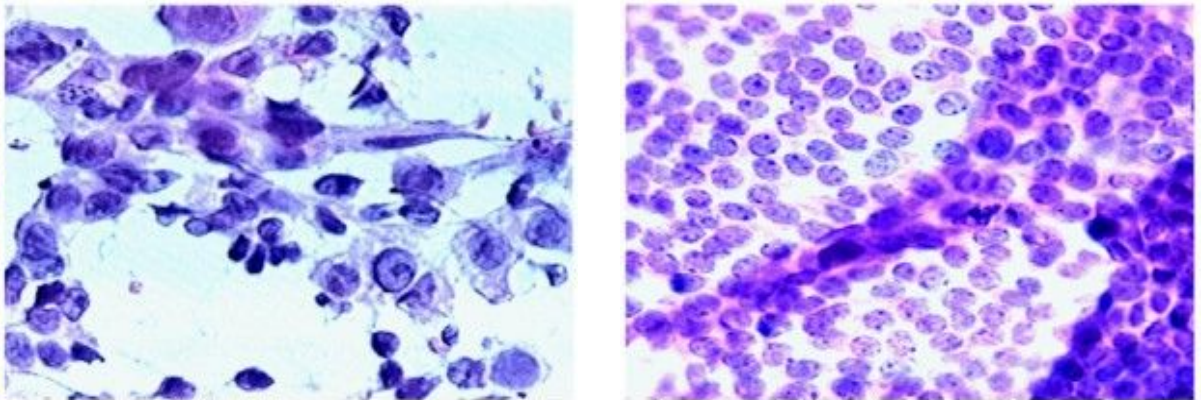
```
In [11]: datapoints, class_to_label = split_dataset_to_data_and_labels(test_input)

         params = {
             'problem': {'name': 'svm_classification', 'random_seed': 7070},
             'algorithm': { 'name': 'QSVM.Kernel' },
             'backend': {'name': 'qasm_simulator', 'shots': 1000},
             'feature_map': {'name': 'SecondOrderExpansion',
                             'depth': 2, 'entanglement': 'linear'}
         }

         backend = Aer.get_backend('qasm_simulator')

         algo_input = SVMInput(training_input, test_input, datapoints[0])
         %time result = run_algorithm(params, algo_input, backend=backend)
```

● **The *Wisconsin Breast Cancer Dataset* (WBCD):**



Malignant:
asymmetrical and heterogeneous cells.

Benign:
symmetrical and homogeneous cells.

Image 5: Visual difference between Benign and Malignant Breast Cancer cells.
Credit to: Glaucia Sizilio
https://www.researchgate.net/figure/Captured-images-of-layers-of-glass-with-smears-of-breast-mass-obtained-by-FNA-the-parts_fig3_232811011

The first version of the *Wisconsin Breast Cancer Dataset*[39] was developed between January 1989 and November 1991, by the American surgical oncologist William H. Wolberg of the *Department of Surgery and Human Oncology*, in collaboration with the Mathematics and Computer Sciences' Professor Olvi Leon Mangasarian and two of his students.
Wolberg was driven to undertake this research by three goals closely related to each other:

- improve the diagnosis and prognosis of breast cancer, based on minimally invasive fine needle aspirates, avoiding surgical biopsy
- discriminate benign breast lumps from malignant ones
- predict the recurrence of malignant samples without extracting lymph node.

His team (in the original version) analyzed and classified 699 cancer cell samples, after having carefully calculated their characteristics with real values for each cell nucleus: radius, texture, perimeter, area, smoothness, compactness, concavity, concave points, symmetry and fractal dimension.
In addition to this first version, the oncologist periodically developed other

[39] https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Original)

versions of the *Breast Cancer Dataset*, slightly modified. In my research I study the second version of the dataset (1995), which comprises 30 features and a target (type of cancer).

Table 3: Description of the *Breast Cancer dataset*

Number of Samples Number of Classes Number of Features	569 2 30
Samples	569 samples, collected by Dr. Wolberg among his patients.
Classes	Malignant (target==0) Benign (target==1)
Features: Features computed by a non-invasive diagnostic test, based on digitized image of a fine needle aspirate (FNA).	mean radius mean texture mean perimeter mean area mean smoothness mean compactness mean concavity mean concave points mean symmetry mean fractal dimension radius error texture error perimeter error area error smoothness error compactness error concavity error concave points error symmetry error fractal dimension error worst radius worst texture worst perimeter worst area worst smoothness worst compactness worst concavity worst concave points worst symmetry worst fractal dimension
Class Distribution	212 Malignant (37,26%) 357 Benign (62,74%)

By applying a linear programming based on classification methods, Mangasarian' s team created a diagnostic software, named *Xcyt*[40], which Dr. Wolberg currently uses in the practice of his medical profession. The program marked a turning point both in the treatment and in the prevention of cancer[41].
Wolberg's dataset has become very famous and it's widely used to test supervised Machine Learning algorithms.

Feature Scaling

I use the functions StandardScaler() and MinMaxScaler(-1,1) from *sklearn.preprocessing* in order to transform the features.
The first command standardizes the features to a gaussian distribution with mean zero, while the second sets the range of all features between minus one and one. Feature Scaling is a fundamental step in data preprocessing, because it prevents a feature with a wider range than the others to account for most of the distance.

Pairplot

I use the *Pairplot graph* from the *Seaborn library* in order to show the affinity between different features of the Breast Cancer Dataset.
This is useful to visualize pairwise relationships in the dataset, pointing up the type of correlation that exists between the selected variables.
Since I'm working on a supervised learning problem, the dataset is provided with target labels, so that I can plot the data points highlighting the distinction between classes by marking them with different colors (blue for **malignant**: target==0.0, orange for **benign**: target==1.0).

[40] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.45.8794&rep=rep1&type=pdf
[41] Mangasarian O. L., Wolberg W. H. , *Cancer diagnosis via linear programming*, SIAM News, Volume 23, Number 5, September 1990, pp 1 & 18.

