Dipartimento
di Impresa e Management

Cattedra: Artificial Intelligence and Machine Learning

Quantum Computing Explained in a (Qu)Bit!
Potentialities, Present Fragilities and Future Prospects.
Practical implementation using the Python language.

Relatore:

Prof. Giuseppe Francesco Italiano

Candidata:

Annalisa Feliziani
Matricola: 235651

Anno Accademico 2020/2021

Abstract

Machine learning aims at automatically identifying structures and patterns in large data sets. However, the large dimensionality of the vector spaces involved in operations makes the execution of machine learning algorithms at large scale very resource-intensive, thus motivating the development of innovative methods to lower their computational cost. Among these methods, the employment of quantum computing solutions is gaining particular interest among academics.

The discovery of new quantum algorithms triggered a lot of research efforts at the intersection of machine learning and quantum information processing, leading to important developments in the field which culminated in the construction of the first quantum simulators and quantum computers, operated by the most advanced technological firms (above all, Google and IBM). Other important achievements include the implementation of quantum versions of well-known classical algorithms, which are then applied to problems pertaining to a variety of fields, from physics to finance.

The financial applications of machine learning and quantum machine learning methods are of particular interest for the present thesis and will be examined more in detail. However, the realization of these quantum algorithms on pure quantum computers is still years away, as the resource requirements for their implementation is unfortunately prohibitive for devices that are and will be available in the short term (Henriet et al.2020), which are known as NISQ (Noisy Intermediate-Scale Quantum).

Nonetheless, active research is exploring the capabilities of currently available quantum machines, albeit imperfect especially in terms of error-correction and of number of qubits. This thesis intends to dive into the research around quantum computation in order to write a compendium explaining its main concepts, by consulting the latest research on the one hand, and by practically implementing a quantum machine learning algorithm on a financial dataset on the other. This is done in order to make comparisons between traditional machine learning algorithms and their quantum counterparts and to make inferences about their differences in an attempt to understand all the potentialities of quantum computation, as well as the fragilities and the future prospects within this exciting new field of computation.

Acknowledgements

To my thesis Supervisor, Prof. Giuseppe F. Italiano for his help and support while writing this thesis and throughout these years.
To my parents and my family for their unconditional love.
To the best part of me, the person I admire the most: my sister Arianna.
To all my friends and colleagues, for the incredible journey we shared.
FInally, to my lover, my best friend, my partner in crime: you are my one and only.
Thank you for all the support you showed me during the darkest times: I love you all.

Contents

Introduction

The discipline of quantum computing is growing fast, gaining progressively more attention thanks to the high potentials it promises.
This is especially true in the field of artificial intelligence and machine learning, where quantum computing pledges to significantly reduce runtimes on larger datasets compared to classical computing. As a matter of fact, "quantum computation and machine learning are nowadays recognized as two closely-related connected research fields" (Sergioli et al. 2019).
However, an important distinction regarding quantum algorithms needs to be made: pure quantum vs. machine learning quantum. The former are algorithms specifically designed to run on a quantum computer, which makes their practical implementation still intractable in the present era (NISQ); the latter are quantum versions of well-known classical machine learning algorithms, and have been developed so that they can be run on NISQ devices, hence accounting for errors coming from both the devices themselves and the environment.
Following this distinction, the thesis is developed by first explaining the concepts of classical and quantum machine learning algorithms and their properties, which are described in detail from a mathematical perspective. After introducing the main theoretical concepts, comparisons are carried out and conclusions are drawn with respect to the main differences among these methods, as well as the implications of these differences from a computational perspective. This is the first contribution of this work, which is very important in order to better understand all the potentialities of this new computational field. This also helps understand why the efforts towards the design of new quantum machines should benefit the research world and more generally the future technological advances. For this purpose, a table summing up all the findings stemming from these comparisons was created. It can be found in Chapter 5. Another main contribution of this thesis is the practical implementation of a quantum version of the classical K-nearest neigbours machine learning algorithm, namely the Quantum K-nearest neighbours algorithm. The aim is to perform a binary classification task using these two models on a financial pre-processed dataset in order to understand the quantum advantage of the latter method over the former.
As a matter of fact, today it is possible to simulate quantum systems on classical computers thanks to Software Development Kits (SDK). An example is Qiskit, provided by IBM Q and implemented using the Python environment. In particular, a quantum simulator will be used to test the Qknn on a financial, pre-processed dataset. The choice of carrying out this task on this specific dataset stems from previous research: Daniel Kok (2020) managed to implement a quantum version of the knn algorithm, but when he tried to test it on a dataset containing information about German credit, it did not perform well compared to the classical machine learning version. He then suggested one possible explanation: it could be that, since the dataset was not previously pre-processed, this could have influenced the results. Therefore, the present thesis follows his work up with the intent to find an optimal solution to this problem. The aim is to verify this theoretical advantage of the quantum method by comparing the results obtained from the QKNN and the knn. This has also very important implications because, by comparing the results, the differences between the classical and the quantum methods will be better analysed. But there is also another reason. This new algorithm will be designed based on the representation of classical patterns in terms of quantum objects, with the aim to increase the computational efficiency of the classical counterpart. This thesis will show that this approach can be made theoretically possible thanks to the versatility of the quantum states properties, which in turn allows to design and exploit quantum algorithms that boost the time complexity of the standard

algorithms, potentially leading to computational advantages from a quantum perspective.
Finally, all the conclusions from these comparisons are used to reason about the important implications and the possible future directions researches will take in this field.
As a result, the thesis can be regarded as a roadmap to dive into the field of quantum computing theory, with the purpose to render it easily comprehensible for every reader. This is a very important contribution because it allows to better understand where the future of computation is going, and what will be the main drawbacks in the foreseeable future. In particular, from the analysis carried out in this work, it is clear that the devices available nowadays, namely the NISQ, are predominant. This implies that all the quantum algorithms that are being developed today will be influenced by quantum error in the near future. On the other hand, gaining an in-depth understanding of quantum mechanisms is the only way in which improvements can be made, in favour of a shift from the NISQ era to Fault-Tolerant era: that is, the stage in quantum computing advances in which devices will be completely free from quantum error, therefore representing the arrival point in the development of quantum devices.
Important discoveries towards the achievement of such "pure" quantum machines are being made, as the thesis will explain in detail. However, it is clear that much still needs to be done, as the research is still moving its first steps in a field of endless possibilities.
François Chollet gives a definition of artificial intelligence in his book "Deep Learning for Python": it is "the effort to automate intellectual tasks normally performed by humans." As such, artificial intelligence encompasses both machine learning and deep learning, where the latter is a subfield of the former. In the same book mentioned above, the author recalls a quote by a pioneer of computer science, Alan Turing, in an attempt to explain the concept of machine learning: "machine learning arises from this question: could a computer go beyond "what we know how to order it to perform" and learn on its own how to perform a specified task? Could a computer surprise us? Rather than programmers crafting data-processing rules by hand, could a computer automatically learn these rules by looking at data?" (Turing 1950).
Of course, the answer to Turing's question is: yes! In particular, the peculiarity of a machine-learning system is that it is trained rather than explicitly programmed. The training consists of feeding a computer with many examples relevant to a task, in order for it to find statistical patterns which allow the system to finally output the rules for automating such task.
This process, as well as the comparison with classical computing, can be visualized in the image below:

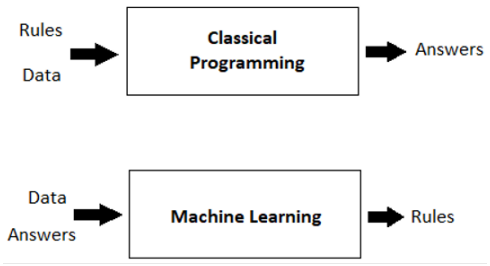


Figure 1: the upper diagram shows that classical programming comprises two inputs, the data and the rules to be applied to the data, and the system generates the answers, whereas the lower diagram represents the process happening in machine learning: the data and the answers are fed to the system, which in turn will output the rules to apply to some new data.

Although machine learning started to flourish only in the 1990s, it has quickly become the most popular and most successful subfield of AI.
This trend was driven by the availability of faster hardware and larger datasets. As a matter of fact, machine learning tends to deal with large, complex datasets (such as a dataset of millions of images, each consisting of tens of thousands of pixels) for which it is necessary to overcome classical statistical analysis such as Bayesian analysis, as this would be impractical for the manipulation of such a big mole of data. As a result, machine learning exhibits comparatively little mathematical theory and is more engineering oriented (Chollet 2017).
The potentialities of artificial intelligence, and especially machine learning, are demonstrated by the many successful applications which have been taking place throughout the years: from self-driving cars which project us directly into the future, to more daily instances such as the recommendation systems on which Netflix or Spotify are based (among others), the revolution realized by artificial intelligence is impacting our lives in a wide number of fields, from marketing to finance to medicine and so forth.
What about quantum computing? Where does its innovation come from?
Loosely speaking, the fact that some very powerful principles drawn from quantum mechanics can be used, leading to an improvement in terms of computational time and also error minimisation, with respect to classical computing. Such principles include the entanglement, the superposition and the interference, and they are discussed in more details in Chapter 3. This gives a completely new perspective on the development of the already promising fields of artificial intelligence and machine learning, as it opens to the possibility of exploiting the principles of quantum mechanics in order to reduce the resource complexity of classical methods exponentially.
As a matter of fact, in the field of computer science and algorithms, problems are catalogued following their computational complexity, which is the analysis of the efficiency of a given algorithm based on the resources it needs in order to run, both in terms of time and space. In the next chapter, some of the main classical and quantum algorithms are presented and compared, following a theory which is known as the “Computational Complexity Theory”.

1 Chapter 1 - Preliminaries

1.1 Introduction

This chapter will provide the basic theory on which the thesis is built. It will dive into the concepts of computational complexity, classical and quantum algorithms and the comparisons among them; finally, it will outline the structure of the thesis, the research questions, the method and the original contributions.

1.1.1 Computational Complexity Theory

The Computational Complexity Theory (or simply "Complexity Theory") analyses the computational resources which are necessary and sufficient to solve certain computational problems (Köbler 1993).
Problems are traditionally divided into decision problems and search problems (also known as relation problems).
The structure of a general decision problem is the following: "given an input x, say whether F(x) holds." These types of problems can be classified in P (Polynomial time), NP (Nondeterministic Polynomial time), NP-hard, NP-complete and BQP (Bounded error Quantum Polynomial time).

- P problems: decision problems that can be solved by a deterministic Turing machine in polynomial time. An example is sorting.

1.1.2 Sorting

In sorting, a disordered list of elements is given, and we want to order it. We can use one of a several number of sorting algorithms. This task is relatively easy for a classical computer, since there exists an algorithm which can solve this problem in a relatively small number of steps (polynomial of n).
For example, one of the most efficient sorting algorithms is MergeSort, which has a time complexity of $O(n \log(n))$. Developed by John Von Neumann in 1945, it follows a general, three-steps pattern called "Divide and Conquer":

1. First, it divides the input into two subarrays of roughly equal size;
2. Then it recursively merges all sorted elements of each subarray;
3. Finally, it merges the two subarrays into a single sorted array.

The sorting problem is also efficiently solved by a quantum computer, but without outperforming classical computing (the running time remains at least $(n \log(n))$).
Therefore, in general, there is no real advantage in using quantum computers to solve these types of tasks. This is true for all problems in P.

- NP problems (Nondeterministic Polynomial time): problems that can be solved by a non-deterministic Turing machine in polynomial time, but which can be verified by a deterministic Turing machine in polynomial time. This category comprises both NP complete problems and NP hard problems:

- NP hard problems: the class of decision problems to which all problems in NP can be reduced in polynomial time by a deterministic Turing machine.

- NP-complete problems: the intersection of NP-hard and NP problems.
  Equivalently, NP-complete is the class of decision problems in NP to which all other problems in NP can be reduced to in polynomial time by a deterministic Turing machine. There is no known algorithm that solves an NP-complete problem in less than exponential time, but a solution can be verified in polynomial time.

However, quantum computers are neither deterministic nor Turing. Indeed, there exists a different class which contains all the problems that can be efficiently solved by a quantum machine in polynomial time:

- BQP: problems which can be solved by a quantum computer in polynomial time. The idea is to make use of a quantum system (the quantum computer) to simulate another quantum system: nature. In particular, some quantum algorithms for Hamiltonian simulation have been developed, which have an exponential vantage in describing natural phenomena with respect to classical algorithms.

A problem which is not known to be contained in P but is contained in BQP is factorization.

1.1.3 The Factorization Problem

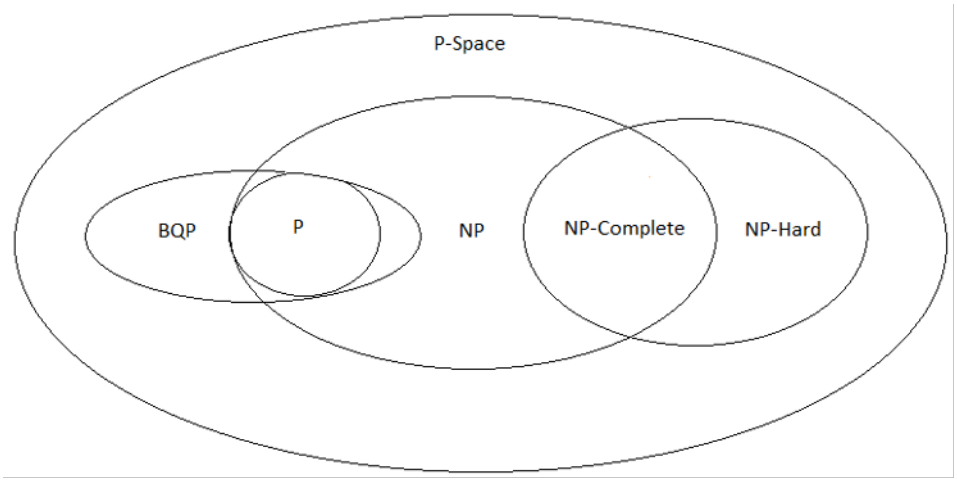
We need to find two prime numbers p and q whose product is n. This is a difficult task for a classical computer, since the best algorithm able to solve it, which is the number field sieve method, has an exponential complexity of $O(exp[c(\ln n)^{1/3}(\ln \ln n)^{2/3}])$. This limitation of classical computers, which are only able to process a reduced number of ciphres, is exploited in the safety protocol called RSA, from its inventors Rivest, Shamir and Adleman. It can be schematized as follows:

1. Choose two prime numbers p and q;
2. Calculate their product N = $p * q$, called module;
3. Choose a number e (public exponent), smaller than N and prime with respect to: $\Phi(N) = (p-1) * (q-1)$ where Φ is the Euler function[1].
4. 4. Calculate the number d (private exponent) such that $e * d = 1$.

A message m is coded through the operation $c = m^e$. To decode a message, it is therefore necessary to know p and q through the factorization of N. This is impractical for a classical computer, which would take so long to find that number that, in the meantime, the password that protects the coded message has been already changed several times.
This problem is extremely overcome by the Shor quantum algorithm; indeed, it is able to factor large numbers in polynomial rather than exponential time. Just to give an idea, the most powerful supercomputer would take millions of years to find the factorization of a 2048-ciphres number, whereas a quantum computer would take only a few seconds. The possible future implementation of the Shor algorithm on a quantum computer would therefore threaten the safety of a myriad of information. A number of solutions were already found: among them, quantum cryptography, which is a protocol based on peculiar properties of quantum mechanics, where it is not possible to realize a quantum measure on a particle without altering its state. In the context of cryptography, this means that, once the photon which brings information inside an optic fiber is identified, it is not possible to clone it in

[1] function which associates to each number n the prime numbers p such that MCD(n, p) = 1

any way. This guarantees the safety of all the users who will rely on this new, sophisticated technology. A full description of Shor's algorithm is provided in Chapter 3.
It may be useful to visualize the concepts described so far with a simple Euler diagram I realized, representing the intersections of the various classes in the P-space as follows:



Along with this categorization of decision problems, the theory of complexity also includes various classes of search problems: FP, FNP, and FBQP, where "F" stands for "Function". The structure of a general search problem is: "given an input x and a binary relation R, find a y such that R(x,y) holds, if such a y exists." An example is the unstructured search problem.

1.1.4 The Unstructured Search Problem

In this problem, we want to find a particular item x^* in a sequence (e.g. a list) of N items, such that $f(x^*) = 1$. Unstructured means we have no prior knowledge regarding the order of the items.
Indeed, if we knew the sequence was ordered, we could simply apply a classical searching algorithm such as Binary Search, which works by breaking down the list in half on every iteration and outputs the result in O(logN) in the worst case. However, when the sequence is unstructured, like in this case, the worst-case time complexity of any classical searching algorithm would be O(N).
The quantum counterpart is Grover's algorithm, which in order to find the element x^* makes a number of $\sim \sqrt{N} = 2^{\frac{n}{2}}$ calls to the "oracle" (which is a "black box" system that can be viewed in terms of its inputs and outputs, without actually knowing its inner working). The final solution is obtained in $0(\sqrt{N})$ steps, which is a quadratic speed-up compared to the classical solutions. Therefore, this problem is in FPN, and in particular in FBQP.
It is also possible to use this algorithm to solve decision problems, by reducing them to search problems. As a matter of fact, Grover's algorithm "can immediately be applied to any problem in the complexity class NP" (Montanaro 2016).
Its applications range from database search to optimization problems such as routing or scheduling, as well as portfolio optimization or machine learning with the Knn.
The table I realized below aims at summarizing the comparison among all the algorithms presented thus far:
It would appear clear from the discussion above regarding the potentialities of quantum computing of outperforming classical computing in certain tasks (although not all tasks) that quantum computers are essential to building such quantum programs in order for the

Problem Name	Algorithm	Categorization	Problem(s) solved					Time-Complexity	Quantum Advantage
			P	BQP	NP	FP	FBQP		
Sorting	MergeSort	Decision Algorithm	Yes	/	No	/	/	O(n log(n))	/
	Quantum Sorting	Decision Algorithm	/	Yes	No	/	/	O(n log(n))	None
Factorization	Number Field Sieve Method	Decision Algorithm	Yes	/	No	/	/	$O[c(\ln n)^{1/3}(\ln \ln n)^{2/3}]$	/
	Shor	Decision Algorithm	/	Yes	/	/	/	$O(\log n)^2(\log\log n)$	Exponential
Unstructured Search	Binary Search	Search Algorithm	/	/	/	Yes	No	O(log(n))	/
	Grover	Search Algorithm	/	/	Yes	Yes	Yes	$O(\sqrt{N})$	Quadratic

quantum algorithms to run properly. As a matter of fact, nowadays, the most advanced firms (in terms of technological capabilities and customer reach) are starting to sense this shift towards quantum-based solutions, and they have been working towards the creation of a quantum computer and relative libraries to write quantum code.
For instance, Google recently announced TensorFlow Quantum (TFQ), an open-source library for quantum machine learning. Other examples include IBM, which has established "IBM Q", the first 20-bit commercial quantum computer offering cloud access to the most advanced quantum computers available.
However, the quantum algorithms I presented up to now need to be run on a particular type of quantum computer, namely an "error corrected" one. This is because qubits contain probabilities, which makes them especially prone to measurement errors and noise from the environment, as opposed to bits.
These types of computers are not yet available, as it is extremely challenging for researchers to achieve a level of error correction which would allow the quantum algorithms to run smoothly. The quantum computers which have been built up to know, including Google's and IBM's, are still considered very "noisy" in terms of error handling, in addition to the problem that they still comprise a little number of qubits compared to what would really be necessary.

1.2 Quantum Supremacy and Error Correction

"Quantum supremacy is so 2019. In quantum computing, error correction is the next hot thing." (Cho 2020).
By quantum supremacy (or quantum advantage) it is intended the goal of demonstrating that a programmable quantum device can solve a problem that no classical computer can solve in any feasible amount of time, irrespective of the usefulness of the problem.
Harrow et al. (2017) have set four (plus one) requirements to experiments that show quantum supremacy:

1. A well-defined computational task (it does not need to be of any practical use except for showing quantum supremacy, which is, in its own right, a worthy goal (Preskill 2018)).

2. A quantum algorithm able to solve the problem, which can be run on near-term quantum devices (NISQs);

3. An amount of resources allowed to any classical competitor;

4. An assumption from complexity-theory, supporting a quantum supremacy hypothesis in the given experiment;

   along with the optional requirement:

5. A verification method that can easily distinguish the quantum algorithm from its classical competitors.

As of today, the competition for quantum supremacy is between China and the US: in 2019, Google announced to have reached it with its 54-qubit Sycamore processor that was able to perform a calculation in 200 seconds. That task would have taken the world's most powerful supercomputer 10,000 years. Now, a team in China has demonstrated that it has built the world's most powerful quantum computer, surpassing Google.
However, as stated by the physicist Adrian Cho, error correction is the real deal. As a matter of fact, the biggest challenge which are facing researchers today is related to the problem of building a machine which is able to deal with errors efficiently, but problems would not even end here: as Cho argues, "manipulating individual qubits can introduce errors, and unless that error rate falls below a certain level, then entangling more qubits with the original one only adds more noise to the system".
Another daunting challenge that experimenters face regarding error correction is that "once scientists have begun to master error correction, they'll have to repeat nearly every discovery so far in quantum computing with the more robust but highly complex logical qubits" (Cho 2020). Therefore, the problem of error correction represents the main obstacle towards the achievement of a really efficient quantum device. Nonetheless, a solution exists, as Sergioli et al. suggest: up to now, they argue, the best practice is to use hybrid solutions, which make use of quantum formalism to obtain significant benefits in the classical computation contexts. As they write, 'he recourse to quantum states to represent classical patterns is naturally motivated by the possibility of exploiting the potential of quantum algorithms to boost the classification process".

1.3 Problems on The Feasibility of Quantum Computing

There is a general polarization of ideas among researchers in the field of quantum computing, stemming from the fact that, despite the theoretical proofs which would favour quantum computing as opposed to classical computing, there are still many major challenges awaiting, which are fundamental to make progresses towards a shift in the paradigm of the world of computing. An example is the fact that, as of today, quantum machines are very noisy and must operate in extreme physics conditions (-273 degrees Celsius), due to the presence of qubits. This is also considerably expensive in terms of economic resources. So, despite the many advances that have already been made, scientists are still debating whether it is actually true that quantum computing will outperform classical computing, or whether it will be really possible to construct a quantum computer in practice.
Some of the main critics are related to the technical challenges that would have to be overcome, which irremediably prevent the creation of a useful quantum computer 'in the foreseeable future" (Mikhail Dyakonov). Dyakonov points out that the number of continuous parameters describing the state of such a useful quantum computer at any given moment must be at least 21 000, arguing that "a useful quantum computer needs to process a set of continuous parameters that is larger than the number of subatomic particles in the observable universe". Finally, the author stresses with emphasis the already mentioned problem of error

correction, highlighting his uncertainty towards the existence of feasible solutions to it, in opposition to the almost exhausting endeavours of many researchers.
"While I believe that such experimental research is beneficial and may lead to a better understanding of complicated quantum systems, I'm skeptical that these attempts will ever result in a practical quantum computer" he concludes.
On the other hand, Jay Gambetta, who leads IBM's quantum computing efforts, showed all his optimism by stating "in the next couple of years, you'll see a series of results that will come out from us to deal with error correction."
As a matter of fact, error correction is one of the main concerns which need to be addressed in order for the research in the field of quantum computing to keep progressing, and this is what the main leading firms in the field are doing.
Quantum computing has a great appeal to scientists and researchers, and many advances are being made; however, they are still at an early stage, and much needs to be done to eventually overcome classical computing.

