Feature Importance & Explainability in Quantum Machine Learning

A Final Year Project submitted
in Partial Fulfilment of the Requirements
for the Degree of

Bachelors of Science
in
Data Science and Analytics

by

Luke Power
(Student No. 120371316)

Supervised by Dr. Krishnendu Guha

College Of Science, Engineering, and Food Science
Scool of Computer Science & Information Technology
University College Cork
April 23, 2024

Abstract

Many Classical Machine Learning (ML) models are referred to as black-box models, providing no real insights into why a prediction is made. Feature importance and explainability are important for increasing transparency and trust in ML models. With quantum computing's unique capabilities, such as leveraging quantum mechanical phenomena like superposition, this can be combined with ML techniques to create the field of Quantum Machine Learning (QML), and such techniques may be applied to QML models. This project explores the insights of feature importance and explainability in QML compared to Classical ML models. Utilizing the widely recognized Iris dataset, we examine classical ML algorithms— Support Vector Machine (SVM) and Random Forests, against hybrid quantum counterparts, implemented via IBM's Qiskit platform: the Variational Quantum Classifier (VQC) and Quantum Support Vector Classifier (QSVC). This project includes implementing Deutsch-Jozsa and Grover's algorithms, providing demonstrations of the 'quantum advantage'. This project aims to provide an in-depth comparison of the insights generated in ML by employing permutation and leave-one-out feature importance methods, alongside ALE (Accumulated Local Effects) and SHAP (SHapley Additive exPlanations) explainers.

ucc

Coláiste na hOllscoile Corcaigh, Éire
University College Cork, Ireland

Declaration of Originality

In signing this declaration, you are conforming, in writing, that the submitted work is entirely your own original work, except where clearly attributed otherwise, and that it has not been submitted partly or wholly for any other educational award. I hereby declare that:

- This is all my own work, unless indicated otherwise, with full and proper accreditation.
- With respect to my own work: none of it has been submitted at any educational institution contributing in any way to an educational award.
- With respect to another's work: all text, diagrams, code, or ideas, whether verbatim, paraphrased or otherwise modified or adapted, have been duly attributed to the source in a scholarly manner, whether from books, papers, lecture notes or any other student's work, whether published or unpublished, electronically or in print.

Degree:	B.Sc. Data Science and Analytics
Title:	Feature Importance and Explainability in Quantum Machine Learning
Candidate:	Luke Power (120371316)
Signature of Student:	_____
Date:	April 23, 2024

Acknowledgements

I would like to thank my supervisor Dr Krishnendu Guha, for his help, guidance and for setting the wheels of this project in motion. I would also like to give a heartfelt thank you to my friends, family, and classmates for their unwavering support, criticism, and craic that made this project possible.

Contents

List of Figures

List of Tables

List of Abbreviations

QML Quantum Machine Learning i
ML Classical Machine Learning i
SHAP SHapley Additive exPlanations 8
SVM Support Vector Machine i
SVC Support Vector Classifier 43
VQC Variational Quantum Classifier i
COBYLA Constrained Optimization By Linear Approximation optimizer 42
SLSQP Sequential Least SQuares Programming optimizer 42
PDP Partial Dependency Plot 8
ALE Accumulated Local Effects 8
XAI Explaiable AI 30
XML Explainable ML
QSVC Quantum Support Vector Classifier i
LOO Leave-One-Out 50
XQAI Explainable Quantum Artificial Intelligence 73
QSVM Quantum Support Vector Machine 27

1

Introduction

1.1 | Motivation

This project seeks to provide an understanding of quantum computing, and how it takes quantum mechanical phenomenon and integrates with classical machine learning to create the domain of QML.
Given how many current ML models are considered 'black box' or 'opaque models', meaning that there is no real way to truly understand how or why an output is generated, there is a rise in the necessity to implement ways of explaining a models inner workings and rationalising its outputs. Currently there are several methodologies to determine a models' most important features and explainability of individual predictions, and this project seeks to apply these methods to QML models and make an in-depth comparison of the results.

1.2 | Project Summary & Document Outline

1.2.1 | Project Summary

The main objective of this project is to ascertain what new inferences could be made about a dataset via QML when compared to ML and attempt to explain how these differences come about. For this project, we are interested in using classical data, but quantum algorithms, in another way; the top-right quadrant

of the image 1.1. This was achieved by using a common ML method, building an equivalent QML model, and making inferences about the data by measuring feature importance - i.e. how important is a feature in making an accurate classification by using feature omission and permutations to quantify the differences, and also applying explainability methods to the models, which can provide greater insights into a singular prediction.
As of writing, this paper is the first to provide a comprehensive comparison of the results of applying machine learning explainability to QML on multi-class data.
This project will also delve into the theory behind quantum computing and the implementations of some basic quantum computing models do give clear demonstrations of the 'quantum advantage'.

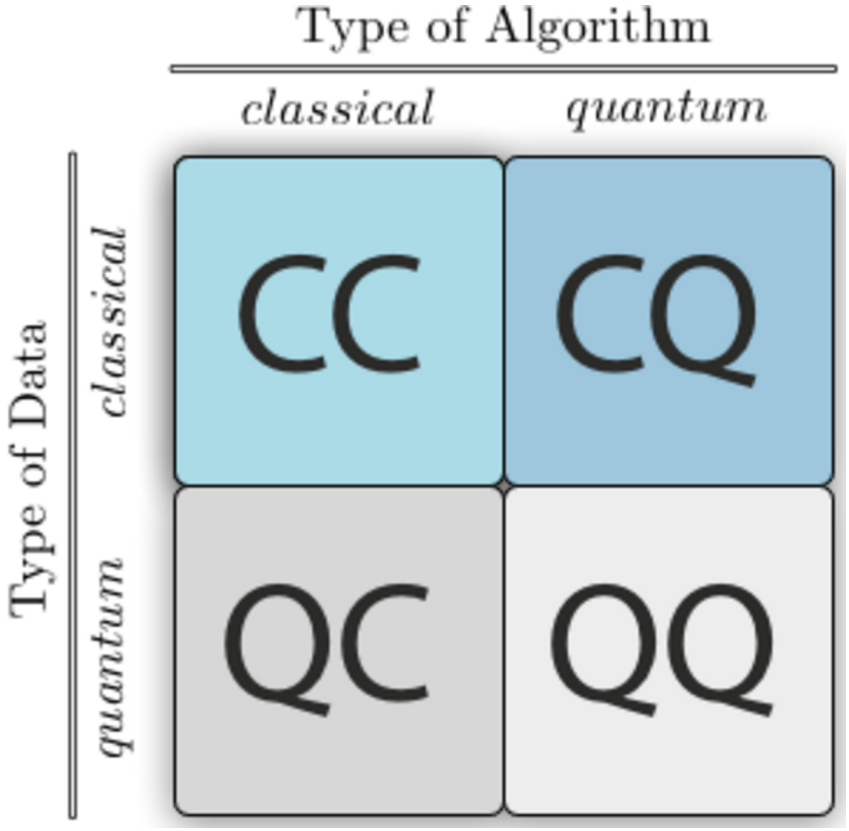

Figure 1.1: Classical Data on Quantum Algorithms. We are using Classical data with Quantum algorithms. Image adapted from (1).

1.2.2 | Report Outline

The rest of the Introduction will provide a general overview of the project; the main objectives as well as some theoretical background on the main concepts of quantum computing and ML and a brief overview of the primary technology and tools that were used over the course of completing this project.
Chapter Two, Theoretical Background, will provide a more in-depth exploration of the machine learning models used, and explore the theory and mathematics behind implementing the QML models. This chapter will also demonstrate the implementation of quantum algorithms that provide some examples of quantum advantage in some computing tasks. The end of this chapter will also contain a literature review of the current research and papers in the field.
Chapter Three, Implementation goes through the actual implementation of the models on the Iris dataset and provide a comparison on the models' predictive performance, in the subsequent chapter, Results, the results of the implementations of the Feature Importance and Explainability methods are revealed, analysed and compared.
The Conclusions chapter will provide a personal reflection on the undertaking of this project, reviewing the experience in completing this project, how it was completed along with the future plans of the project.

1.3 | Methodology

The approach to this project involved partaking in a course on quantum computing, familiarization with concepts of quantum theory, utilization of IBM's Qiskit platform which provides access to quantum hardware and building the necessary circuits and models. After several exploratory scripts and implementing algorithms which can provide proven demonstrations of the quantum advantage. After this, the initial QML models were built. The basic models were then expanded with different permutations of optimisers and ansatzes, to gain a level of accuracy comparable to the classical counterparts. After some initial model comparisons, we can then implement several methods of feature importance and explainability to both sets of models and compare the results. An overview

of this process is displayed in the flow chart 1.2

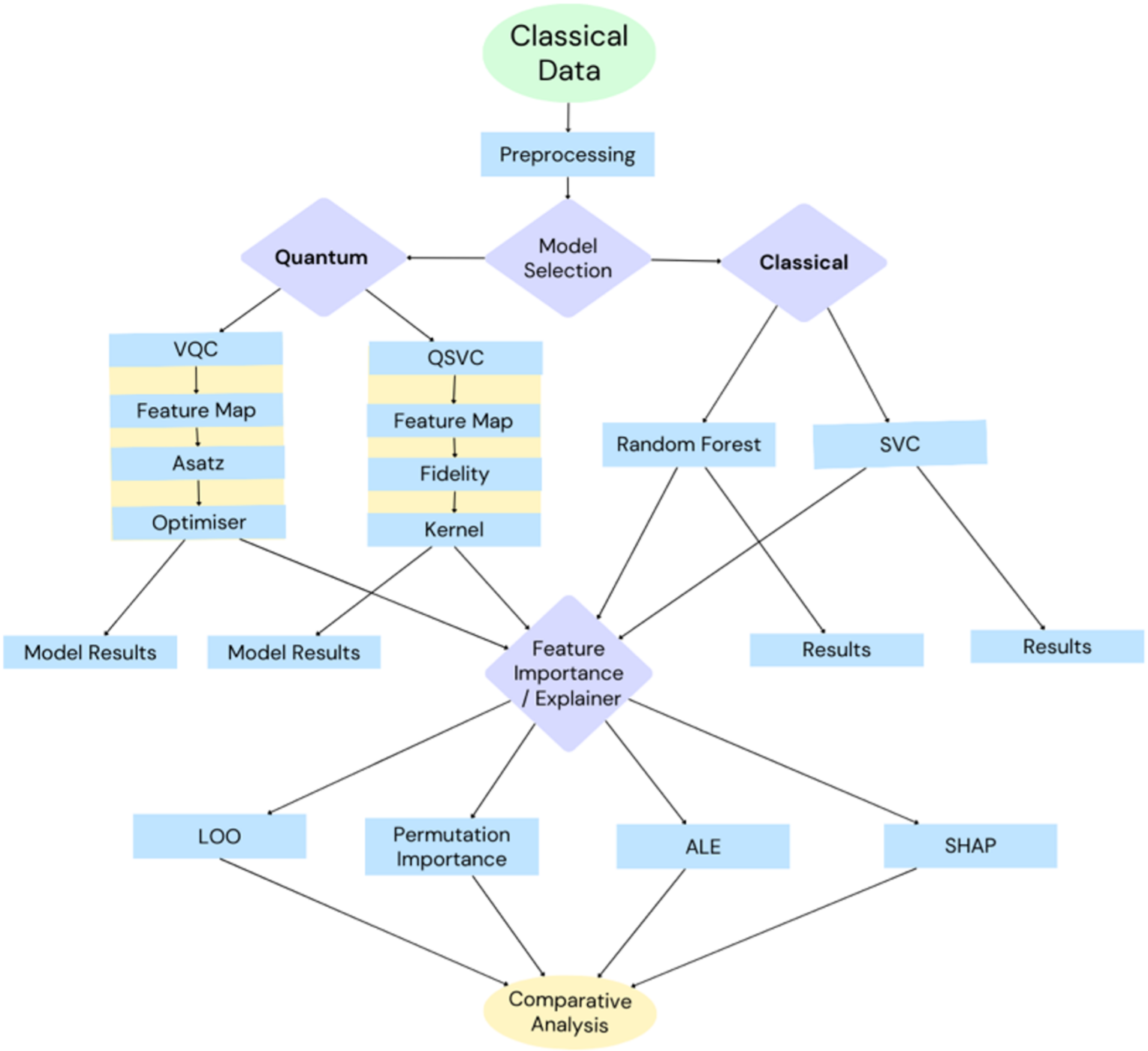

Figure 1.2: Data Flow of the project, centring around the application of Feature Importance and Explainability.

1.4 | Concepts and Technologies

1.4.1 | Machine Learning

Machine Learning (ML) is a subsection of computer science and **AI!** (**AI!**) that utilizes statistical models to make inferences about the underlying patterns in a dataset and build algorithms that can make accurate predictions or classifications when presented with new, unseen data, without the need of explicit programming. (2).

Machine Learning can be broken down into three main 'flavours':

- **Supervised Learning:** Models are trained on a labelled dataset, learning to predict outcomes for new data based on this training.
- **Unsupervised Learning:** Algorithms work with unlabeled data to find structure or patterns, without explicit instructions on processing it.
- **Reinforcement Learning:** An agent learns to make decisions by taking actions in an environment to achieve some objectives, learning from rewards or penalties.

Supervised learning is the most widely used form of machine learning, and is the underlying method that was followed in the course of this project. Supervised learning involves feeding a model some labelled data which it will train its parameters on to minimise some cost function.

1.4.2 | Qubits and Superposition

QML is a branch of quantum computing that uses phenomena seen in quantum physics, such as superposition and entanglement which allows for multiple computations to be conducted simultaneously.(3) The '**qubit**' is the quantum equivalent of the bit in classical computing. What is special about the qubit is that unlike bits, which can only ever have the states one or zero, a qubit can be in both states simultaneously. This phenomenon is known as **superposition**.

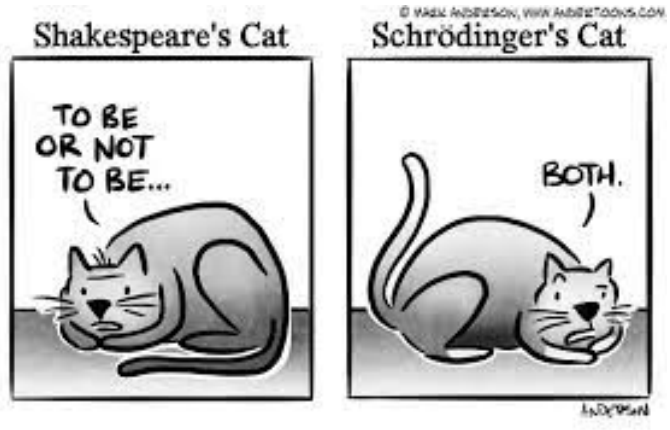

Figure 1.3: Schrodinger's Cat is a thought experiment to help conceptualize superposition. Image adapted from (1).

Schrodinger's Cat is a thought experiment to describe the idea of superposition, which is explained by thinking of a cat that is placed in a container containing a radioactive element that has a 50% probability of releasing and killing the cat. It is said that until the box is opened and the state of the cat is observed (the state being either alive or dead), the cat is said to be in a state of superposition of both alive and dead at the same time(4). A more feline-friendly way to think about superposition is a coin that has been tossed in the air and is 'simultaneously heads and tails'. The ability to have data in superposition can allow for quantum algorithms to perform computations on data in multiple states simultaneously. This can lead to exponential speedups in computation and the ability to perform tasks on highly complex data.

1.4.3 | Feature Importance & Explainable ML

Although ML models have provided great utility to many people and companies across a multitude of sectors, many of them are often regarded as 'black boxes'. The data is passed in and a result is returned, the actual inner workings of the model are not known to the user. This leads to an investigation into 'Explainable AI' XAI and 'Explainable ML', XML, which aims to shine a light on the inner workings of a model. Reasons for this could be in a denied loan ap-

plication, and the ability to explain clearly why the model decided to deny the application, to a doctor or patient, explaining why a model predicted a certain medical diagnosis, or perhaps what settings are causing a machine to malfunction. Good explainability can also be utilised by data scientists to provide sanity checks or debugging on the model (what if someone's name was the reason behind the denied loan?). (5) (6)

1.4.4 | Methods for Assessing Feature Importance:

- **Leave-One-Out (LOO) Feature Importance**: This method assesses the impact on model performance when each feature is individually left out from the model. A significant decrease in performance upon leaving out a feature implies increased importance.
- **Permutation Feature Importance**: This method assesses feature importance by measuring the increase in the model's prediction error after permuting the feature's values, which breaks the relationship between the feature and the outcome.(7)
- **SHapley Additive exPlanations (SHAP) Values**: Based on cooperative game theory, and Shapely values which in short, calculate how the winnings should be split among the team players, based on predictions on the outcomes had the players played independently or some permutations of them. This can be applied to ML through SHapley Additive exPlanations (SHAP) (8) values to explain the prediction of an instance by computing the contribution of each feature to the prediction. SHAP values offer both global and individual explanations of feature importance. (9)
- **Accumulated Local Effects (ALE)**: Used to interpret the influence of features in a model, especially in the context of prediction tasks. Unlike other feature importance metrics like in a Partial Dependency Plot (PDP), Accumulated Local Effects (ALE) values focus on the local effects of features, by splitting up the feature space into a number of 'windows' and

calculating the average change in the model's predictions over the intervals of the feature's values. (10)

1.4.5 | Tech & Tools

The primary tools used for this project are as follows:

- **IBM Qiskit**
  - Qiskit is an open-source Software Development Kit (SDK), managed by IBM.
  - It provides tools for users to create and run quantum computing programs on actual quantum hardware or simulators.