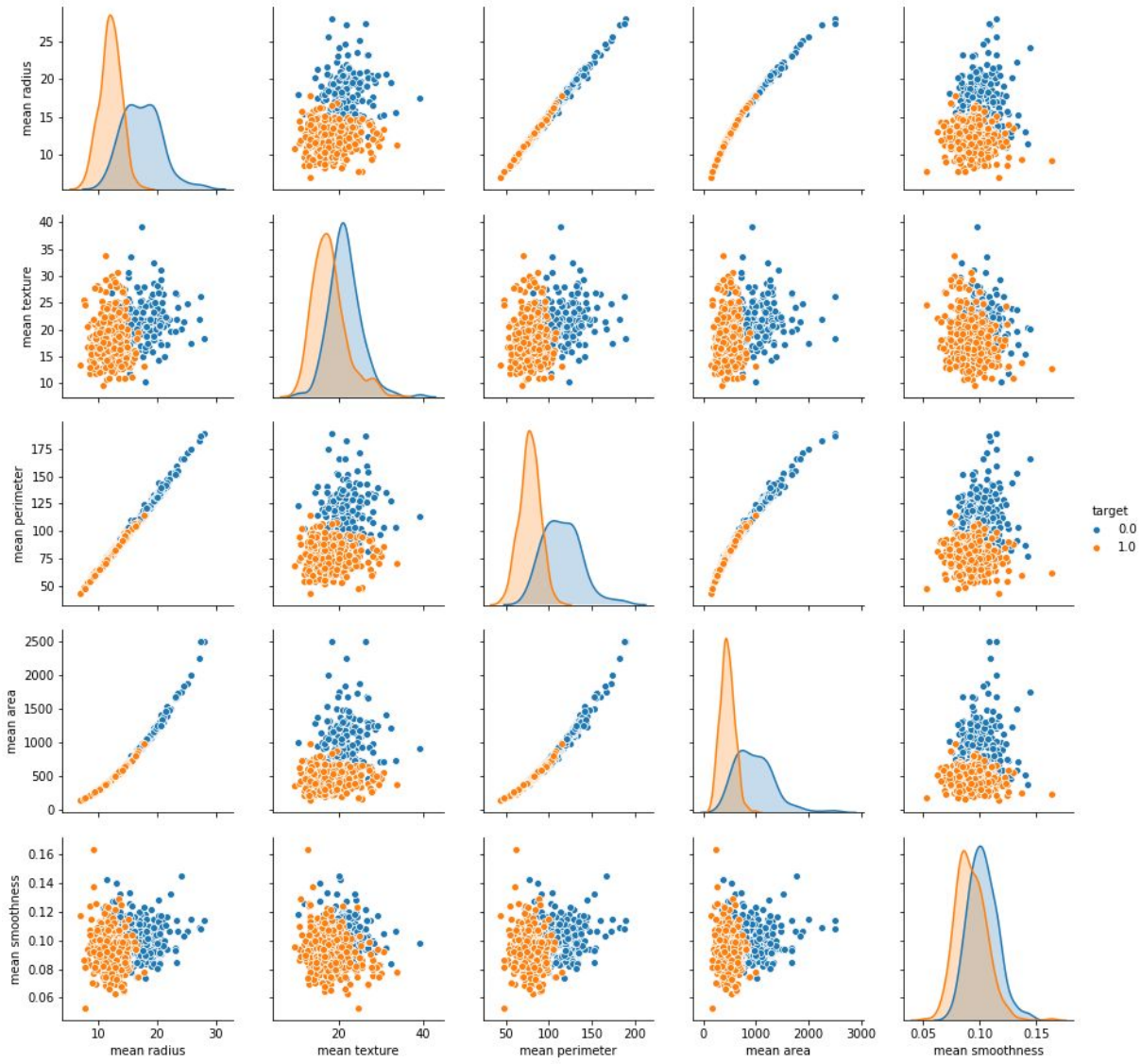


Image 6: Pairwise relationship between different features of the *Breast Cancer Dataset.*
malignant: target==0.0 **benign**: target==1.0

Principal Component Analysis (PCA)

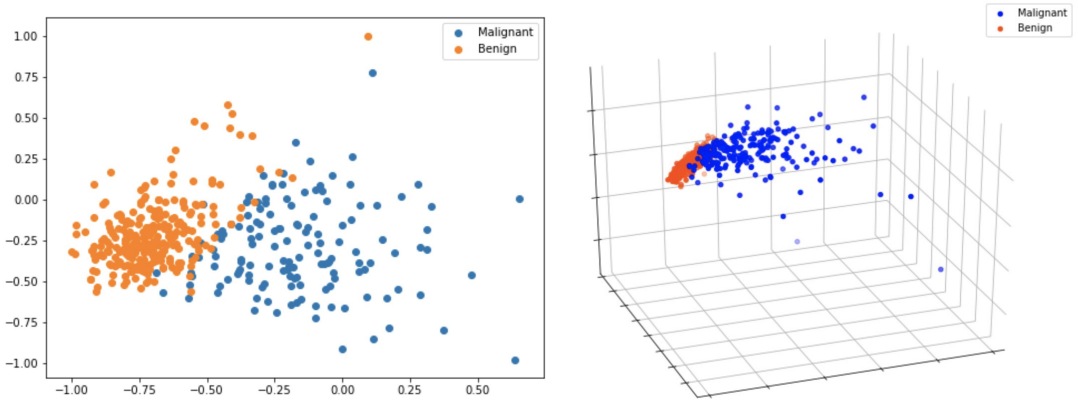


Image 7: Matplotlib.pyplot graphs showing the 2D and 3D PCA for the *Breast Cancer dataset* (left to right).

I use Principal Component Analysis (PCA) to plot the dataset on a two-dimensional and a three-dimensional graph, retaining as much information as possible about the features: the 2D PCA explains 63.3% of the total variation of the dataset, while the 3D PCA explains 72.69%.
In the 2D PCA-space, we can almost draw a line between the two groups: the two classes of breast cancer cells appear as pretty well-divided overall, with the points in the top-left part of the graph being easy to categorize as **benign**, while the points on the bottom-right are likely to be predicted as **malignant** by our classification algorithm. But there is some overlap in the middle. Obviously, the points located where the two classes overlap are the most difficult to classify, but we can use a learning model to train a classifier with just two principal components.
I will analyze the precision of the classifier on the dataset, first with the 2D PCA data, then with the 3D PCA data.
Since the 3D PCA retains more information about the 30 features in the dataset I expect the accuracy of the classifier to be higher in the 3D PCA case, but the linear connections between qubits will also increase, so that the algorithm will take more time to run.

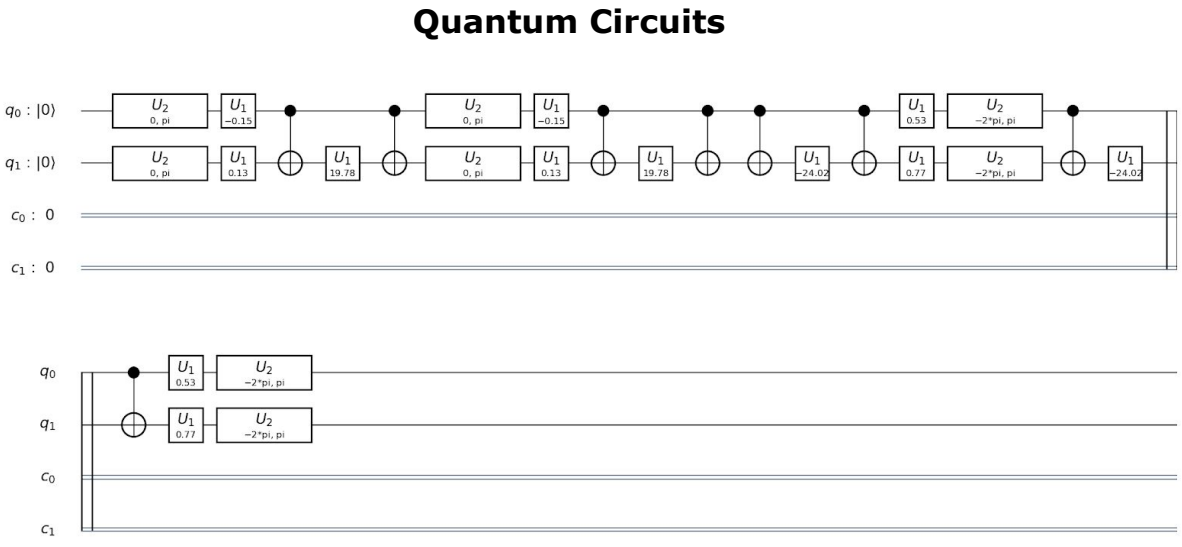


Image 8: Quantum Support Vector Machine circuit on the first two data points using the *Breast Cancer dataset* reduced to 2 principal components (2D PCA).