1.4 Research Questions

Due to the fact that developments in quantum computing are still at an early stage, it is difficult to dive into this topic without generating a great deal of confusion, as the opinion of academics and researchers are very polarized.
In order to better visualize the potentialities of this emerging field, this thesis will dive into several quantum and quantum machine learning algorithms, and will try to optimize the existing quantum counterpart of a simple, supervised machine learning algorithm (the knn), that will perform classifications on a financial dataset representing the German credit, which will be pre-processed before trials. This is done based on previous research (Afham et al. 2020, Kok 2020, Svore et al. 2014) as well as through the development of a Python code with the aid of the Qiskit tools, released by IBM, in an attempt to compare the Qknn algorithm to the classical knn, which is a classification algorithm explained more in detail in Chapter 2.
The purpose of this thesis is therefore to answer the following questions: what is the current state of the art in quantum research? What are the advantages of such technology, the present fragilities and the future prospects? Is classical computing doomed to be definitely overcome by the emergent field of quantum computing, or will they both play a role in the future of computation?
And, last but not least, how does the performance of the Qknn algorithm change when fed with a pre-processed dataset rather than a non-modified dataset? What is the comparison between the Qknn and the knn? What are the implications of these differences?

1.5 Literature Review

In order to answer the questions posed in the section above, I referred to an extensive literature, prevalently on the subject of quantum computing and quantum machine learning. All the academic articles which were relevant to this thesis are listed in alphabetical order and are summarized in the following table, while some of them are explained in more detail in the next section, "Research and Research Gaps".

Title	Authors	Purpose	Method	Major Findings
"A concise review of Rydberg atom-based quantum computation and quantum simulation"	Xiaoling Wu, Xinhui Liang, Yaoqi Tian, Fan Yang, Cheng Chen, Yong-Chun Liu, Meng Khoon Tey, Li You (2021)	To acknowledge about the current state-of-the-art in Rydberg atom-based quantum computation and simulation.	Highlight of the recent experimental progresses within the field over the last years.	Achievements such as the realization of high-fidelity quantum gates, or the simulation of quantum spin models represent a starting point towards continuous successes in quantum computing with Rydberg atoms.
"A Tutorial on Quantum Convolutional Neural Networks (QCNN)"	Seunghyeok Oh, Jaeho Choi, Joongheon Kim (2020)	To propose a model to effectively solve the classification problem in quantum physics and chemistry using a quantum computing environment and to compare this model to the classical CNN.	The TensorFlow Quantum platform is used to implement the model by applying the structure of the CNN to the quantum computing environment. the MNIST dataset is used to test the QCNN and the CNN in order to compare them.	The model results in a O(log(n)) depth using Multi-scale Entanglement Renormalization Ansatz (MERA). A better performance of the QCNN is obtained over the CNN.
"A new quantum approach to binary classification"	Giuseppe Sergioli, Roberto Giuntini, Hector Freytes (2019)	The paper proposes a new quantum-based method to classify classical datasets.	The authors take inspiration from the Helstrom measurement by applying density patterns that are the	A new classifier, the Helstrom Quantum Centroid, is defined and compared with 12 linear and non linear classifiers, outperforming all of them.

			quantum encoding of classical patterns.	
"Credit Risk Analysis using Quantum Computers"	Daniel J. Egger, Ricardo García Gutiérrez, Jordi Cahué Mestre, Stefan Woerner (2019)	To present and analyze a quantum algorithm to estimate credit risk more efficiently than Monte Carlo simulations can do on classical computers.	The authors estimate the economic capital requirement, i.e. the difference between the Value at Risk and the expected value of a given loss distribution. In particular, they provide estimates of the total number of required qubits, the expected circuit depth, and how this translates into an expected runtime under reasonable assumptions.	The developed quantum algorithm estimates economic capital requirements with a quadratic speedup.
"Constructing exact representations of quantum many-body systems with deep neural networks"	Giuseppe Carleo, Yusuke Nomura, Masatoshi Imada (2018)	The development of a constructive approach to generate artificial neural networks representing the exact ground states of a large class of many-body lattice Hamiltonians.	Deterministic approach based on the deep Boltzmann machine architecture, in which two layers of hidden neurons mediate quantum correlations among physical degrees of freedom in the visible layer.	This approach is an alternative to the standard path integral, and it could also be useful to improve numerical approaches based on the restricted Boltzmann architecture.
"Deep learning and the renormalization group"	Cédric Bény (2013)	To explain the link between RG (renormalization group) and deep machine learning.	Review of MERA, a numerical method based on RG which stands for "multiscale entanglement renormalization ansatz" and its conversion into a learning algorithm based on a generative	Under the assumption that the distribution to be learned is fully characterized by local correlations, this algorithm involves only explicit evaluation of probabilities, hence doing away with sampling.

			hierarchical Bayesan network model.	
"Deep Learning in Finance"	J. B. Heaton, N.G. Polson, J. H. Witte (2016)	To apply deep learning methods to financial prediction problems (such as designing and pricing securities, constructing portfolio or risk management).	Training of a deep architecture and predicting with the maximum a posteriori (MAP) estimator.	Deep learning methods applied to financial prediction problems produces more useful results than standard methods in finance by detecting and exploiting patterns invisible to any financial economic theory.
"Machine learning explainability in finance: an application to default risk analysis"	Philippe Bracke, Anupam Datta, Carsten Jung and Shayak Sen (2019)	To address the "black box" problem present in some Machine Learning applications in order to find a solution to financial predicting questions.	The approach is based on the QII (Quantitative Input Influence) method, which is applied to a real-world example (predicting mortgage defaults).	This method estimates key drivers of mortgage defaults such as the loan-to-value ratio and current interest rate, which are in line with the findings of the economics and finance literature. The authors developed a systematic analytical framework that could be used for approaching questions in real world financial applications.
"Machine Learning for Quantitative Finance Applications: A Survey"	Francesco Rundo, Francesca Trenta, Agatino Luigi Stallo, Sebastiano Battiato (2019)	To surpass traditional approaches to financial data representation, such as autoregressive integrated moving average (ARIMA) and the exponential smoothing model, in order to deal	A review of some of the most significant works providing an exhaustive overview of recent machine learning techniques in the field of quantitative finance.	The machine learning techniques presented outperform traditional approaches in applications related to quantitative finance, such as HFT trading systems or financial portfolio allocation and optimization systems.

		with the challenge of analyzing financial data and developing innovative models to understand financial assets.		
"Machine Learning for Temporal Data in Finance: Challenges and Opportunities"	Jason Wittenbach, Brian d'Alessandro, Bayan Bruss (2020)	To account for the temporal richness of financial data in machine learning, such as bank account transactions and modern data sources like website clickstreams.	A review of the current machine learning approaches in this area, stressing both opportunities and especially challenges (such as including explicitly prohibited features regarding individuals, which is strictly forbidden by law) for researchers.	- Pre-processing is a necessary step to solving the core modelling financial tasks. - The solution to the challenges posed by the law regarding sensible data being used in machine learning programming may not be in the deep learning models themselves, but in coupling these models with auxiliary techniques for discrimination detection.
"Machine Learning in Energy Economics and Finance: A Review"	Hamed Ghoddusi, Germán G. Creamer, Nima Rafizadeh (2019)	To critically review the literature dedicated to energy economics and finance applications of machine learning.	More than 130 relevant articles published between 2005 and 2018 on the matter of interest are reviewed in terms of the methods used and the findings.	Their analysis suggests that Support Vector Machine (SVM), Artificial Neural Network (ANN), and Genetic Algorithms (GAs) are among the most popular techniques used in energy economics papers.
"Machine learning solutions to challenges in finance: An application to the pricing of	Lirong Gan, Huamao Wang, Zhaojun Yang (2020)	This paper proposes a machine-learning method to price arithmetic and geometric average options accurately and in particular quickly.	A model-free method based on deep learning is implemented to predict Asian option prices and the result is verified by empirical applications as well as	The numerical results and empirical analysis show that no matter which set of data is used to train the deep learning model, it can predict the Asian option prices with high accuracy. Compared with the three

financial products"			by numerical experiments.	traditional methods, the speed of the trained deep learning model is extremely fast.
"Multiple-Particle Interference and Quantum Error Correction"	Andrew Steane (1996)	To discuss the concept of multiple-particle interference using insights provided by the classical theory of error correcting codes.	Methods of error correction in the quantum regime are presented, and their limitations assessed.	In a quantum channel, there is an exponential reduction of decoherence with only a polynomial increase in the computing resources required. So, quantum computation can be made free of errors in the presence of physically realistic decoherence levels.
"Quantum Algorithms for Nearest-Neighbor Methods for Supervised and Unsupervised Learning"	Nathan Wiebe, Ashish Kapoor, Krysta M. Svore (2014)	To present new quantum algorithms for performing k-nearest-neighbor algorithm and k-means clustering based on the Euclidean distance measurement.	Computing the Euclidean distance to determine the nearest neighbors both directly and via the inner-product along with methods for performing amplitude estimation.	Polynomial reductions in query complexity relative to Monte Carlo algorithms and competitive classification accuracy with respect to classical knn methods.
"Quantum Computing in the NISQ era and beyond"	John Preskill (2018)	General overview of the state-of the-art of NISQ era in quantum computing.	Comparison of literature review.	- The design of noise-resilient algorithms may extend NISQ devices computational power - Improved quantum gate accuracy will lower the cost of quantum error correction when it will be eventually implemented.
"Quantum Computing with Neutral Atoms"	Loïc Henriet, Lucas Beguin, Adrien Signoles, Thierry Lahaye, Antoine Browaeys,	To review the main characteristics of atom-based systems based on previous researches.	The authors illustrate how applications ranging from optimization challenges to simulation of quantum systems can be	Interesting developments include the ability to engineer relatively simple error-correction procedures. Many of the recent advances in this field of

	Georges-Olivier Reymond, Christophe Jurczak (2020)		explored either at the digital level (programming gate-based circuits) or at the analog level (programming Hamiltonian sequences), giving evidence of the intrinsic scalability of neutral atom quantum processors in the 100-1,000 qubits range and introducing prospects for universal fault-tolerant quantum computing and applications beyond quantum computing.	research will be key for the advent of the second quantum revolution.
"Quantum Generative Adversarial Networks for learning and loading random distributions"	Christa Zoufal, Aurélien Lucchi, Stefan Woerner (2019)	To employ quantum simulation to demonstrate the use of a trained quantum channel in a quantum finance application.	Use of quantum Generative Adversarial Networks (qGANs) to facilitate efficient learning and loading of generic probability distributions, implicitly given by data samples, into quantum states. Through the interplay of a quantum channel, such as a variational quantum circuit, and a classical neural network, the gGAN can learn a representation of the probability distribution underlying the data samples and load it	The work successfully presents a hybrid quantum-classical algorithm for efficient, approximate quantum state loading.

			into a quantum state.	
"Quantum k nearest neighbors algorithm"	Afrad Basheer, A. Afham, Sandeep K. Goyal (2020)	To present a new quantum algorithm to perform the knn algorithm using the fidelity as similarity measurement.	Encoding the fidelity information between the test state and all the train states as amplitudes of a quantum state and then converting this amplitude to a digital format to enable an efficient comparison.	The QkNN algorithm can be reduced to an instance of the quantum k maxima algorithm; hence the query complexity of QkNN is $O(\sqrt{kM})$ using the SWAP test and encoding using the fidelity measure.
"Quantum Support Vector Machine for Big Data Classification"	Patrick Rebentrost, Masoud Mohseni, Seth Lloyd (2014)	Show that the support vector machine, an optimized binary classifier, can be implemented on a quantum computer.	A least-squares formulation of the support vector machine allows the use of phase estimation and the quantum matrix inversion algorithm. The speed of the quantum algorithm is maximized when the training data kernel matrix is dominated by a relatively small number of principal components	Their model obtains a complexity which is logarithmic in the size of the vectors and the number of training examples. So, an exponential speedup is obtained relative to the classical sampling algorithms requiring polynomial time.
"Solving the Optimal Trading Trajectory Problem Using a Quantum Annealer"	Gili Rosenberg, Poya Haghnegahdar, Phil Goddard, Peter Carr, Kesheng Wu, Marcos López de Prado (2016)	To solve a multi-period portfolio optimization problem using D-Wave Systems' quantum annealer.	The authors derive a formulation of the problem, discuss several possible integer encoding schemes, and present numerical examples that show high success rates.	The experiment demonstrated the potential of D-Wave's quantum annealer to achieve high success rates when solving an important and difficult multi-period portfolio optimization problem.

"Towards Quantum Machine Learning with Tensor Networks"	William Huggins, Piyush Patil, Bradley Mitchell, K. Brigitta Whaley, E. Miles Stoudenmire (2018)	To overcome the challenges derived from near-term devices.	The authors build circuits based on tree and matrix product state tensor networks in an attempt to find quantum computing approaches to both discriminative and generative learning. In particular, they train a discriminative model to perform handwriting recognition and testing the noise.	The result is a unified framework where a model can be trained in a classical way and then optimized in a quantum setting. Tensor network circuits also allow the number of physical qubits needed to scale logarithmically independently of the data size.
"Training deep quantum neural networks"	Kerstin Beer, Dmytro Bondarenko, Terry Farrelly, Tobias J. Osborne, Robert Salzmann, Daniel Scheiermann, Ramona Wolf (2020)	To introduce a new quantum machine learning algorithm based on the neural-network architecture.	The authors successfully train a deep quantum neural network by creating a quantum analogue of the backpropagation algorithm based on the discovery of the fundamental property of the QNN defined by them: information propagates from input to output and hence naturally implements a quantum feedforward neural network.	This method allows for fast optimisation with reduced memory requirements and robustness to noisy training data, revealing as particularly fit to the NISQ era.

1.6 Previous Research and Research Gaps

The main starting point is the research conducted by Daniel Kok in 2020: in his work, he was able to design a quantum knn with Python, but with some limitations.
In particular, the realized Qknn represents a hybrid, due to the fact that it is only possible to run the quantum algorithm on a classical hardware, using a quantum simulator available online. The most relevant consequence of this is that the final accuracies reached by the Qknn are either low or unstable, and the simulated execution times show that the promised reduction in complexity is currently prevented by the limitations of the hardware. However, it is unclear whether the pre-processing of one of the datasets used, namely the financial dataset, could play some role in the result achieved and, if so, what would that result be. By verifying this, it could be possible to obtain a better performance of the Qknn compared to that presented by Kok.
Afham et al. (2020) also work towards the development of a Qknn. Their research suggests that the Qknn algorithm can be reduced to an instance of the quantum k maxima algorithm, by means of Swap test and generalizations of quantum analog to digital conversion algorithms to construct an oracle, and of an encoding of the information between the test state and

all the train states, which is carried out using the fidelity as a similarity measure of the amplitudes of a quantum state. The algorithm requires $O(\sqrt{k}M)$ calls to obtain the identities of the k-nearest neighbours of the test, where M is the number of train states. As stated by the authors, "the advantage of the proposed algorithm is its ability to classify quantum states without their explicit classical description."
Svore et al. (2014) managed to obtain polynomial reductions in query complexity with their quantum knn for supervised and unsupervised learning relative to Monte Carlo algorithms and competitive classification accuracy with respect to classical knn methods. Their approach is based on using the Euclidean distance as a measurement of the distance between a vector u and the cluster centroids, in order to determine the nearest neighbours both directly and via the inner-product along with methods for performing amplitude estimation.
Other successful implementations of quantum machine learning algorithms that are carried out by researchers include: the Quantum Support Vector Machine (Rebentrost et al. 2014); the Quantum Neural Networks (Beer et al. 2020) and the Quantum Convolutional Neural Networks (Choi et al. 2020), as Chapter 3 will describe in detail.
A number of recent papers incorporate machine learning methods within the financial sector, such as: risk analysis (Bracke et al. 2019), product pricing (Gan et al. 2020) or corporate finance (Rundo et al. 2019). Among all of these papers, there is one concluding that "pre-processing is a necessary step to solving the core modelling financial tasks" (Wittenbach et al 2020). However, this may not apply to the case of quantum computing. The verification and eventual confirmation of this assertion in the case of the financial dataset applied to the Qknn developed by Kok would therefore help in answering the question whether or not the efficiency of the Qknn will increase after the pre-processing of the data fed to the system.
For what pertains quantum machine learning applied to finance, some examples are: the use of Monte Carlo simulations for risk analysis (Woerner et al. 2019), combinatorial optimization for optimal trade opportunities (Rosenberg et al. 2015) and quantum deep learning for option pricing (Zoufal et al. 2019).
Finally, some works deal with the recent developments that have been made in quantum experiments, including the recent advances in the employment of Rydberg atoms (Wu et al. 2020), while others highlight the present fragilities, first and foremost regarding quantum error correction (Preskill 2018). One of the aims of this thesis is to summarize all the findings enunciated thus far, in an attempt to draw the complete picture of the state-of-the-art in quantum computation up to now.
As a matter of fact, all these researches are relevant to gaining knowledge about the general outlook of quantum computing in the future, in terms of the potential drawbacks it may be faced with on the one hand, and of all the potentialities it is likely to bring on the other. The other aim is to compare the performance of four machine learning algorithms with their quantum implementation. The results of these comparisons are summarized in a table in Chapter 5, while the fourth, that is, knn vs. Qknn, is also analyzed in a separated section.

1.7 Method

Following the research gaps presented thus far, this is how the thesis is structured:

- The introduction, preceded by an abstract and the acknowledgements, which presents the main content of the thesis, namely the definition of the problem and its importance, the possible future applications and the original contributions of this work, as well as a panoramic of the topics that will be approached later in the thesis;

- The present chapter (Chapter 1), a preliminary introduction to the researches around

quantum computing which also describes the purpose and methods of the thesis;

- In Chapter 2, an overview to the current state of machine learning and its applications within the financial sector is made, as well as a step-by step explanation of the following classical algorithms: the SVM, the neural networks, the convolutional neural networks and the knn;

- In Chapter 3, the main theoretical concepts behind quantum computing are introduced, along with the following quantum algorithms: the Quantum SVM, the Quantum Neural Networks, the Quantum Convolutional Neural Networks and the Quantum knn.

- Chapter 4 focuses on the implementation of the Python code in order to perform the data pre-processing on the German credit dataset;

- The analysis of the results and the comments are carried out in Chapter 5;

- Future research directions based on the main gaps of this study are identified.

- Conclusions on the findings of this thesis are drawn in the final part.

First of all, the main theory regarding both machine learning and quantum computing are presented, as well as an analysis of the potentialities and fragilities of these fields. After a "panoramic" of the current state-of-the-art research, the thesis focuses on the knn algorithm and explains in detail the main steps towards designing its quantum counterpart: the Qknn. The designing part is done by implementing a Python code using the Qiskit language, developed by IBM in order to foster the production of quantum programs. Based on the assertion made by Sergioli et al. (2019), which opens to the possibility of implementing quantum code on a classical hardware, I rely on such hybrid solution for the purpose of this thesis: I will run a quantum code on my computer using a quantum simulator, hence combining quantum and classical computing.
The ultimate goal of the thesis is to improve an already implemented version of the Qknn and to use the results to compare a classical machine learning algorithm to the quantum counterpart.

1.8 Contributions

The main contributions achieved with this thesis are:

- To render the theory from quantum literature easily comprehensible by enunciating the main theories from a simple, mathematical perspective.

- To outline the current state-of-the-art in quantum research, in order to point out both the potentialities and the fragilities of this new frontier, in relation with the fields of artificial intelligence and machine learning.

- To develop the quantum version of a machine learning classification algorithm, namely the Quantum knn, performing data pre-processing on an extensive financial dataset in order to employ it in the execution part.

- To verify that the Qknn algorithm works in an acceptable way on a pre-processed financial dataset.

- To compare such algorithm to the classical counterpart and make inferences about the results.
- To compare other three relevant machine learning algorithms to the corresponding quantum implementations and make inferences about the results.
- To utilize all the information gained throughout the writing of this thesis to draw significant conclusions on the future developments of the emerging field of quantum computation.

2 Chapter 2 - Machine Learining Theory

2.1 Introduction

This chapter focuses on machine learning, its history, its main aspects and its latest applications within finance. The knn and other machine learning algorithms are then presented and explained in detail.

2.1.1 Machine Learning: A Definition

The field of machine learning has gained much attention since the last decade.
Among the many definitions that have been provided to describe it, I chose the one enunciated by Tom Mitchell (1997): "a computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."
Indeed, machine learning relies on the learning principle; that is, a computer program will perform better with experience, which is gained via learning.
Therefore, in order for this discipline to be efficient, the right mole of data must be fed to the system which is intended for training, both in terms of quality and quantity: machine learning algorithms will work much more effectively with a higher quantity of data, as this assures a higher number of examples from which the machine can learn. The other focal point is the information provided by such data, which must be significant towards the achievement of the given task. If, for example, the goal is to forecast the weather in Rome, historical data about the weather in the last months will be much more significant than historical data about traffic in the centre in the same period. This is intended as "quality data".

2.1.2 Machine Learning Algorithms

Training a machine learning model amounts to splitting the dataset on which we want to train the model into two sets, namely the training and the test set.
First of all, for a machine learning algorithm to work properly, a huge amount of data to train it is needed. These are called training data and must be as generalized as possible, which refers to the completeness and representativeness of the data. However, when the model is too complex relative to the amount and noisiness of the training data, we may incur in overfitting, which means that the model performs well on the training set, but it does not generalize well to new data. This is because an overfitted model is likely to detect patterns in the noise itself (if the training set is noisy, or if it is too small, which introduces sampling noise), so the patterns will not generalize to new instances.
Some possible solutions to this problem include applying regularization, that is, reducing the number of attributes in the training data. The amount of regularization to apply during learning can be controlled by a parameter of a learning algorithm (not of the model) which must be set prior to training and remains constant during training, called a hyperparameter. It is important to find an appropriate value for it, which can be done for example by training multiple models until the one which yields the lowest error rate on the test set is found.
The test set is defined as the data set which is used to evaluate the trained model. Indeed, once we have generalized a model to new instances we need to test how well it actually performs. Usually, models are configured in such a way that 80% of the data is used for training, while 20% for testing. The testing is carried out through the error rate, which is a value for estimating our model performance on new data. Other configurations include:

75% training, 25% testing or, 70% training and 30% testing. A popular method used on the validation set is cross validation, which consists in creating many small validation-sets and averaging all single evaluations.
Machine learning models can be trained to perform both classification and regression tasks, which fall under the category of prediction problems. The steps that these types of problems comprise are:

1. Modelling: consider several different models and different parameter settings.
2. Model selection: choose the type of model and fully specify its architecture.
3. Training: run an algorithm to find the model parameters that will make it the best fit for the training data and will make good predictions for new data.
4. Predictions: apply the selected model on new data.