  (11)
- **Python**
  - Python is a high-level, interpreted programming language that is compatible with IBM Qiskit.
  - It is highly versatile and commonly used in data science, featuring numerous libraries and an easy-to-follow syntax.
- **Jupyter Notebook**
  - Jupyter Notebooks are used for their integrated coding and documentation environment.
  - They allow for code execution in segments along with markdown sections, enabling reproducible results and a streamlined reporting process.

The main dependencies in the project can be found in table 1.1.

Table 1.1: Project Git Dependency

Package Name	**Version**	**License**
attrs	21.4.0	MIT
brotlipy	0.7.0	MIT
configparser	6.0.1	MIT
cryptography	37.0.1	Apache-2.0 and others
cython	0.29.32	Apache-2.0
dl	0.1.0	BSD-2-Clause
docutils	0.18.1	BSD-2-Clause
htmlparser	0.0.2	BSD-2-Clause
importlib-metadata	7.0.1	Apache-2.0
ipython	8.12.3	BSD-2-Clause and BSD-3-Clause
ipywidgets	7.6.5	BSD-2-Clause and BSD-3-Clause
jinja2	3.1.3	BSD-2-Clause and BSD-3-Clause
jnius	1.1.0	MIT
keyring	23.4.0	MIT
lockfile	0.12.2	MIT
lxml	4.9.1	BSD-2-Clause and BSD-3-Clause
matplotlib	3.5.2	PSF-2.0
numpy	1.23.5	BSD-2-Clause
ordereddict	1.1	MIT
pandas	1.4.4	BSD-2-Clause and BSD-3-Clause
pillow	10.2.0	HPND
pillow	9.2.0	HPND
protobuf	4.25.3	BSD-3-Clause
pyopenssl	22.0.0	Apache-2.0
pyopenssl	24.0.0	Apache-2.0
qiskit	0.44.2	
qiskit-aer	0.12.2	
qiskit-ibmq-provider	0.20.2	Apache-2.0
qiskit-terra	0.25.2.1	
railroad	0.5.0	MIT
scikit-learn	1.4.0	BSD-2-Clause and BSD-3-Clause
seaborn	0.11.2	BSD-2-Clause and BSD-3-Clause
sphinx	5.0.2	BSD-2-Clause
thread	0.1.3	BSD-2-Clause and BSD-3-Clause
toml	0.10.2	MIT
tornado	6.1	Apache-2.0
xmlrpclib	1.0.1	
zipp	3.8.0	MIT

2

Theoretical Background

2.1 | Supervised Machine Learning

Supervised ML consists of first gathering the labelled data, which the ML model will be trained and tested on. For classification tasks, this often consists of collecting data and filling in a number of feature vectors and a classifying label. Examples of this could be spam mail, where the features could consist of time, sender, and content keywords, among other possible predictors, and the classification label could be a simple 1 = spam, 0 = not spam. Another example could be a collection of photos, along with labels, classifying the image subjects, for example, differentiating cats and dogs. The feature vector in this regard would be the values of the image pixels' RGB and opacity values. (2)

There are a number of ML algorithms, but the one used for this project was the SVM.

2.1.1 | Support Vector Machines

Support Vector Machine are a powerful ML tool that can be used for both regression and classification but is more often used in classification tasks. A SVM works by finding an optimal hyperplane that separates the data points with the best probability of correctly separating classes where there is overlap by maximising the margins of the hyperplane.

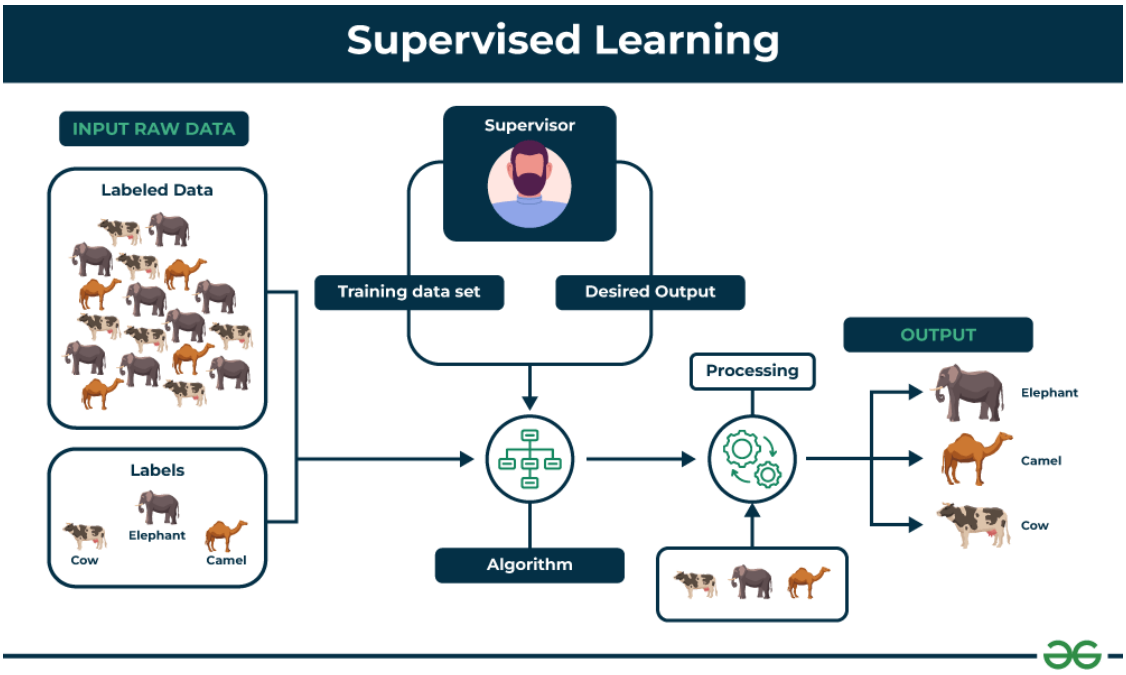

Figure 2.1: Supervised ML Process. Image adapted from (12).

The separating hyperplane $f(x)$, defined by:

$$f(x) = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \ldots + \beta_p X_{pi}$$

for a given point (x_i, y_i) is so that:

$$f(x_i) = \begin{cases} > 0, & \text{if } y_i = 1 \\ < 0, & \text{if } y_i = 0 \end{cases}$$

And we can use this as a classification rule for the data; classifying x^* with respect to $(sign) f(x^*)$

The objective of the SVM is to calculate the optimal hyperplane to separate the data defined by

$$M = argmax_{\{\beta,\epsilon,m\}} \{y_i(\beta_0 + \beta_1 x_{1i} + \ldots + \beta_p x_{pi}) \geq m(1 - \epsilon_i\}$$

Subject to:

$$\sum \beta_j^2 = 1, \quad \epsilon_i \geq 0, \quad \sum_{i=1}^{N} \epsilon_i \leq c$$

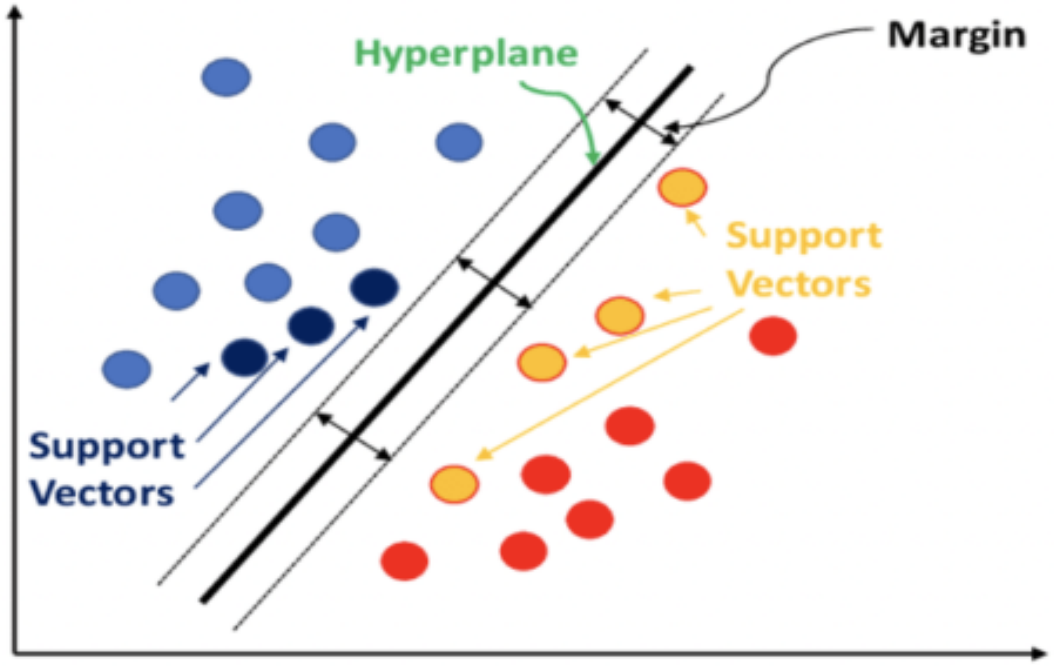

Figure 2.2: Support Vector Machine. Image adapted from (13).

For some tuning parameter $c > 0$, also known as a 'budget' which is the allowed rate for margin violation, and ϵ_i or 'slack variable', where:

$$\epsilon_i > 0 \quad \text{implies point is classified on the wrong side of the margin,} \tag{2.1}$$

$$\epsilon_i > 1 \quad \text{implies point is classified on the wrong side of the hyperplane.} \tag{2.2}$$

In real life, data is often non-linearly separable, and the formula for the hyperplane needs to be expanded to account for non-linearity by incorporating functions known as kernels on the predictors, as seen in figure 2.3.

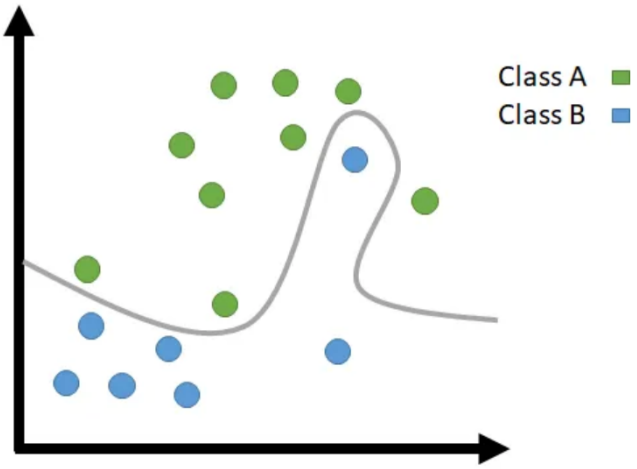

Figure 2.3: Non-Linear Support Vector Machine. Image adapted from (14).

This introduction of kernel function is known as the 'kernel trick', which updates the formulas for the hyperplane as follows:

$$M = argmax_{\{\beta,\epsilon,m\}}\{y_i(\beta_0 + \sum_{j=1}^{P}(\beta_j x_{i_i} + \alpha_j x_{j_i}^2) \geq m(1-\epsilon_i\}\forall i$$

Subject to:

$$\sum \epsilon_i \leq c, \quad \epsilon_i \geq 0, \quad c > 0, \quad \sum \beta_j^2 + \sum_j \alpha_j^2 = 1$$

We can then establish that an SVM uses kernel functions

$$f(x) = \sum_{h=1}^{H} \alpha_h K_h(x, x') + \alpha_0$$

$$= \alpha_0 + \sum_{i \in \delta} \alpha_i k(x, x_i)$$

Where δ is the set of Support Vectors. (15)

2.1.2 | Random Forest

A random forest is an ensemble model that samples from a number of different decision trees and aggregates the output of the trees. This is due to the fact that while growing a tree, the optimal cutoff is on a split-by-split basis. This greedy approach can result in a sub-optimal tree. This strategy aims to reduce this effect, reduce variability and to preserve a low bias. In order to aggregate the results from the trees requires resampling. For example: using a bootstrap resampling scheme:

$$\hat{f}_B^*(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}_b^*(x)$$

Where B represents the number of bootstrap samples and $\hat{f}_b^*(x)$ represents an estimate from each bootstrap sample. The notation $\hat{f}_B^*(x)$ denotes the averaged estimate across all bootstrapped samples.(16)

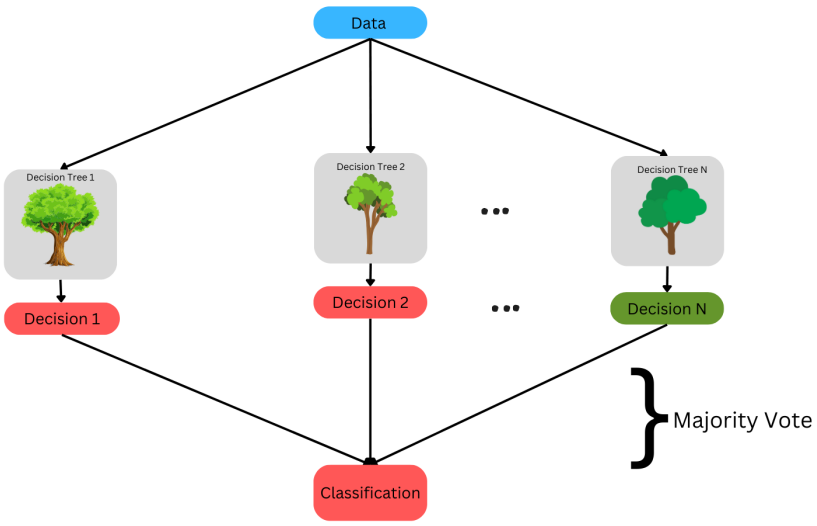

Figure 2.4: How a Random Forest Classifier Works

2.2 | Quantum Machine Learning

Quantum Machine Learning can enhance machine learning by leveraging quantum mechanics to identify patterns in data that classical computing struggles with. Research suggests that quantum computers might excel in tasks like pattern recognition and optimization due to their ability to process complex, counter-intuitive patterns. Although some literature indicates that quantum-enhanced machine learning could unlock new paradigms in data analysis and problem-solving, the biggest advantage that is expected of quantum computing is the speed-ups that can be expected, particularly through quantum computers ability to handle large matrix operations. (17)(18)

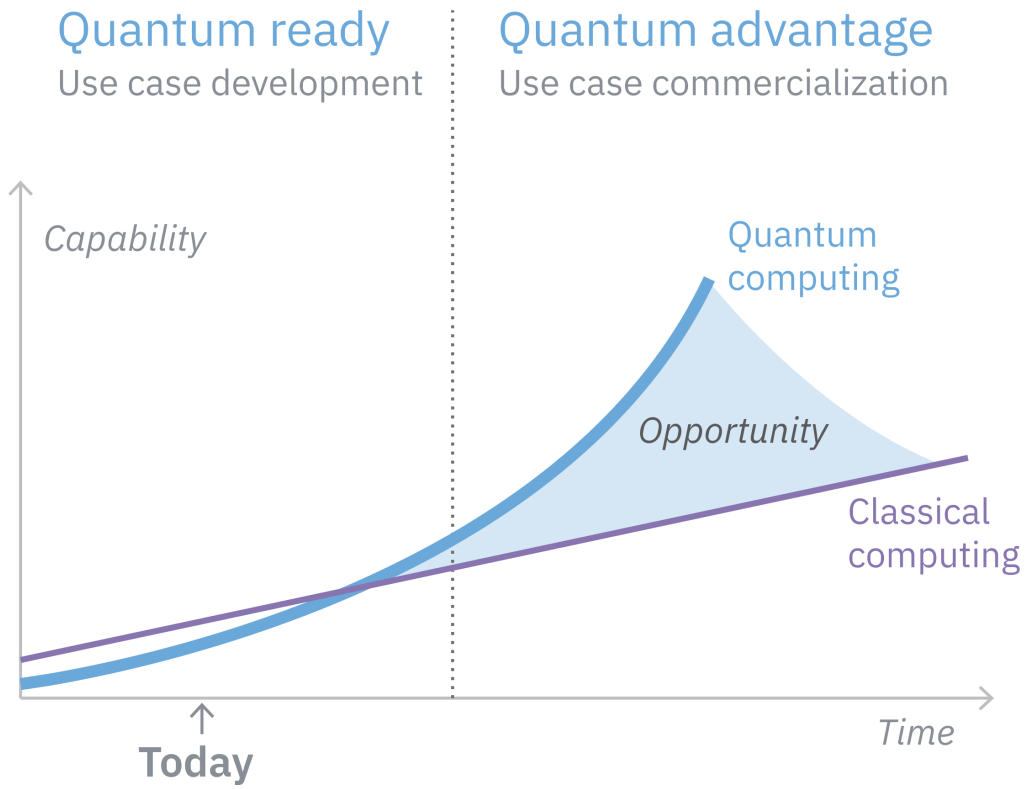

Figure 2.5: Quantum Advantage. Image adapted from (19).