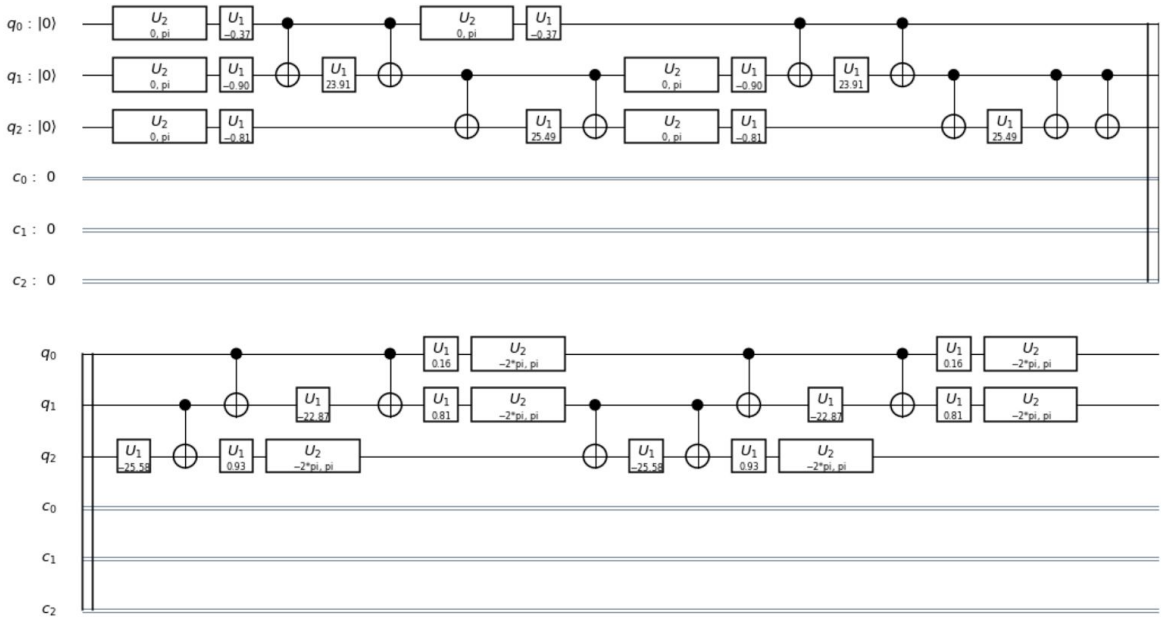


Image 9: Quantum Support Vector Machine circuit on the first two data points using the *Breast Cancer dataset* reduced to 3 principal components (3D PCA).

The circuits above show the quantum gates the QSVM applied on the first two data points of the *Breast Cancer* dataset in the cases of 2D and 3D PCA respectively. Both graphs were obtained using the function: qsvm.construct_circuit(x1=datapoints[0][0], x2=datapoints[0][1])[42]

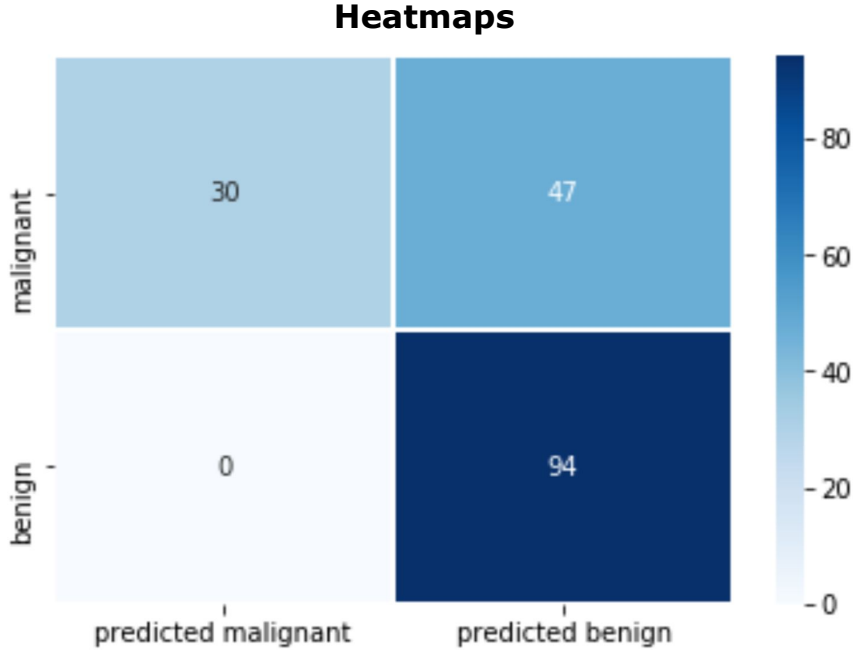


Image 10: Heatmap showing the precision of the classifier on each class, using the *Breast Cancer dataset* reduced to 2 principal components (2D PCA).

[42] https://Qiskit.org/documentation/_modules/Qiskit/aqua/algorithms/many_sample/qsvm/qsvm.html

	precision	recall	f1-score	support
malignant	1.00	0.39	0.56	77
benign	0.67	1.00	0.80	94
accuracy			0.73	171
macro avg	0.83	0.69	0.68	171
weighted avg	0.82	0.73	0.69	171

Image 11: Precision scores obtained on the *Breast Cancer dataset*, reduced to 2 principal components (2D PCA).

The Quantum Support Vector Machine I initialized with the training data was able to carry out a reasonably accurate classification, with an overall precision of 72.51%.

The Heatmap above shows the performance of the classical-quantum hybrid classification algorithm in predicting each class.

The algorithm obtained a better result in correctly predicting the class of malignant cancer cells, with an accuracy score of 100%. The precision of the classifier on benign cancer cells is lower with a score of 67%.

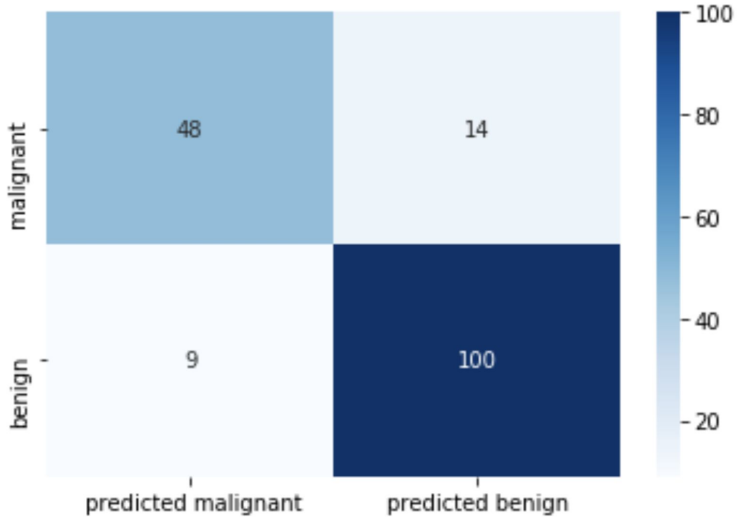


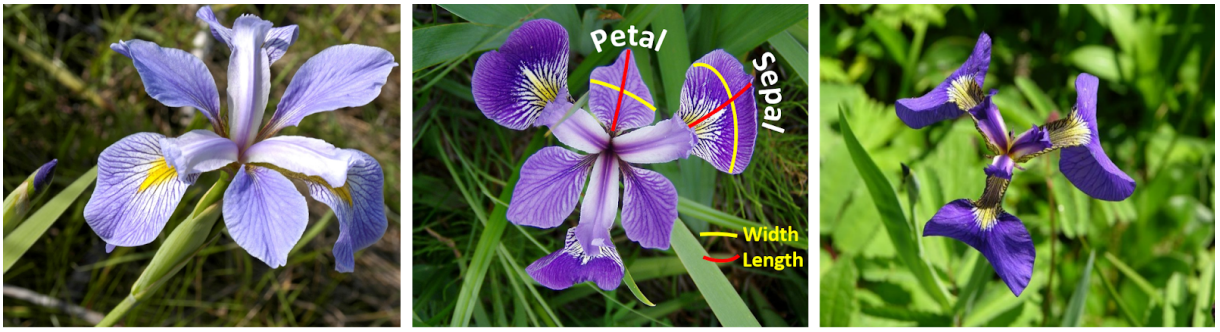
Image 12: Heatmap showing the precision of the classifier on each class, using the *Breast Cancer dataset* reduced to 3 principal components (3D PCA).

	precision	recall	f1-score	support
malignant	0.84	0.77	0.81	62
benign	0.88	0.92	0.90	109
accuracy			0.87	171
macro avg	0.86	0.85	0.85	171
weighted avg	0.86	0.87	0.86	171

Image 13: Precision scores obtained on the *Breast Cancer dataset*, reduced to 3 principal components (3D PCA).