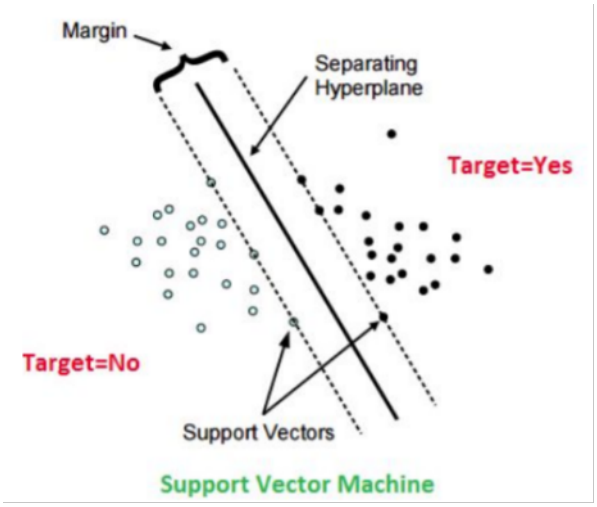
The main drawback of this approach is the risk of overfitting if the test data are similar to the training data.
In this context, classification refers to a predictive modelling problem where a class label is predicted for a given example of input data, while in a regression task the model learns to predict numeric scores, therefore the output is a number instead of a class. If classification is about separating data into classes, regression is about fitting a shape that gets as close to the data as possible.
This thesis focuses on classification problems starting from labelled data, which fall into the category of supervised machine learning problems.

2.1.3 Supervised Machine Learning

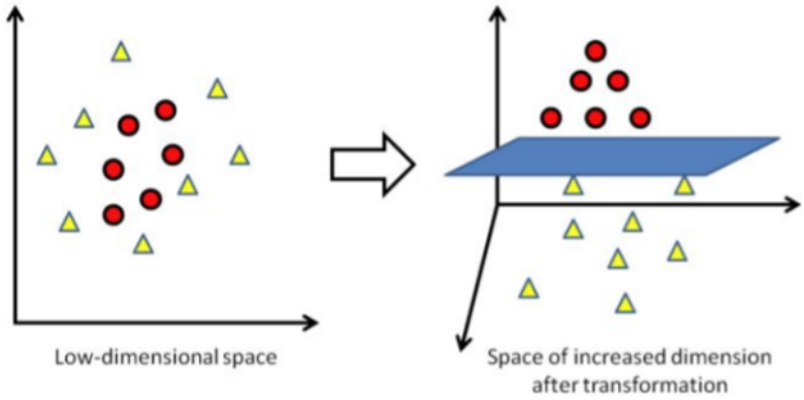
The aim of a supervised machine learning task is to map inputs (such as images) to targets (such as the label "cat"), which is done by observing many examples of inputs and targets. So, the main feature of this type of learning is the use of classified or labelled data, as opposed to unsupervised machine learning where algorithms work with data that are neither classified nor labelled. Other types of learning include reinforcement learning, on which sophisticated systems such as autonomous cleaning robots are based.
Three famous supervised machine learning algorithms used for classification tasks are shown **explained** below.

2.1.4 Support Vector Machine Algorithm

Support Vector Machine, or simply SVM, is a popular machine learning model performing linear and nonlinear problems, especially classification ones.
It works by looking for the separating hyperplane with the help of several support vectors. Such separating hyperplane is the decision boundary: a 2-dimensional line which is able to separate different classes and must be furthest from the closest training instance of both classes. To classify new data points, it suffices to check which side of the decision boundary they fall on.
The procedure comprises two steps: first, the data is mapped to a new high-dimensional representation where the decision boundary can be expressed as a hyperplane (a straight line for two-dimensional data); then, a good decision boundary (a separation hyperplane) is computed by trying to maximize the distance between the hyperplane and the closest data points from each class. This is knows as maximizing the margin.



However, mapping data to a high-dimensional representation is often intractable from a computational viewpoint. The solution, which is the key idea on which the so-called kernel methods like SVR are based, is the "kernel trick".
To find good decision hyperplanes in the new representation space, we do not have to explicitly compute the coordinates of the points in the new space; we just need to compute the distance between pairs of points in that space, which can be done efficiently using a kernel function, an operation that maps any two points in the initial space to the distance between these points in the target representation space, completely bypassing the explicit computation of the new representation. Such functions are typically crafted by hand.



SVMs became extremely popular in the field for a long time. However, a drawback is the fact that they proved hard to scale to large datasets and did not provide good results for perceptual problems (such as image classification): in fact, this would require first extracting useful representations manually (a step called feature engineering), which is not a trivial task.
The first drawback can be overcome by the quantum theoretical version of the SVM, as will be explained in Chapter 3. As for image classification, it is a complex problem which requires more sophisticated machine learning solutions with respect to SVM. An example are neural networks.

2.1.5 Neural Networks

The architecture of neural networks models pertains the field of deep learning, which is a subfield of machine learning that deals with successive layers of representation. The depth of the model is given by the number of its layers. For instance, in modern deep learning models there may be tens of successive layers (as opposed to shallow learning, with only one or two layers of representation). These are all learned automatically from training data via neural networks models.
The main idea behind the functioning of neural networks is to imitate the human brain to perform prediction tasks. Indeed, the term "neural network" comes from neurobiology, although these are not models representing the brain. Instead, these machine learning models take inspiration from the processes happening in the brain, which involve billions of neurons linked to each other and electric or chemical signals sent via the axon to each other when stimulated.
One of the simplest neural networks architectures was developed in 1957 by Frank Rosenblatt, and it is based on an artificial neuron called Threshold Logic Unit (TLU)., where inputs and outputs are numbers and each input connection is associated with a weight. In particular, TLU computes a weighted sum of its inputs:

$$z = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n = x^T w$$

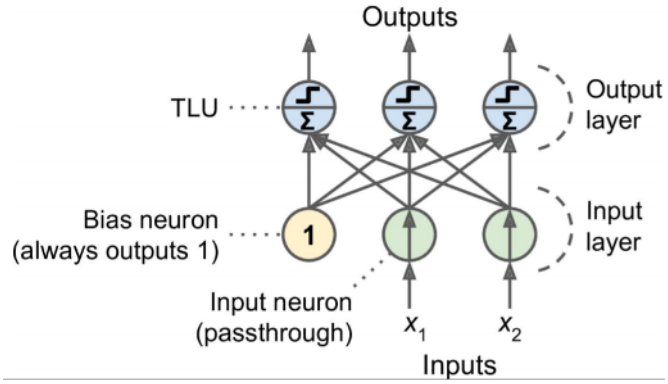
Then it applies a step function to the output of the linear combination z and outputs the result:

$$h(x) = step(z), where z = x^T w$$

The most common step functions used with perceptrons are:

$$\text{heaviside}\ (z) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases} \qquad \text{sgn}\ (z) = \begin{cases} -1 & \text{if } z < 0 \\ 0 & \text{if } z = 0 \\ +1 & \text{if } z > 0 \end{cases}$$

A single TLU can be used for simple linear binary classification: it computes a linear combination of the inputs and then, if the result exceeds a threshold, it outputs the positive class; otherwise, it outputs the negative class. Training a TLU in this case means finding the right values for the weights of the input features.
The simplest model involving perceptrons is composed of a single layer of output nodes (TLU):

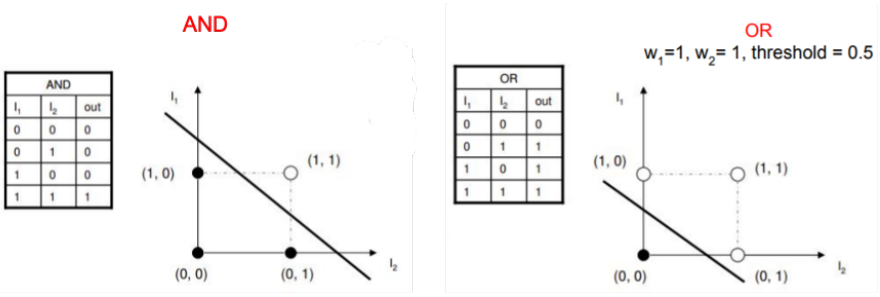


The inputs are fed directly to the outputs via a series of weights (i.e. each TLU is connected to all the weighted inputs). When all the neurons in a layer are connected to every neuron in

the previous layer (i.e. the input neurons), the layer is called a fully connected layer (dense layer), whose output is computed as follows:

$$h_{w,b}(X) = \Phi(XW + b)$$

where X represents the matrix of input features (it has one row per instance and one column per feature); W is a weight matrix containing all the connection weights (except for the ones from the bias neuron), with one row per input neuron and one column per artificial neuron in the layer; b is the bias vector containing all the connection weights between the bias neuron and the artificial neurons: it has one bias term per artificial neuron; and finally, the function Φ is called the activation function (when the artificial neurons are TLUs, it is a step function).
The Perceptron training algorithm proposed by Rosenblatt was inspired by Hebb's rule (1949): the connection weight between two neurons tends to increase when they fire at the same time. This rule was later summarized in a phrase by Siegrid Lowel: "cells that fire together, wire together". This meant that when a biological neuron triggers another neuron often, the connection between these two neurons grows stronger.
Training is made using a variant of this rule, that takes into account the error made by the network when it makes a prediction. The process is based on reinforcement.
A drawback of this simple effective model is that it cannot solve some very simple problems like the XOR (Exclusive OR) classification problem (this is true of any other classification model).
In this problem, given two binary inputs, it is required to train the network to calculate the appropriate weights and thresholds in order to classify correctly the different classes. In other words, we should create decision boundaries between classes, represented as decision surfaces (the surface at which the output of the unit is precisely equal to the threshold: in 1D it is just a point, while in 2D it is always a straight line).
The threshold may be defined with a AND relation or a OR relation, as shown in the following figures:



In the first case, only when I1 AND I2 both have a weight 1, they output the value 1 (above the threshold, ie. the straight line). In all the other cases (either I1 or I2 are 1, or they are both 0) the output is classified as 0 (below the straight line).
In the second case, In this case, only when EITHER I1 OR I2 are 1, the output is 1. Also when they're both 1. Since the threshold is at 0.5, all these cases will fall above the line.
The XOR condition adds an additional constraint to the OR one: only when EITHER I1 OR I2 are 1 will the output be 1. When they are both 1, it will be zero. This is how the XOR gate below works.

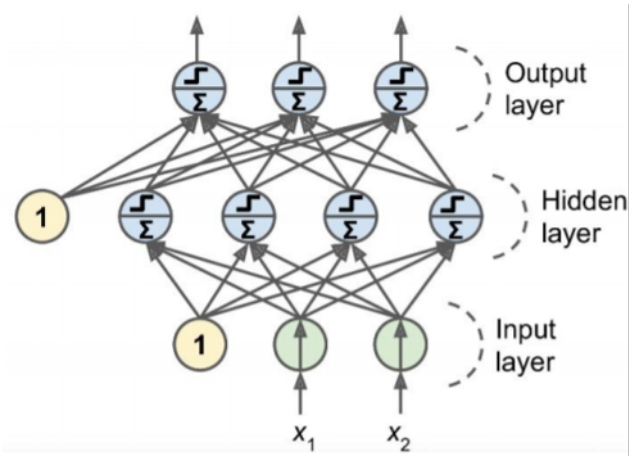
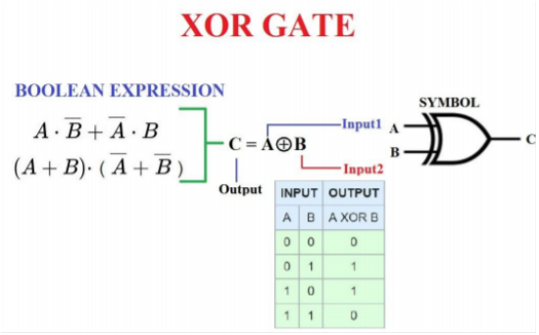
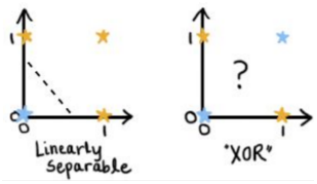


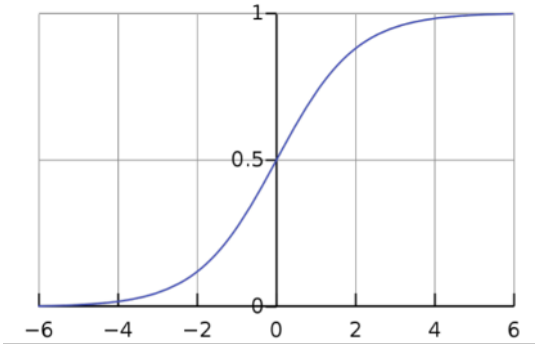
Figure 2: Multi-layer perceptron (MLP) architecture representing a feedforward neural network (FNN) with one hidden layer: the signal flows in one direction, from the inputs to the outputs.



We apply the XOR condition to the dataset below:



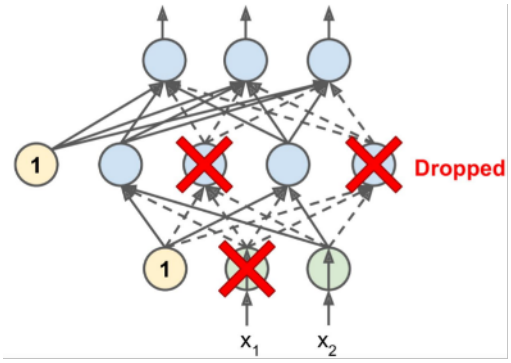
We seek to separate the yellow stars from the blue stars. We just need a single line in the figure on the right, because the data is linearly separable and we are in 2-dimension.
On the other hand, XOR is not linearly separable: we would need more than a line in order to separate the two classes (in this case, we need two lines).
This is where non-linear machine learning methods, like multi-layer perceptrons (MLP) come into play. An example is found below: With this type of structure it is possible to solve the XOR problem: with inputs $(0, 0)$ or $(1, 1)$ the network outputs 0; with inputs $(1, 0)$ and $(0, 1)$ it outputs 1.
Binary classification problems can be carried out with the MLP using the logistic activation function, which is a continuous and differentiable function used to create some nonlinearity among layers and has is S-shaped, with the output value ranging from 0 to 1:



Then, the output will be a number between 0 and 1, which we can interpret as the estimated probability of the positive class, while the estimated probability of the negative class is equal to one minus that number.
Multiclass and multilabel binary classifications are also possible. In general, a typical classification MLP architecture may be as follows:

Hyperparameter	Binary classification	Multilabel binary classification	Multiclass classification
Input neurons	One per input feature	One per input feature	One per input feature (e.g., 28 x 28 = 784 for MNIST)
Hidden layers	typically 1 to 5	typically 1 to 5	typically 1 to 5
Neurons per hidden layer	typically 10 to 100	typically 10 to 100	typically 10 to 100
Output neurons	1	1 per label	1 per class
Hidden activation	ReLU	ReLU	ReLU
Output activation	Logistic	Logistic	Softmax
Loss function	Cross entropy	Cross entropy	Cross entropy

where the ReLU and the Softmax are other two commonly used activation functions.
Finally, the testing is carried out through the dropout, a popular regularization technique for deep neural networks. It was proposed by Geoffrey Hinton in 2012 and it was proven to be highly successful: even the best constructed neural networks get plus 1-2% accuracy simply by adding dropout.
"When a large feedforward neural network is trained on a small training set, it typically performs poorly on held-out test data. This "overfitting" is greatly reduced by randomly omitting half of the feature detectors on each training case" (Hinton 2012). At each training step, every neuron except the output neurons has a probability p of being temporarily "dropped out": it will be entirely ignored during this training step, but it may be active during the next step. The hyperparameter p is called the dropout rate, and it is typically set between 10% and 50%.



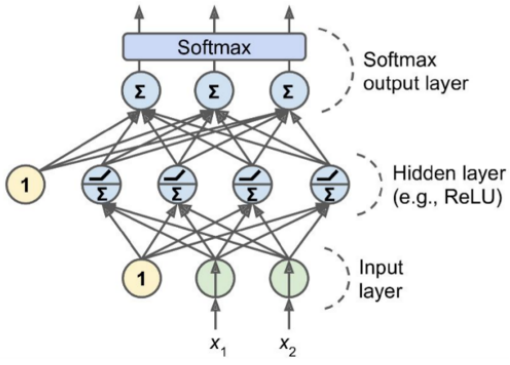
After training, neurons do not get dropped anymore.

2.1.6 Convolutional Neural Networks

An application of the neural-network architecture consists in the processing and classifications of images through the CNN (Convolutional Neural Networks). A popular example is the hand-digits classification problem: we want to classify images representing hand-written numbers from a dataset of 60000 small square 28x28 pixel grayscale images into one of ten classes representing integer values from 0 to 9 included. In particular, each instance only belongs to a single class. This is a multiclass problem.
In this case, we need to have one output neuron per class, and use the softmax activation function for the whole output layer.
This function will ensure that all the estimated probabilities are between 0 and 1 and that they add up to 1 (which is required if the classes are exclusive):

$$\hat{p}_k = \sigma(s(x))_k = \frac{exp(s_k(x))}{\sum_{j=1}^{k} exp(s_j(x))} \tag{2.1}$$

where K is the number of classes, s(x) is a vector containing the scores of each class for the instance x and $\sigma(s(x))k$ is the estimated probability that the instance x belongs to class k, given the scores of each class for that instance:



Both neural networks and convolutional neural networks are highly employed to carry complex computational tasks. An implementation of these methods using quantum solutions promises even more appealing results when it comes to very resource-expensive programs, as Chapter 3 will explain in detail.

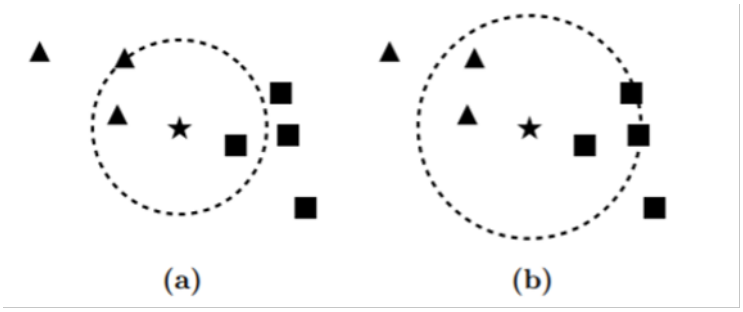


Figure 3: In the above figures, a binary classification task is shown. The two classes are the triangle and the square, and the task is to ascertain whether the star element belongs to one class or the other. Two different range values for the amount of nearest neighbours are chosen for this purpose: in the left-hand figure, it is set to 3, while on the right-hand side it is equal to 5. We immediately notice how the classification of the star changes in the two cases: it will be assigned to the triangles class in (a), and to the squares class in (b). This denotes the importance of the determination of the number k.

2.1.7 The K-Nearest Neighbours Algorithm

The k-nearest neighbours algorithm, also called knn algorithm, is a basic yet powerful supervised machine learning algorithm used for both classification and prediction tasks on labelled data. However, we will only consider the classification version of this algorithm in this thesis.
The steps the knn takes to classify a data point are the following:

1. Determination of the distance of all N points with dimension D to an unclassified data point (this is usually done by handling the space as a Euclidian space and using the Euclidian metric for distance calculation. However, it can be calculated in other ways too, as we will see in a bit).

2. Detection of the k points closest to the data point (i.e. with D lowest);

3. Identification of the class of the data point based on a majority vote.

The figure below visually displays this process. The best value which k should take on, although still an object of study and debate (Zhang et al. 2017), can be found out via learning, but this is not in the scope of this thesis. Let's focus on the determination of the distance instead.
For the purpose of this thesis, the most suitable method for calculating the distance involves the use of the cosine similarity as a distance measurement:

$$(x, y) = \frac{xy}{\|x\|\|y\|}$$

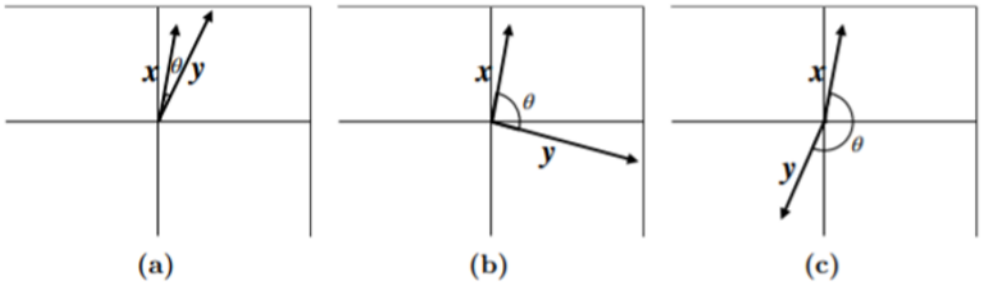
This equation will take a value between -1 and 1, and it is an indication of how much the vectors x and y overlap with each other.
By means of example: consider some two-dimensional vectors x, y. Then:

- If $(x, y) = 0$ it means that a full overlap has happened. The angle θ between the two vectors is 0 (a).

- If $(x, y) = 1$, the vectors are opposite to each other. The angle between them is 180° (b).

- If $(x, y) = 0.5$, θ forms a 90° angle. This is the case of two orthogonal vectors (c).

The graphical representation can be found below.



The distance D can therefore be determined by computing 1 – (x, y). Then we can proceed with the other two steps.
This algorithm can be easily implemented using the sci-kit learn library on Python, which is done in Chapter 4 in order to test in on the pre-processed financial data set, while in the next chapter, after an introduction to the main theoretical concepts behind quantum computing, we will see how the same k-nearest neighbours algorithm is structured in its quantum counterpart.