The core advantage of quantum computing lies in its ability to process infor-

mation in ways that classical systems cannot match, offering exponential speed-ups for certain types of problems. For example, quantum algorithms have been demonstrated to outperform classical counterparts in oracle-based problems, showing that a significant quantum advantage emerges even in existing noisy systems.(20) Furthermore, quantum technology can revolutionize how we learn from experiments, as shown in a study where quantum machines learned from exponentially fewer experiments than required by conventional means, leading to dramatic reductions in the number of necessary experiments.(18) Quantum machine learning also benefits from the ability to process atypical patterns produced by quantum systems, offering potential speed-ups in learning tasks.(17) Additionally, the use of quantum-enhanced feature spaces in machine learning can provide advantages in solving classification problems where classical feature spaces become computationally prohibitive (21).

2.2.1 | The Qubit & Superposition

As mentioned in the introduction, quantum computing utilises qubits which differ from classical bits by being able to exist as both zero or one simultaneously in a state called superposition.
This means that n bits can represent n^2 possible states, but qubits can be in 2^n states simultaneously. Superposition allows operations to be applied on all the possible states simultaneously, and combined with interference effects which can cancel out 'wrong' results can provide results that cannot be achieved with classical computing. (22). Quantum computing also relies on linear algebra to mathematically represent states as vectors. For example, the classical bit's states of zero or one would be represented as the vectors (using Dirac notation):

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

In quantum computing, these certain states would be referred to as the two 'pure states'. A qubit, on the other hand, would be in a combination of the two states simultaneously, in a phenomenon known as superposition taking on coefficients α and β which represent the probabilities of being in their respective states. A

qubit in a state $|\psi\rangle$ would be written as

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, alpha, \beta \in C$$

Where C represents complex numbers; numbers that have an imaginary component i where $i = \sqrt{-1}$. It is not possible to measure α or β, as the observation or measurement of a qubit will cause it to collapse into one of its pure states. However since α and β are still probabilities their total magnitude must be equal to one.

$$|\alpha|^2 + |\beta|^2 = 1$$

(23)

A superpositioned state can be achieved by passing qubits through a unitary operation known as a Hadamard gate.

- Applied to $|0\rangle$, the Hadamard gate produces the state $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, an equal superposition state where the probabilities of measuring the qubit in state $|0\rangle$ or $|1\rangle$ are both 50%.

- Applied to $|1\rangle$, it produces the state $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, another equal superposition state, but with a phase difference between the $|0\rangle$ and $|1\rangle$ components.

(22)

To better understand the state of a qubit we use what is called a 'Bloch Sphere', depicted in figure 2.6. A qubit's state can be represented geometrically on a Bloch sphere described with the formula:

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle$$

In this representation, $|0\rangle$ and $|1\rangle$ are the standard basis states of the qubit. The angles θ and ϕ are real numbers, corresponding to the point on the Bloch sphere, where θ ranges from 0 to π and ϕ ranges from 0 to 2π. The state $|\psi\rangle$ is

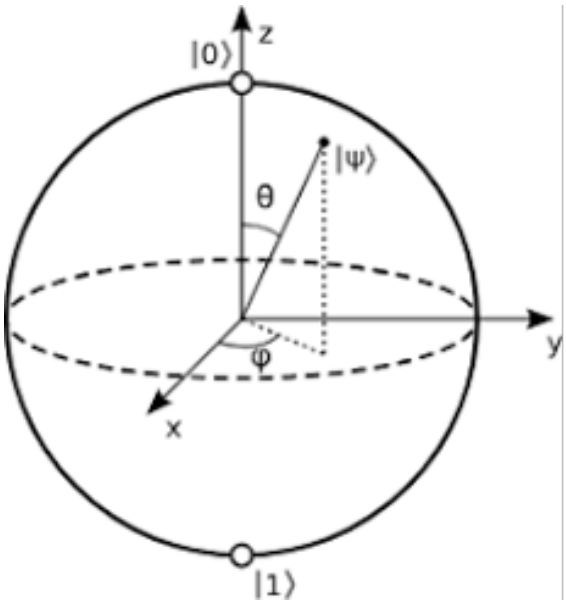

Figure 2.6: 3D representation of a qubits' possible states.

thus a point on the surface of the sphere, where the north pole corresponds to $|0\rangle$ and the south pole to $|1\rangle$. (24)

2.2.2 | Deutsch-Jozsa Algorithm

Deutsch-Jozsa Algorithm was initially proposed by Deutsch and Jozsa in 1992 with improvements made by Cleve, Ekert, Macchiavello, and Mosca in 1998, is one of the first algorithms to demonstrate the quantum advantage(25).
The algorithm addresses a problem involving an 'oracle'; a hypothetical black-box device capable of processing inputs and generating outputs, but its internal workings are inaccessible. This oracle accepts strings of boolean values (1 or 0) as inputs and produces outputs following one of two distinct properties: it either returns a constant value (the same output) for any input or yields a balanced output (equal numbers of zeros and ones across all inputs). The goal is to determine what property the oracle demonstrates using the minimal number of queries possible. Classically this would take at least two queries(if there are two different outputs) but up to $2^{n-1} + 1$ queries in the worst case,(if the first

two outputs are zero, we will need to try again to check if the next one is different, etc.) but in the quantum solution, by utilising superposition and quantum interference effects, requires only one(22).
Figure 2.7, shows a diagram of the Deutsch-Jozsa Algorithm on a quantum circuit with a size of five qubits, with the algorithm in 1 showing the steps of the algorithm. Running this circuit and taking the measurements results in the histogram in figure 2.8.

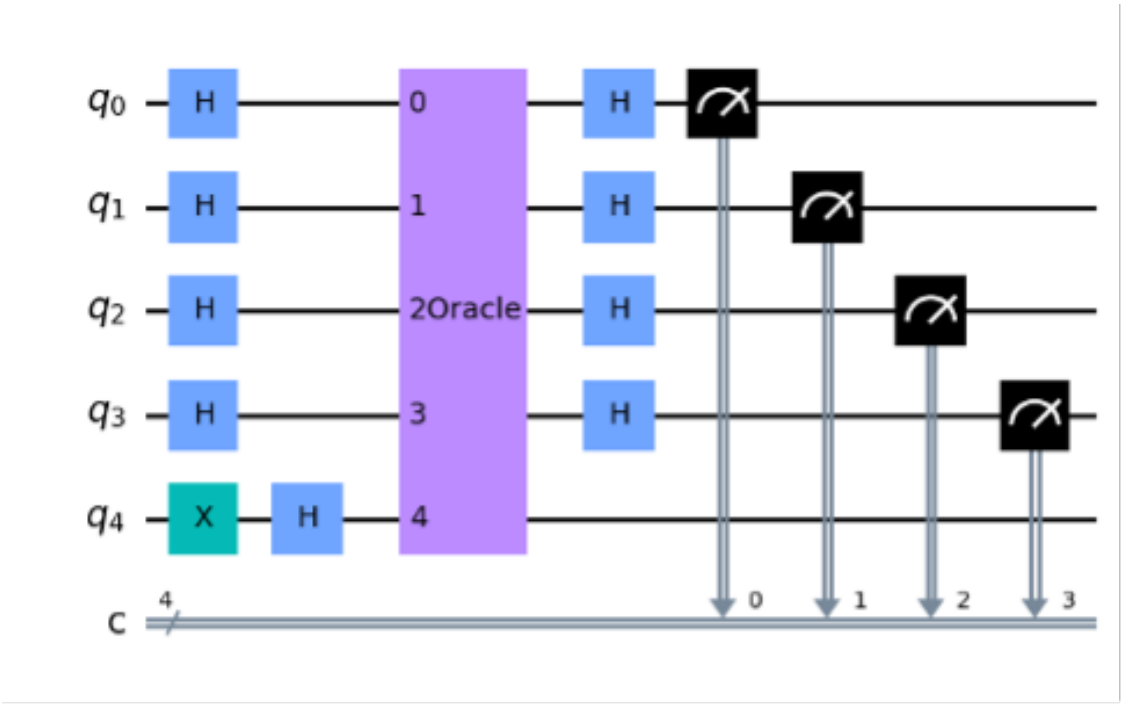

Figure 2.7: Example Deutsch-Jozsa Algorithm applied in a quantum circuit.

(22)

Algorithm 1 Deutsch–Jozsa Algorithm

Require: An oracle function $f : \{0,1\}^n \to \{0,1\}$ that is guaranteed to be either constant or balanced.

Ensure: Determines whether the function is constant or balanced.

1: Prepare n qubits in state $|0\rangle^{\otimes n}$ and one auxiliary qubit in state $|1\rangle$.
2: Apply Hadamard gate H to each of the n qubits and the auxiliary qubit to create a superposition state.
3: Apply the Oracle to the qubits.
4: Apply Hadamard gate H to the n qubits again.
5: Measure the n qubits.
6: **if** the measurement result is $|0\rangle^{\otimes n}$ **then** ▷ All qubits are 0
7: **return** The function f is constant.
8: **else**
9: **return** The function f is balanced.
10: **end if**

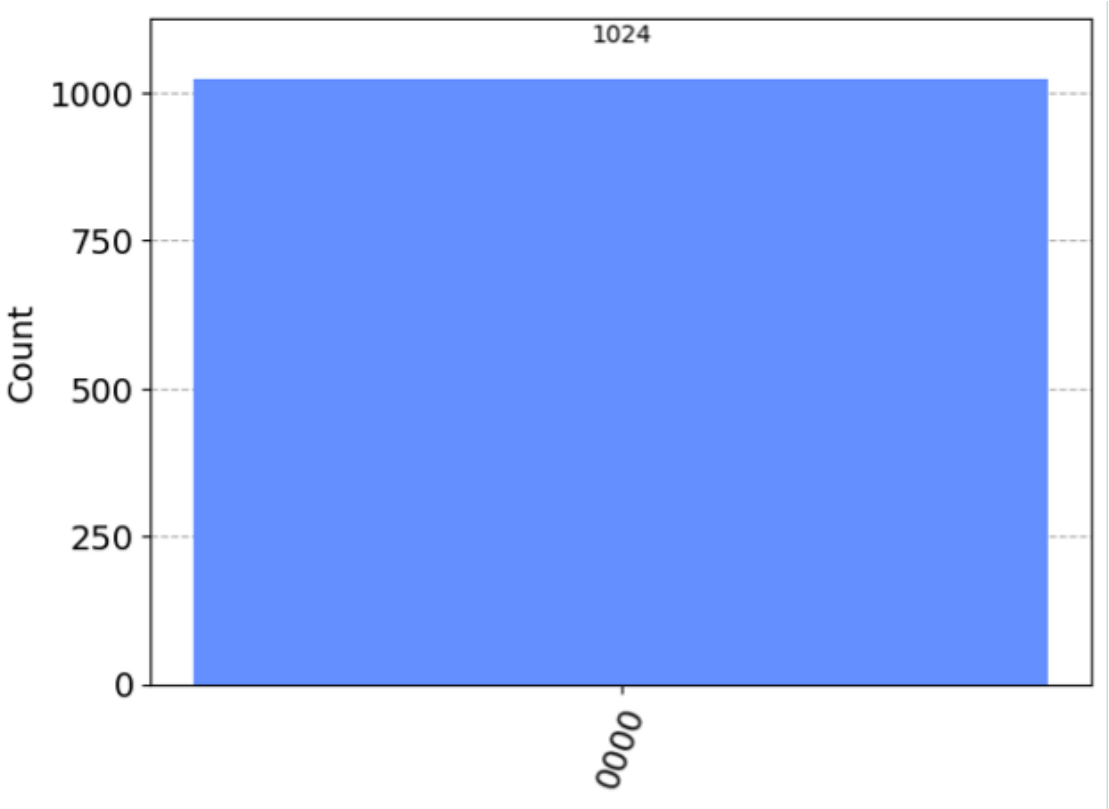

Figure 2.8: Deutsch-Jozsa Result.

This shows that out of 1024 runs or 'shots', the algorithm correctly identified that the oracle returns a constant stream every time.(Note: This was ran on the qasm simulator, which mitigates any quantum noise)

2.2.3 | Grover's Algorithm

Grover's algorithm is a quantum algorithm developed by Lov Grover in 1996 that provides a way to speed up the search for a specific item within an unsorted database. It achieves a quadratic speedup compared to classical algorithms. In essence, Grover's algorithm can search through an unsorted list of N items in $O(\sqrt{N})$ time, which is substantially faster than the classical $O(N)$ time. This makes it one of the cornerstone algorithms demonstrating the quantum advantage(22)(26).

Grover's algorithm operates on a quantum system initialized in an equal superposition of all possible states. It then uses a sequence of unitary operations, often referred to as Grover iterations or Grover operators, which amplify the probability amplitude of the desired state. Each Grover iteration consists of the following steps:

1. The algorithm applies an oracle function that inverts the sign of the amplitude of the desired item in the superpositioned state.
2. It then applies the Hadamard gates to convert the superposition back to the basis state representation.
3. Next, it performs an inversion about the mean operation on these amplitudes.
4. The Hadamard gates are applied once more to return to the superposition representation.

This sequence of steps gradually increases the probability of measuring the desired state. After $O(\sqrt{N})$ repetitions of the Grover iteration, the system is measured, resulting in measuring the desired items' index with the highest

probability.

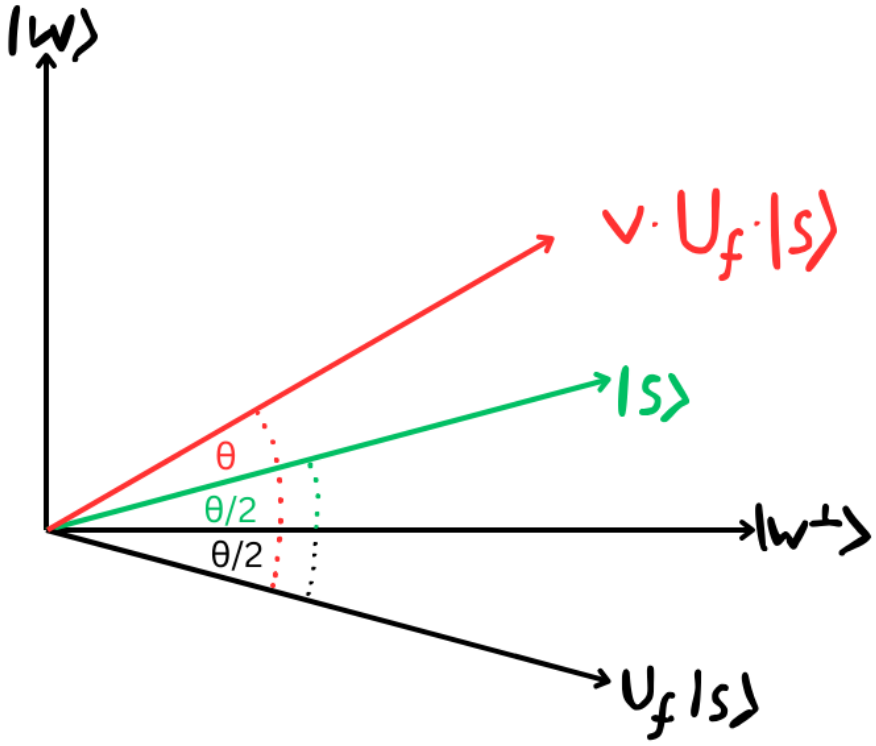

Figure 2.9: How Grover's Algorithm uses superposition $|S\rangle$ to cast a reflection V around the oracle U_f on the basis of the marked state $|W\rangle$ and its orthogonal basis. This process is repeated r times to maximise θ.