With the 3-components PCA, the classification result improves, achieving an overall accuracy of 87%. In order to obtain more accurate measures I would need to increase the number of features as well as the sizes of both the training and the test sets. But the quantum technology provided by IBM is still extremely susceptible to errors, so that in order to obtain a meaningful result I need to run the computations several times, which increases the computational time. Hopefully in the future researchers will discover better solutions to the problem of decoherence, which will improve the performance of all quantum circuits.

- **The Iris Dataset:**



Iris Virginica Iris Versicolor Iris Setosa

Image 14: Visual Differences among the three species of Iris flowers.
Credit to: Karlijn Willems, data scientist.
Picture from: https://towardsdatascience.com/keras-deep-learning-in-r-b0be9dc726ff

The Iris dataset, published by the British mathematical and geneticist Ronald Fisher in *The Use of Multiple Measurements in Taxonomic Problem (1936)*[43], is a well known test case, often used to analyze the efficiency of supervised Machine Learning techniques.

The dataset[44] consists of a collection of measurements, carried out by the American botanist Edgar Anderson on a sample of 150 flowers. It is a multivariate dataset, which contains 50 observations for each of its three classes, which correspond to three different species of Iris flowers: Iris Setosa, Iris Virginica and Iris Versicolor.

The classification criterion found by the botanist takes into account four morphological variables, which are the length and width of the sepal and of the petal. Using the information derived from these four features, Fisher developed a discriminant model that can be used to distinguish the species from each other:

- Iris *Setosa* if petal-length ≤ 1,9 cm
- Iris *Virginica* if petal-width > 1,7 cm
- Iris *Versicolor* if petal-length ≤ 4,9 cm and petal-width ≤ 1,7 cm.

Based on these four characteristics only, it's possible to train a Machine Learning classifier in order to tell which type the observed Iris flower belongs to. Here I import the Iris dataset provided in the Sklearn library[45], splitting it in order to have 70% of the data for the training set and 30% for the test set.

Table 4: Description of the *Iris Dataset*[46]

Number of samples	150
Distribution	50, 50, 50
Number of Dimensions	4
Features	sepal length [cm] sepal width [cm] petal length [cm] petal width [cm]
Classes	Setosa (target==0) Versicolor (target==1) Virginica (target==2)

[43] http://www.comp.tmu.ac.jp/morbier/R/Fisher-1936-Ann._Eugen.pdf
[44] https://www.kaggle.com/arshid/iris-flower-dataset
[45] https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_iris.html
[46] https://www.math.umd.edu/~petersd/666/irisdata.txt

Feature Scaling

I use the functions StandardScaler() and MinMaxScaler(-1,1) from sklearn.preprocessing in order to transform the features.
The first command standardizes the features to a gaussian distribution with mean zero, while the second sets the range of all features between minus one and one. Feature Scaling is a fundamental step in data preprocessing, because it prevents a feature with a wider range than the others to account for most of the distance.

Pairplot

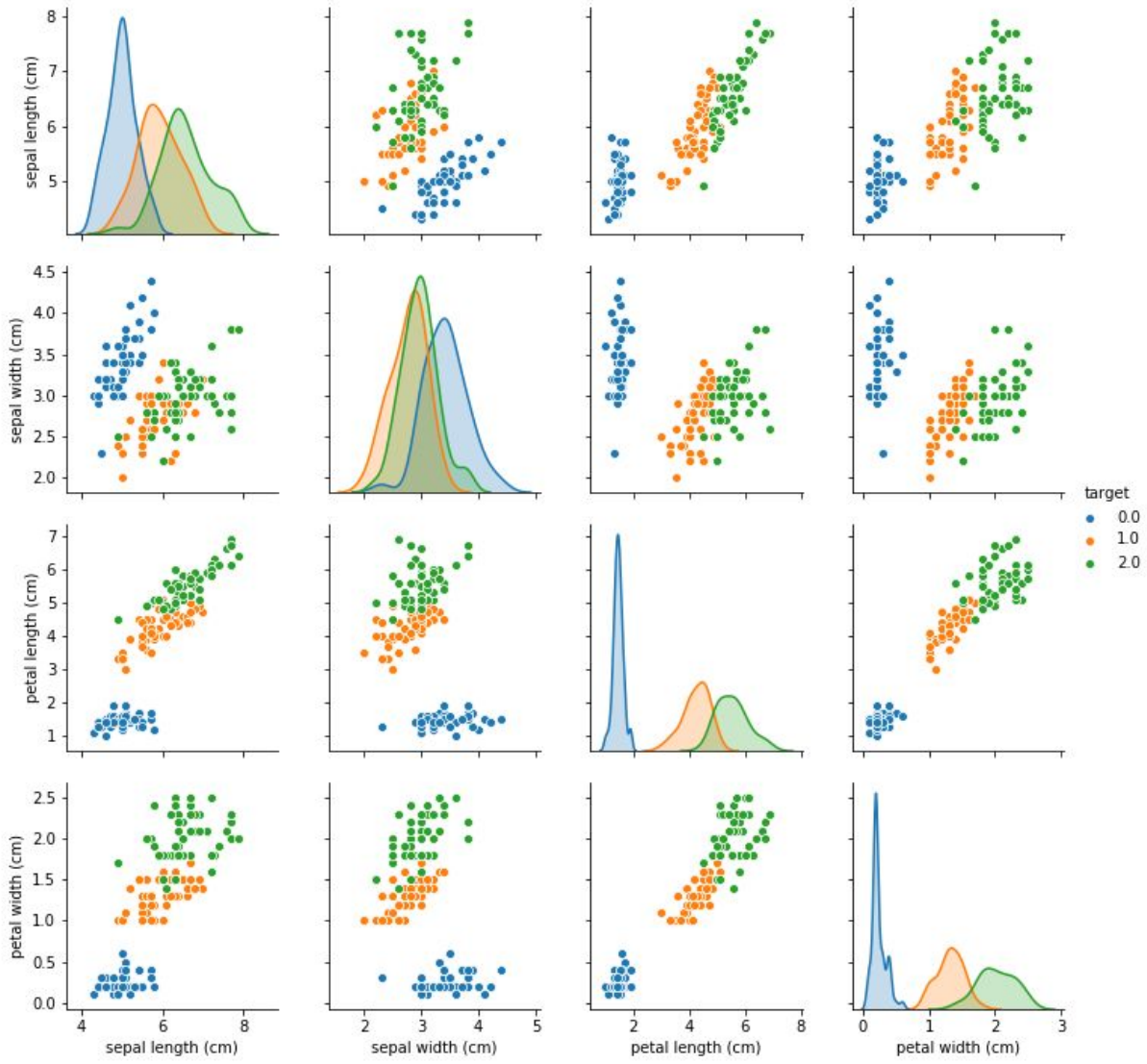


Image 15: Pairwise relationship between different features of the *Iris Dataset*.