2.2 Applications of Machine Learning in Finance

The ever-growing interest in machine learning is mainly justified by its versatility: as a matter of fact, it can be applied to a variety of disciplines. One very interesting application is within the financial sector.
A number of recent papers discuss the potential use of applying data science and machine learning within banking: Heaton et al. (2018) applied a deep machine learning architecture to financial prediction problems such as portfolio or risk management, by means of the MAP estimator. They suggest that training a financial economic model with deep learning methods is actually more efficient than standard methods. Ghoddusi et al. (2019) reveal that there are many domains in which the application of machine learning techniques has not been fully utilized yet: for instance, trading strategies still make little or no use of these methods, which could be employed in forecasting and trading in energy markets, among other uses. On the other hand, they found that machine learning techniques are combined with optimization techniques mostly within the field of finance, from portfolio optimization to supply chain (as is the case of the aforementioned work of Heaton et al.).
Other works incorporating machine learning methods within the financial sector are: risk analysis (Bracke et al. 2019), product pricing (Gan et al. 2020) or corporate finance (Rundo et al. 2019).
Among all of these papers, there is one concluding that "pre-processing is a necessary step to solving the core modelling financial tasks" (Wittenbach et al 2020). This will in part be useful in chapter 4, where I will perform some data pre-processing on a financial dataset.

3 Chapter 3 - Quantum Computing Theory

3.1 Introduction

In this chapter we will learn about quantum mechanics, which is at the core of the functioning of theoretical quantum computers. This is done by first introducing some basic notions on classical computing, as well as the limitation of classical computers, which are two fundamental aspects to bear in mind when trying to understand the differences with respect to quantum computation. A detailed explanation of the main quantum algorithms then follows, including a comparison of the best-known quantum machine learning algorithms. The step-by-step process to obtain a quantum knn is also described in detail, and finally an overview of the current developments in quantum computing is carried out.

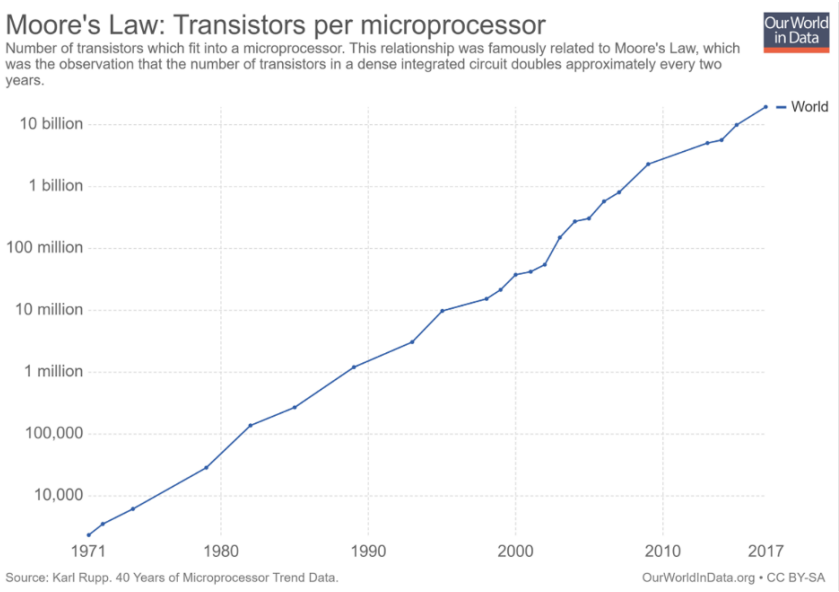
3.2 The Limitations of Current Classical Computers

Classical computers rely on both hardware and software components which work together towards the elaboration of a great mole of data. The main hardware components of a computational device include the CPU (Central Processing Unit), a control centre that converts input data to output information, and the RAM (Random Access Memory), which stores the data that the CPU will process, coming from the operating system and the software applications.
For what concerns the software, it is the set of instructions that allow the functioning of the hardware. It divides into operating systems, which manage the hardware and create the interface between the hardware and the user, and application software, the category of programs specifically thought for the users. The first computer program was an algorithm written by mathematician Ada Lovelace in 1843.
The basic components of classical computers, which link the hardware to the software are essentially just switches, known as transistors. Electric current goes through these transistors, representing the data in units called bits, and they are either turned off, denoted by 0 or turned on, denoted by 1. The precursors of these components are traced back to 1904, when John A. Fleming introduced the oscillation valves, electronic devices which are able to either allow or hinder the passage of electric current. This two states (passage/no passage) were associated to two symbols, 0 and 1 respectively, which are the basic elements in the binary system on which is based the functioning of classical computers. The valves were replaced first by transistors and then by integrated circuits (the so-called “chips”), which are made of a number of transistors (the higher their number, the better the performance of the electronic device).
Switches are used to perform logical operations on the bits: one is a single-bit operation, which negates a single bit, changing its value from 0 to 1 or vice versa. This is called the NOT-gate.
The others are two-bit operations, which means they output one bit for two input bits. One gate, called the AND-gate, takes two inputs, each having a value of either 1 or 0, and outputs the value 1 if and only if both inputs have value 1, otherwise the output value is 0. The other gate, called the OR-gate, outputs value 0 if and only if both inputs have value 0, otherwise the output value is 1. In reality, computer chips only implement one type of logic gate, a so-called universal gate, since any of these three logic gates can be broken down into sequences consisting entirely of either NAND (not-AND) gates or NOR (not-OR) gates. This is known as NAND-logic or NOR-logic respectively, and has the benefits of smaller and faster circuits.
Historically, the theorization of the first modern computer is attributed to Alan Turing,

who presented the notion of a universal machine capable of computing 'anything that is computable" in 1936. Technological advances have allowed to move from this first theoretical model to the powerful modern computers we are used to nowadays.
Despite the seemingly infinite potential of classical computers, Gordon Moore identified a big limitation, which contributed to laying down the foundations for the emergence of quantum computing theory.

3.2.1 Moore's Law

Ever since the discovery of the integrated circuits, technological development started growing exponentially. In 1965, Gordon Moore elaborated what is known as Moore's Law: he claimed that the number of electronic components inside these circuits would double from year to year, for the following 10 years. This law eventually proved right, as the graph below shows, and it is still valid today (although now the number doubles every 18 months).
'Our World in Data" provided the following graph representing Moore's law: it displays the number of transistors in the world which fit into microprocessors as a function of time, from 1971 to 2017. It is striking to see the exponential trend within the data.



In particular, the law which allows to estimate the number of transistors after x months from the release of the Pentium II (Intel microprocessor) is the following:

$$f(x) = 2^{\frac{x}{18} * 7{,}5}$$

However, this law has a limitation, as Moore himself pointed out: since the reduction of the silicon bases of the integrated circuits cannot last forever, at some point transistors will no longer be useful, since it will be necessary to reason in terms of atomic order of magnitudes. On this matter, Richard Feynman stated "nature isn't classical, dammit, and if you want to make a simulation of nature, you'd better make it quantum mechanical, and by golly it's a wonderful problem, because it doesn't look so easy". A few years later, University of Oxford physicist David Deutsch formally described a general-purpose quantum computer, a quantum analogue of the universal Turing machine (1985).

3.2.2 The Classical Bit

The word "Bit" (for "Binary digit") is used to indicate the basic unit of information, which can be represented by either of two possible and equiprobable states: 0 and 1. The transistors, which are the elementary components of an integrated circuit, can assume one of these two states, which mean: the electric current is passing through the circuit (1) or, the electric current is not passing through the circuit (0). So, information is represented by a classical calculator in a finite sequence of bits, and with N transistors, there are 2N possible states for the computer to be in.
The Alan Turning machine works obeying the principles of classical physics, in the same way as modern computational devices do.

3.3 Quantum Computing

IBM defines quantum computing as "harnessing the phenomena of quantum mechanics to deliver a huge leap forward in computation to solve complex problems that today's most powerful supercomputers cannot solve, and never will".
The first theoretical model of a quantum computer was developed in 1980 by Paul Benioff, which can be considered the quantum counterpart of the Turing machine (Benioff 1980). The turning point towards the development of such a revolutionary device was actually thanks to Erwin Schrödinger, who derived what is known today as "Schrödinger equation" back in 1925. As Richard Feynman claimed later, "it is not possible to derive it from anything you know. It came out of the mind of Schrödinger". When he published it the next year, he hypothesized it described a matter wave; however, different interpretations were given to this extraordinary equation, and the most relevant one was formulated in the same year by Max Born. Known as Born law, which is at the basis of the famous Copenhagen interpretation, it introduced the idea of a wave function representing probability amplitudes, that is, the probability to obtain a certain result when an observation is being made, as we will see later.
Other important contributors include Tommaso Toffoli, who introduced the reversible Toffoli gate (see sections "Quantum Logic Gates" and "Qiskit"); Richard Feynman, who had stressed the need for a device which does not only simulate the rules of quantum mechanics, but obeys them: a quantum simulator. In 1982 he conjectured that quantum computers can be programmed to simulate any local quantum system, which proved to be right (Lloyd 1996). Last but not least, David Deutsch, who described the first universal quantum computer in 1985, as I have already mentioned, and is also the author of two quantum algorithms which will be presented later in this chapter and which are named after him (Deutsch algorithm and Deutsch-Josza algorithm).
Now let's dive into the mathematical explanation of the theoretical functioning of quantum computers.

3.3.1 How Quantum Computers Work

Thanks to their computing power, quantum computers represent a new paradigm for processing information and making faster and more complicated calculations. They are not to be mistaken for a simple improvement of classical computers.
The fundamental difference between classical vs. quantum computers is the fact that the latter rely on the principles of quantum mechanics rather than classical physics.
The three principles which constitute the basis for the functioning of this technology are quantum superposition, quantum entanglement and quantum interference.

3.3.2 Superposition of Quantum Bits

As I previously stated, the main difference between a classical computer and a quantum computer is how they represent information. In particular, quantum computers use the qubit as the basic unit of information, rather than the bit, where qubit stands for QUantum BIT. Mathematically, a qubit can be defined as a unitary vector described in the bidimensional complex Hilbert vector space $\mathbb{C}^2$ In order to represent elements in a complex vector space, it is convenient to use Dirac's notation, which is a standard notation in quantum mechanics: If we define two vectors:

$$\begin{pmatrix}1\\0\end{pmatrix} \quad \begin{pmatrix}0\\1\end{pmatrix}$$

and we associate them to the states 0 and 1 respectively, they form an orthonormal base, that is, an orthogonal space with vectors of length 1, known as standard computational base. Moreover, we can define states using column vectors, so as to obtain:

$$|0\rangle = \begin{bmatrix}1\\0\end{bmatrix} \qquad |1\rangle = \begin{bmatrix}0\\1\end{bmatrix}$$

These two vectors correspond to the classical states 0 and 1.
A qubit can be represented by any linear combination $|\Psi\rangle$ of the basic states:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{3.1}$$

where alpha and beta are two complex numbers, such that:

$$|\alpha|^2 + |\beta|^2 = 1 \tag{3.2}$$

In order for this to happen, the vectors and must be normalized.
That's because those two squared magnitudes correspond to the probabilities for the spin of the electron to be in the basic states $\uparrow$ and $\downarrow$ when measured, and since these are the only two possible states for $|\Psi\rangle$ to be in, they must add up to 1. For example, if the probability of finding the electron in the $\uparrow$ state is 0.6 (60 percent), then the probability of finding it in the $\downarrow$ state must be 0.4 (40 percent).
Then, $|\Psi\rangle$ is a possible state of the qubit, which is algebraically expressed as:

$$|\psi\rangle = \alpha \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \tag{3.3}$$

This is equivalent to saying that $\|Psi\rangle$ is in superposition of states.
Only when we effectively measure the system, will we obtain a discrete value of the qubit, since the state will collapse in either $|0\rangle$ (with probability $|\alpha|^2$) or $|1\rangle$ (with probability $|\beta|^2$), as explained in the abovementioned example. α and β are called probability amplitudes for this reason.
Therefore, after the measurement, the system is in the measured state. This is a fundamental aspect in which the quantum bit differs from a classical bit, which only comprises two precise physical states (0 and 1).

3.3.3 Geometric Interpretation: The Bloch Sphere

A quantum bit can be represented via a Bloch Sphere, as shown below.

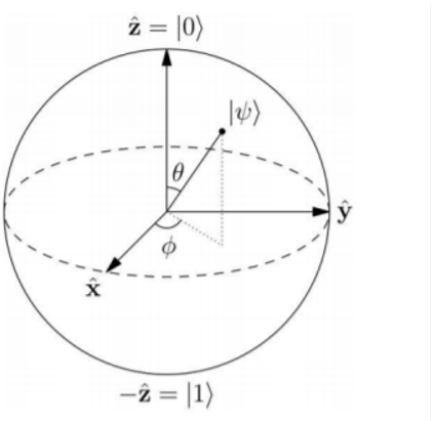


Figure 4: The Bloch Sphere is used for the spatial representation of the quantum bit. The zx-plane shows the real values of the exemplary quantum state Ψ, whereas the xy-plane show its complex values. The quantum state is added by means of example, to show that it is defined by the angles θ with respect to the z-axis, and ϕ with respect to the x-axis.

This is a three-dimensional sphere with unitary-ray, where the two poles correspond to a pair of mutually orthogonal standard-basis vectors representing the two fundamental states, 0 and 1. We can imagine that all possible states of a qubit are positioned on the surface of this sphere, where a qubit is represented as a line pointing in a direction indicated in spherical coordinates $(1, \theta, \phi)$.
This representation is made possible by the bijective correspondence that exists between the description of a generic qubit state:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

and the description of a restricted qubit state, that is, its representation on the surface of the sphere in $\mathbb{R}^3$:

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\Phi}sin\frac{\theta}{2}|1\rangle$$

where Ψ and θ represent the real numbers which identify coordinates of a point on the sphere.
In particular, the angle θ indicates the state of the qubit, whereas the angle Ψ indicates its phase.
As a matter of fact, since the amplitudes of the states of a qubit are complex numbers such that the sum of their squared norm equals one, we can use Euler's formula to represent them in polar coordinates:

$$e^{i\phi} = cos(\phi) + i\sin(\phi)$$

A qubit represented in a Bloch sphere would be in state $|0\rangle$ with$\alpha = 1$and $\beta = 0$ if the line is pointing to the north of the sphere ($+\hat{z}$, θ = 0°), and in state $|1\rangle$, with $\alpha = 0$ and $\beta = 1$, if the line is pointing to the south of the sphere ($-\hat{z}$, θ = 180°).

3.3.4 Quantum Entanglement

Consider two qubits. A generic system of two qubits can be written as:

$$|\Psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$$

This state is called "entangled" since it is not possible to express it as a direct product of two states of the two subsets of the Hilbert space.
Take the superposition state of two qubits as an example:

$$|\Psi\rangle = \frac{1}{\sqrt{2}}|00\rangle - |11\rangle$$

Then, we can never construct this state as a product of two superposition states, but only as a superposition of two product states, no matter the choice of α and β. We call such a state an entangled state (Vestergaard 2020).
A particular entangled state is the following:

$$|\Psi\rangle = \frac{|00\rangle - |11\rangle}{\sqrt{2}}$$

This is a singlet, an eigenvector of the total spin S of two particles.
This particular property of quantum physics implies a strict correlation between the state of the two particles taken into consideration, and it is at the core of the quantum computing advantage in terms of exponential speed.
As a matter of fact, a system of multiple entangled qubits (namely a "quantum register") allows to have a huge computational power. In particular, the information contained in a system with N qubits is 2N, which is more than the estimated number of atoms in the universe.

3.3.5 Quantum Registers

Quantum registers are used to build quantum circuits performing certain tasks; in order to do so, quantum registers are initialized and manipulated into different states using quantum gates, as we will see in the next section. These registers are mathematically represented via the tensor product, e.g. for two qubits forming the quantum register we have:

$$|q_1 q_2\rangle = |q_1\rangle \otimes |q_2\rangle = \begin{bmatrix} \alpha_{q_1} \\ \beta_{q_1} \end{bmatrix} \otimes \begin{bmatrix} \alpha_{q_2} \\ \beta_{q_2} \end{bmatrix} = \begin{bmatrix} \alpha_{q_1} \otimes \begin{bmatrix} \alpha_{q_2} \\ \beta_{q_2} \end{bmatrix} \\ \beta_{q_1} \otimes \begin{bmatrix} \alpha_{q_2} \\ \beta_{q_2} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \alpha_{q_1}\alpha_{q_2} \\ \alpha_{q_1}\beta_{q_2} \\ \beta_{q_1}\alpha_{q_2} \\ \beta_{q_1}\beta_{q_2} \end{bmatrix}$$

where α and β denote all the various amplitudes of the two qubits. The matrices used to manipulate qubits are called quantum (logic) gates.

3.3.6 Quantum Logic Gates

Quantum logic gates are unitary quantum operations performed by quantum computers on qubits, ideally preserving the quantum nature of the quantum bit since they do not collapse the superposition states or break entanglement, unless needed. They can be considered the quantum counterparts of the classical logic gates, since they draw inspiration from the NAND- and NOR-logics of classical computers which are considered universal logic gates as all three basic logic gates can be obtained by combining them. A similar set of universal quantum gates can be chosen for quantum computers, since any unitary quantum operation can be decomposed into a series of two-level unitary operations, i.e. operations which only act on two of the multi-qubit states (Vestergaard 2020). Such set includes the Hadamard gate, the C-NOT gate and the $\frac{\pi}{8}$ gates. This is sometimes known as C-NOT logic.
The Hadamard gate creates superposition states from single qubits states and vice versa:

$$a_0|0\rangle + a_1|1\rangle \xrightarrow{\text{Hadamard}} \sqrt{\frac{1}{2}}\Big((a_0 + a_1)|0\rangle + (a_0 - a_1)|1\rangle\Big)$$

The $\frac{\pi}{8}$ gate gives a complex phase to the $|1\rangle$ part of the state:

$$a_0|0\rangle + a_1|1\rangle \xrightarrow{\pi/8} a_0|0\rangle + e^{i\pi/4}a_1|1\rangle$$

For reasons regarding fault-tolerant implementations, it is sometimes useful to apply this gate twice.
Finally, we have the C-NOT gate, which takes two qubits and changes the state of one depending on the state of the other:

$$a_{00}|00\rangle + a_{01}|01\rangle + a_{10}|10\rangle + a_{11}|11\rangle \xrightarrow{\text{C-NOT}} a_{00}|00\rangle + a_{01}|01\rangle + a_{11}|10\rangle + a_{10}|11\rangle$$

The C-NOT gate can be used to construct the Toffoli gate (also known as CCNOT gate).
The gates which are more relevant for this thesis are the Hadamard gate, the X-gate (or NOT-gate) which flips the state of the qubit from $|0\rangle$ or $|1\rangle$ to respectively $|1\rangle$ or $|0\rangle$, and the SWAP-gate, which "swap" the state of the two qubits involved in the operation, as the name suggests.
The matrix representation for these three operators is the following:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad SWAP = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

As unitary transformations, any single-qubit gate corresponds to a rotation of the Bloch vector onto the Bloch sphere. For example, the Hadamard gate corresponds to a rotation of π around the (x + z) axis.
Applying these gates to the qubits allows to exploit all the potentials of these powerful mechanisms. One of their main features is that, between initialising the qubits and measuring them, the operations (gates) are always reversible. Graphically, these reversible gates can be represented as rotations around the Bloch sphere. For example, the X gate is a rotation by a π angle around the x-axis, while the Hadamard gate is a rotation of π around the (x + z) axis.

3.3.7 Quantum Circuits

Quantum gates that act on systems of n qubits can be thought of as (2n×2n) unitary matrices. However, when working with large systems of qubits, the matrix representation of a quantum gate can be challenging to work with due to an exponentially large number of rows and columns. Because of this, quantum gates are usually represented as diagrams made up of quantum qubits, called quantum circuits. The result of a quantum circuit is then a probabilistic state in itself and can only be determined by running the circuit multiple times. This kind of quantum computer is also referred to as a gated quantum computer since the qubits can be manipulated into certain states with the help of quantum logic gates.
These are the types of computers built by IBM and Google, which were discussed in Chapter 1.
Nowadays, there is also another type of quantum computer, called the quantum annealer,

which uses a large system of qubits brought to a superposition to solve one problem using the annealing method (Boixo et al. 2014).

3.3.8 Quantum Inerference

"Quantum interference phenomena are a key property that enables us to discern classical physics from the quantum realm" (Cimini et al. 2019).
Quantum interference is the last but not least of the properties of quantum computing; in particular, it is a by-product of the property of superposition, and it states that not only can elementary particles be in more than one place at any given time (through superposition), but that an individual particle, such as a photon (light particles) can cross its own trajectory and interfere with the direction of its own path. This concept serves to realize the fundamental idea in quantum computing of controlling the probability a system of qubits collapses into particular measurement states. In other words, this is what allows us to bias the measurement of a qubit toward a desired state or set of states.
However, quantum interference may be disrupted by an incidental measurement of a system qubit. This phenomenon is called quantum decoherence and can be a major source of error when working with physical quantum computers. As we will see later, this problem is determining the current and also future developments in the field of quantum computing.

3.3.9 Quantum Algorithms

A quantum algorithm is a series of manipulations on qubits using quantum gates.
In continuity with the first chapter, which introduced two well-known quantum algorithms, namely Shor's and Grover's, this section explains them more in detail from a mathematical viewpoint, along with other two famous quantum algorithms: Deutsch and Deutsch-Jousza, in an attempt to better understand the quantum advantage which is at the core of these methods and also to show the quantum operations described thus far in a more practical way.

3.3.10 Deutsch Algorithm

The problem solved by the Deutsch algorithm consists in determining whether the function $f : \{0, 1\} \rightarrow \{0, 1\}$ is a constant or balanced function. The approach of this algorithm to solve the problem is based on the property of interference among states, and on the concept of quantum parallelism. Let's consider a superposition of states of two qubits as initial state, such that:

$$|\psi_0\rangle = |0\rangle \otimes |1\rangle$$

An Hadamard gate is applied to both states, so as to transform the initial state into the following state:

$$|\psi_1\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

Another transformation U_f is applied to the state we just obtained, such that the following equivalence holds:

$$U_f|\psi_1\rangle = U_f \left[|x\rangle \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right]$$

Solving for U_f we get:

$$\frac{1}{\sqrt{2}}[|x, 0 \oplus f(x)\rangle - |x, 1 \oplus f(x)\rangle]$$

$$\begin{cases} |x\rangle \dfrac{|0\rangle - |1\rangle}{\sqrt{2}} & se\ f(x) = 0 \\ -|x\rangle \dfrac{|0\rangle - |1\rangle}{\sqrt{2}} & se\ f(x) = 1 \end{cases}$$

which can be rewritten as:

$$|x\rangle \frac{|0\rangle - |1\rangle}{\sqrt{2}} \xrightarrow{U_f} (-1)^{f(x)} |x\rangle \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

Now we consider two different scenarios based on the distinct values that the functions $f(0)$ and $f(1)$ can assume, and by applying the Hadamard gate on the first qubit we finally get:

$$|\psi_3\rangle = \pm \begin{cases} |0\rangle \otimes \dfrac{|0\rangle - |1\rangle}{\sqrt{2}} & se\ f(0) = f(1) \\ |1\rangle \otimes \dfrac{|0\rangle - |1\rangle}{\sqrt{2}} & se\ f(0) \neq f(1) \end{cases}$$

$$\begin{cases} f(0) \oplus f(1) = 0 & se\ f(0) = f(1) \\ f(0) \oplus f(1) = 1 & se\ f(0) \neq f(1) \end{cases}$$

$$|\psi_3\rangle = \pm |f(0) \oplus f(1)\rangle \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

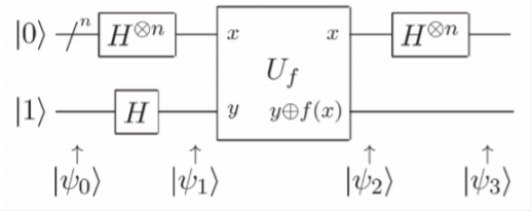
By measuring the first qubit we obtain a result which on the classical counterpart would have required at least two valuations, hence this solution is faster.