A quantum circuit implementing Grover's algorithm typically involves Hadamard gates for state preparation, a Grover oracle to mark the searched item, and the diffusion operator, also known as the Grover operator, to amplify the marked state's amplitude. Below is a depiction of a simple Grover's algorithm quantum circuit.

(22)

Algorithm 2 Grover's Search Algorithm

Require: A black-box oracle function $f : \{0,1\}^n \rightarrow \{0,1\}$ that evaluates to 1 for the desired search element and 0 otherwise.

Ensure: Identifies the desired element from the unsorted database.

1: Prepare a superposition of all possible states using Hadamard gates.

2: Repeat the following steps $O(\sqrt{N})$ times:

   1. Apply the oracle function f to invert the phase of the desired element.
   2. Apply Hadamard gates to all qubits.
   3. Perform an inversion about the mean on the amplitudes.
   4. Apply Hadamard gates again.

3: Measure the qubits to obtain the index of the desired element.

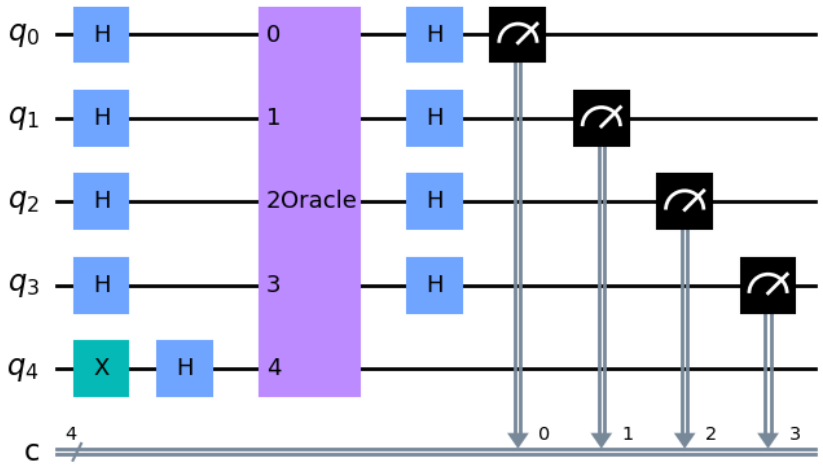

Figure 2.10: A quantum circuit representation of Grover's algorithm for a single search iteration, with depth five.

Running on the qasm simulator results with the following histogram:

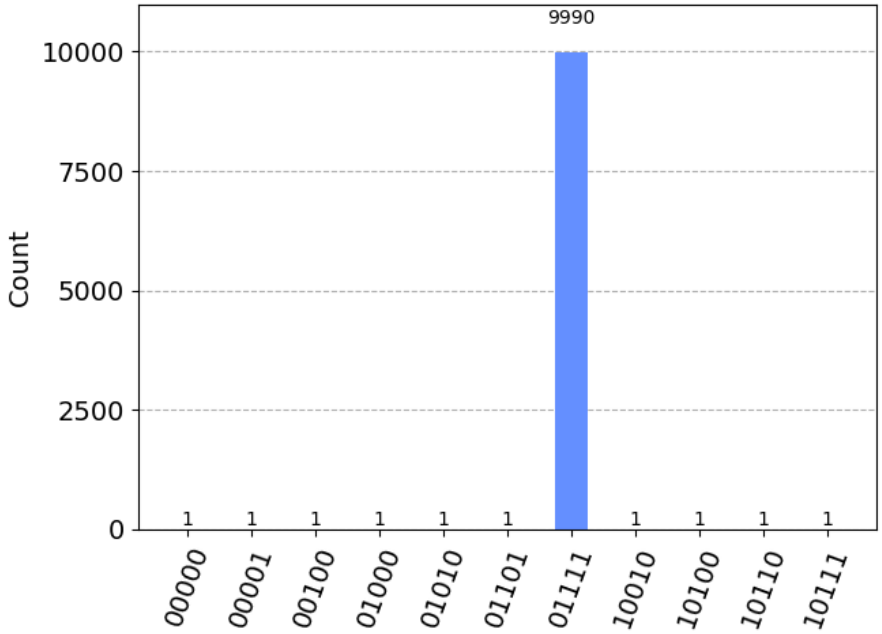

Figure 2.11: Histogram of Grover's algorithm for a single search iteration, with index 5 marked.

The circuit was also ran on the ibm_nairobi backend 2.1, with two states marked.

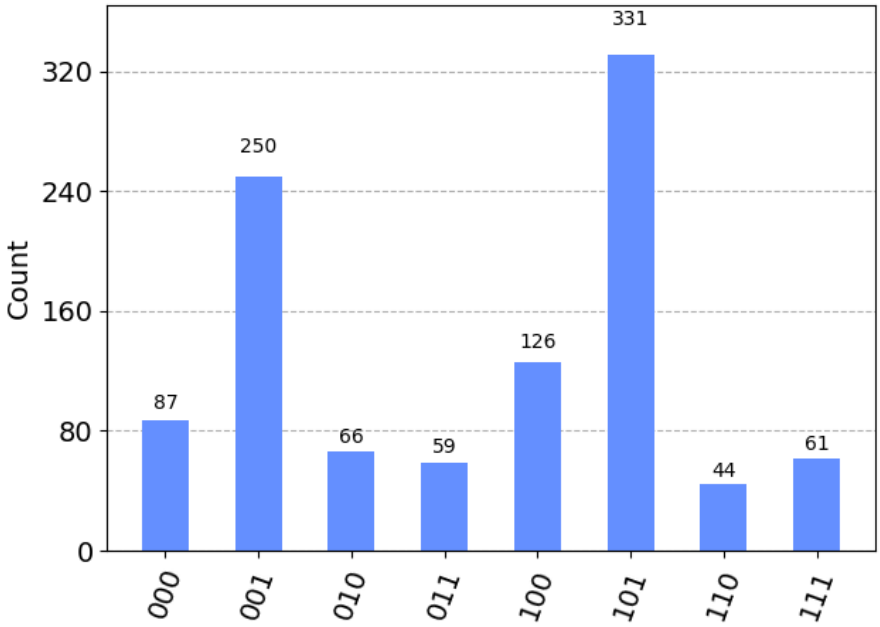

Figure 2.12: 3 qubits, basis state [1, 5] marked, 1 rounds.

In figure 2.12 we can see the the counts for states 1 and 5 are significantly higher than the other states. This result is no quite perfect but still shows the application of this quantum algorithm.

Backend	**Info**
Backend Name	ibm_nairobi
Backend Version	1.3.3

Table 2.1: Backend Information

2.2.4 | Variational Quantum Classifier

A VQC represents an integration of quantum computing principles into machine learning. The process of implementing a VQC involves encoding classical data to a quantum feature space via a feature map, and then using this new quantum data in the quantum circuit. This allows for classical data to be used in a quantum circuit. A VQC uses variational quantum algorithms which follow hybrid

quantum-classical schemes, involving parameterised circuit and gates with parameters optimised through a classically-based optimisation loop to minimise some cost function. These steps are repeated in a quantum-classical hybrid loop that eventually terminates when the classical optimization has found optimal parameters. The typical cost function is a measurement between the actual outputs and the desired outputs for training data. (22) (27)

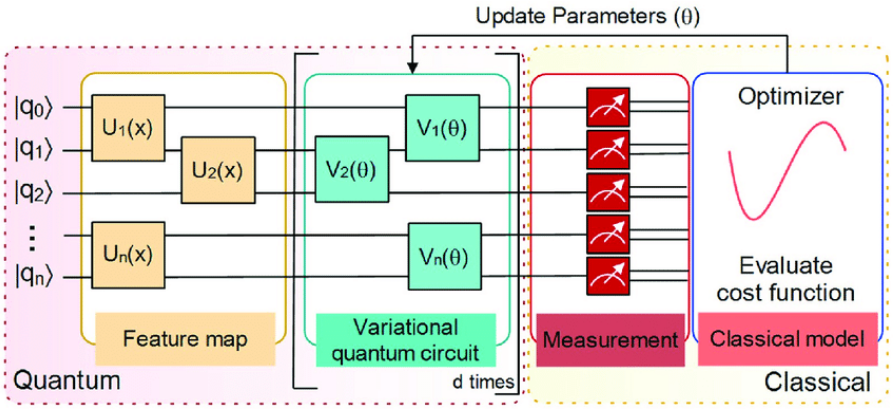

Figure 2.13: Schematic representation of a variational quantum circuit (VQC). Image from (28)

.

2.2.5 | Quantum Support Vector Machines

Quantum Support Vector Machine (QSVM)s are the quantum extension to SVMs discussed earlier by offering a natural and efficient framework for linear algebra operations, including those required for SVMs. QSVMs exploit quantum algorithms for linear systems of equations and quantum kernel estimation to calculate the inner products in a high-dimensional feature space more efficiently. This approach leverages the concept of quantum feature maps and kernels to implicitly perform these calculations in a Hilbert space represented by quantum states. What this suggests is that QSVMs' quantum kernels are expected to do better than classical if they are hard to estimate classically.

The quantum kernel is created by mapping a classical feature vector $\vec{x}$ to a

Hilbert space using a quantum feature map $\phi(\vec{x})$.

$$K_{ij} = |\langle \phi(\vec{x}_i) | \phi(\vec{x}_j) \rangle|^2$$

where K_{ij} is the kernel matrix, $\vec{x}_i, \vec{x}_j$ are n dimensional inputs $\phi(\vec{x})$ is the quantum feature map $|\langle a|b\rangle|^2$ denotes the overlap of two quantum states a and b.(29)

2.3 | Encoding Classical Data

One of the first steps in building the QML model is encoding (or 'embedding) the classical data into a quantum state. This step allows us to utilise phenomena like superposition and entanglement on the embedded classical data. (22) there are a number of ways of implementing this;

- **Basis Encoding:** Basis (or computational) encoding maps each bit of classical binary data to the corresponding quantum state. A classical bit value of 0 or 1 is represented by the quantum state $|0\rangle$ or $|1\rangle$ respectively. (30)
- **Amplitude Encoding** Where data vector $\vec{x}$ in $\mathbb{R}^N$ is normalised and then encoded into the amplitudes of a quantum state $|\psi\rangle$, represented as:

  $$|\psi\rangle = \sum_{i=0}^{N-1} x_i |i\rangle ,$$

  where $|i\rangle$ denotes the computational basis states. This method allows for encoding N-dimensional data into $\log_2(N)$ qubits. (31)
- **Angle Encoding:** Utilizes the values of classical data to define the angles in quantum rotations about the Bloch Sphere (22). For instance, a real number θ is encoded as $|\psi(\theta)\rangle = R(\theta)|0\rangle + R(\theta)|1\rangle$, letting R represent the rotation procedure. For example a rotation about the Z axis would be represented as: The rotation matrix around the Z-axis is given by:

  $$R_z(\theta) = e^{-i\theta Z/2} = \begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix}$$

  (30)

2.3.1 | ZZ-Feature Map

The ZZFeatureMap, is a second-order Pauli-Z evolution circuit,(32) The Pauli Expansion circuit is a data encoding circuit that transforms input data $\vec{x} \in \mathbb{R}^n$, where n is the number of feature dimensions, as

$$U_\Phi(\vec{x}) = \exp\left(i \sum_{S \in I} \phi_S(\vec{x}) \prod_{i \in S} P_i\right).$$

Here, S is a set of qubit indices that describes the connections in the feature map, I is a set containing all these index sets, and $P_i \in \{I, X, Y, Z\}$. Per default the data-mapping ϕ_S is

$$\phi_S(\vec{x}) = \begin{cases} x_i & \text{if } S = \{i\} \\ \prod_{i \in S}(\pi - x_i) & \text{if } |S| > 1 \end{cases}$$

(21)

The construction of the ZZFeatureMap involves applying a series of controlled rotation gates that correlate qubit states in a manner that reflects the underlying structure of the classical data.

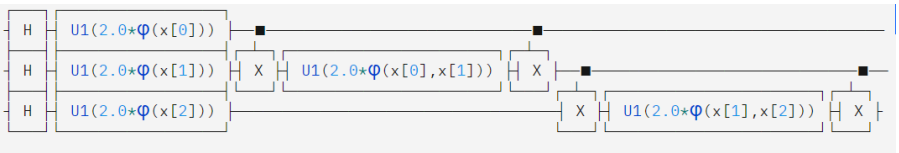

Figure 2.14: 3 qubits and 1 repetition and linear entanglement circuit.

2.14 represents the circuit of the ZZFeatureMap where where ϕ is a classical non-linear function, which defaults to $\phi(x) = x$ if $\phi(x, y) = (\pi - x)(\pi - y)$.(32)

2.4 | Feature Importance & Explainability

Feature importance plays a pivotal role in the domain of ML, serving as a bridge to understanding the inner workings of black box predictive models, which are often hidden from the user. This concept highlights the relative

significance of each input feature in contributing to the model's ability to make accurate predictions. The importance of being able to understand a model spans various aspects of development and application, emphasizing the necessity for interpretability, transparency, and trust in machine learning outcomes.
In many critical sectors such as healthcare, finance, and more, decisions made by machine learning models can have profound implications(6). For example in healthcare, a model might be used to predict patient outcomes based on various clinical parameters, and this outcome needs to be fully understood by not only the doctor but also to be explained to the patient. Understanding which features most significantly influence the predictions can provide insights into diseases, patient management, and treatment planning. Similarly, in finance, models predicting credit risk can benefit from clear insights into which factors most influence a person's likelihood of qualifying for a loan, aiding in fair and transparent decision-making.
The emphasis on feature importance thus directly ties into the goal of Explaiable AI (XAI). XAI seeks to peel back the layers of complex ML models to offer human-understandable explanations for their predictions.
With the increasing deployment of ML models in sensitive and impactful domains, regulatory bodies are mandating greater transparency and explainability in automated decision-making systems. For instance, the European Union's General Data Protection Regulation (GDPR) introduces the right to explanation, where individuals can ask for explanations of automated decisions that affect them. Feature importance metrics can help in fulfilling such regulatory requirements by providing a basis for explaining decisions made by ML models (33).

From a technical perspective, understanding feature importance is integral to refining and optimizing ML models. By identifying and focusing on the most features, data scientists can streamline models, reducing their complexity without sacrificing performance. This process of feature selection helps in mitigating overfitting, enhancing the model's ability to generalize to unseen data. Moreover, it can lead to more efficient models by reducing the dimensionality of the data, thereby lowering computational costs and improving runtime efficiency.

For machine learning solutions to be widely adopted, especially in high-stakes domains, it is crucial for end-users and stakeholders to trust the decisions made by these systems. Feature importance metrics facilitate this trust by offering transparency in the decision-making process. When users understand why a model makes a certain prediction, their confidence in the system's reliability and fairness should increase. This trust is paramount, especially in critical applications, where the cost of errors is high.

Feature importance and explainability are cornerstones of ethical and effective machine learning practice. It not only fosters trust, transparency, and ethical decision-making but also contributes to the technical robustness and efficiency of predictive models. As machine learning continues to permeate various sectors of society, the role of feature importance in ensuring responsible AI cannot be overstated. This understanding is achieved through various methods, including:

- **Model Interpretability and Transparency**: By understanding which features significantly influence the model's output, stakeholders can gain insight into the model's decision-making process, leading to greater trust.
- **Improved Model Performance**: Identifying and focusing on important features can lead to more efficient models by reducing dimensionality, which can, in turn, lower computational costs and mitigate the risk of overfitting.
- **Feature Engineering**: Knowing which features are important can guide the feature engineering process, where new features are created and redundant or irrelevant features are removed.
- **Domain Insight**: Feature importance can reveal unexpected relationships between features and the target variable, providing new insights into the domain area being studied.

- **Regulatory Compliance**: In many regulated industries, the ability to explain decisions made by machine learning models is a legal requirement. (5) (6) .

2.4.1 | Leave-One-Out (LOO) Feature Importance

Leave-one-out feature importance is an intuitive approach where the importance of a feature is determined by the impact on model performance when that feature is left out of the training process. In this method, the model is trained multiple times, each time leaving out a different feature. The change in model performance (such as accuracy or F1 score) indicates the importance of the omitted feature. If the performance drops significantly without a particular feature, it suggests that the feature is important for the model to make accurate predictions. However, this is a very time-consuming method as it requires retraining a model for each feature to be tested.