The Pairplot shows the pairwise relationships between the four features in the dataset; petal width, petal length, sepal width and sepal length.

The three classes of flowers are denoted by different colors, blue for ***Setosa***, orange for ***Versicolor*** and green for ***Virginica***.
From these graphs we can already see how the features of the Setosa class are extremely different from the ones of the other two classes.
This makes us think that the setosa class will be easier for our algorithm to spot and classify, even if provided with a small training set.

Principal Component Analysis (PCA)

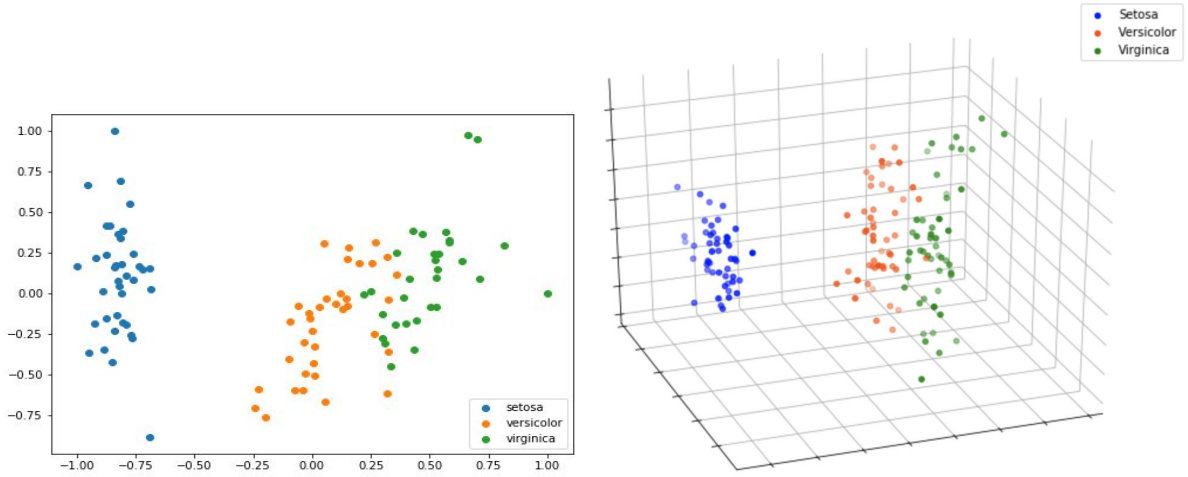


Image 16: Matplotlib.pyplot graphs showing the 2D and 3D PCA for the *Iris dataset* (left to right).

The 2-Dimensional PCA explains 97.76% of the total variation of the dataset, while the 3-Dimensional PCA explains 99.48%.
The PCA plots show that the points belonging to the class ***Setosa*** are situated in a distinct area of the graph, far away from the one where the data belonging to the other two Iris classes are located.
As a matter of fact, the *Setosa* class is linearly separable from the other two, meaning that its data points can be separated from the others through a simple straight line (or a hyperplane in the 3D case).
Instead, the ***Versicolor*** and ***Virginica*** data points are well mixed together, showing that those two classes are not linearly separable and thus will be harder to classify correctly.
The structure of the dataset requires the use of non-linear classification techniques in order to obtain a high accuracy in predicting the classes.

The Quantum-enhanced Kernel Support Vector Machine classifier I'm using uses a process similar to Nonlinear Embedding to separate the points so that different classes occupy distinct high dimensional spaces. This high level classification process should be able to tell the difference between Versicolor and Virginica, achieving a good classification accuracy.

Quantum Circuits

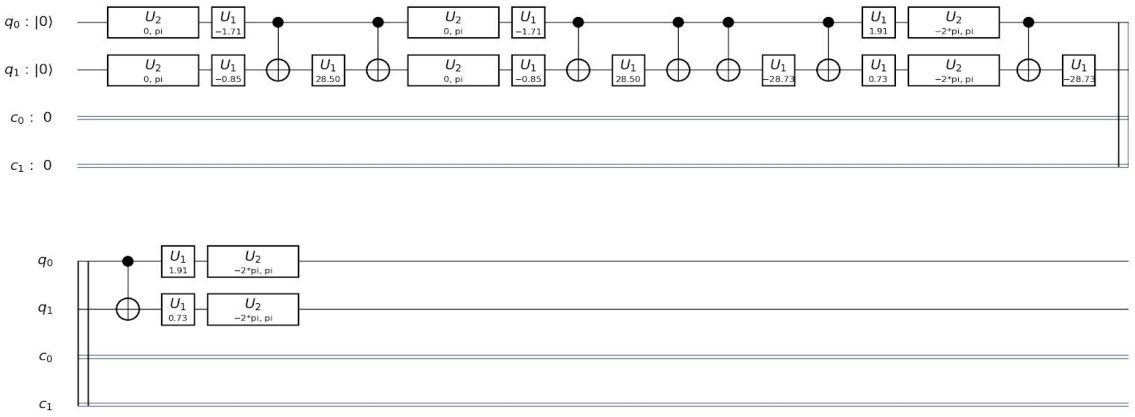
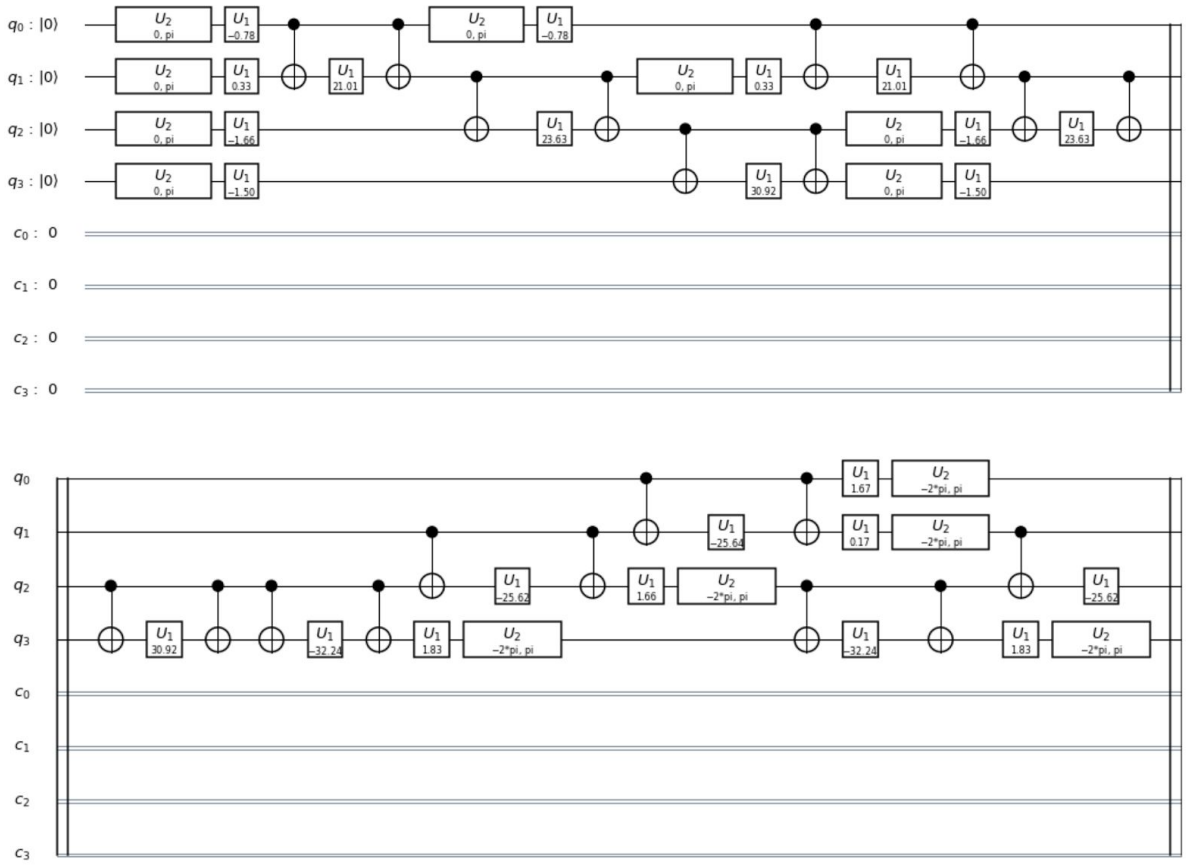