3.3.11 Deutsch-josza Algorithm

The algorithm we just analyzed may be extended to boolean functions on n bits of the form:

$$f : \{0, 1\}^n \rightarrow \{0, 1\}$$

Suppose we know that the function f can be constant or balanced with value 1 in half of the inputs and 0 in the other half. A classical algorithm would employ $2^{n-1} + 1$ steps in the worst case to classify the function. With this algorithm this can be done in one step. The circuit representation of the algorithm is the following:



which indicates an input x characterized by n qubits in the initial state $|0\rangle$ which will form the quantum register, while a qubit in the state $|1\rangle$ will be used as a target to contain the

result of f(x):

$$|\psi\rangle_0 = |0\rangle^{\otimes n}|1\rangle$$

A Walsh-Hadamard gate is applied to this state, which allows to realize an equiprobable superposition of the 2^n states. The states of the system at every step will be:

$$|\psi_1\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

$$|\psi_2\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{f(x)}|x\rangle \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

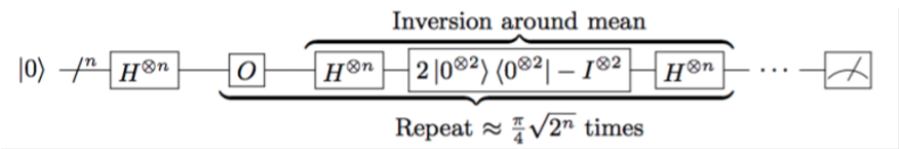
where $\{0,1\}^n$indicates the bit string x, y. After re-applying n Hadamard gates we get:

$$|\psi_3\rangle = \frac{1}{2^n} \sum_{y\in\{0,1\}^n} \sum_{x\in\{0,1\}^n} (-1)^{x\cdot y + f(x)}|y\rangle \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

Since the amplitude of the state $|0\rangle^{\otimes n}$ is given by $\frac{1}{2^n}\sum_x (-1)^{f(x)}$, it will be equal to +1 if the function is constant and equal to 0, while it will be -1 if the function is constant and equal to 1.
As opposed to the classical case, where the solution to this problem is deterministic and exponentially depends on the number of bits, here we obtain a speed-up thanks to one single and effective execution.

3.3.12 Grover's Algorithm

Grover's algorithm is employed in the resolution of searching problems, as we anticipated in the first chapter. In particular, it is involved in searching an element x in a set of possible solutions such that a given function or condition $P(x)$ holds. This method is considered optimal for unstructured search problems, and it makes use of an oracle which has to determine whether a given sequence of n bits is a solution or not. Mathematically, such oracle can be represented as a unitary transformation implementing a Boolean function defined as $f : \{0,1\}^n \rightarrow \{0,1\}$, where $f(x) = 1$ means that x is a solution, while $f(x) = 0$ means it is not.
Grover's algorithm can be represented by the following circuit:



where the steps consisting in the inversion around the mean and the entire passage to reiterate to augment the value of the solution are explicitly highlighted.
We can re-define the problem using Dirac notation:

$$O : |x\rangle|y\rangle \rightarrow |x\rangle|y \oplus f(x)\rangle$$

where x belongs to the set $\{0,1\}^n$ while represents a single qubit which, if initialized to 0, becomes 1 when $f(x) = 1$. If, instead, $|t\rangle$ is posed in a superposition of states, then the

oracle O will invert the amplitudes of the states $|y\rangle$ which are a solution, while those which are not are left unchanged. Mathematically, this can be written as:

$$O : |x\rangle \frac{|0\rangle - |1\rangle}{\sqrt{2}} \rightarrow (-1)^{f(x)|x\rangle} \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

Here, qubit $|y\rangle$ is never modified by the oracle, therefore we can equivalently write:

$$O: \sum_{x=0}^{n-1} \alpha_x |x\rangle \longmapsto \sum_{x=0}^{n-1} (-1)^{f(x)} \alpha_x |x\rangle$$

By initializing the algorithm, we want to prepare the input in the following state:

$$|\psi\rangle = |0\rangle^{\otimes n} = |00\cdots0\rangle$$

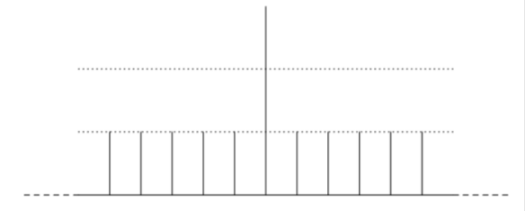
So, a Walsh-Hadamard transformation is applied to get an equiprobable superposition of states:

$$|\psi\rangle H^{\otimes n} = \frac{1}{\sqrt{N}} \sum_{x=0}^{n-1} |x\rangle$$

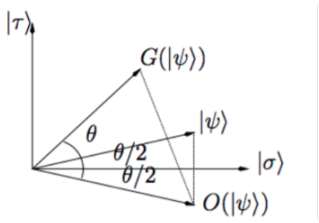
At this point, we iteratively apply the Grover operator G, defined as:

$$G = OH^{\otimes n} P_0 H^{\otimes n}$$

The operation $H^{\otimes n} P_0 H^{\otimes n}$ is called "inversion around the mean" because its effect is to amplify the amplitudes of the states of the solutions which were previously inverted by the oracle. It can be visualized as follows:



The geometric interpretation of the iteration of the G operator is shown below:



We repeat the application of G, that is the oracle and the inversion around the mean for i_max times, where:

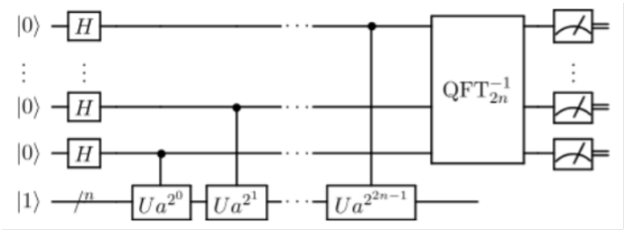
$$i_max = |\sqrt{N}|$$

By doing so, we augment the amplitude of the solution state, and consolidate the probability of obtaining it through measurement.

The complexity of Grover's algorithm was analyzed in the first chapter, but it may be useful to recall it: in fact, it is based on the chosen number of iterations of the G operator, providing a quadratic speedup compared to a classical research algorithm.

3.3.13 Shor's Algorithm

As we have seen in the introduction, the most important application of Shor's algorithm is related to the possibility to solve the factorization problem. We will now analyze how this works in more detail.
The circuit representing Shor's algorithm is:



As the image shows, the algorithm starts from an initial quantum state, defined as:

$$|00\cdots0\rangle|00\cdots01\rangle$$

which can be properly modified by substituting the value of the qubit in the state $|1\rangle$ in order to apply the Hadamard gates to get the system to the following state:

$$\frac{1}{\sqrt{2^t}}\sum_j |j\rangle \frac{1}{r}\sum_{s=0}^{r-1}\sum_{k=0}^{r-1} e^{-\frac{2\pi isk}{r}} |x^k (mod\ N)\rangle$$

We can now apply the inverse of the Fourier transform, a mathematical method used to get from a time domain to a frequency domain, in order to transform the content between parentheses in $s\tilde{/}r$. Using the methods of continuous fractions, which is a way of representing real numbers by taking the integer part of a number and the reciprocal of its fractional part, we can finally find the value of r:

$$\frac{1}{r}\sum_{s=0}^{r-1} |\tilde{s/r}\rangle e^{-\frac{2\pi isk}{r}} |x^k (mod\ N)\rangle$$

At this point, the two possible states are the following:

$$\tilde{s/r} = 0.010110 = \frac{32*0+16*1+8*0+4*1+2*1+1*0}{64} = \frac{22}{64}$$

$$\frac{22}{64} = \frac{1}{2+\frac{20}{22}} = \frac{1}{2+\frac{1}{1+\frac{1}{10}}} \cong \frac{1}{3} \Rightarrow r = 3;$$

$$\widetilde{s/r} = 0.010011 = \frac{32*0+16*1+8*0+4*0+2*1+1*1}{64} = \frac{19}{64}$$

$$\frac{19}{64} = \frac{1}{3+\frac{7}{19}} = \frac{1}{3+\frac{1}{2+\frac{5}{7}}} = \frac{1}{3+\frac{1}{2+\frac{1}{1+\frac{2}{5}}}} = \frac{1}{3+\frac{1}{2+\frac{1}{1+\frac{1}{2+\frac{1}{2}}}}}$$

$$\cong \frac{8}{27} \Rightarrow r = 27$$

The last passage requires to check whether the following verification holds:

$$x^r = 1(modN)$$

In case it does not, the algorithm will need to be run again from the first step.
All the steps are summarized as follows:

1. If N is even, return the factor 2 and repeat;
2. Choose a casual number x, such that $1 < x < N$;
3. Calculate $f_{nb} = MCD(x, N)$ by means of the Euclidean algorithm. If $f_{nb} > 1$, then f_{nb} is a non-trivial factor of N, otherwise proceed with step 4;
4. Apply Shor's algorithm to find r of x module N such that r verifies $x^r = 1(modN)$;
5. If r is odd or $x^{\frac{r}{2}} = N–1(modN)$, go back to step 1;
6. If r is even and $x^{\frac{r}{2}} \neq N–1(modN)$, calculate MCD and MCD with the Euclidean algorithm; if either of the calculated integers results in a non-trivial factor N, verify:

$$x^r - 1 = (x^{\frac{r}{2}} - 1)(x^{\frac{r}{2}} + 1) = N = 0(modN)$$

   If this is verified, the algorithm successfully terminates, otherwise start again from step 1.

Shor's algorithm is a perfect example of a hybrid solution between using a classical computer and a quantum computer. However, this is also the reason why this algorithm cannot be implemented as of today. With the aid of technological advances in the research, it could nonetheless be possible that in the future the concept of information security as we know it today will be completely revolutionized by it.
This section was useful to understand the theoretical advantage of quantum algorithms compared to classical ones, and also to get acquainted with how quantum algorithms work. A table summarizing the main findings of this comparison can be found in chapter 5.

3.4 Quantum Machine Learning

As we have seen at the beginning of Chapter 2, machine learning programs will work more efficiently when fed with a good mole of data, both in terms of quality and quantity.
Quantum machine learning algorithms are similar to the extent that they need an even higher amount of data in order to work better.
The same machine learning algorithms explained in the second chapter are presented below in their quantum counterpart.

3.4.1 Quantum Support Vector Machine

We left the discussion regarding the classical SVM algorithm by stating that it proved hard to scale to large datasets. In particular, calculating the kernel may become extremely expensive in terms of computational resources, as the number of instances inside the data set grows. Therefore, given the usefulness of the SVM to perform a variety of tasks on smaller data sets, it is particularly important to find an efficient solution to be able to apply it on larger data sets, based on the evaluation of the scalar product. This becomes possible thanks to quantum computing.
Rebentrost, Mohseni and Lloyd (2014) demonstrated that, given the quantum state:

$$|\psi\rangle = \frac{1}{\sqrt{N_\psi}} \sum_{i=1}^{N} |\vec{x}_i|\,|i\rangle|x^i\rangle; \quad N_\psi = \sum_{i=1}^{N} |\vec{x}_i|^2$$

it is possible to perform such evaluation faster on a quantum computer, where the initial state can be built using a QRAM (Quantum Random Access Memory) and $|v^c\rangle$form a 2n-dimensional computational base in the training vectors space, so that each training vector can be represented via a superposition of the form:

$$|v^c\rangle = \sum a_i |x^i\rangle$$

Then, bearing in mind that the quantum states are normalized as follows:

$$\langle x^i|x^j\rangle = \frac{\vec{x_i}\vec{x_j}}{|\vec{x_i}|\,|\vec{x_j}|}$$

it is possible to use this to evaluate the scalar product necessary for the classification.
The kernel matrix K of the scalar products can indeed be calculated by computing a partial trace of the corresponding matrix with density $|\psi\rangle\langle\psi|$ on the states $|x^i\rangle$:

$$tr_x[|\psi\rangle\langle\psi|] = \frac{1}{\sqrt{N_\psi}} \sum_{i,j=1}^{N} \langle x^i|x^j\rangle |\vec{x}_i|\,|\vec{x}_j|\,|i\rangle\langle j| = \frac{K}{tr[K]}$$

The evaluation of the scalar product can now be used for both for the construction of the kernel matrix, and for the classification of patterns, according to Rebentrost, Mohseni and Lloyd.
This evaluation method, which must be calculated during the optimization phase of the loss function, is effective in solving the problem of the classical SVM of scaling to larger datasets, hence constituting an important theoretical advantage of quantum computing over classical computing.
In addition, the authors suggest that their implementation of the quantum SVM also provides advantages in terms of data privacy: “The quantum algorithm never requires the explicit O(MN) representation of all the features of each of the training examples, but it generates the necessary data structure, the kernel matrix of inner products, in quantum parallel. Once the kernel matrix is generated, the individual features of the training data are fully hidden from the user” (Rebentrost et al. 2014).

3.4.2 Quantum Neural Networks

The growing interest in quantum computing applied to the Neural Network architecture allowed to make remarkable progresses in this field.
Beer et al. (2020) recently proposed a fully quantum analogue of classical neurons, which constitute quantum feedforward neural networks capable of universal quantum computation. In particular, they use the fidelity as a cost function, providing both classical and efficient quantum implementations, where the fidelity is a number between 0 and 1 that measures the closeness between the state created by a quantum operation and the theoretically expected state. Their method allows for fast optimisation with reduced memory requirements: the number of qubits needed scales with only the width, allowing deep-network optimisation.
In building their network architecture, they employ the quantum perceptron. Then, they propose that a QNN is a quantum circuit of quantum perceptrons organised into L hidden layers of qubits, acting on an initial state ρ^{in} of the input qubits, and producing a mixed state ρ^{out} for the output qubits, according to:

$$\rho^{\text{out}} \equiv \text{tr}_{\text{in,hid}}\Big(\mathcal{U}(\rho^{\text{in}} \otimes |0\cdots 0\rangle_{\text{hid,out}}\langle 0\cdots 0|)\mathcal{U}^{\dagger}\Big)$$

where $\mathcal{U} \equiv U^{out}U^{L}U^{L-1}\cdots U^{1}$ is the QNN quantum circuit.
The most crucial property defined by the authors is that the network output may be expressed as the composition of a sequence of completely positive layer-to-layer transition maps. This characterisation of the output of a QNN highlights a key structural characteristic: information propagates from input to output and hence naturally implements a quantum feedforward neural network. This key result is the fundamental basis for the quantum analogue of the backpropagation algorithm, which comprises the following steps (from Beer et al. 2020):

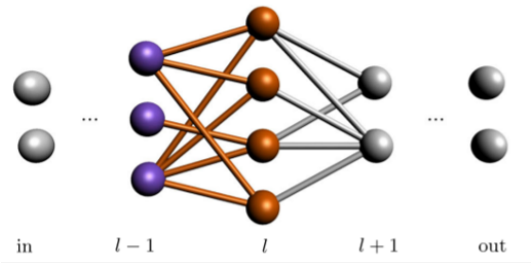
1. Initialization

   Choose the initial U_j^l randomly for all j and l, where are the layer unitaries, comprised of a product of quantum perceptrons acting on the qubits in layers $l-1$ and l.

2. Feedforward step

   For every training pair $\left(|\phi_x^{in}\rangle, |\phi_x^{out}\rangle\right)$ and every layer l, perform the following steps:

   2a. Apply the channel ε^l to the output state of layer 1-1: Tensor ρ_x^{l-1} with layer 1 in state $|0\cdots 0\rangle_l$ and apply $U^l = {U^l}_{m_l}\cdot U_1^l$:



   2b. Trace out layer l–1 and store ρ_x^{l-1}

3. Network update

Calculate the parameter matrices given by

$$K_j^l = \eta \frac{2^{m_l - 1}}{N} \sum_{x=1}^{N} \mathrm{tr}_{\mathrm{rest}} M_j^l$$

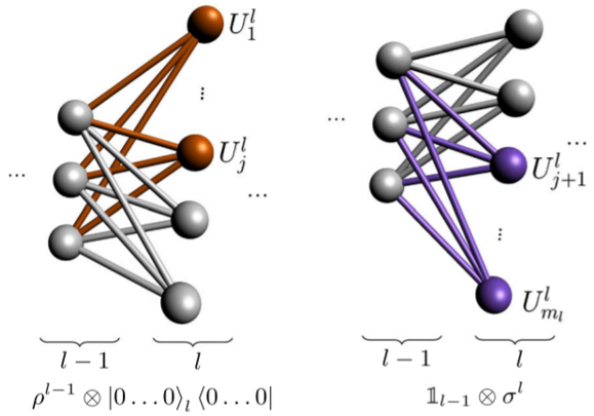
where the trace is over all qubits that are not affected by U_j^l, η is the learning rate,

$$M_j^l = \Big[\prod_{\alpha=j}^{1} U_\alpha^l \left(\rho_x^{l-1,l}\right) \prod_{\alpha=1}^{j} U_\alpha^{l\,\dagger}, \prod_{\alpha=j+1}^{m_l} U_\alpha^{l\,\dagger} \left(\mathbb{I}_{l-1} \otimes \sigma_x^l\right) \prod_{\alpha=m_l}^{j+1} U_\alpha^l \Big]$$

$$\rho_x^{l-1,l} = \rho_x^{l-1} \otimes |0 \ldots 0\rangle_l \langle 0 \ldots 0|$$

$$\sigma_x^l = \mathcal{F}^{l+1} \left(\ldots \mathcal{F}^{\mathrm{out}} \left(|\phi_x^{\mathrm{out}}\rangle \langle \phi_x^{\mathrm{out}}| \right) \ldots \right)$$

and $\mathcal{F}^l$ is the adjoint channel to ε^l, i.e. the transition channel from layer $l+1$ to layer l. Below, the two parts of the commutator are depicted:



3a. Update each unitary according to $U_j^l \to e^{i \in K_j^l} U_j^l$

4. Repeat steps 2 and 3 until the cost function reaches its maximum.

Finally, the authors conclude that the QNN training algorithm they presented is promising with respect to the NISQ (Noise Intermediate-Scale Quantum) era. However, they state that 'a crucial problem that has to be taken into account with regard to NISQ devices is the inevitable noise within the device itself." (Beer et al. 2020). Although they obtained the remarkable result of QNN robustness with respect to approximate depolarising noise, advances in the field of quantum error correction are required to completely overcome the problem.
Among the other questions remaining in the study of QNNs the authors include: generalising the quantum perceptron definition further to cover general CP maps (thus incorporating a better model for decoherence processes), studying the effects of overfitting, and optimised implementation on the next generation of NISQ devices.

3.4.3 Quantum Convolutional Naural Networks

A study by Choi et al. (2020) describes an approach to apply the CNN structure to a quantum system to efficiently solve quantum physics problems.

In particular, the QCNN presented in this research was proposed by Cong et al. (2019), and it extends the main features and structures of the CNN to quantum systems. This is done because "in moving the quantum physics problem defined in the many-body Hilbert space to the classical computing environment, the data size exponentially increases according to the system size, so it is not suitable to solve efficiently" (Choi et al. 2020). By applying a CNN structure to a quantum computer, this problem is avoided.
The structural design of the QCNN by Cong et al. is based on the MERA model (Multi-Scale Entanglement Renormalization Ansatz) and comprises 4 steps:

1. The convolution circuit finds the hidden state by applying multiple qubit gates between adjacent qubits.

2. The pooling circuit reduces the size of the quantum system by observing the fraction of qubits or applying 2-qubit gates.

3. Repeat the convolution circuit and pooling circuit defined in 1)-2).

4. When the size of the system is sufficiently small, the fully connected circuit predicts the classification result.

Quantum noise is also addressed, by proposing an improvement to the MERA model (which is an efficient method that reduces the size of the quantum system exponentially from the given data): the quantum error correction is applied to give additional degrees of freedom to the model.
The performance of the resulting circuit is tested on the MNIST dataset using the TensorFlow Quantum platform: the QCNN makes use of only $O(\log(N))$ variational parameters for input sizes of N qubits. This allows for efficient training and implementation on realistic, near-term quantum devices.
However, there are some limitations to this simulation: first of all, the 28x28 MNIST dataset was downscaled to 10x10 size; the filter size of the quantum convolutional layer was limited to 2x2 and, finally, in each epoch 2500 random images out of 60000 are selected for learning. However, the authors explicitly state that they intend to apply this QCNN model to more complex data (Choi et al. 2020).

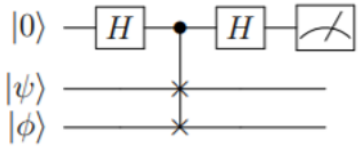
3.5 Quantum Knn

The knn is translated step by step to a quantum algorithm, namely the Qknn, based on the works by Afham et al. (2020) and Kok (2020), to perform classification on the German credit dataset. For the purpose of this thesis, I referred to the code provided by Kok, including tutorials and documentation, on PyPI, ReadTheDocs, and GitHub. The Qknn is developed within Qiskit.
As Kok states in his research, "the results show that the Qknn performs similar, although often a bit worse, than the knn. There is also a constant variability in the accuracies of the Qknn for the finance and physics data. However, the variability in accuracy for the Iris data reduces as the number of shots of the qasm_simulator increases." Among all three datasets he used in his research, the financial and the physics ones were not pre-processed before the trials, which may be why the quantum knn performed better on the Iris dataset compared to the other two.
In order to overcome this problem, some data pre-processing will be performed on the financial dataset in the next chapter before implementing the Qknn. But in order to do that, first we need to dive into the construction of the quantum knn algorithm to better understand its functioning.