2.4.2 | Permutation Importance

Permutation importance ranks the importance of a feature by shuffling the values of the column in question and recalculating the accuracy of the model. The permutation importance algorithm 3 shows how it is implemented. Permutation Importance breaks the relationship between the feature and the true outcome, thus the change in model error after permutation is indicative of the feature's predictive power. The main advantage of permutation importance is that it can be applied to any model and is less likely to be biased towards features with a high number of categories.(7)

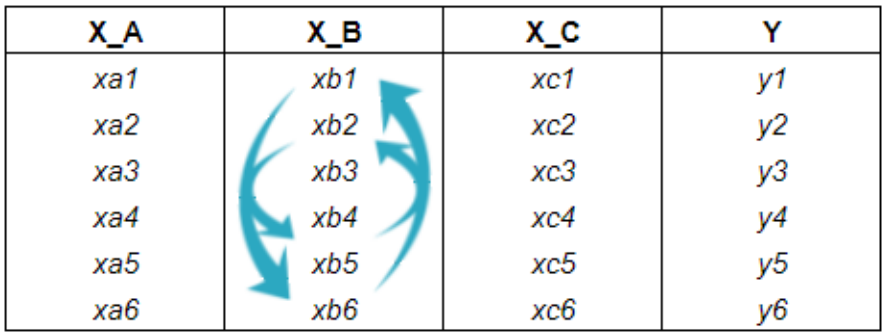

Figure 2.15: How permutation importance works. Image from (34)

Algorithm 3 Permutation Importance

Require: fitted predictive model m, tabular dataset D (training or validation)

Compute the reference score s of the model m on data D
for each feature j in columns of D **do**
    for each repetition $k = 1, \ldots, K$ **do**
        Randomly shuffle column j of dataset D to generate a corrupted version of the data named $\widetilde{D}_{k,j}$
        Compute the score $s_{k,j}$ of model m on corrupted data $\widetilde{D}_{k,j}$
    end for
    Compute importance i_j for feature f_j defined as:
    $i_j = s - \frac{1}{K} \sum_{k=1}^{K} s_{k,j}$
end for

2.4.3 | SHAP Values

The SHAP method uses game theory to explain a machine learning model's output. Through the use of the traditional Shapley values from game theory and their related extensions, it links local explanations for optimal credit allocation.SHAP values give an overall measure of feature importance but also show the direction and magnitude of a feature's effect on individual predictions (8). This method assigns each feature an importance value for a particular prediction by comparing what a model predicts with and without the feature. The calculation considers all possible combinations of features, which ensures fair at-

tribution since it accounts for interaction effects between features. The formula can take the form:

$$\mathbb{E}[f(X) \mid do(X_S = x_S)]$$

This tells us how the model would behave if the inputs were changed, and also because it is much easier to compute as computing SHAP values are generally NP-hard (9).

2.4.4 | Accumulated Local Effects

Accumulated Local Effects (ALE) plots calculate the local impact of a feature by evaluating the change in predictions over finely segmented windows of the feature space. This approach allows for a more accurate representation of the feature's effect, taking into account the natural interactions and dependencies among features. ALE plots achieve this by focusing on the differences in model predictions within these intervals, thus isolating the specific contribution of each feature to the prediction outcome. ALE plots average the changes in the predictions and accumulate them over the grid. This algorithm was introduced to improve upon the Partial Dependency Plot (PDP) method, which can introduce problems when features are correlated (10).

$$\hat{f}_{S,ALE}(x_S) = \int_{z_{0,S}}^{x_S} \mathbb{E}_{X_C|X_S=x_S}\left[\hat{f}^S(X_S, X_C)|X_S = z_S\right] dz_S - \text{constant}$$

$$= \int_{z_{0,S}}^{x_S} \left(\int_{X_C} \hat{f}^S(z_S, X_C) d\mathbb{P}(X_C|X_S = z_S)\right) dz_S - \text{constant}$$

The ALE method calculates the differences in predictions by replacing the feature of interest with grid values z. The difference in prediction is the effect the feature has for an individual instance in a certain interval. The sum on the right adds up the effects of all instances within an interval, which appears in the formula as the neighbourhood $N_j(k)$. We divide this sum by the number of instances in this interval to obtain the average difference of the predictions for this interval. This average in the interval is what the term *Local* in the name ALE covers. The sum symbol on the left means that we accumulate the average

effects across all intervals. The uncentered ALE of a feature value that lies, for example, in the third interval, is the sum of the effects of the first, second, and third intervals. This is what is reflected by the word *Accumulated* in ALE.

This effect is centred so that the mean effect is zero.

$$\hat{f}_{j,ALE}(x) = \hat{f}'_{j,ALE}(x) - \frac{1}{n}\sum_{i=1}^{n}\hat{f}'_{j,ALE}\left(x_j^{(i)}\right)$$

(10)(35)

2.5 | Literature Review

There are several papers and resources explore the implementation of QML algorithms, including guides provided through IBM Qiskit (27) and papers such as 'Quantum variational multi-class classifier for the iris data set' (36), This paper confirmed a method implemented to check optimising combinations of ansatzs and optimisers. However, there is only a handful of papers that explore the implementation of feature importance and explainability for QML.

There is an abundance of papers and resources about machine learning, feature importance, and quantum computing and quantum machine learning, however, there is an apparent gap in combining these topics. The first paper to compare feature importance between quantum and classical machine learning models was a paper titled 'Study of feature importance for quantum machine learning models' (37) was conducted by researchers at IBM using ESPN Fantasy Football data and claims to be the first paper exploring feature importance in QML. However, this paper only investigated two methods of feature importance; Permutation importance and ALE (37). This project expands on this paper by providing a much more in-depth report on the implementation of QML and by both exploring more methods of feature importance and also venturing into methods of explainability.
There are fewer papers exploring the area of explainable QML; 'eXplainable AI for Quantum Machine Learning' (38) Implements a variation of SHAP on basic classifiers, ranging from a single qubit classifier two a four qubit classifier, using

the binary 'bars and stripes' data. However, this paper does not make any comparisons to the results from a classical counterpart.
Last year, a paper published 'Explainable heart disease prediction using ensemble-quantum machine learning approach' (39) utilised various QML methods on UCI's Heart Disease data set, with a binary classification of diagnosis or not. This paper implemented SHAP predictions but does not make clear comparisons between the classical and quantum results concerning their SHAP values or comparable metrics.

What this project does is implement classical and quantum machine learning models on the multi-class Iris dataset, which is a well-known data set used to classify flowers, and implement several methods of feature importance and explainability techniques and provide some direct comparisons between the results.

Table 2.2: Literature Comparison

Paper	SVM/C	RF	VQC	QSVC	ALE	LOO	Perm	SHAP
Study of Feature Importance for Quantum Machine Learning Models. (37)	Yes		Yes		Yes		Yes	
Quantum variational multi-class classifier for the iris data set. (36)			Yes					
Supervised learning with quantum-enhanced feature space. (21)	Yes		Yes	Yes				
eXplainable AI for Quantum Machine Learning. (38)	Yes		Yes					
Principles and practice of explainable machine learning. (5)	Yes	Yes					Yes	Yes
Explaining Quantum Circuits with Shapley Values (pre-print) (40)			Yes	Yes				Yes
Explainable heart disease prediction using ensemble-quantum machine learning approach (39)			Yes	Yes				Yes
This Project	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

3

Implementation

3.1 | Technology Overview

This project was conducted through Python Jupyter Notebooks. This decision was made as Jupyter Notebooks allow for Python code to be run in individual cells, allowing for a much more interactive and dynamic analysis workflow. The notebooks also support markdown text and formatting which aid further in explaining analysis and charts in detail in the file itself. They also allow for reproducibility and are a good tool for presenting and sharing analysis results.

The main notebooks follow as sample training classical data on QML models, Optimizing VQC configurations for each dataset, a notebook running SVMs, random Forrest, VQC and QSVM for each dataset, which also gathers the feature importance data from each of the models. Then the final notebook contains the implementations of Feature Importance and Explainability, generating the visualizations and analysis of the feature's importance. In the repository, there will also be a directory containing many exploratory and research scripts made with the purpose of learning the technology, and other example implementations adopted from online resources and tutorials.

3.2 | The Iris Dataset

The Iris dataset, introduced by Ronald Fisher in 1936, is a foundational dataset used in statistical learning and machine learning. It comprises 150 samples from three species of Iris flowers, each described by four features: sepal length, sepal width, petal length, and petal width. The dataset's simplicity yet capability to demonstrate the effectiveness of various algorithms has made it a staple for teaching purposes. And so this dataset was chosen to be used in this project(41). For a clear breakdown, of the Iris dataset, refer to table. 3.2.

3.2.1 | Data Preprocessing

The data was normalised with scikit-learns MinMaxScaler, where features are scaled to fall from zero to one. This step was taken to ensure better comparability between the classical and quantum models, as this step was necessary to encode the data for the quantum classifiers. All the models were trained and scored on an 80% - 20% train-test split. The breakdown of the test group can be found in table 3.2.1.

Table 3.1: Test Case Distribution

Species	Label	Test Count
Setosa	0	7
Versicolor	1	9
Virginica	2	14

Plot 3.1 shows a pairs plot of the Iris dataset, which visualises the relationship between the features. The consistent separation of the setosa data points suggests that it will be more easily classified than versicolor and virginica, which have several overlapping points. It is also worth noting the linear structure between peal length and width which shoes the correlation between the two features, which can indicate that one of the features may be redundant.

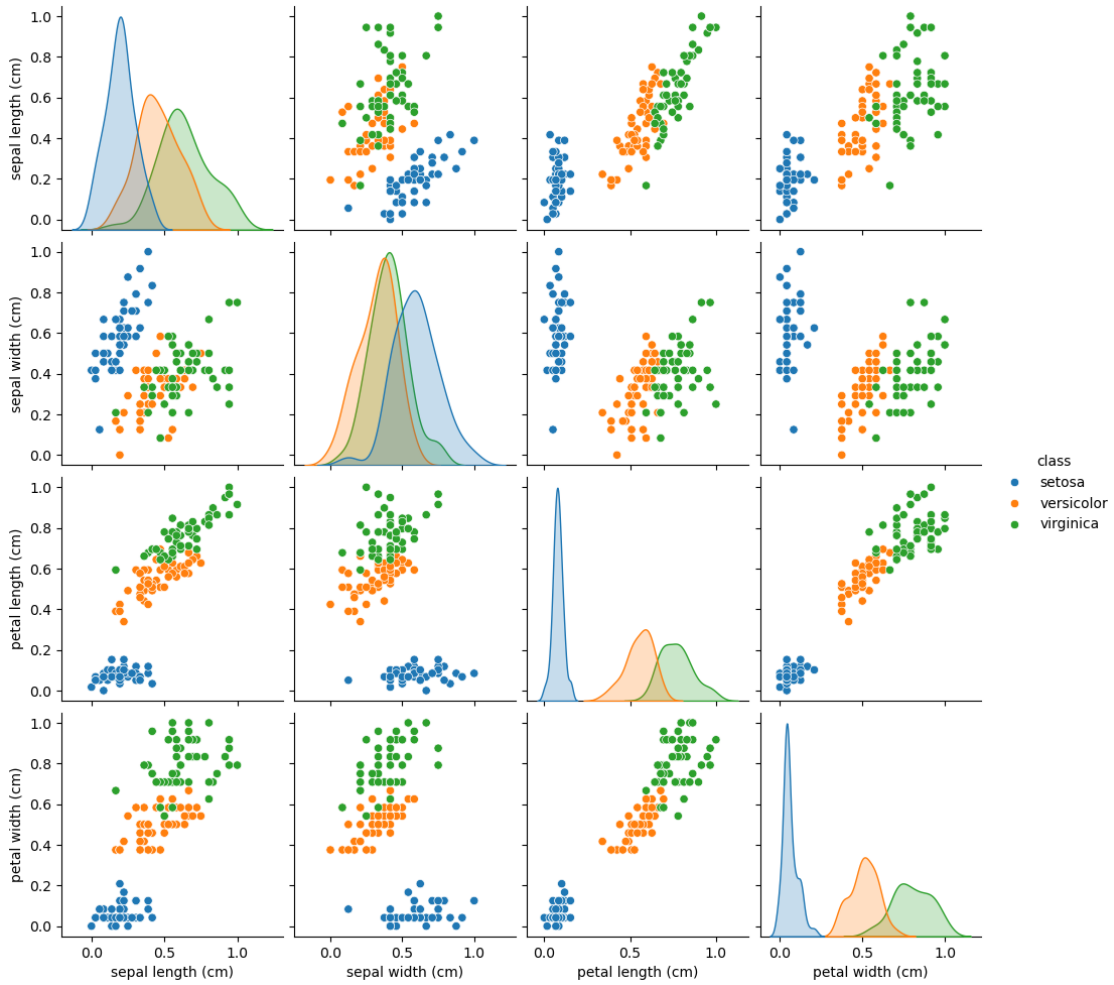

Figure 3.1: Iris Pairs Plot.

3.2.2 | VQC Implementation

The first and most basic QML model implemented was the VQC, which could be described as the quantum equivalent of a simple linear classifier. A ZZFeatureMap was used on the dataset to use the classical data in the quantum algorithm. However, to maximise the performance of the model we also needed to select what combination of ansatz and classical optimizer should be utilised.

Table 3.2: Iris Dataset Summary

Characteristic	**Detail**
Number of Instances	150 (50 in each of three classes)
Number of Attributes	4 numeric, predictive attributes and the class
Attribute Information	Sepal length in cm Sepal width in cm Petal length in cm Petal width in cm
Classes	Iris-Setosa Iris-Versicolour Iris-Virginica
Missing Attribute Values	None
Class Distribution	33.3% for each of 3 classes
Creator	R.A. Fisher
Donor	Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
Date	July, 1988

VQC Optimisations

Choices of Ansatz

- **RealAmplitudes** circuit is a heuristic trial wave function used as Ansatz in chemistry applications or classification circuits in machine learning. The circuit consists of alternating layers of Y rotations and CX entanglements. The entanglement pattern can be user-defined or selected from a predefined set. It is called RealAmplitudes since the prepared quantum states will only have real amplitudes, the complex part is always 0. (42)

- **EfficientSU2** circuit consists of layers of single qubit operations spanned by SU(2) and CX entanglements. This is a heuristic pattern that can be used to prepare trial wave functions for variational quantum algorithms or classification circuit for machine learning.
  SU(2) stands for special unitary group of degree 2, its elements are 2×2 unitary matrices with determinant 1, such as the Pauli rotation gates. (43)

Choices of Optimizer

- **Constrained Optimization By Linear Approximation optimizer (COBYLA)** is a numerical optimization method for constrained problems where the derivative of the objective function is not known. (44)
- **Sequential Least SQuares Programming optimizer (SLSQP)** minimizes a function of several variables with any combination of bounds, equality and inequality constraints. SLSQP is ideal for mathematical problems for which the objective function and the constraints are twice continuously differentiable. (45)

These were combined and iterated from one to four repetitions of the ansatz. As seen in 3.2, SLSQP and EfficentSU2 have the best overall performance, with an initial score of 90%, the SLSQP and EfficientSU2 with three and four reps would take between one and one and a half hours respectively to train on a local simulator, while the COBYLA and EfficientSU2 with three reps only took a minute and a half, with a bootstrapped accuracy of 87%. Due to the necessity of high numbers of repetitions in the experiments, this model was selected for the experiments.