Image 17: Quantum Support Vector Machine circuit on the first two data points, using the *Iris dataset* reduced to 2 principal components (2D PCA).



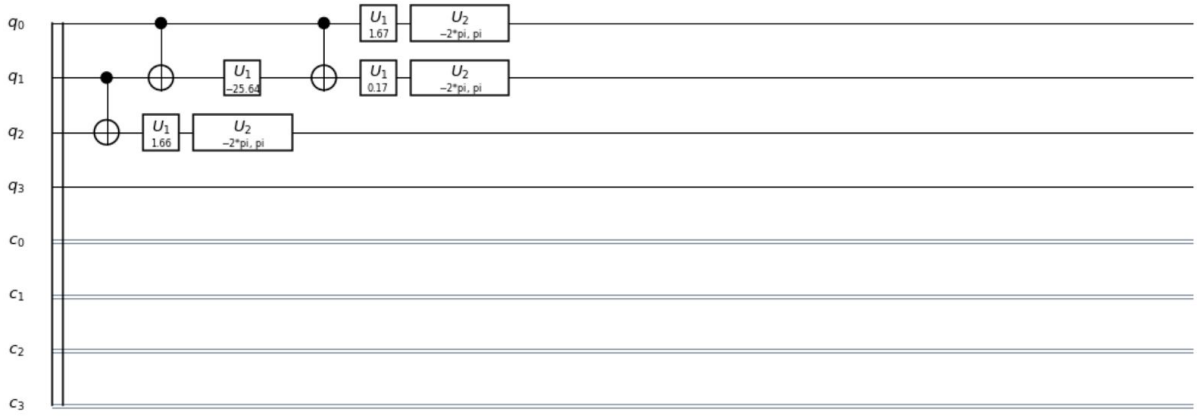


Image 18: Quantum Support Vector Machine circuit on the first two data points, using the *Iris dataset* reduced to 3 principal components (3D PCA).

The circuits above show the quantum gates the QSVM applied on the first two data points of the Iris dataset in the cases of 2D and 3D PCA respectively. Both Graphs were obtained using the function: qsvm.construct_circuit(x1=datapoints[0][0], x2=datapoints[0][1])

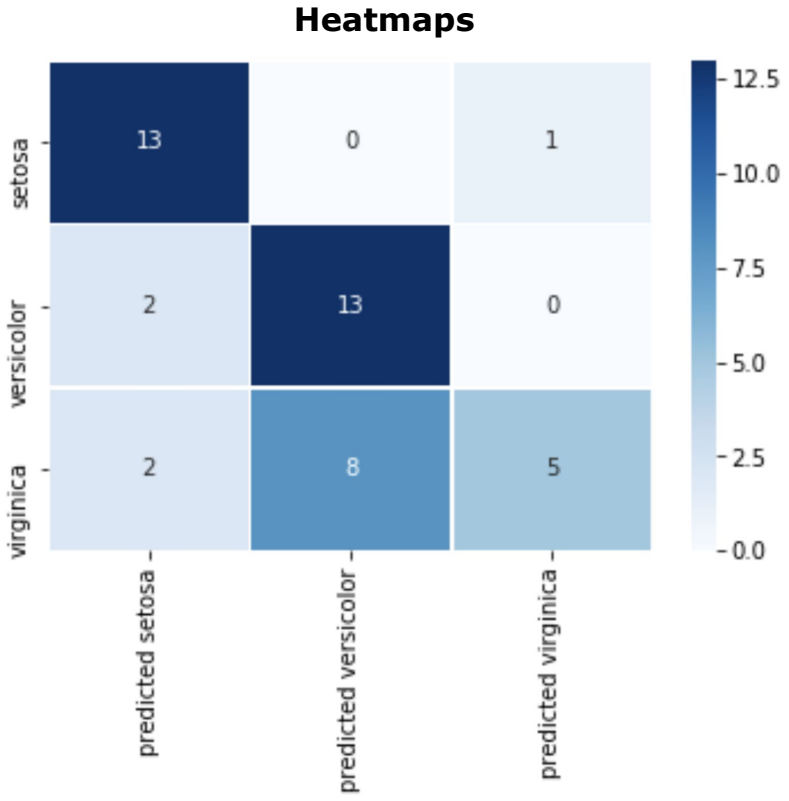


Image 19: Heatmap showing the precision of the classifier on each class, using the *Iris dataset* reduced to 2 principal components (2D PCA).

	precision	recall	f1-score	support
setosa	0.76	0.93	0.84	14
versicolor	0.62	0.87	0.72	15
virginica	0.83	0.33	0.48	15
accuracy			0.70	44
macro avg	0.74	0.71	0.68	44
weighted avg	0.74	0.70	0.68	44

Image 20: Precision scores obtained on the *Iris Dataset*, reduced to 2 principal components (2D PCA).

The QSVM on the Iris dataset gets an average of 70% on the test set. This precision may seem low if compared with a traditional Machine Learning classifier, but we know that the Quantum device is still subject to several decoherence errors that inevitably lower the accuracy. Hopefully in the near future Quantum classifiers will be able to match up with the existing traditional computing technology and maybe even surpass it.

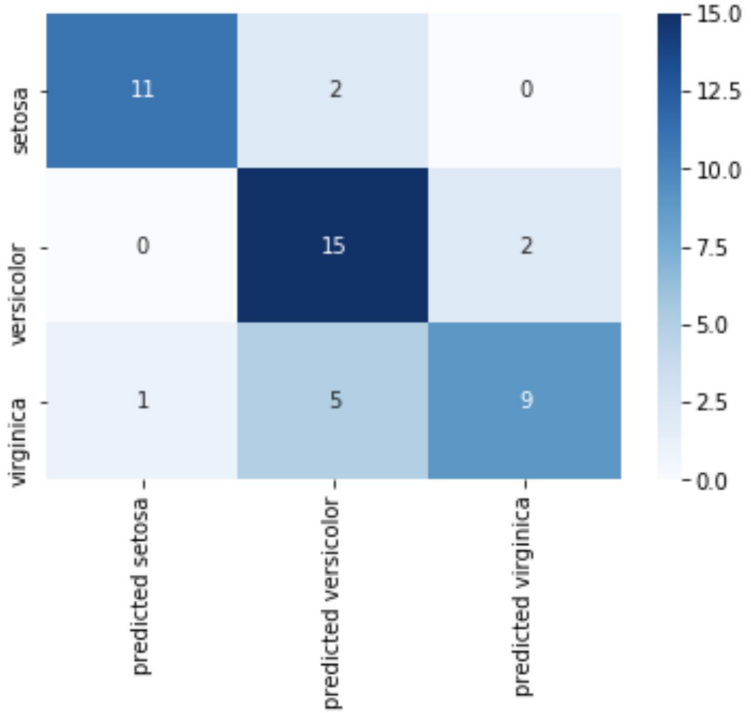


Image 21: Heatmap showing the precision of the classifier on each class, using the *Iris dataset* reduced to 3 principal components (3D PCA).

	precision	recall	f1-score	support
setosa	0.92	0.85	0.88	13
versicolor	0.68	0.88	0.77	17
virginica	0.82	0.60	0.69	15
accuracy			0.78	45
macro avg	0.81	0.78	0.78	45
weighted avg	0.80	0.78	0.78	45

Image 22: Precision scores obtained on the *Iris dataset*, reduced to 3 principal components (3D PCA).