Distance determination is the first step, and it is achieved using the fidelity measure between two quantum states, which represents the overlap between these two states. In particular, the fidelity of two states ϕ and ψ is given by:

$$F = \|\langle\psi|\phi\rangle\|^2$$

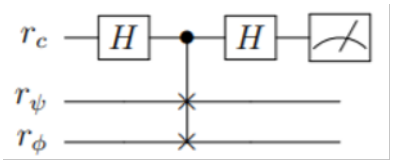
where the qubits $|\psi\rangle$ and $|\phi\rangle$ represent the states of which the fidelity will be measured and $\langle\psi|\phi\rangle$ represents the inner product between two vectors and is directly relatable to the cosine similarity.
By following the procedure from Buhram et al. (2001), this equation can be translated into the following quantum circuit:



This method is called SWAPTest-method (Fredkin and Toffoli 1982), since it works by the controlled SWAP gate (also known as cSWAP gate or Fredkin gate).
The probabilities of measuring the qubit in either of the two states and are determined by Afham et al. (2020) as:

$$\mathbb{P}(0) = \frac{1}{2} + \frac{1}{2}\|\langle\psi|\phi\rangle\|^2,\ \mathbb{P}(1) = \frac{1}{2} - \frac{1}{2}\|\langle\psi|\phi\rangle\|^2$$

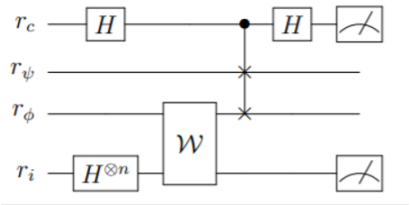
for which the circuit must be executed multiple times. These probabilities contain the fidelity values that can be retrieved by calculating $\mathbb{P}(0) - \mathbb{P}(1)$.
In the case of multiple qubits, we will describe the states as quantum registers r_ψ, $r_\phi$and r_c (the control qubit $|0\rangle$ consisting of only one qubit):



However, it is necessary to modify this circuit according to Afham et al. (2020), since this circuit for the measurement of the fidelity only measures the overlap between two states (in this case $|\psi\rangle$and $|\phi\rangle$) and therefore it does not account for databases with N data points, all described by:

$$Phi = \{\phi_1 \cdots \phi_N\}$$

The modified circuit will be:



which adds to the previous circuit the register r_i, an oracle $\mathcal{M}$ and a measurement on r_i, that is a register with n qubits representing the computational basis, where $n = log_2 N$ and

N the number of data points in the training data set.
All n qubits are brought in superposition via the $H^{\otimes}n$gate creating the state$|i\rangle$with $i \in [1, N]$.
The oracle identifies which state from Φ needs to be applied to r_ϕ , by using the state $|i\rangle$as index:

$$\mathcal{M}|i\rangle|0\rangle = |i\rangle|\phi_i\rangle$$

The question is how does the oracle perform such operation.
The circuit described above applies the method of quantum parallelism: it brings into superposition of all states $|i\rangle$ at once, causing the oracle to initialize r_{phi} into a superposition of all data points $\phi_i \in \Phi$ with the equation we just saw. So, the overlap between ψ and all states ϕ_i is measured simultaneously.
The second step is to find the value of the closest k. From Afham et al. (2020) we know that the probability densities of the r_c register can be defined by:

$$p(0) = \frac{1}{2} + \frac{1}{2N}\sum_{i=1}^{N}\|\langle\psi|\phi\rangle\|^2, \quad p(1) = \frac{1}{2} - \frac{1}{2N}\sum_{i=1}^{N}\|\langle\psi|\phi\rangle\|^2$$

and that the probability density of measuring the i-th outcome in the register is described by:

$$p_0(i) = \frac{1 + \|\langle\phi_i|\psi\rangle\|^2}{N + \sum_{j=1}^{N}\|\langle\phi_j|\psi\rangle\|^2}, \quad p_1(i) = \frac{1 - \|\langle\phi_i|\psi\rangle\|^2}{N - \sum_{j=1}^{N}\|\langle\phi_j|\psi\rangle\|^2}$$

They define a new variable, the contrast, which is the difference between these two probabilities for the i-th outcome, and is directly proportional to the fidelity. This is the variable that will be used fot the knn classification. So, now, the fidelity can be rewritten as:

$$F_i = \frac{N}{2}q(i)(1 - R^2) + R$$

where $R = p(0) - p(1)$
Finally, we can define the distance as $D = 1–F$, sorting the list to determine the majority vote, which will assign a qubit to the most occurring class among the k number of data points with the lowest D. According to Kok (2020), translating the knn algorithm to a quantum algorithm promises a complexity of $log_2(ND)$ (instead of ND for classical methods) for determining k nearest neighbors in a data set with N vectors with dimensionality D. The data are also encoded using the analog (amplitude) encoding method before loaded onto the quantum circuit.
Furthermore, the runtime as a function of complexity $log_2(ND)$ shows a quadratic increase for the Qknn against an almost constant runtime for the knn.
A final consideration made by Kok is that ‘the present-day use of the Qknn is not advised until the accessibility to the SDK increases, the noise on a quantum computer reduces, and algorithms of larger scale can run better” (Kok 2020).
After consulting all the latest and most relevant research, I was able to sum up the findings concerning the comparison between classical machine learning algorithms and their quantum counterpart in a table, stressing the pros and cons of both categories. This is done in Chapter 5.

3.6 Quantum Machine Learning in Finance

Despite being at an early stage of development, quantum computation theory can already be found at the basis of many researchers' works, also at the intersection with other disciplines. In particular, a number of applications of quantum machine learning in finance exists. Some examples include the use of Monte Carlo simulations for risk analysis (Woerner and Egger 2019), combinatorial optimization for optimal trade opportunities (Rosenberg et al. 2016) and quantum deep learning for option pricing (Zoufal et al. 2019).
In the next chapter, a more theoretical approach to the concepts elaborated thus far is adopted. In particular, I will try to overcome the problem faced by Kok in his attempt to design a quantum knn algorithm, namely the contradictory results he obtained when comparing the quantum knn to the classical one. Contrarily to what one would expect, he did not observe an improved performance of the quantum algorithm over the classical counterpart, and he hypothesizes the reason could be the absence of data pre-processing on the financial dataset. The purpose of the next chapter is therefore to perform such pre-processing on the same dataset he used, and to compare the result both to the performance of the classical knn, and to that of Kok's quantum algorithm.

4 Chapter 4 - Python Implementation Using Qiskit

4.1 Introduction

This chapter focuses on the experimental part of the thesis: first, an introduction to Qiskit is made, as well as an overview of common techniques for data pre-processing; then the focus will be on the Python code used to perform the pre-processing (including feature engineering) on the financial dataset, in order to achieve a good performance of the Qknn.

4.1.1 Qiskit

As the official Qiskit website recites, "Qiskit is an open-source software development kit (SDK) for working with OpenQASM and the IBM Q quantum processors".
It was released by IBM and it can easily be implemented using the Python language.
After installing Qiskit and all the relevant libraries from the Python shell, I started running some preliminary lines of code.
An examplary code of how to run a basic quantum circuit using Qiskit can be found below:

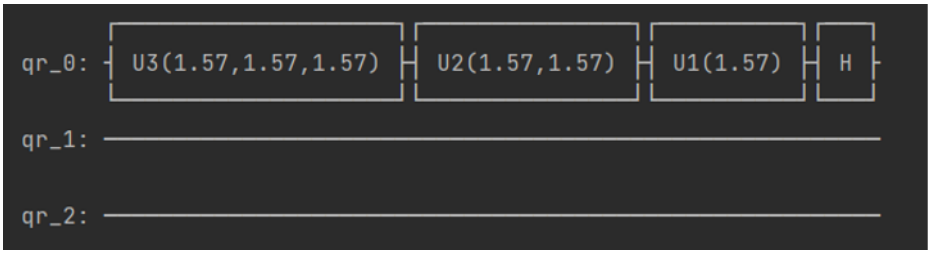
```
from qiskit import QuantumCircuit, QuantumRegister, ClassicalRegister
# Define quantum register
qr = QuantumRegister(3, "qr")
# Define quantum and classical registers on circuit
qr = QuantumRegister(3, "qr")
cr = ClassicalRegister(3, 'cr')
circuit = QuantumCircuit(qr, cr, name="quantum_circuit")
# Define quantum circuit
circuit = QuantumCircuit(qr, name="quantum_circuit")
pi = 3.14
# u3 gate
circuit.u3(pi/2, pi/2, pi/2, qr[0])
print(circuit)
# u2 gate
circuit.u2(pi/2, pi/2, qr[0])
print(circuit)
# u1 gate
circuit.u1(pi/2, qr[0])
print(circuit)
# Hadamard gate
circuit.h(qr[0])
print("Hadamard")
print(circuit)
# X gate
circuit.x(qr[0])
print("X-gate")
print(circuit)
```

```
# X gate
circuit.x(qr[0])
print("X-gate")
print(circuit)
# Y gate
circuit.y(qr[0])
print(circuit)
# Z gate
circuit.z(qr[0])
print(circuit)
# Rx gate
circuit.rx(pi/2, qr[0])
print(circuit)
# Ry gate
circuit.ry(pi/2, qr[0])
print(circuit)
# Rz gate
circuit.rz(pi/2, qr[0])
print(circuit)
# Controlled-NOT o Controlled-X gate
circuit.cx(qr[0], qr[1])
print(circuit)
# Controlled-Y gate
circuit.cy(qr[0], qr[1])
print(circuit)
# Controlled-Z gate
circuit.cz(qr[0], qr[1])
print(circuit)
```

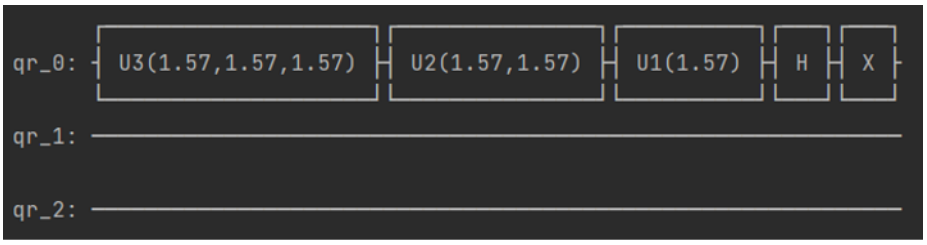
```
# Controlled-Hadamard gate
circuit.ch(qr[0], qr[1])
print(circuit)
# Controlled-Z_Rotation gate
circuit.crz(pi/2, qr[0], qr[1])
print(circuit)
# Controlled-u1 gate
circuit.cu1(pi/2, qr[0], qr[1])
print(circuit)
# Controlled-u3 gate
circuit.cu3(pi/2, pi/2, pi/2, qr[0], qr[1])
print(circuit)
# SWAP gate
circuit.swap(qr[0], qr[1])
print("SWAP gate")
print(circuit)
# Toffoli gate
circuit.ccx(qr[0], qr[1], qr[2])
print(circuit)
```

We can now visualize, for example, the following gates:

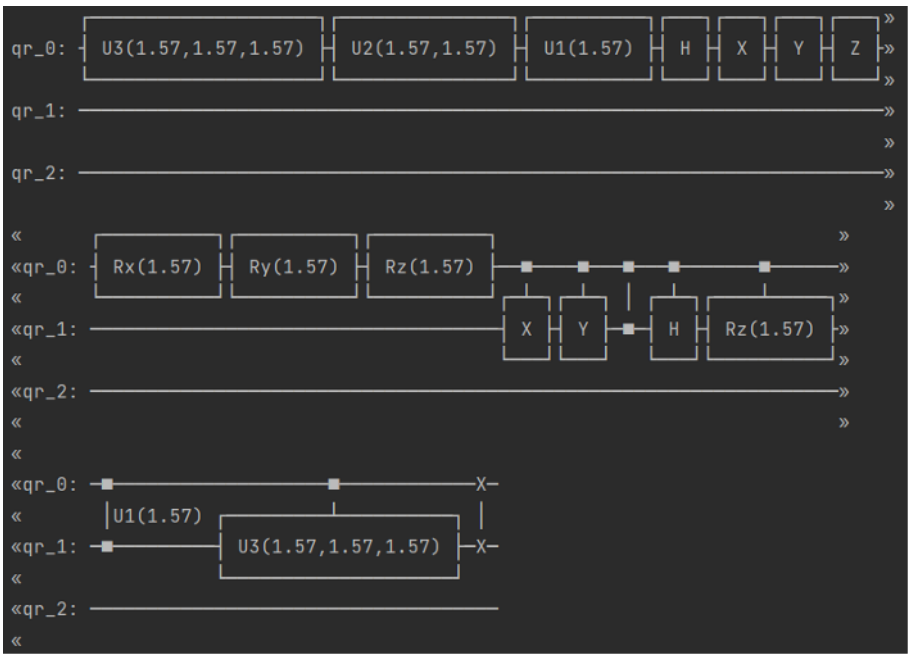
- **Hadamard Gate**



- **X-gate**



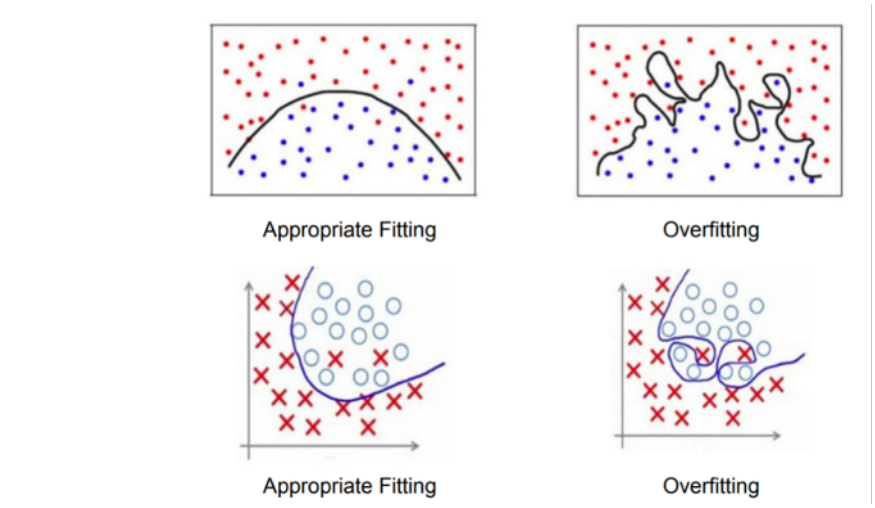
- **SWAP-gate**



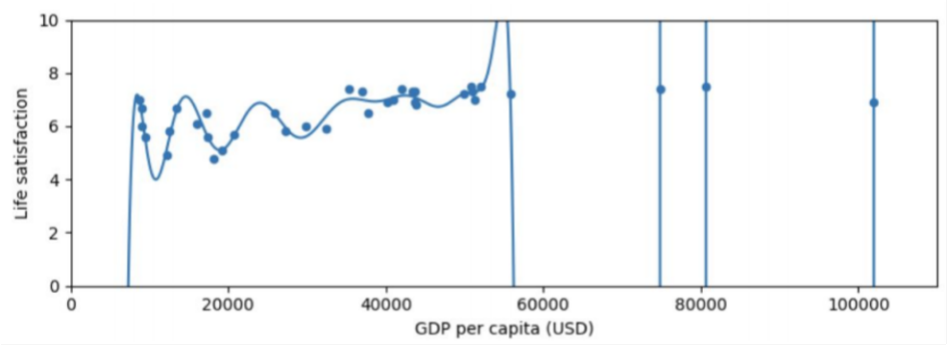
Qiskit Aqua (Algorithms for QUantum Applications) provides a library of quantum algorithms to build quantum applications and leverage near-term devices. Although it has been deprecated (meaning some of its core algorithms and functions have been moved to another library, Qiskit terra) I still managed to use it for the purposes of this thesis.
After familiarizing with Qiskit and its tools, I will use it more thoroughly to carry out the development of a Qknn able to perform a classification task on financial pre-processed data. Before doing that, a little introduction to the main steps in data pre-processing are enunciated.

4.2 Data Pre-Processing

The importance of data pre-processing stems from the fact that the results we obtain can vary significantly, depending on whether we perform it or not.
As we have already mentioned in Chapter 2, an over-generalization of the training data may lead to overfitting, as shown in the image below:



For example, the following high-degree polynomial model representing life satisfaction as a function of the GDP per capita in USD currency strongly overfits the training data:



As a result, the overfitted model will likely detect patterns in the noise, thus not being able to generalize to new instances.
A possible solution, which is implemented in this thesis, is to make use of feature engineering methods, as explained below.

4.2.1 Feature Engineering

An effective way in which we could improve the training set is through feature engineering, which refers to the process of selecting a good set of features to train on, and it comprises two steps: feature selection, that is, selecting the most useful features to train on, among a set of existing features; and feature extraction, which consists in combining existing features to produce more useful ones. An example of a feature selection method is the analysis of the principal components, which is carried out in this thesis and for this reason will now be shortly introduced.

4.2.2 Principal Component Analysis

Principal Component Analysis (or PCA) is a dimensionality-reduction method which comprises five steps:

1. Standardize the range of continuous initial variables;
2. Compute the covariance matrix to identify correlations;

3. Compute the eigenvectors and eigenvalues of the covariance matrix to identify the principal components;
4. Create a feature vector to decide which principal components to keep;
5. Recast the data along the principal components axes.

This method is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set.

4.2.3 Cross-Validation

Cross validation is a process where we create many small validation sets, which must always be as representative as possible of the data we expect to use in production in order to avoid mismatches with future datasets.
Each single model is evaluated once per validation set after it is trained on the rest of the data; we then average all these evaluations to get a more accurate measure of its performance. For more accurate predictions, we may want to re-train our model on another validation set: if it performs well on this new validation set, then the model is not overfitting the training set; if it performs poorly, then the problem must be coming from the data mismatch.
A drawback of this method is that the training time increases.

4.2.4 Evaluation

After obtaining the validation sets, the evaluation part can be carried out using various metrics. For the purpose of this thesis, the chosen metric is the AUC (Area Under the Curve). An important distinction in this regard is between the concepts of "true/false positives" and "true/false negatives": these categorizations are useful to evaluate the ability of a machine learning model to identify the true or false class of an instance, and also give an idea of how many misclassifications it performed. From this information, one can construct evaluation metrics such as the recall, which will be used for the performance evaluation of both the knn and the Qknn on the German credit dataset. In particular, the recall metric indicates the number of positive cases which were correctly classified by the model. This metric seems the most fit, since the aim of the model is to identify the risk of default from the viewpoint of "bad" risk, which corresponds to the positive class, as the section below will explain.

4.3 The Code

The remaining part of this chapter will focus on the main steps of the code which was elaborated to accomplish the objective of the thesis: to test the efficiency of the Qknn algorithm on a pre-processed dataset, and to compare the results with respect the classical knn on the same, pre-processed dataset.

4.3.1 The Dataset

The German dataset was provided by Dr. Hans Hofmann of the University of Hamburg. It documents several financial and demographic information about checking accounts, their owners (including personal info like sex and age), their housing and employment status. This is how the original dataset looks like:

Attribute	Description	Type
1	Status of existing checking account	Categorical
2	Duration in month	Numerical
3	Credit history	Categorical
4	Purpose	Categorical
5	Credit account	Numerical
6	Savings account/bonds	Categorical
7	Present employment since	Categorical
8	Installment rate in percentage of disposable income	Numerical
9	Personal status and sex	Categorical
10	Other debtors/guarantors	Categorical
11	Present residence since	Numerical
12	Property	Categorical
13	Age	Numerical
14	Other installment plans	Categorical
15	Housing	Categorical
16	Number of existing credits at this bank	Numerical
17	Job	Categorical
18	Number of people being liable to provide maintenance for	Numerical
19	Telephone (yes/no)	Categorical
20	Foreign worker	Categorical

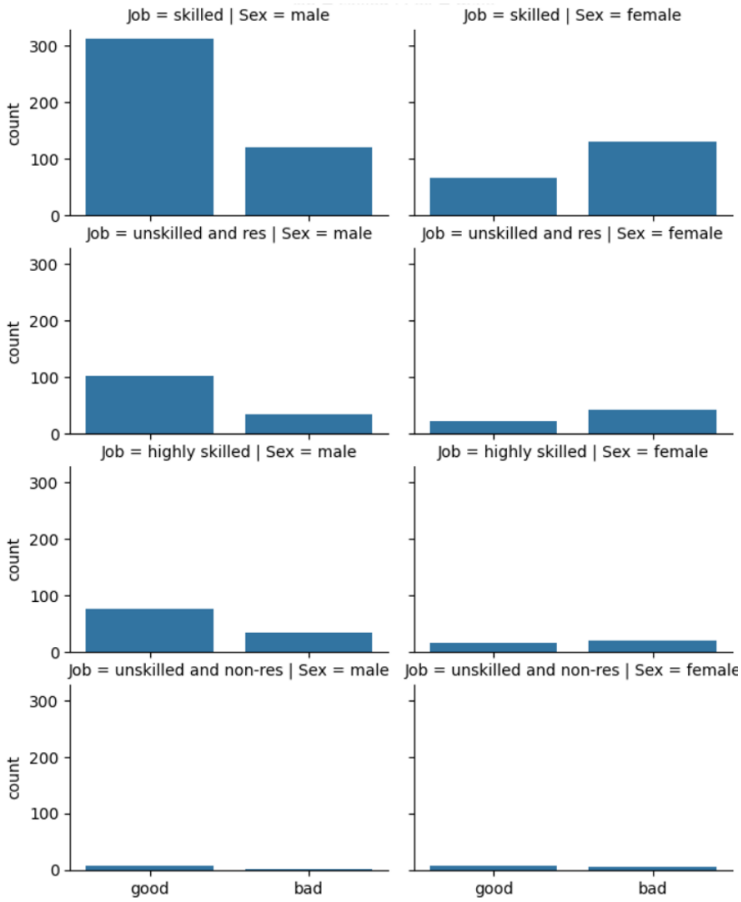
The dataset used in the present project, however, is simplified and comprises 10 columns, of which the target variable or class is the "Risk" column and contains the two values: "good" and "bad". These will need to be label encoded to 0 and 1, respectively, to meet the general expectation for imbalanced binary classification tasks where 0 represents the negative case and 1 represents the positive case.
The task is to determine whether the customer is good or bad based on risk. The assumption is that the task involves predicting whether a customer will pay back a loan or credit, since this is of particular interest from the viewpoint of a bank. Good customers are the default or negative class, whereas bad customers are the exception or positive class. A total of 70 percent of the examples are good customers, whereas the remaining 30 percent of examples are bad customers.
Finally, a cost matrix is provided with the dataset that gives a different penalty to each misclassification error for the positive class. Specifically, a cost of five is applied to a false negative (marking a bad customer as good) and a cost of one is assigned to a false positive (marking a good customer as bad). This suggests that the positive class is the focus of the prediction task as it is more costly to the bank or financial institution to give money to a bad customer than to not give money to a good customer. This explains the reason behind the choice of the recall metric for evaluation.