3.2.3 | QSVC Implementation

The Quantum Support Vector Classifier is the other quantum model implemented in this project, which is the quantum equivalent of the Support Vector Classifier. To build the QSVC, we need to specify a quantum kernel. The FidelityQuantumKernel class in Qiskit provides a quantum kernel implementation based on the principle of state fidelity. This takes in a feature map, for which we again use the ZZFeatureMap and a fidelity instance; ComputeUncompute which uses the sampler primitive to calculate the state fidelity of two quantum circuits following the compute-uncompute method. (29)

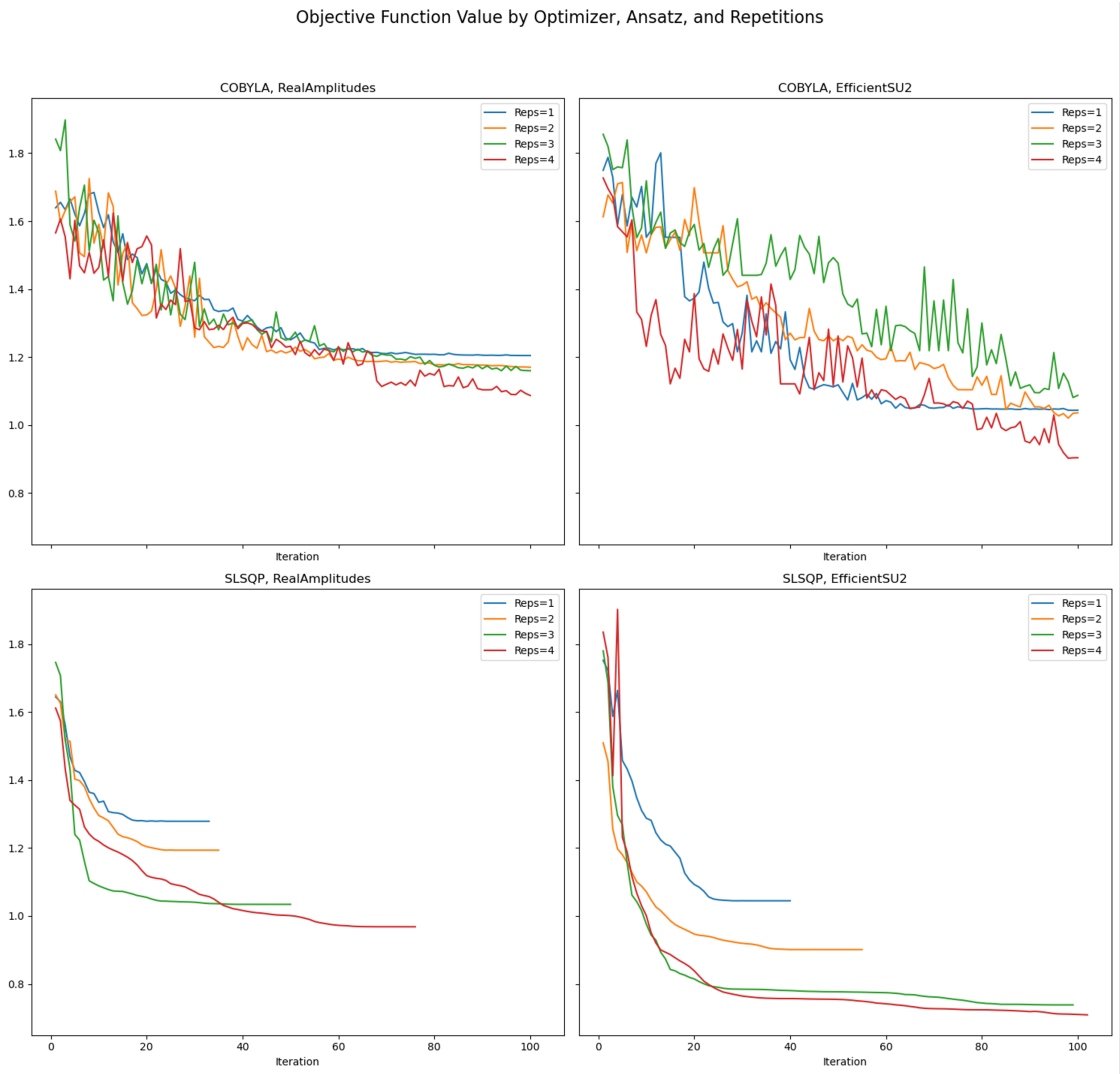

Figure 3.2: Combinations of Optimiser and Ansatz for the VQC.

3.3 | Model Comparisons

Table 3.3 provides the classification report generated from each of the models; the Support Vector Classifier (SVC) showcases high precision across all classes, achieving perfect precision and recall for class 0 (setosa). Its performance remains strong for classes 1 (Versicolor) and 2 (virginica), with high f1-scores and

overall accuracy of 93%.
The VQC had decent precision for classes 1 and 2, with scores of 90% and 92% respectively, with a high f1-score for class 1, but across almost all metrics, VQC appears to be the weakest classifier.
The QSVC presents excellent precision and perfect recall for classes 0 and 2, and a recall of 86% for class 1. The f1-scores are notably high, particularly for class 2, with an overall accuracy reaching 97%. This model demonstrates the best performance among the other classifiers, indicating its possibly superior capability to manage and balance dataset complexities.
The Random Forest classifier model exhibits 100% precision for classes 0 and 2 but slightly lower precision for class 1. While it achieves perfect recall for class 0 and class 1, its recall for class 2 drops, reflecting some difficulties in consistently identifying this class. The f1-scores are high for class 0 but moderate for classes 1 and 2, culminating in an overall accuracy of 90%. Although the model performs well, it shows specific weaknesses that could be mitigated through further optimization.
Overall, the QSVC stands out as the superior model, offering the highest accuracy and robust performance across all evaluated metrics, making it the preferred choice when quantum resources are available. The SVC follows closely behind, providing a highly effective and reliable alternative without the need for quantum enhancements. Both the SVC and QSVC exhibit strong capabilities in handling the classification challenges presented by the Iris dataset, whereas the RF, despite its strengths, shows areas for potential improvement.

To compare the models' overall accuracy against the test data, a bootstrap implementation was used using 1000 bootstrap resamples from the test data, as seen in table 3.3. The QSVC and SVC had average bootstrap results of 97% and 93% respectively, outperforming the random forest classifier and VQC. The boxplot 3.3 shows how the distributions of the results overlap, which can indicate that these models could be suitable for comparison.

For a clearer idea of how the models performed with respect to the individual test case classes, we can look at the confusion matrix in 3.5, which shows how the four models accurately classified all the test setosa (class 0) samples. This was expected and was also a handy sanity check as the setosa flower is clearly

Table 3.3: Summary of Classification Results by Model

Model	**Metric**	**Class 0**	**Class 1**	**Class 2**	**Overall**
VQC	Precision	0.90	0.71	0.92	
	Recall	1.00	0.71	0.86	
	F1-Score	0.95	0.71	0.89	
	Accuracy				0.87
SVC	Precision	1.00	0.78	1.00	
	Recall	1.00	1.00	0.86	
	F1-Score	1.00	0.88	0.92	
	Accuracy				0.93
QSVC	Precision	1.00	1.00	0.93	
	Recall	1.00	0.86	1.00	
	F1-Score	1.00	0.92	0.97	
	Accuracy				0.97
RF	Precision	1.00	0.70	1.00	
	Recall	1.00	1.00	0.79	
	F1-Score	1.00	0.82	0.88	
	Accuracy				0.90

Table 3.4: Bootstrapped Accuracy Results

Classifier	Mean Accuracy	25th Percentile	75th Percentile
SVC	0.932233	0.900000	0.966667
QSVC	0.965567	0.933333	1.000000
Random Forest	0.900333	0.866667	0.933333
VQC	0.864833	0.833333	0.900000

separable from the other two, which can be seen in the pairs plot 3.1. The most interesting observation is that the QSVC could accurately classify all 14 test instances of virginica (class 2). The QSVC also misclassified one of the versicolor (class 1) samples as virginica while both the SVC and Random Forest correctly classified all seven samples. The VQC had the poorest results by comparison, misclassified two of the versicolor samples, one as setosa and one as virginica. A breakdown of the misclassification can be found in table 3.5.

From table 3.5 we can see that there are five of the thirty test cases were mis-

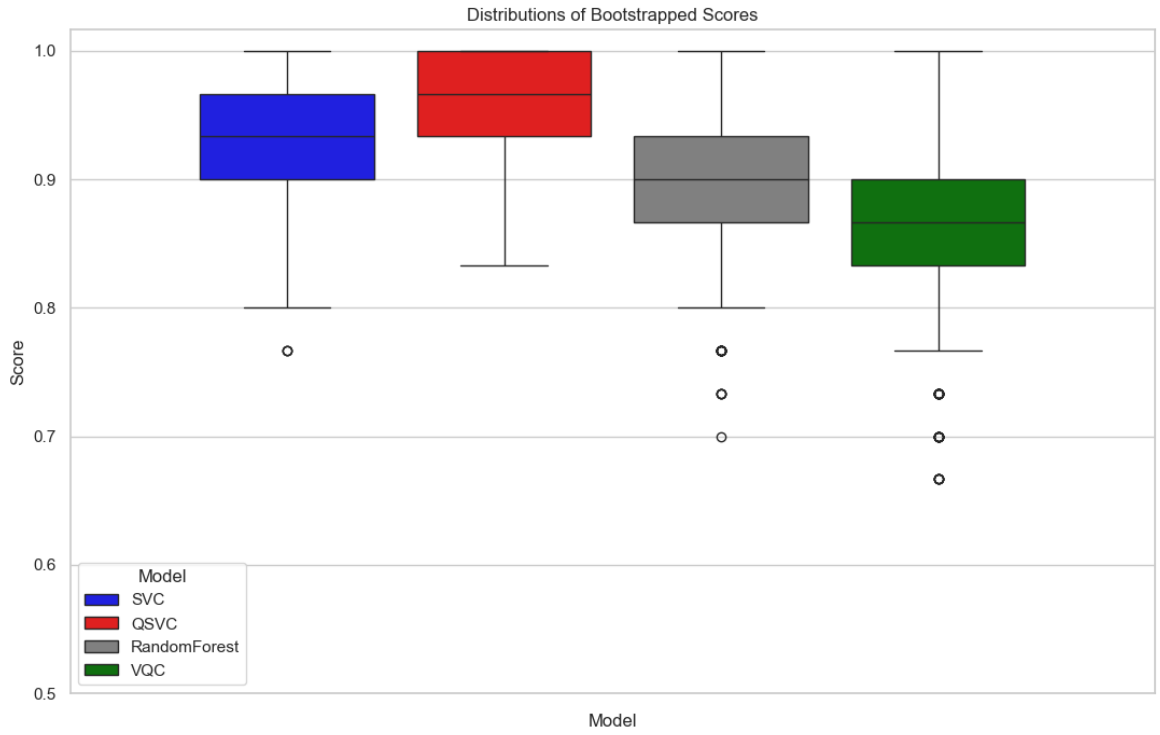

Figure 3.3: Comparison of bootstrapped accuracy scores.

classified, with sample 4, which has a true label of 2 (virginica) misclassified as 1 (versicolor) by all models except the QSVC. For a view of what points were misclassified, refer to the plot 3.4, which highlights the five misclassified points. We can see that four of the points occur where there are some overlaps of the classes. We can see in this plot that point 15 (highlighted in green), which was misclassified by the VQC, seems to be clearly grouped with other versicolor samples. This suggests that there is still further optimisation that could be implemented in the VQC.

Table 3.5: Misclassification Details

Test Sample	**Label**	**Misclassified As**
4	2	SVC: 1, Random Forest: 1, VQC: 1
10	1	QSVC: 2, VQC: 2
15	1	VQC: 0
16	2	Random Forest: 1
17	2	SVC: 1, Random Forest: 1, VQC: 1

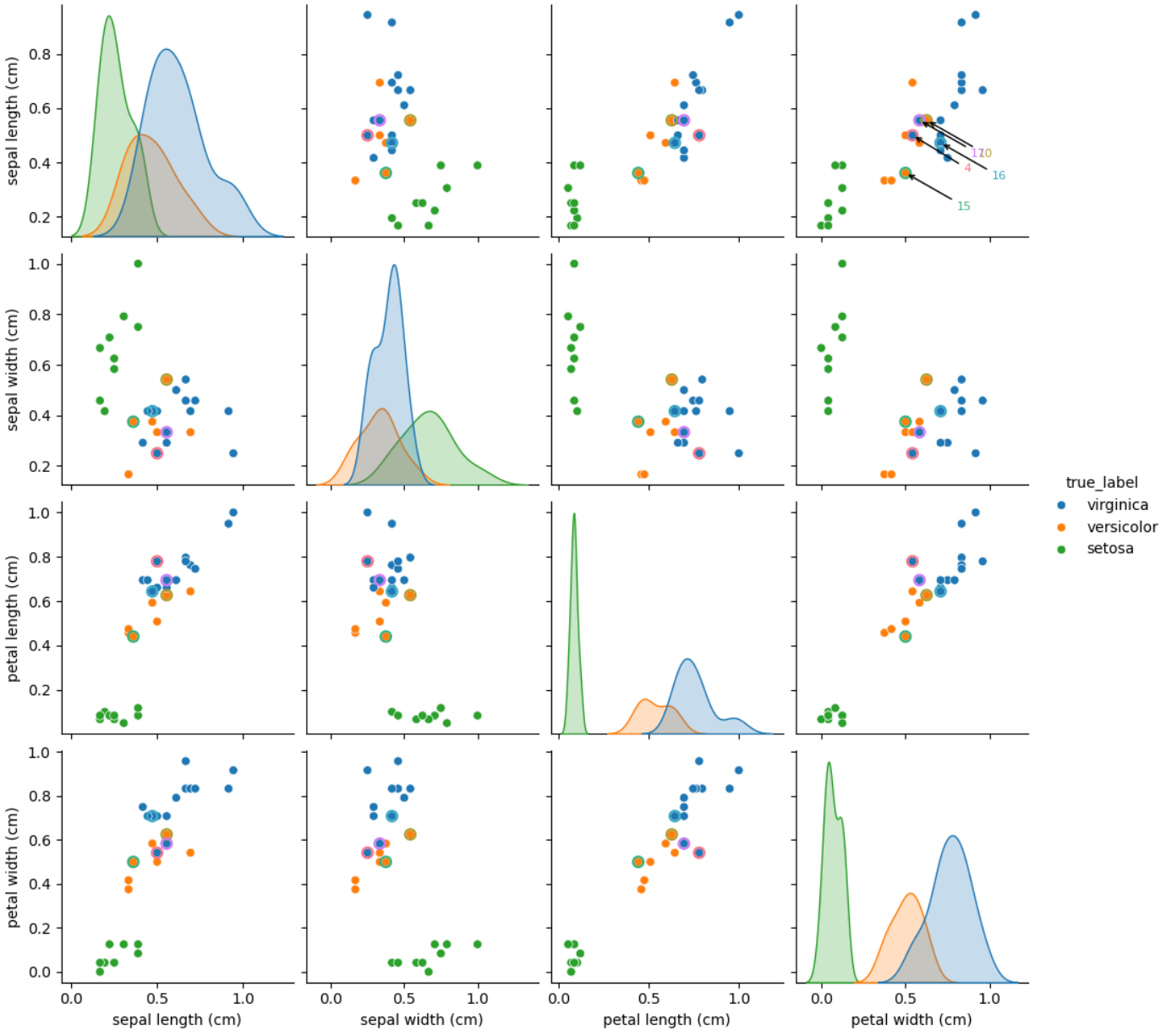

Figure 3.4: Pairs plot of the test data points, highlighting the misclassified points. For misclassification breakdown refer to table 3.5

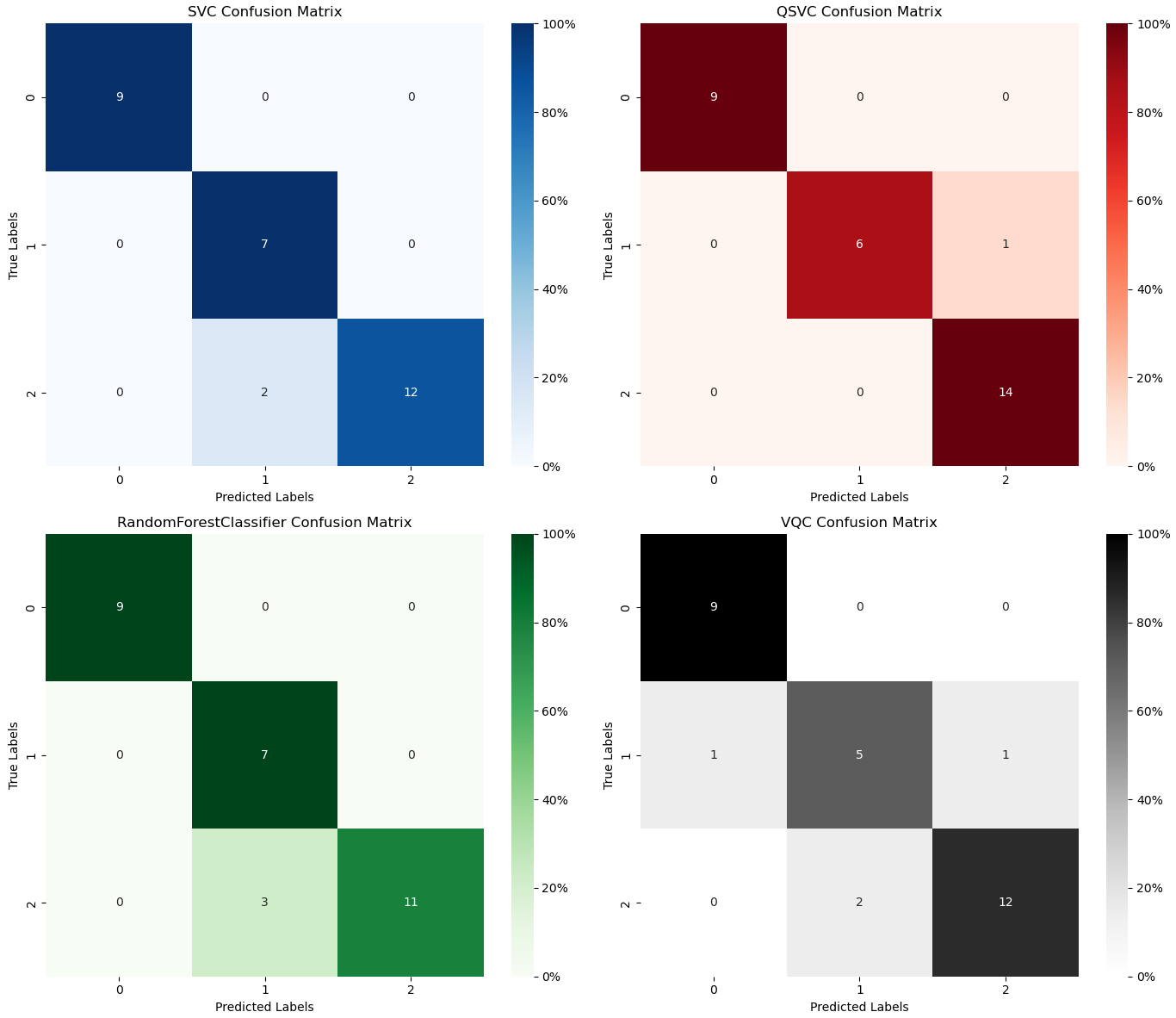

Figure 3.5: Confusion matrices of models.