As expected, increasing the number of principal components to three, rises the performance, giving an overall accuracy of 78%.
Hopefully in the near future, better quantum devices will allow us to process large amounts of data of several dimensions, opening up quantum technology to Big Data applications.

Conclusions:

Quantum Machine Learning is a promising emerging field, which has a lot to offer for the future of data analysis. Both the hardware technology and the research are still in their infancy, but recent discoveries in quantum computation have been so fenomenal as to convince several companies and governments to invest in this field. Giant as *Google*, *Microsoft* and *IBM* and some startups, such as *Rigetti* and *Zapata* are racing to get quantum supremacy. As a consequence, there were significant achievements in regression, classification and reinforcement learning tasks, as well as in quantum error correction. All this innovation makes Quantum Computing a really interesting field of study that might revolutionize several aspects of our life, thanks to its contributions to Machine Learning, data encryption and Chemistry.
In my final work I have focused on Quantum Machine Learning, implementing a Support Vector Machine Algorithm using the *Qiskit* development kit from *IBM*. Tackling a quantum algorithm seemed daunting at first but, after understanding the basic theory of quantum computation, I happily surprised myself and managed to create a working quantum code.

The *IBM* technology is still very limited: the quantum devices can only process a limited number of features before they run out of memory, so that the dataset has to be reduced in dimensionality before it can be used to train the classifier. If the quantum technology was powerful enough to allow for a greater number of features, the results would certainly be much better, as adding principal components directly increases the total variation captured by the data and thus the overall accuracy score. Despite this limitation, the algorithm obtained a reasonably good result on the data I provided it with, but the greatest achievement for me is that the algorithm is general enough to be exploited on nearly any dataset, in order to run a classification procedure. This work was so fascinating to me that I will certainly continue to study Quantum Computing and I hope I will be able to become a Quantum Computing expert someday.
I hope you enjoyed reading this research as much as I enjoyed writing it.

List of Tables and Images:

Short Glossary:

Ancilla Qubit: an auxiliary qubit necessary either for quantum error correction or to store entangled states.

Bloch Sphere: an intuitive geometrical representation of the quantum state of a qubit as one or more complex vectors.

Data Mining: a Knowledge discovery process, based on the analysis and comparison of a lot of data, in order to extrapolate new information and patterns.

Decoherence: the loss of quantum coherence due to errors caused by interference with the surrounding environment.

Entanglement: the ability of a qubit to link with another qubit, so that they both react automatically to any change of the partner qubit.

Gate-based Quantum Computer: a quantum device which operates on a small number of qubits, using gate operations.

Kernel Method: algorithm that exploits a kernel function to perform a process that is similar to nonlinear embedding but less computationally expensive.

Nonlinear Embedding: nonlinear transformation of the data onto a high dimensional feature space.

Qiskit (Quantum Information Science Kit): an innovative modular open-source Quantum Computing framework, developed by *IBM*.

Qiskit-Aqua: a modular and flexible open-source library, written in *Python*, which includes advanced quantum algorithms and tools.

Quantum Algorithm: an algorithm designed to work on a quantum computer.

Quantum Annealing (QA)*:* an algorithm for finding the global minimum for an objective function.

Quantum Circuit: an algorithm that can run on a quantum device, it's composed of a sequence of quantum gates.

Quantum Circulator: a device for controlling the flow of a microwave signal in a quantum computer.

Quantum-Classical Interface: the software and hardware needed for a classical computer to communicate with and control a quantum computer.

Quantum Code: a quantum program or circuit, the implementation of an algorithm designed to run on a quantum computer.

Quantum Computer: an innovative and futuristic computing device that can quickly process huge and complex data sets.

Quantum Computing: a kind of computation based on quantum mechanical phenomena, such as superposition and entanglement.

Quantum Computing Device: either a quantum computer or a quantum computing hardware component.

Quantum Computing Technology: any hardware or software relevant to the design, development, construction, deployment or use of a quantum computer.

Quantum Computing Toolkit: a collection of software tools and libraries which enable and facilitate the development of quantum programs.

Quantum Dot Technology: any device based on quantum dots, which are artificial atoms made up of nanocrystals of semiconductor material.

Quantum Kit (Q-Kit): a graphical user interface, consisting of a quantum circuit simulator which shows the effect of quantum gate operations.

Quantum Interference: a physical phenomena related to the wave nature of particles, where two or more independent qubits have an interaction, that can have constructive or destructive effects.

Quantum Gates: the building blocks for quantum circuits, unitary transformations chosen from a continuum of possible values.

Quantum Error Correction (QEC): a set of codes, techniques and methodologies to reduce decoherence errors.

Quantum Machine Learning: field of Computer Science which studies how to perform machine learning techniques on quantum computers.

Quantum Mechanical Properties: Superposition, Entanglement, and Interference.

Quantum Parallelism: the ability of a quantum computer to perform an operation on a quantum memory register which results in the simultaneous calculation of a function of 2^n different values.

Quantum Simulator: a device that actively uses quantum effects to answer questions on model systems.

Quantum Volume: a measurement system for testing the performance of quantum devices, based on the number and quality of qubits, circuit connectivity and error rates of operations.

Qubit (Quantum Bit): the basic unit of information of a quantum computer.

Superposition: qubits' ability to be in more than one quantum state at the same time.

Supervised Learning: a Machine Learning technique to make an algorithm learn from example. It uses labeled data to train the algorithm to perform its task.

Support Vector Machine (SVM): an algorithm (or recipe) that learns by example to assign labels to objects.

Unsupervised Learning: the collection of Machine Learning techniques that analyze unlabeled data and process them without any guidance or prior information.

Topological Quantum Computer: a quantum computer that is still in development, which would use anyons, in order to to avoid environmental perturbations.

Sources:

Bibliography:

Alicki R., Lendi K., *Quantum Dynamical Semigroups and Applications*, Springer, Berlin, 2007.

Benenti G., Casati G., Rossini D., Strini G., *Principles of Quantum Computation and Information:A Comprehensive Textbook*, World Scientific Pub Co Inc, 2019.

Breuer H.P., Petruccione F., *The Theory of Open Quantum Systems*, Oxford University Press, New York, 2006.

Brown J., *Minds, Machines, and the Multiverse: The Quest For The Quantum Computer,* Simon & Schuster, New York, 2002.

Hirvensalo M., *Quantum Computing*, Springer-Verlag, Berlin, 2001.

Kaye P., Laflamme R., Mosca M., *An Introduction to Quantum Computing*, Oxford University Press, 2007.

Kraker J., Valle C., *Shor's algorithm and Grover's algorithm in Quantum Computing*, Proquest, 2011.