4.3.2 Data Visualization

After loading the csv file as a data-frame and examining its shape, the first step is to look at the distribution of the numerical input variables by creating a histogram for each one of them:

The age and the duration clearly display a skewed distribution, whereas the distribution of jobs appears to be discrete. As for the credit amount, it follows a seemingly exponential distribution. Based on these histograms, we would expect scaling the distributions to the same range to be useful later. The next step is to create plots which help to better visualize the data we are dealing with:

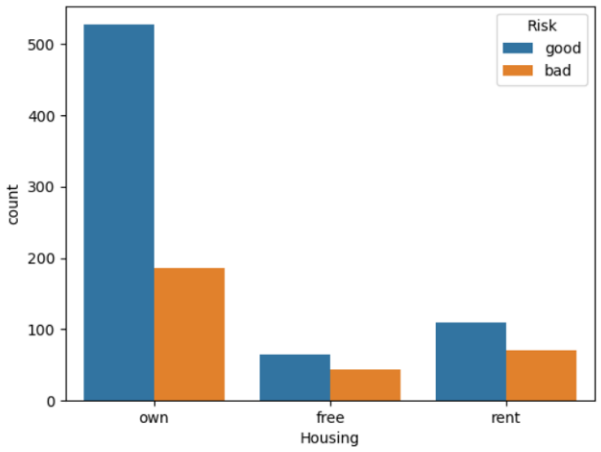
1. Distribution of Risk across Job Type and Gender:



Observations:

- Skilled workers are more likely to get ‘good’ risk rating. This makes sense, as skilled workers tend to earn more and have more job security.
- Men, throughout all job types, tend to get a ’good’ rating much more often than women. Skilled female workers are more commonly classified as ’bad’ than ’good’. This hints towards institutional sexism.

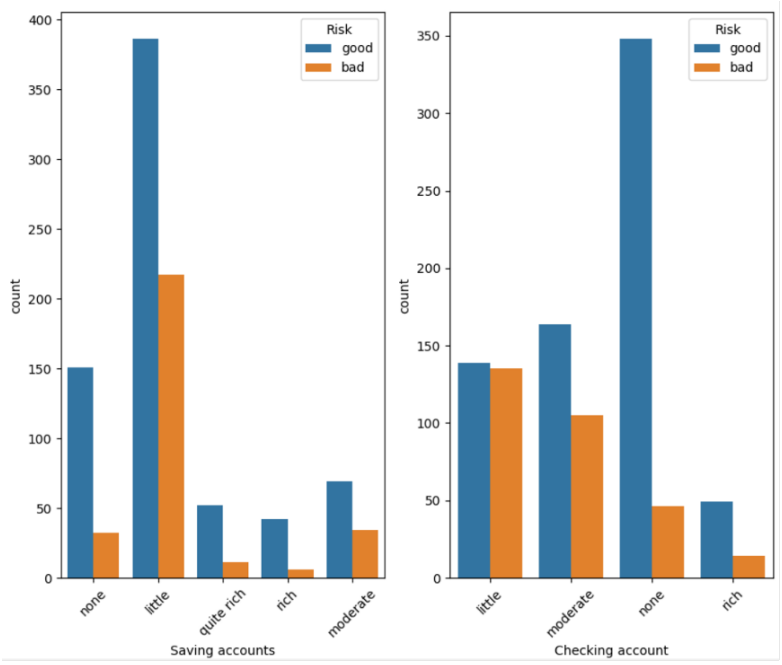
2. Distribution of Risk across Housing Status:



Observations:

- People who own their own home are more likely to get a “good” risk rating on their credit compared to people who rent. This is understandable, as credit worthiness goes hand in hand with asset ownership.

3. Distribution of Risk across Bank Account Status:

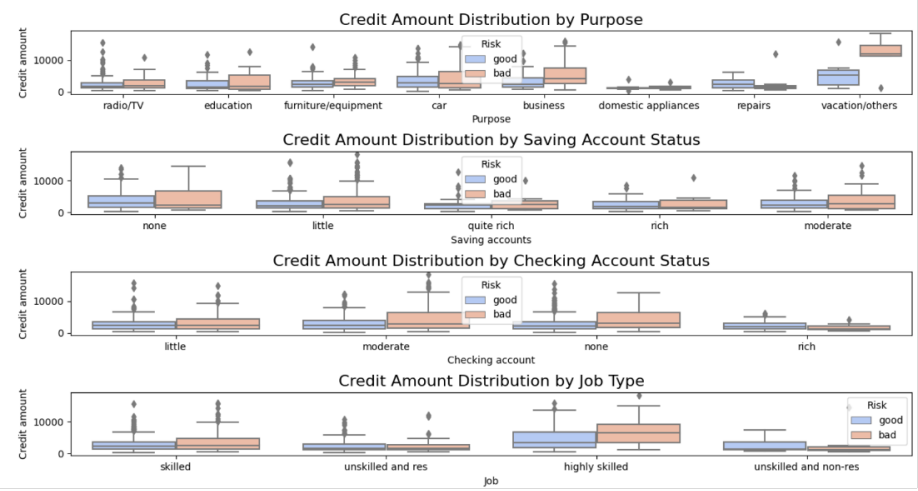


Observations:

- When it comes to saving accounts, richer individuals seem more likely to be classified as “good” with respect to others. However, there are visibly more “good” ratings than “bad” ones. This makes sense since the very existence of a savings account implies some degree of financial security.

- As for checking accounts, again richer individuals seem more likely to be classified as “good”. Those with ‘little’ money have an equal distribution of good and bad ratings. It is also astounding to note the disparity between ‘good’ and ‘bad’ risk ratings when it comes to people whose checking account value was “none”.

4. Boxplots of Distribution of Credit Amounts Across Several Variables:



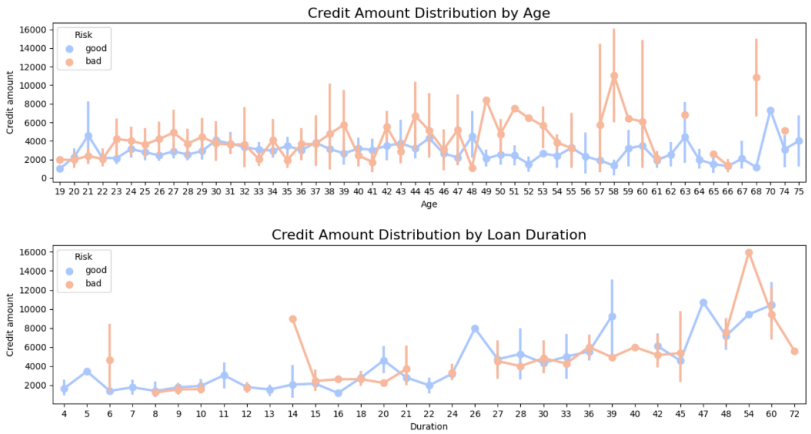
Observations:

- Quite a number of interesting things can be noted when looking at the “Credit Amount Distribution by Purpose”: firstly, certain “purposes” are more likely to have “bad” ratings, especially if the loan amount is high. For example, if we look at vacation/others, we can see that “bad” rated loans usually consist of a higher amount. The same trend can be observed for business loans. Generally, if we exclude furniture/equipment and repairs categories, almost all other categories have a larger interquartile range when it comes to the credit amounts of “bad” loans. Also, the radio/TV category has a lot of outliers, especially for loans classified as “good”. This is quite interesting, as it is hard to imagine a loan for buying a radio or a TV to be approved by a bank.

- When we look at the amount distributions according to bank account statuses, we observe that rich people tend to borrow less compared to poorer people. On one hand, this makes sense as poor people would need more money from a bank. On the other hand, it doesn’t make much sense for a relatively rich person to borrow a very small amount from the bank.

- When it comes to job types, skilled workers tend to have a bigger and higher interquartile for bad loans than for good loans. This makes sense since larger amounts are more likely to be classified as “bad”. However, if we look at unskilled borrowers, this disparity doesn’t hold true.

5. Credit Amount Distribution by Age and Loan Duration:

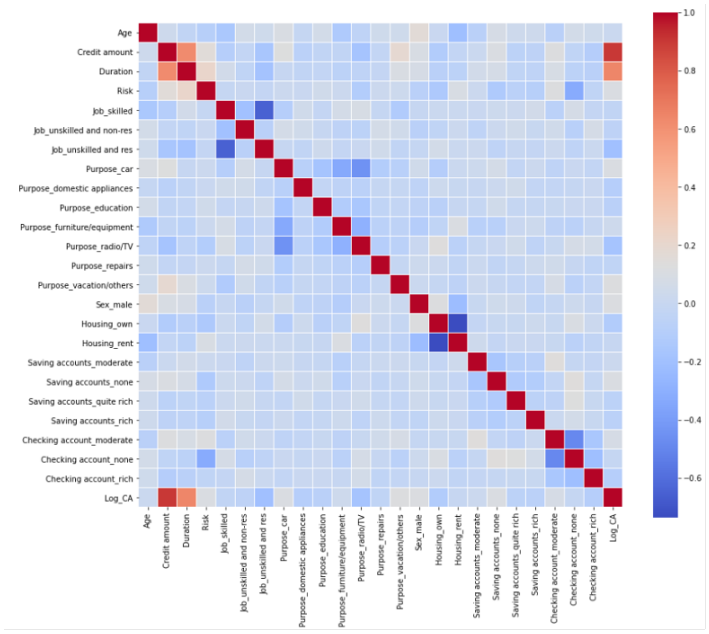


Observations:

- Generally, throughout all age gaps, bad loans tend to be of a higher amount; however, when it comes to people over 55, this gap widens.
- When it comes to loan duration, bad and good loans amounts tend to move in tandem. Moreover, larger loans over a longer duration are more likely to be classified as "good". After gaining many useful insights on the data, the pre-processing may begin.

4.3.3 Data Pre-Processing: Feature Engineering

The first thing to do is to generate dummy variables for all the categorical variables and then drop the redundant ones. Moreover, as "Risk" is a categorical value, it needs to be encoded as binary: specifically, I will assign 0 to "good" loans and 1 to "bad" loans, since from a bank's perspective it is more important to identify bad loans.
Then, instead of using the raw credit amount as a dependent variable, it is more useful to take the log of these amounts, in order to make the distribution more normal, as the graph below shows:

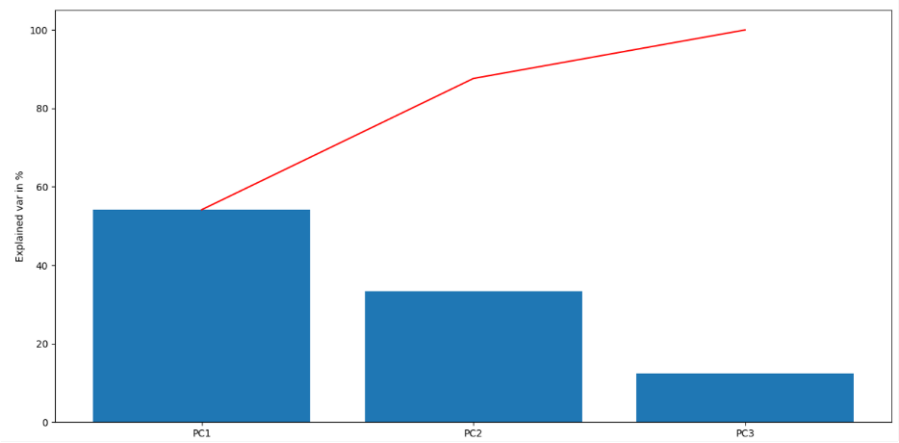


4.3.4 Principal Component Analysis

After standardizing all the data of a smaller set of features (“Age”, “Credit Risk” and “Duration”) using a Standard Scaler (provided by the scikit-learn python library), Principal Component Analysis is performed on the new data-frame. In particular, the following three operations are carried out:

- Eigendecompostion of the covariance matrix;
- Eigendecomppostion of the correlation matrix;
- Eigendecompostion of the correlation matrix.

The following plot shows that around 58% variance can be explained by the first component and around 32% variance is explained by the second component:



So, the first and second components both cover around 90% of the variance and therefore the third component can be dropped without losing too much information. This information can be used to finally implement the projection onto the new feature space.

4.3.5 Evaluation

As anticipated, the measure used for the evaluation is the recall: the closer the recall value is to 1, the more accurate the prediction performed by the tested model.
After splitting the dataset into training set and test set, hence creating validation sets to test both the knn and the Qknn models, we are ready to perform the classification.

4.3.6 Knn

The kNN classifier that is used in this thesis is the KNeighborsClassifier provided by scikit-learn library.
The model has been designed in order to account for the cosine similarity as a distance measurement.

4.3.7 Qknn

The quantum circuit will be simulated as if it was run on a quantum computer using the qasm_simulator. This means that it will have to run multiple times to approach the state vector produced by the quantum circuit.

The first step is encoding: all data provided to the quantum circuit is saved to the qubits in the circuit, which is done by translating numbers into sequences of 0s and 1s as described below:

$$\vec{d} = d_0 d_1 \cdots d_n \in \{0,1\}^n$$

In turn, this value describes a sum which is calculated via:

$$\begin{pmatrix} d_0 & d_1 & \cdots & d_n \end{pmatrix} \cdot \begin{pmatrix} 2^0 \\ 2^1 \\ \vdots \\ 2^n \end{pmatrix}$$

Any real number can now be obtained from this operation.
This method is also carried out on classical computers, whenever data are encoded to binary. In particular, data will be encoded via the analog or amplitude encoding, a built-in method in Python Numpy and achieved by using the einsum method which performs the Einstein summation.
Finally, the Oracle is initiated. Recall that an Oracle is an operation that has some unknown property which allows it to get a desired result. In particular, it needs to initialise a register r_ϕ into the following state:

$$\mathcal{M}|i\rangle|0\rangle = |i\rangle|\phi_i\rangle$$

This is accomplished by Kok (2020) by means of the "where_to_apply_x()" method he implemented: "since the computational basis r_i is brought into a superposition of all states $|i\rangle$, the register is brough to a superposition of all training states $\phi_i \in \Phi$ because of the Oracle".
Now that both models are ready, the classification task can be performed. Results are shown and commented in the next chapter.

5 Chapter 5 - Results and Limitations

5.1 Introduction

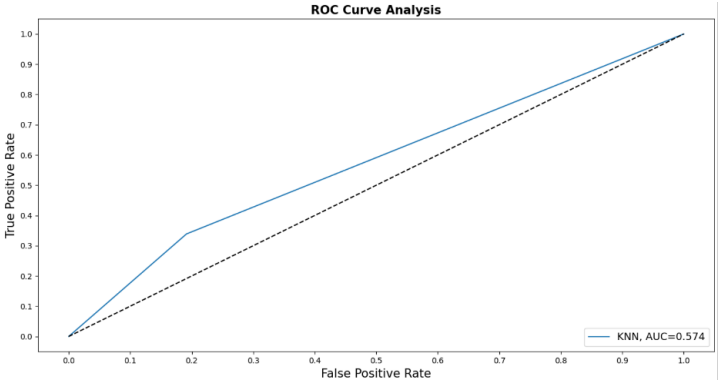
This chapter comments the results obtained from:

- The execution of both the classical knn and the quantum knn on the pre-processed German credit dataset.
- The comparison between other three classical machine learning algorithms and the quantum counterpart.
- The analysis of the current situation regarding quantum computing, with a particular focus on Rydberg atoms and quantum error correction in the NISQ era.

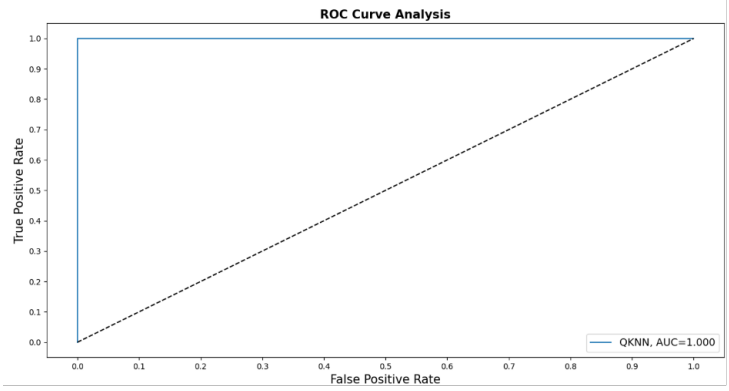
It also defines the limitations of the study and future research directions that may be followed.

5.2 QKnn vs. Knn

After feeding the knn algorithm with the pre-processed German credit dataset, the performance of the model is explained with the following plot:



The AUC value is slightly above 0.5, meaning that the model manages to classify true positives better than it misclassifies false positives. This result is not optimal, and may be improved as the ROC curve for the Qknn confirms:



Below the comments regarding these results.

5.2.1 Comments

The comparison between the two ROC curves shows the superiority of the Qknn with respect to the knn model. Incredibly, the Area Under the Curve in this case results as exactly 1. This implies that the classifier is perfectly able to distinguish between true positives and false positives.
Despite the seemingly encouraging result, some observations are worth noting.
First of all, the same method for testing the accuracy was employed for both models. However, it seems reasonable to assume that different and more reliable methods should be used to benchmark the results regarding the quantum algorithm.
Another point is that, in spite of the many researches highlighting the possibility to run quantum programs on classical devices, the latter are still affected by noise. For this reason, the results presented thus far still may be biased. An improvement in this sense would be to apply error correction techniques, specifically designed for these types of quantum algorithms, which however is out of scope for this thesis.
Nonetheless, data pre-processing succeeded in providing good results, which is expectable since it is what gives significance to an analysis, by selecting only the data which are most relevant. As Wittenbach et al. (2020) stated, "pre-processing is a necessary step to solving the core modelling financial tasks". It was reasonable to expect a correspondence of this statement also in the quantum reign, especially once the proper hardware on which to run quantum algorithms will have been developed, which however will not happen in the next years.
Although it is not possible to overcome the abovementioned hardware problem for now, using a quantum simulator on a classical hardware still may be useful to understand the mechanisms implied by quantum mechanics, in order to better address limitations regarding noise in future and more specialized studies.
For what pertains the other limitations, it may be interesting to continue this work by focusing on which value k should take on. As a matter of fact, the optimal choice of the number k could significantly improve both the classical knn algorithm and the quantum one.

5.3 QML vs. ML

The results of the comparison between the quantum machine learning algorithms presented in the "Quantum Machine Learning Algorithms" section of Chapter 3 and the classical counterpart can be summarized in the following table, including the comparison between the Qknn and the knn explained in the preceding section.

Algorithm	Type	Advantages	Disadvantages	Implementations
Support Vector Machines	Classical machine learning algorithm for classification tasks.	- Able to perform efficiently a variety of tasks on smaller data sets. - Kernel trick to deal with high-dimensional data	- Hard to scale to large datasets. - Expensive in terms of computational resources - Bad results on perceptual problems such as image classification	Binary classification tasks. Examples: email spam detection. Non-linear classification.

Quantum Support Vector Machines	Quantum machine learning algorithm for classification tasks.	- Exponential speed-up with respect to classical sampling algorithms requiring polynomial time. - Can be implemented on a classical hardware. - Efficient solution to the problem of classical SVMs of scaling to larger datasets. - Optimisation of the kernel trick thanks to the construction of a kernel matrix. - Data privacy advantage.	- Affected by noise from both the environment and the NISQ devices available as of today. - Theoretical gains cannot be put to practice on a real quantum computer yet.	Classification of patterns. Non-linear classification.
Neural Networks	Classical machine learning algorithm for classification tasks.	- Can carry out complex computational problems that are infeasible for other classical machine learning methods thanks to their architecture based on deep layers.	- Very resource-intensive problems are still hard to solve with this architecture.	XOR problem. Binary classification problems. Multi-class classification. Multi-label classification.
Quantum Neural Networks	Quantum machine learning algorithm for classification tasks.	- Theoretical advantage of quantum computing over classical computing	- Affected by noise from both the environment and the NISQ devices available as of today. - Theoretical gains cannot be put to practice on a real quantum computer yet.	Universal quantum computation. Fast optimisation.
Convolutional Neural Networks	Classical machine learning algorithm for classification tasks.	- Can solve image classification problems that other classical methods are unable to solve.	- Very resource-intensive problems are still hard to solve with this architecture.	Multi-class classification: - Face classification. - Optical character recognition. Image processing.
Quantum Convolutional Neural Networks	Quantum machine learning algorithm for	- Theoretical advantage of quantum computing over classical computing	- Affected by noise from both the environment and the NISQ devices available as of today.	Multi-class classification: -Face classification -Optical character recognition. Quantum physics problems. Image processing.

	classification tasks.		- Theoretical gains cannot be put to practice on a real quantum computer yet.	
K-Nearest Neighbours	Classical machine learning algorithm for classification tasks.	- Simple and interpretable algorithm - Quick calculation time - No assumptions about the data - Ideal for non-linear classification	- Accuracy depends on the quality of the data - Slow or intractable predictions on larger datasets - Sensitive to data scaling and to irrelevant features - Computationally expensive for high amounts of data	Binary classification Multi-class classification Imbalanced classification: -Fraud detection -Outlier detection Image/video recognition
Quantum K-Nearest Neighbours	Quantum machine learning algorithm for classification tasks.	- Theoretical advantage of quantum computing over classical computing.	- Accuracy depends on the quality of the data - Affected by noise from both the environment and the NISQ devices available as of today. - Theoretical gains cannot be put to practice on a real quantum computer yet.	Binary classification Multi-class classification Imbalanced classification: -Fraud detection -Outlier detection Image/video recognition
Shor	Pure quantum algorithm for decision problems.	- Exponential advantage in solving the factorization problem with respect to classical methods.	- Able to break the RSA protocol. - Cannot be implemented on a classical hardware. - Needs a quantum error corrected computer in order to properly run. - Theoretical advantage cannot be proven yet.	Factorization problem.
Grover	Pure quantum algorithm for search problems.	- Quadratic speedup in solving the unstructured search problem with respect to classical methods.	- Cannot be implemented on a classical hardware. - Needs a quantum error corrected computer in order to properly run. - Theoretical advantage cannot be proven yet.	Unstructured search problem. Decision problems in NP.
Deutsch	Pure quantum algorithm.	- Exponential acceleration of the Deutsch problem with respect to the classical counterpart	- Cannot be implemented on a classical hardware. - Needs a quantum error corrected computer in order to properly run.	Deutsch problem. Determine whether f: {0, 1} → {0, 1} is a constant or balanced function.

			- Theoretical advantage cannot be proven yet.	
Deutsch-Jousza	Pure quantum algorithm.	- Allows quantum amplitudes to take both positive and negative values, as opposed to classical probabilities that are always non-negative.	- Cannot be implemented on a classical hardware. - Needs a quantum error corrected computer in order to properly run. - Theoretical advantage cannot be proven yet.	Deutsch-Jouzsa problem, where the Deutsch problem is extended to boolean functions on n bits; specifically designed problem to be easy for a quantum algorithm and hard for any classical computer to show the efficiency of quantum computing solutions over classical approaches.