4

Results

4.1 | Feature Importance

Three methods of feature importance were implemented; Leave-One-Out, Permutation Importance, and Accumulated Local Effects. We will compare the results on a model by model basis, and an overall summary of each method results.

4.1.1 | Leave One Out

To calculate Leave-One-Out (LOO) feature importance, we rerun the model omitting the feature of interest, and check how the performance of the model is affected.

SVC Leave-One-Out

The effects of the feature omissions in the SVC can be seen in the plot 4.1, and in the table 4.1. We can see from the results that when either petal length or petal width is missing, the accuracy of the model increases by 3.3%. This may indicate that petal length or petal length could be poor predictors for the SVC. This counterintuitive increase in test scores upon removing 'Petal Width' and 'Petal Length' could indicate an over-reliance on these features or it could be the re-

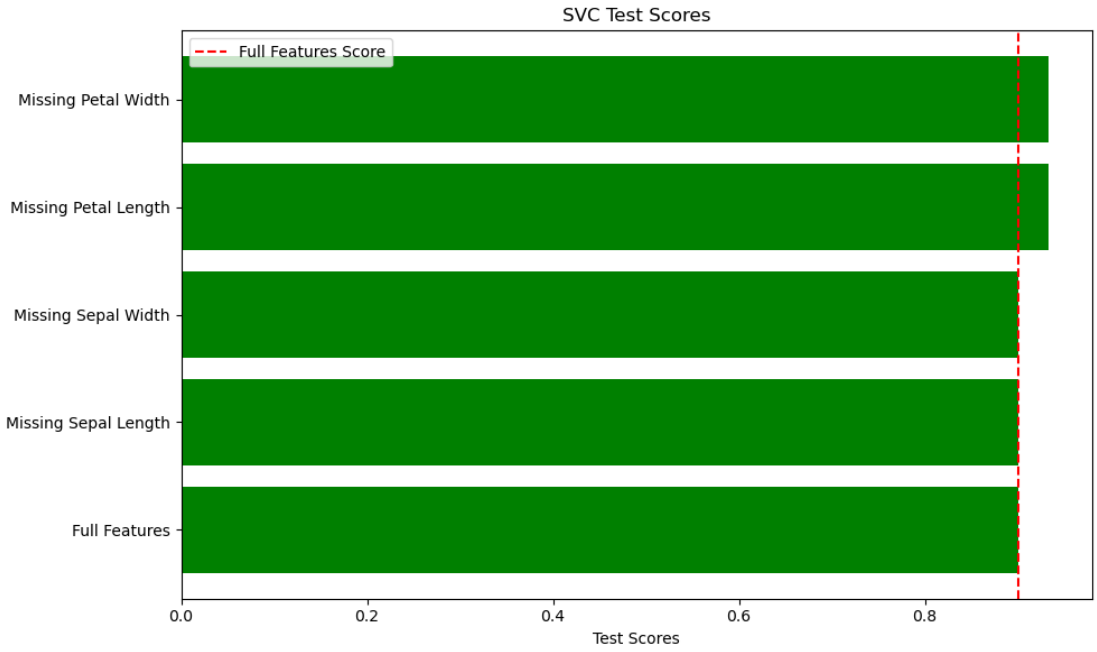

Figure 4.1: SVC Leave One Out Feature Importance.

moval of an interference effect due to the high correlation between these two feature, suggesting that only one of them are really needed in the model. Accuracy remains unchanged when the sepal length or width is omitted. This suggests that the sepal length or width may not be a critical feature for the model, possibly because other features provide similar or sufficient information for making accurate classifications.
Due to the minimal differences, this analysis suggests that SVC regards all of the features to a similar importance.

Random Forest Leave-One-Out

The effects of the feature omissions in the Random Forest can be seen in the plot 4.3, and in the table 4.2. We can see from the results that when either petal length is missing, the accuracy of the model increases by 3.3%. This may indicate that petal length could be a poor predictor compared to the others. This is interesting

Table 4.1: SVC LOO Results

Scenario	Accuracy Score
Full Feature	0.90
Missing Sepal Length	0.90
Missing Sepal Width	0.90
Missing Petal Length	0.93
Missing Petal Width	0.93

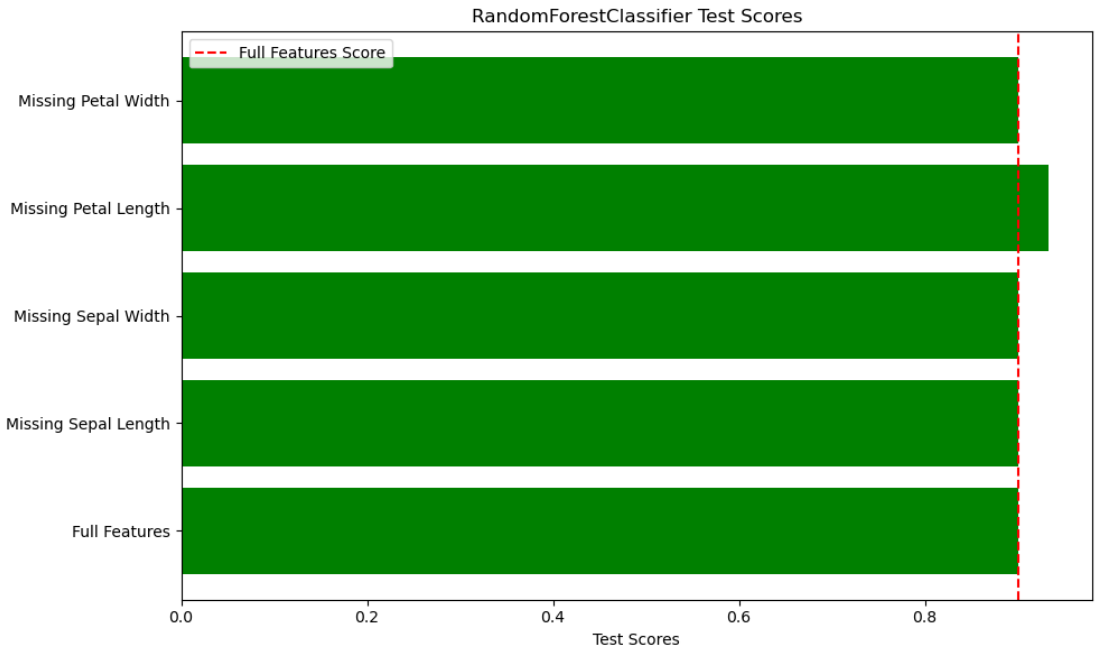

Figure 4.2: Random Forest Leave One Out Feature Importance.

due to the high correlation between petal length and petal width, which would suggest that if petal length has an effect, petal width would cause a similar effect. Accuracy remains unchanged when the sepal length or width is omitted. This suggests that the sepal length or width may not be a critical feature for the model, possibly because other features provide similar or sufficient information for making accurate classifications.

Due to the minimal differences, this analysis suggests that the Random Forest

regards all of the features to similar importance like the SVC possibly assigning petal length with the lowest importance.

Table 4.2: Random Forest LOO Results

Scenario	Accuracy Score
Full Feature	0.90
Missing Sepal Length	0.90
Missing Sepal Width	0.90
Missing Petal Length	0.93
Missing Petal Width	0.90

VQC Leave-One-Out

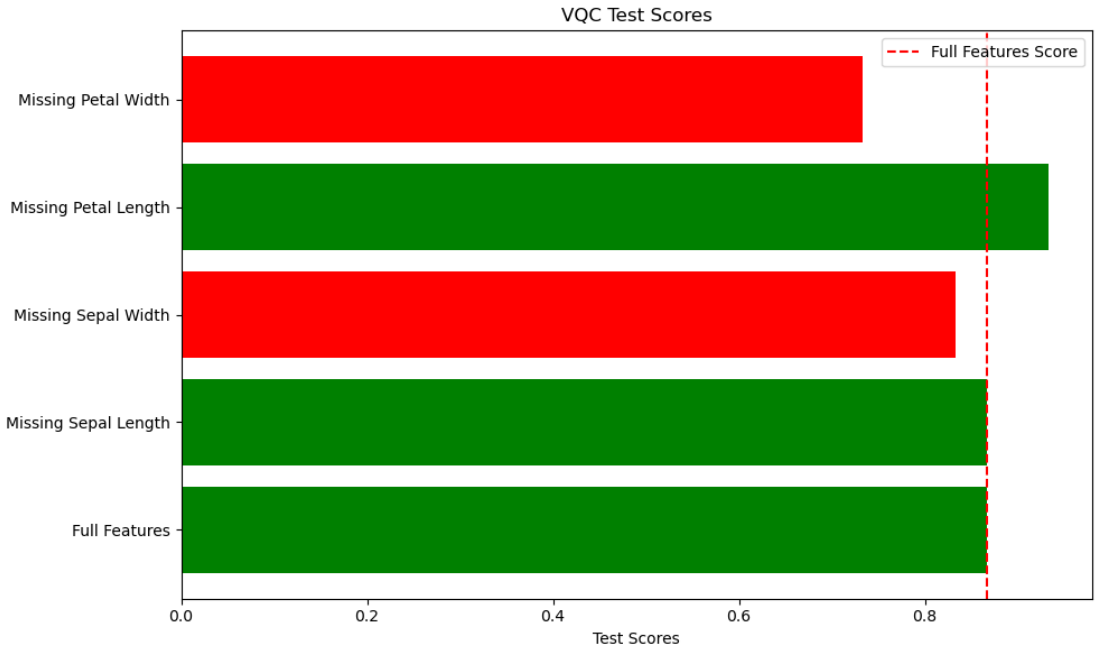

Figure 4.3: VQC Leave One Out Feature Importance.

The effects of the feature omissions in the VQC can be seen in the plot 4.3, and in the table 4.3. The LOO results are much different to the SVC and Random For-

est. The full-featured model has an accuracy of 87%, but this time the removal of petal width caused a drop to 73%, a 16% decrease from our full model, a much bigger difference to the 3.3% difference we found with the SVC and no change in the Random Forest classifier. This causes the effect of removing petal length to be somewhat paradoxical, increasing the model's performance by 6.9%. Due to the highly correlated nature of petal length and width, they would be expected to have similar predictive effects in a model, however, they have opposite effects on the model's performance.

We can also see that the effect of removing sepal width, which had no effect in either the SVC or Random Forest, caused a 4.6% decrease in the VQC's accuracy. These results demonstrate that the VQC has highly different levels of feature importance, with petal width being considered the most important, as its absence causes the biggest decrease in accuracy, followed by sepal width for the same reason, then sepal length and petal length being the least important as its absence improves the models' accuracy.

Table 4.3: VQC LOO Results

Scenario	**Accuracy Score**
Full Feature	0.87
Missing Sepal Length	0.87
Missing Sepal Width	0.83
Missing Petal Length	0.93
Missing Petal Width	0.73

QSVC Leave-One-Out

The effects of the feature omissions in the VQC can be seen in the plot 4.4, and in the table 4.4. We can see that the QSVC had similar accuracy to the VQC, but ended up with highly different results, more in line with what we observed in the SVC and Random Forest, the removal of individual features had very little overall effect on the model's accuracy. What has remained consistent across all the results was the removal of petal length slightly increased the model's accuracy, in this case by 3.5%. There was also in identical effect with the removal of

sepal length, indicating that sepal length introduces some noise into the model. According to this LOO analysis, we can see that QSVC acts somewhat similarly to the classical models, regarding all the models with similar importance.

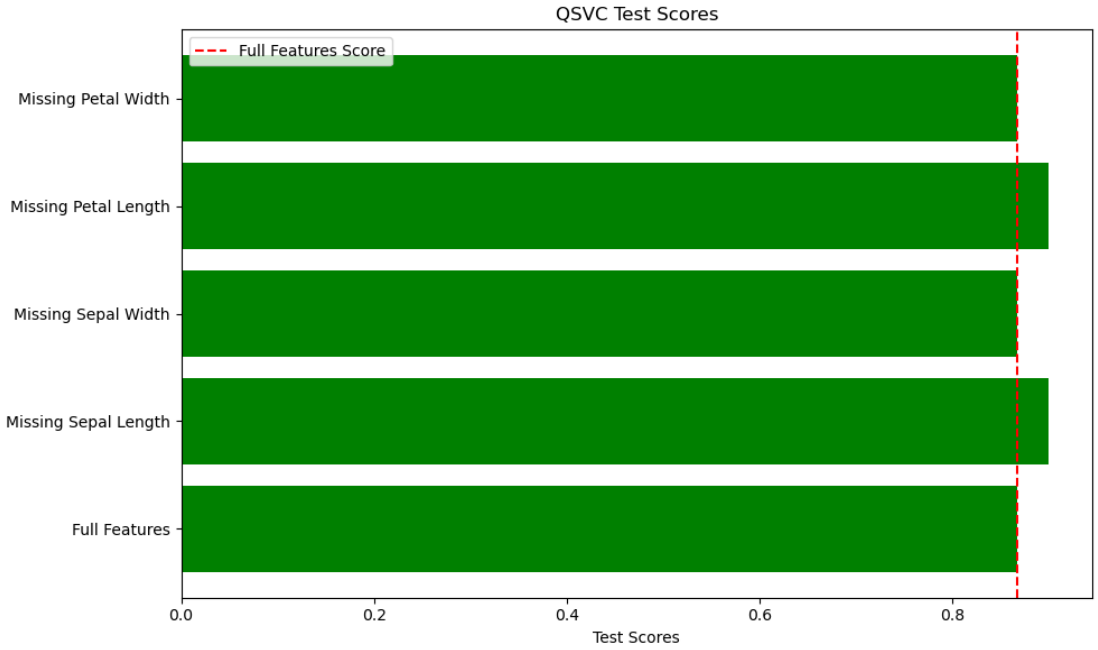

Figure 4.4: QSVC Leave One Out Feature Importance.

Table 4.4: QSVC LOO Results

Scenario	**Accuracy Score**
Full Feature	0.87
Missing Sepal Length	0.90
Missing Sepal Width	0.87
Missing Petal Length	0.90
Missing Petal Width	0.87

LOO Summary

Figure 4.5, shows a full comparison of LOO feature importance across all the models. The VQC model exhibits a highly different approach to assessing feature importance, particularly diverging from the other models in its valuation of Petal Width. The VQC was also the only model to demonstrate a decrease in performance at the removal of any features. The removal of a feature in the other three models either did not affect accuracy or increased it. Interestingly, the QSVC, while also a quantum model, seems more aligned with the classical SVC and RF models in terms of feature impact, except with Sepal width, who's removal increases accuracy. Both quantum models demonstrate a sensitivity to Sepal features, yet they differ in which specific Sepal features they find more important. The QSVC shows a marked increase in importance for Sepal Length, a trend not observed in the VQC model. Meanwhile, the VQC assigns considerable importance to Sepal Width, unlike the QSVC. This difference demonstrates the unintuitive ways in which quantum models process feature importance compared to the classical models.

4.1.2 | Permutation Importance

Permutation importance differs from LOO by instead of removing the feature entirely, the feature values will be repeatedly shuffled to break the relationship between the feature and the class. This method will allow us to get a gauge of the variation of the importance.