Hurwitz J., Kirsch D.*, Machine Learning For Dummies*, IBM Limited Edition Published by John Wiley & Sons, 2018.

Nakahara M., Ohmi T., *Quantum Computing: From Linear Algebra to Physical Realizations,* CRC Press, 2008.

Narayanan A., Meneer T., *Quantum Artificial Neural Network Architectures and Components*, in Information Sciences, October 2000.

Nielsen M.A., Chuang I.L., *Quantum Computation and Quantum Information*, Cambridge University Press, 2010.

Rezzani A., *Big Data Analytics. Il manuale del data scientist.,* Apogeo Education, Milano, 2017.

Schumacher B., Westmoreland M., *Quantum Processes Systems and Information*, Cambridge University Press, 2010.

Schlosshauer M., *Decoherence And the Quantum-To-Classical Transition,* Springer, Berlin, 2007

Veldhorst M. et al., *Silicon CMOS Architecture for a Spin-Based Quantum Computer*, Sept. 2016.

Wittek P., *Quantum Machine Learning: What Quantum Computing Means to Data Mining,* Elsevier, 2016.

Witten I.H., Frank E., Hall M.A. , Pal C. J. , *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufmann Publishers, 4th Edition, 2016.

Yanofsky N.S., Mannucci M.A., *Quantum Computing for Computer Scientists*, Cambridge University Press, 2008.

Sitography:

https://www.xanadu.ai/?fbclid=IwAR2rMaikICO6Ax62sKOVuwUSn03fdyDV8vXSGC4XxM_uneOIPiIYoh5mqm0

https://www.quantum-inspire.com/kbase/introduction-to-quantum-computing/

https://www.quantum-inspire.com/?fbclid=IwAR1YhtmaWnUHL0rn2MVFdHAMbUIAJdeEFQ_fIxoPSfEMwQD47YmQVL0CByY

https://docs.microsoft.com/en-us/quantum/concepts/?view=qsharp-preview&fbclid=IwAR0MiVybn6OEyCQuYR3KJpa6AOCXdVMmdoyHT1CqN2uU3jEIdtPZXx-9SoU

https://docs.microsoft.com/en-us/quantum/concepts/circuits?view=qsharp-preview

https://www.xanadu.ai/software/?fbclid=IwAR2rbXkzbPiZ4Q9n1wWsNXS6LdKtEQiKlSb7oUUu4WRXxImaJ-pME-r3Ngk#pennylane

http://quantum.ustc.edu.cn/web/index.php/en/node/1

https://cambridgequantum.com/cqc-and-jsr-corp-issue-statement-on-their-quantum-computing-project/

https://docs.microsoft.com/en-us/quantum/concepts/dirac-notation?view=qsharp-preview

https://phys.org/news/2019-01-quantum-chemistry.html

https://plato.stanford.edu/entries/computation-physicalsystems/

http://khaledelleithy.org/Conferences/Quantum%20Computing%20Hardware%20Implementation%20Methods

https://qt.eu/understand/underlying-principles/gate-based-qc/

https://newsroom.intel.com/press-kits/quantum-computing/#gs.w57km0

https://en.wikipedia.org/wiki/Quantum_computing

https://plato.stanford.edu/entries/qt-quantcomp/
https://cloudblogs.microsoft.com/quantum/2018/09/06/developing-a-topological-qubit/
https://ai.googleblog.com/2018/03/a-preview-of-bristlecone-googles-new.html
https://quantumcomputingreport.com/our-take/better-qubits-versus-more-efficient-error-correction/
https://en.wikipedia.org/wiki/Topological_quantum_computer
https://medium.com/swlh/topological-quantum-computing-5b7bdc93d93f
https://qosf.org/project_list/
ttps://quantiki.org/wiki/list-qc-simulators
https://www.scientificamerican.com/article/quantum-computers-compete-for-supremacy/
https://syncedreview.com/2019/04/03/ibm-proposes-quantum-enhanced-feature-space-for-ml/
https://Qiskit.org/documentation/aqua/library.html#aqua-library
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.45.8794&rep=rep1&type=pdf
http://www.comp.tmu.ac.jp/morbier/R/Fisher-1936-Ann._Eugen.pdf
https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Original)
https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_iris.html
https://www.octimine.com/patents-of-quantum-computing-companies/
https://cloudblogs.microsoft.com/quantum/2018/09/06/developing-a-topological-qubit/
https://cloudblogs.microsoft.com/quantum/2018/06/06/the-microsoft-approach-to-quantum-computing/
https://cloudblogs.microsoft.com/quantum/2018/07/23/learn-at-your-own-pace-with-microsoft-quantum-katas/
https://www.research.ibm.com/ibm-q/
https://www.research.ibm.com/ibm-q/learn/what-is-quantum-computing/
https://www.theverge.com/2019/1/8/18171732/ibm-quantum-computer-20-qubit-q-system-one-ces-2019
https://developer.ibm.com/dwblog/2017/quantum-computing-coding-ibm-q-experience/
https://www.research.ibm.com/ibm-q/technology/experience/
https://www.allaboutcircuits.com/news/quantum-composer-ibm-5-qubit-quantum-computer-cloud/
https://ai.google/research/teams/applied-science/quantum-ai/
https://interestingengineering.com/googles-quantum-processor-may-achieve-quantum-supremacy-in-months
https://www.21stcentech.com/2019-year-quantum-computer/
https://www.dwavesys.com
https://spectrum.ieee.org/tech-talk/computing/hardware/dwave-launches-free-quantum-cloud-service
https://Qiskit.org/
https://arcb.csc.ncsu.edu/~mueller/qc/qc18-2/qc18/readings/gb_svm_risk.pdf
https://people.eecs.berkeley.edu/~christos/classics/Feynman.pdf
http://docs.rigetti.com/en/stable/qvm.html

https://medium.com/rigetti/the-rigetti-128-qubit-chip-and-what-it-means-for-quantum-df757d1b71ea
https://www.intel.com/content/www/us/en/research/quantum-computing.html
https://spectrum.ieee.org/nanoclast/computing/hardware/intels-new-path-to-quantum-computing
https://www.pcgamesn.com/quantum-computing-researchers-development-where-are-we-now
https://www.techradar.com/news/best-cloud-computing-service
http://www.quantumplayground.net/#/home
https://www.researchitaly.it/en/projects/the-future-is-in-quantum-technology/
https://hackernoon.com/quantum-machine-learning-d0037f59f31a
https://medium.com/Qiskit/Qiskit-aqua-a-library-of-quantum-algorithms-and-applications-33ecf3b36008
https://Qiskit.org/documentation/aqua/algorithms.html
https://Qiskit.org/documentation/_modules/Qiskit/aqua/algorithms/many_sample/qsvm/qsvm.html
https://phys.org/news/2019-03-machine-path-quantum-advantage.html
https://dotscience.com/product/?gclid=Cj0KCQjw4s7qBRCzARIsAImcAxYDGC1kdYhsYCjHwfOIcu1mND-n90u7SCWqdxs_9au_nGENCrq5pMQaAmJgEALw_wcB
https://www.zhinst.com/products/quantum?gclid=Cj0KCQjw4s7qBRCzARIsAImcAxaPdePH3cYOYR4c5NbgDHEEhGpFG7V7HVglPMlluoGWHfQxwkBKjWkaApwcEALw_wcB#
https://www.geeksforgeeks.org/classifying-data-using-support-vector-machinessvms-in-r/
https://www.geeksforgeeks.org/learning-model-building-scikit-learn-python-machine-learning-library/