5.3.1 Comments

A first striking difference between pure quantum algorithms and quantum machine learning algorithms is that the former are specifically designed to show the theoretical quantum advantage over classical solutions, while the latter are used to solve a variety of practical problems with a computational advantage over classical solutions.
In addition, it is possible to note that quantum machine learning algorithms can optimize more complex problems from a computational viewpoint, hence providing a solution to the limitation of classical computers when dealing with larger datasets for machine learning programs, while being able to perform also the same tasks as their classical counterparts.
On the other hand, a common disadvantage that both categories of quantum algorithms display is that they are still negatively influenced by the noise of current quantum devices. In particular, the theoretical advantage of pure quantum algorithms cannot even be proven yet, while in the case of quantum machine learning methods they may present biased results without proper error-correcting techniques applied to them, which are currently at early stages of development and therefore are not so effective as of today.
We can conclude that quantum computing still has a great appeal to scientists and researchers, and many advances are being made; however, they are still at an early stage, and much needs to be done to eventually overcome classical computing.
The next section will help understand what is the current state of development of quantum devices and the future prospects.

5.4 Current State of Development of Quantum Computers and Simulators: An Overview

The properties of quantum mechanics, namely superposition, entanglement and interference and all their applications promise a revolution in the world of computing, from a theoretical viewpoint (Brassard et al. 1998).
Let's analyze more in detail the potentialities of each property:

- Superposition provides the method of saving 2^n numbers on n qubits, reducing the number of bits required exponentially (Svore et al. 2014) and hence counterbalancing the big limitation of classical computing that was identified by Moore's law, namely the fact that as the number of bits required to solve problems increases, it will be infeasible to solve them on classical computers. Quantum computing theory reveals us that this may be overcome thanks to the exponential speedup it provides.

- Entanglement is what allows these n qubits to be manipulated simultaneously using quantum parallelism, a way to perform a quantum operation on a superposition of inputs to produce a superposition of outputs.

- Interference can be used to reinforce the probability of obtaining the desired result via constructive interference or reduce the unwanted result via destructive interference.

On the other hand, these methods present some drawbacks which stem from the nature of quantum mechanics itself: the problem of measurement. Once an observation is made and the state collapses, the superposition is lost. This has important implications in the development of quantum computers, and it is one of the main concerns of researchers today. In fact, despite the theoretical appeal of these methods, most of their applications cannot be practically proven yet, due to the fact that the appropriate quantum devices are still to be constructed.

5.4.1 Quantum Computing Stages and The NISQ Era

In order to better understand where the main critical issues reside, it is useful to consider the timeline which is conventionally used when referring to advances in quantum computing. It comprises three stages:

1. Infancy, which was the old stage where quantum operations were performed only on a small number of qubits, but no quantum circuit could be executed entirely. This was the early phase in quantum computing research;

2. Noisy Intermediate Scale Quantum (NISQ), the stage where quantum computing is now: in this era, the problem of noise interferences, known as quantum decoherence is predominant. In fact, as of today each qubit in a quantum computer has a probability of losing information to the environment over time. The more qubits are in a system, the higher the probability that one qubit has lost too much information, causing the entire circuit to become useless. Therefore, large quantum circuits (constructed by many qubits) are more susceptible to noise.

   Nonetheless, this is also the era in which advances are being made, as quantum operations are being executed on small circuits. The next focal point will be to reduce the noise which interferes with the results achieved in this stage.

3. Fault Tolerant (FT), the ultimate stage researchers are desperate to reach. At this point, large quantum circuits are expected to be successfully executed, and errors are expected to be corrected to the extent that results are no longer significantly influenced by them.

The big obstacles faced by quantum computing in the present NISQ era seem to have created a discrepancy in the scientific world: many argue that these problems will never be overcome, meaning that the Fault-Tolerant stage will never be reached in practice.
However, there are reasons to say that this is not entirely true. As a matter of fact, the latest research (Wu et al. 2021) shows how the main drawback characterizing the NISQ era, namely quantum decoherence, may be overcome in the future thanks to the application of Rydberg atoms to quantum computing.

5.4.2 Rydberg Atoms

Rydberg atoms belong to the category of neutral atoms, that is, atoms displaying a total electric charge equal to zero. In particular, they are found in a highly-excited state, as opposed to ground atoms in which all electrons are in the lowest possible energy levels. This physical characterization of Rydberg atoms makes them ideal choices for the implementation of controllable quantum many-body simulators, as it allows interactions among them to be controlled and manipulated as needed, which makes them especially prone to neutral atom-based platforms for quantum information processing.
In fact, manipulating the interactions among Rydberg atoms amounts to implementing the interaction among qubits, hence allowing for the construction of quantum gates (as opposed to neutral atoms in their electronic ground state, which can only interact via contact collisions).
Neutral-atom quantum processors are based on configurable arrays of single neutral atoms (in this case, Rydberg atoms) which can be seen as registers, where each single atom plays the role of a qubit.
For the sake of completeness, I will now provide a quick overview of the properties of Rydberg atoms.
In 1888, the Swedish physicist Johannes Rydberg extended the Balmer formula, used to describe a series of spectral lines in hydrogen atoms, to a more general Rydberg formula which identifies any atom in the Rydberg state:

$$\frac{1}{\lambda} = R_H \left(\frac{1}{n^2} - \frac{1}{m^2} \right)$$

In particular, any atom (molecule or semiconductor quantum dot) in a state with a highly-excited electron is a Rydberg atom, as opposed to an atom in the ground state. Such property gives rise to dipole-dipole interactions.
This kind of interactions happen between two Rydberg atoms which are very near to each other (a few micro-meters) and are strong enough to shift significantly the energy of the doubly excited state, preventing the excitation of two atoms at the same time. This effect is called Rydberg blockade and is the basic mechanism to achieve a quantum logic: the excitation of a first atom to a Rydberg state conditions the excitation of a second one (Henriet et al. 2020).
This first property has relevant practical implications. For instance, when implementing SWAP gates on quantum devices this comes at high error costs, as each additional gate introduces errors along the computation. The relatively large connectivity of neutral atom devices allows to mitigate this effect.
In addition to large connectivity, neutral atom devices present the advantage of natively implementing multi-qubit gates involving more than 2 qubits. Such multi-qubit gates are instrumental for the efficient implementation of several quantum algorithms, including for example Grover search (Henriet et al. 2020) in terms of time-reduction. Another important property of Rydberg atoms is their instability: they decay with a characteristic lifetime τ according to the states they reside in, and eventually can fall to the ground state.
Two physical mechanisms dominate the decay process: spontaneous emission and BlackBody Radiation (BRR). The total lifetime τ of a Rydberg atom is given by:

$$\frac{1}{\tau} = \frac{1}{\tau_0} + \frac{1}{\tau_{bb}}$$

From this, the radiative lifetime of a given Rydberg state n is:

$$\frac{1}{\tau_0} = \sum_{n^1} A_{nn^1}$$

So, the lifetime of Rydberg atoms increases as the quantum number n becomes larger. On top of all properties listed thus far, this is perhaps the most important: scalability.
Major contributions to decoherence come from this property, as proved by Wu et al. (2021) through many meaningful experiments involving Rydberg atoms. As a matter of fact, scalability constitutes one of the central challenges for implementing large-scale quantum computation. Neutral atoms are inherently identical, thus the requirement for physical resources do not grow much with the scaling up of qubits (Wu et al. 2021).
It is established that, thanks to these striking advantages of Rydberg atoms, neutral atom systems allow to reproduce the key quantum phenomena (superposition, entanglement and interference) by manipulating interactions among atoms, hence becoming quantum simulators. This is extremely important to foster scientific discovery: understanding the mechanisms underlying quantum systems will eventually allow to make progress towards the shift from the NISQ era to the Fault Tolerant era. Another near-term application for neutral atom-based platforms is to solve complex optimization problems by measuring the final states of quantum many-body systems. This is especially promising with respect to machine learning tasks. In fact, solving combinatorial optimization problems remains a chief concern in many industrial areas, such as logistics, supply-chain optimization, or network design.

5.4.3 Main Findings

So, to sum up: at what point is the research on quantum computation today? And what are the future prospects? This section finally addresses these questions.
Nowadays, Noisy Intermediate-Scale Quantum devices are predominant. This means that they are subject to noise errors in the system, which prevents analyses regarding quantum simulations and/or computation to be completely reliable and, sometimes, to be carried out at all. Architectures that implement error-correction procedures are called fault-tolerant, and they will not be available in the near-future, due to the large overhead in the number of qubits necessary to implement error correction. Most of the quantum algorithms tackling computational problems, such as the ones we explored in Chapter 3, have been designed to be executed on an ideal quantum computer without errors. The most prominent example is Shor's factoring algorithm. While factoring a large integer N into the product of its prime factors is a very hard task on a classical computer, it can in theory be done quite efficiently on a quantum device. However, the implementation of these algorithms and the consequent analysis of their advantage with respect to classical computers will only be possible in practice with the rise of these error-corrected devices.
Even though fault-tolerant quantum computing is still many years away, recent results from pioneer companies like Google have shown that devices without error-correction and with a relatively modest number of qubits can already outperform any classical supercomputer for specific computing tasks. The emergence of quantum devices with a few hundred physical qubits opens many exciting perspectives in quantum computing and quantum simulation towards a quantum advantage with respect to supercomputers. However, in the longer term, neutral atom platforms exhibit the most interesting features, in that they allow for quantum-accelerated devices to push further their computational power and develop new applications. Their multi-core architecture makes possible to scale up the number of qubits available for computation, which will ultimately give rise to the implementation of quantum

error correcting codes, hence opening up to the era of fault tolerant quantum devices.
In particular, Rydberg atom-based systems are a recent and promising discovery, as they are expected to accomplish several complex computing tasks and exhibit certain quantum features and advantages despite being noisy.
These scalable quantum devices can be constructed thanks to Rydberg atoms with a small dependence on the temperature of the quantum hardware (recall that, as of today, the temperature under which the qubit exhibits its quantum properties must be about -273 degrees Celsius, which is a huge limitation in carrying out researchers on quantum matters), thereby reducing system noise and improving the quality of the quantum operations performed.
Devices based on Rydberg atoms show promise in their scalability and also allow for low error rates thanks to the ease of manipulation which comes with them, first of all the possibility to control at the single particle level, while preserving all the fundamental properties of qubits (coherence, entanglement, superposition), which makes them also particularly fit to simulate quantum environments (hence they are used as quantum simulators).
Even though neutral atom devices can reproduce many models of interest, the range of applications of this approach remains limited. To overcome this limitation, another framework has recently emerged, consisting in using the quantum formalism in tandem with a classical computer. In this hybrid approach, a classical computer is used to optimize a relevant cost function related to a target Hamiltonian, from measurements of highly entangled states. In this procedure, the target Hamiltonian never needs to be physically realized with the device.
In the logic and quantum error correction layer, decompositions of quantum algorithm in terms of quantum circuits, together with data processing of detection results, are processed by classical computer.
The user interfaces in the application layer are also aided by classical computer. These hybrid approaches particularly benefit the field of machine learning: the large dimensionality of the vector spaces involved in these operations makes their implementation at large scale very resource intensive. Examples we have seen include SVMs and neural networks.
The discovery of these algorithms triggered a lot of research efforts at the intersection of machine learning and quantum information processing, but the resource requirements for their implementation in term of numbers of qubits and gate fidelities is unfortunately prohibitive for devices that are and will be available in the short term. Motivated by these developments, researchers tried to find other ways to leverage machine learning methods on available NISQ quantum processors, which now are mainly employed in combinatorial optimization problems (Henriet et al. 2020).
Quantum machine learning has also seen a growing interest around the topics of practical implementations within many fields, including the financial sector.
Therefore, since in general the implementation of quantum code on classical devices is feasible, as we have seen, many argue that NISQ devices, so prone to errors, could just be left in favour of hybrid solutions involving classical devices. However, as we have seen, the NISQ era is a step that precedes the fault tolerant era; therefore, the only way to reach the ultimate stage is by keeping up with researching and making experiments, so as to gain as much knowledge about the functioning of quantum mechanisms as possible. Not only are these studies beneficial to the researcher's world, but also they started delivering concrete results: in very recent times some companies such as the French start-up Pasqal released quantum devices for commercial use. The peculiarity of these devices, apart from the fact that they are comprised of 196 qubits, is that they are precisely based on Rydberg atoms!
As pointed by Wu et al. (2021), one important future direction for neutral atom-based quantum computation and simulation is to improve fidelity for quantum state manipulation. Continued improvements in fidelity, together with persistent efforts to increase the scale

of qubit arrays, would enable to achieve a Rydberg quantum computer/simulator that can outperform the best classical computer in terms of certain computing tasks, thanks to achievements such as the realization of high-fidelity quantum gates, or the simulation of quantum spin models which would represent a starting point towards continuous successes in quantum computing with Rydberg atoms.
This is only one of the many directions that research on quantum computing is taking. What is clear is that academic interest is proliferating, and this can only be beneficial in terms of new discoveries which will bring us one step closer to the revealing the uncanny secrets of quantum computation.

6 Future Research Directions

One of the purposes of this thesis was to stress the differences that exist between classical computing models and theoretical quantum computing ones, in favour of the latter with respect to particularly complex tasks. As a matter of fact, as it appeared clear from the theory, quantum algorithms seem to outperform the classical counterpart.
However, comparing these two approaches is still a hard task as of today. This is because the development of the proper quantum hardware is still far away in the future. So, a first limitation of this study is the fact that this comparison, both involving the knn vs. Qknn algorithms and the other three algorithms may not be completely reliable, especially due to the fact that a hybrid approach comprising the use of a classical hardware was employed. A real turning point in the establishment of quantum supremacy will be when the comparison will be enabled by the use of a pure quantum computer, that is, when the NISQ era will be left behind in favour of the fault tolerant era.
Nonetheless, many advances that involve NISQ devices are being made. One example we have seen is represented by neutral atom-based devices. it will be interesting to follow all the latest developments in this area, as this is a fast-growing field of research, with many academic papers continuously emerging. Despite all the advances involving quantum computation with Rydberg atoms, which significantly aided the main problems arising today from a hardware perspective, quantum error correction remains a long-term goal, considerably out of reach even with scaled up systems based on current platforms. Only time will say whether all this progress will eventually lead to the fault-tolerant devices researchers are yearning for.
A drawback for the QkNN is its variability (or noise). The same solution from Park, Petruccione, et al. (2019a) can be applied here, according to Blank et al. Since they observe the expectation value resulting from the quantum circuit this ‘opens up a possibility to apply error mitigation techniques to improve the accuracy in the presence of noise without relying on quantum error correcting codes”.

7 Conclusions

Quantum computers are to scientists what the siren's song was to Ulysses: an alluring call, promising a future so bright and plentiful that no man can resist it (Vestergaard 2020). My intention was to analyse this exciting new field in detail.
In particular, this thesis set different goals to be accomplished: among all, showing quantum supremacy, which is, in its own right, a worthy goal (Preskill 2018).
This main task was successfully achieved through the implementation of a Qknn algorithm able to classify instances of a German credit dataset based on the risk of default, with an AUC score (Area Under the Curve) seemingly equal to 1 with respect to true positive classifications. This was possible after performing pre-processing methods on the dataset. The comparison with the classical knn evidences the striking superiority of the former with respect to the latter method.
This result allowed to optimize an already existing quantum knn algorithm, where the attention of the developer was mostly diverted to the construction and benchmarking of the Qknn, therefore leaving out the data science aspect with respect to the German credit data.
Another accomplishment in this sense was the ability to prove the statement from Wittenbach et al. (2020): "pre-processing is a necessary step to solving the core modelling financial tasks". It was reasonable to expect a correspondence of this statement also in the quantum reign, which has been confirmed by the results obtained. It may be possible to extend this assertion to the observation that pre-processing is a necessary step to solving any machine learning task in general, and therefore the same should apply to quantum computing.
However, more research in this sense is needed to either confirm or reverse these results.
Moreover, such findings allow to prove the effectiveness of combining quantum programs with classical hardware. As a matter of fact, since larger problems (especially concerning machine learning) are substantially intractable on classical computers, there is much interest in solving them efficiently using quantum hardware (Rosenberg et al. 2015). However, the research towards this goal is still at an early stage, named NISQ era: quantum computers are still very noisy as of today, and do not allow to completely rely on quantum mechanisms in the most efficient way yet.
Despite the promising results, the hardware problem still may prevent reliable results, due to the fact that it is only possible to run the quantum algorithm on a classical hardware, using a quantum simulator available online. It may be useful to understand such limitations more in depth, in order to better address them in future and more specialized studies.
Sergioli et al. proposed a resolution: up to now, they argue, the best practice is to use hybrid solutions, which make use of quantum formalism to obtain significant benefits in the classical computation contexts, which is what is being done now by researchers and also the method that was carried out in this thesis.
It appears clear that, despite the hype that has accompanied quantum researchers in the last few decades, they are also facing a big paradox: many quantum algorithms are currently available, but the challenges related to the development of the quantum hardware are hindering the possibility to actually exploit their potential, albeit the theoretical proofs.
As of today, the best practice is to make use of quantum simulations, which aim at using a synthetic quantum system to emulate a real-world many-body physical problem based on a model Hamiltonian. Recent years have witnessed great successes in simulating many-body physics on various platforms, among which neutral atom system with Rydberg interactions appears to be a promising choice (Wu et al. 2021). The deployment of the Rydberg atoms to shape a new kind of quantum computers seems to be very promising due to the potential

ability to overcome the main problem related to the present quantum devices, namely Random Error Correction, while also providing scalability, which constitutes another central challenge for implementing large-scale quantum computation.
Error correction happens because qubits contain probabilities, which make them especially prone to measurement errors and noise from the environment, as opposed to bits. This is why this problem is regarded as one of the main concerns which need to be addressed in order for the research in the field of quantum computing to keep progressing. In this sense, this thesis provided evidence that the results achieved by NISQ devices based on Rydberg atoms are useful towards a shift in the present computational paradigm, since they hold the promise of scalable quantum devices with small dependence on the temperature of the quantum hardware, thereby reducing system noise and improving the quality of the quantum operations performed.
Another finding is that, although the actual implementation of pure quantum algorithms is still years away, as they would require ideal digital quantum processors, it is important to keep exploring the capabilities of currently available quantum devices, albeit imperfect and with a relatively modest number of qubits. As a matter of fact, these devices have recently shown that their computing capabilities can outperform classical supercomputers for specific computing tasks. What is clear is that academic interest is proliferating, and this can only be beneficial in terms of new discoveries which will bring us one step closer to revealing the uncanny secrets of quantum computation.
In this sense, the words of the roman philosopher Lucius Annaeus Seneca seem a prophesy: "The time will come when diligent research over long periods will bring to light things which now lie hidden. A single lifetime, even though entirely devoted to the sky, would not be enough for the investigation of so vast a subject... And so this knowledge will be unfolded through long successive ages. There will come a time when our descendants will be amazed that we did not know things that are so plain to them... Many discoveries are reserved for ages still to come, when memory of us will have been effaced. Our universe is a sorry little affair unless it has something for every age to investigate... Nature does not reveal her mysteries once and for all".

Bibliography

[1] Afham, Afrad Basheer, and Sandeep K. Goyal (2020). Quantum k-nearest neighbor machine learning algorithm.

[2] Afrad Basheer, A. Afham, Sandeep K. Goyal. "Quantum k nearest neighbors algorithm" (2020)

[3] Andrew Steane. "Multiple-Particle Interference and Quantum Error Correction" (1996)

[4] Cédric Bény. "Deep learning and the renormalization group" (2013)

[5] Christa Zoufal, Aurélien Lucchi, Stefan Woerner. "Quantum Generative Adversarial Networks for learning and loading random distributions" (2019)

[6] Daniel J. Egger, Ricardo García Gutiérrez, Jordi Cahué Mestre, Stefan Woerner. "Credit Risk Analysis using Quantum Computers" (2019)

[7] Francesco Rundo, Francesca Trenta, Agatino Luigi Stallo, Sebastiano Battiato. "Machine Learning for Quantitative Finance Applications: A Survey" (2019)

[8] Gili Rosenberg, Poya Haghnegahdar, Phil Goddard, Peter Carr, Kesheng Wu, Marcos López de Prado "Solving the Optimal Trading Trajectory Problem Using a Quantum Annealer" (2016)

[9] Giuseppe Carleo, Yusuke Nomura, Masatoshi Imada. "Constructing exact representations of quantum many-body systems with deep neural networks" (2018)

[10] Giuseppe Sergioli, Roberto Giuntini, Hector Freytes ."A new quantum approach to binary classification" (2019)

[11] Hamed Ghoddusi, Germán G. Creamer, Nima Rafizadeh. "Machine Learning in Energy Economics and Finance: A Review" (2019)

[12] J. B. Heaton, N.G. Polson, J. H. Witte. "Deep Learning in Finance" (2016)

[13] Jason Wittenbach, Brian d'Alessandro, Bayan Bruss. "Machine Learning for Temporal Data in Finance: Challenges and Opportunities" (2020)

[14] John Preskill. "Quantum Computing in the NISQ era and beyond" (2018)

[15] Kerstin Beer, Dmytro Bondarenko, Terry Farrelly, Tobias J. Osborne, Robert Salzmann, Daniel Scheiermann, Ramona Wolf. "Training deep quantum neural networks" (2020)

[16] Kok, Daniël (2020). "Documentation for Qiskit Quantum kNN: A pure quantum knn classifier for a gated quantum computer." In: Read the Docs. url: https://qiskit-quantum-knn.readthedocs.io/

[17] Kok, Daniel (2020). Building a quantum kNN classifier with Qiskit: theoretical gains put to practice. Radboud University.

[18] Lirong Gan, Huamao Wang, Zhaojun Yang "Machine learning solutions to challenges in finance: An application to the pricing of financial products" (2020)

[19] Loïc Henriet, Lucas Beguin, Adrien Signoles, Thierry Lahaye, Antoine Browaeys, Georges-Olivier Reymond, Christophe Jurczak. “Quantum Computing with Neutral Atoms” (2020)

[20] Nathan Wiebe 2020 New J. Phys. 22 091001

[21] Nathan Wiebe, Ashish Kapoor, Krysta M. Svore. “Quantum Algorithms for Nearest-Neighbor Methods for Supervised and Unsupervised Learning” (2014)

[22] Patrick Rebentrost, Masoud Mohseni, Seth Lloyd. “Quantum Support Vector Machine for Big Data Classification” (2014)

[23] Philippe Bracke, Anupam Datta, Carsten Jung and Shayak Sen. ¡¡Machine learning explainability in finance: an application to default risk analysis” (2019)

[24] Seunghyeok Oh, Jaeho Choi, Joongheon Kim. “A Tutorial on Quantum Convolutional Neural Networks (QCNN)” (2020)

[25] Vestergaard Skannrup, R. (2020). Rydberg atoms for quantum simulation. Technische Universiteit Eindhoven.