SVC Permutation Importance

We can see in the plot 4.6 that petal width and petal length have the biggest impact when their values are permuted, and sepal width seems to have little to no effect on the model accuracy. The permutation of sepal length has no effect whatsoever. The results of petal length and petal width being permutated are extremely interesting as they appear to demonstrate the opposite effect in the LOO analysis, seen in the plot 4.1. This result makes more intuitive sense as petal length and width are considered to have strong predictive power for the

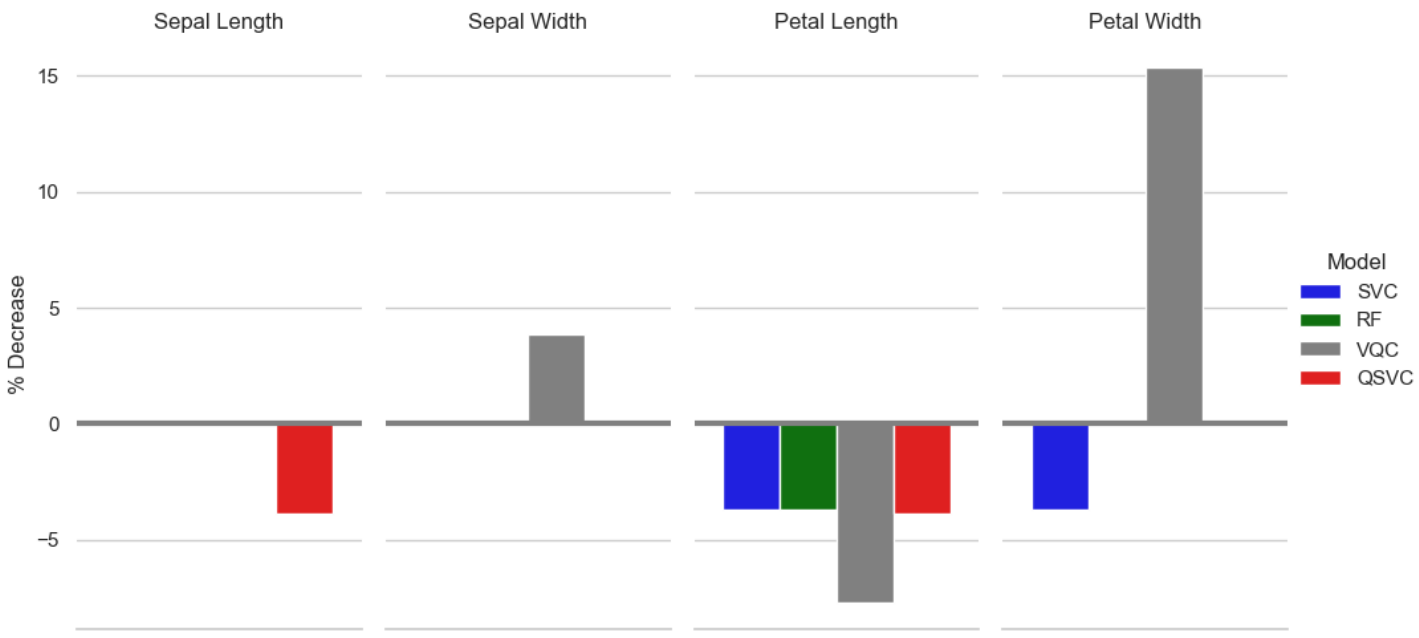

Figure 4.5: Overall LOO results.

Iris dataset. However, this may indicate that the model may be better off with just petal width rather than both petal width and petal length. Another possible cause is the Permutation Importance artificially introduces noise in the data so that it would be highly unlikely that a model's performance would increase when a feature is permuted.

Random Forest Permutation Importance

We can see in the plot 4.7 that petal width and petal length have the biggest impact when their values are permuted, while sepal length and sepal width seem to marginally increase the model's accuracy. These results are also quite different compared to the LOO analysis in plot 4.2, where the removal of petal length causes a slight increase in the model's accuracy. This result closely aligns with the SVC permutation importance results, which also follow the intuition of petal length and width being strong predictors. We could assume the same

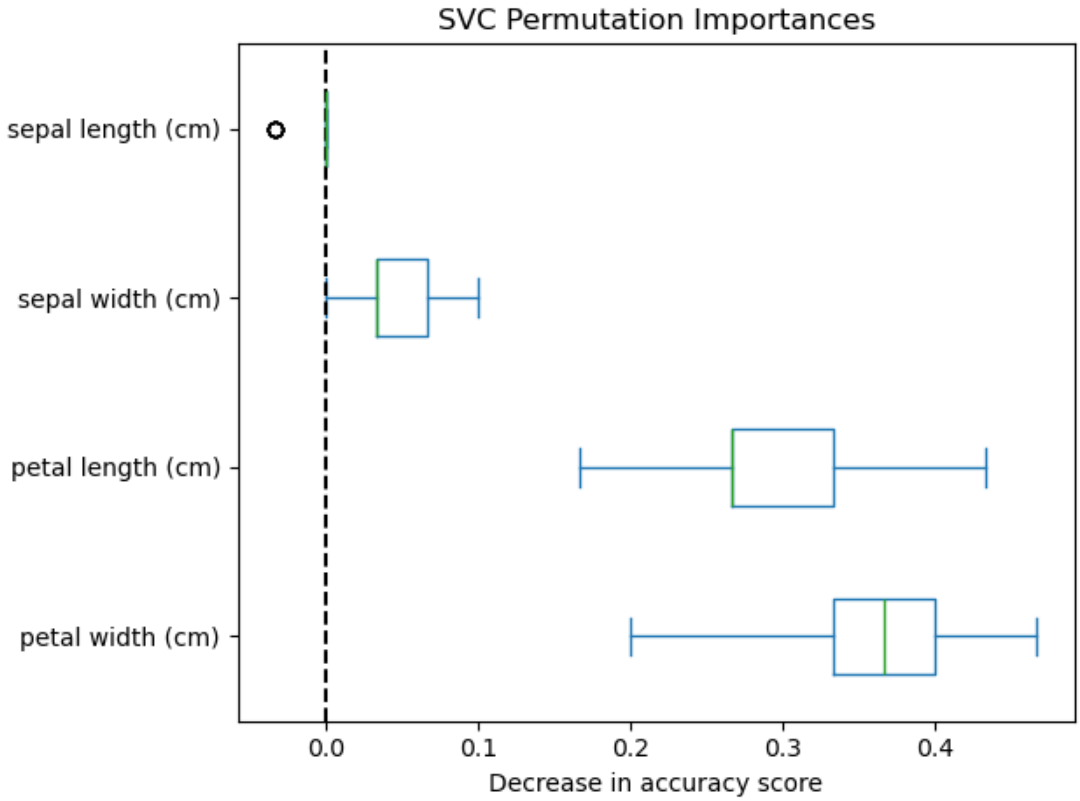

Figure 4.6: SVC Permutation Importance.

reasons mentioned in the SVC Permutation analysis 4.1.2 for the difference seen in this result and the LOO.

VQC Permutation Importance

The results of Permutation Importance, in plot 4.8 shows that the VQC has a much more balanced distribution of the features' importance, with sepal length having the second highest importance after petal width, which differs from the two classical models who both had a very similar ranking of petal width and length being the most importance with sepal length and width having little to none. Due do size of the whiskers on the plots and the high interquartile ranges further show how similarly the VQC considers the features. The permutation

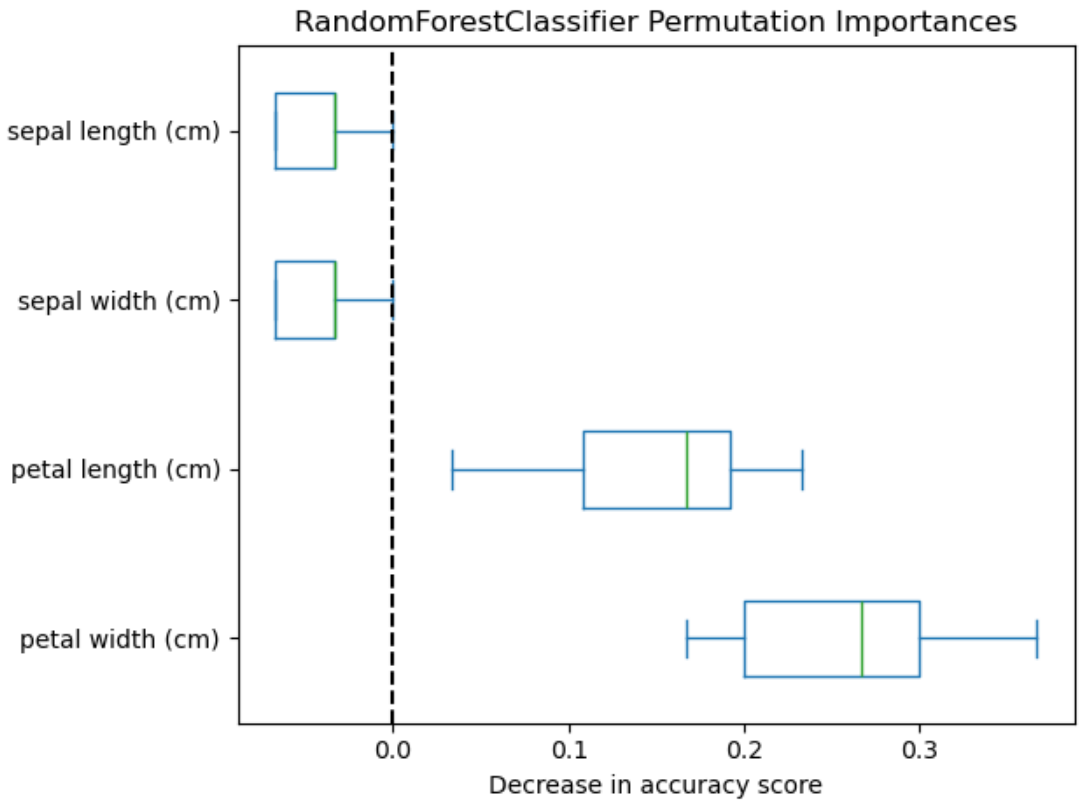

Figure 4.7: Random Forest Permutation Importance.

results still differ from the LOO analysis. Where the removal of petal length increased model performance, and removing sepal length did not affect performance.

QSVC Permutation Importance

We can see that again in plot 4.9 that the QSVC model follows a similar distribution of feature importance to the classical models, with petal length and petal width causing the biggest decrease in model accuracy when their values are permuted, and sepal width and sepal length causing slightly bigger effects but still much lower than the petal features. We can see that the LOO results demonstrated no effects with the removal of petal width and sepal width, and

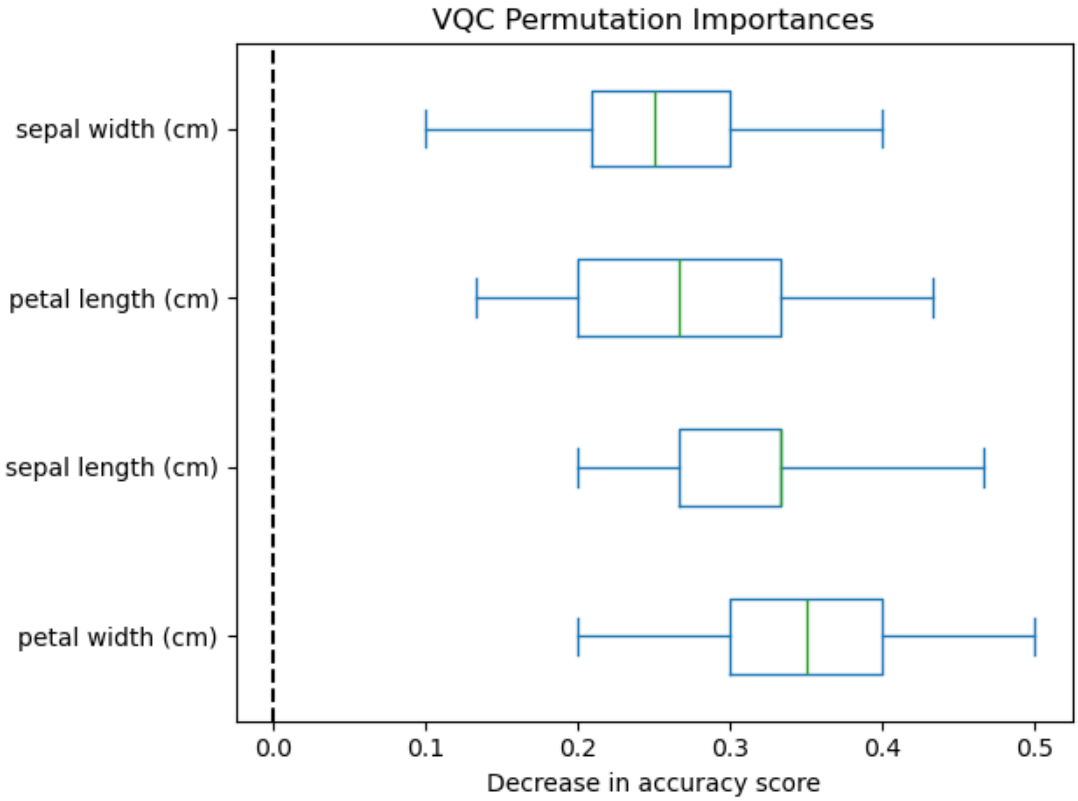

Figure 4.8: VQC Permutation Importance.

removing petal length and sepal length causes a slight increase in model accuracy.

Permutation Importance Summary

The plot 4.10 we can see that the VQC has a fundamentally different approach to feature importance compared to all the other models, including the QSVC, with almost equal weighing across all the features. For the other three models, we can a clear preference for petal length and petal width over sepal length and sepal width, with the lower wicks of the petal features being above the top wicks of the sepal features. The QSVC does seem to place slightly more importance on the sepal features than the classical models, but one should not

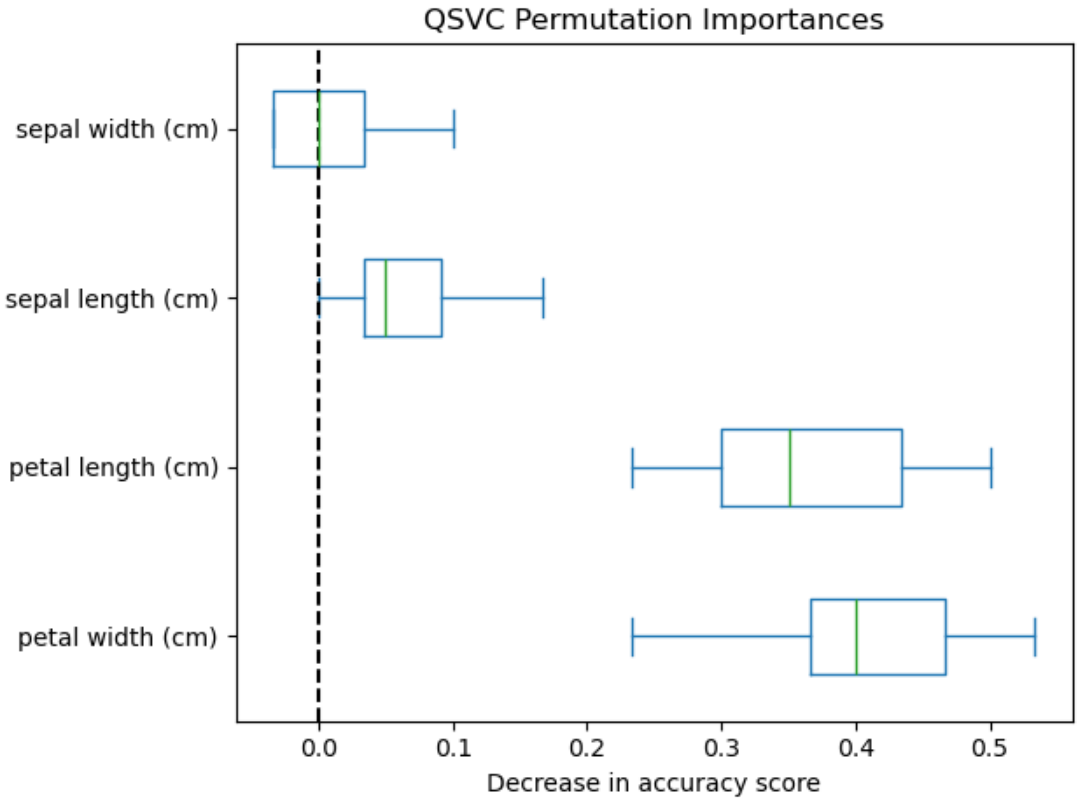

Figure 4.9: QSVC Permutation Importance.

that depending on the permutation of sepal width, the median effect was zero, with some permutations resulting in a better performance, but tending to worse, indicated by the high whisker.
Overall we can see that according to permutation importance analysis, petal features have a higher ranking importance than sepal. This seems to follow LOO results, which shows that petal features have a larger effect than the sepal counterparts, but seemingly in an opposite direction (4.5).

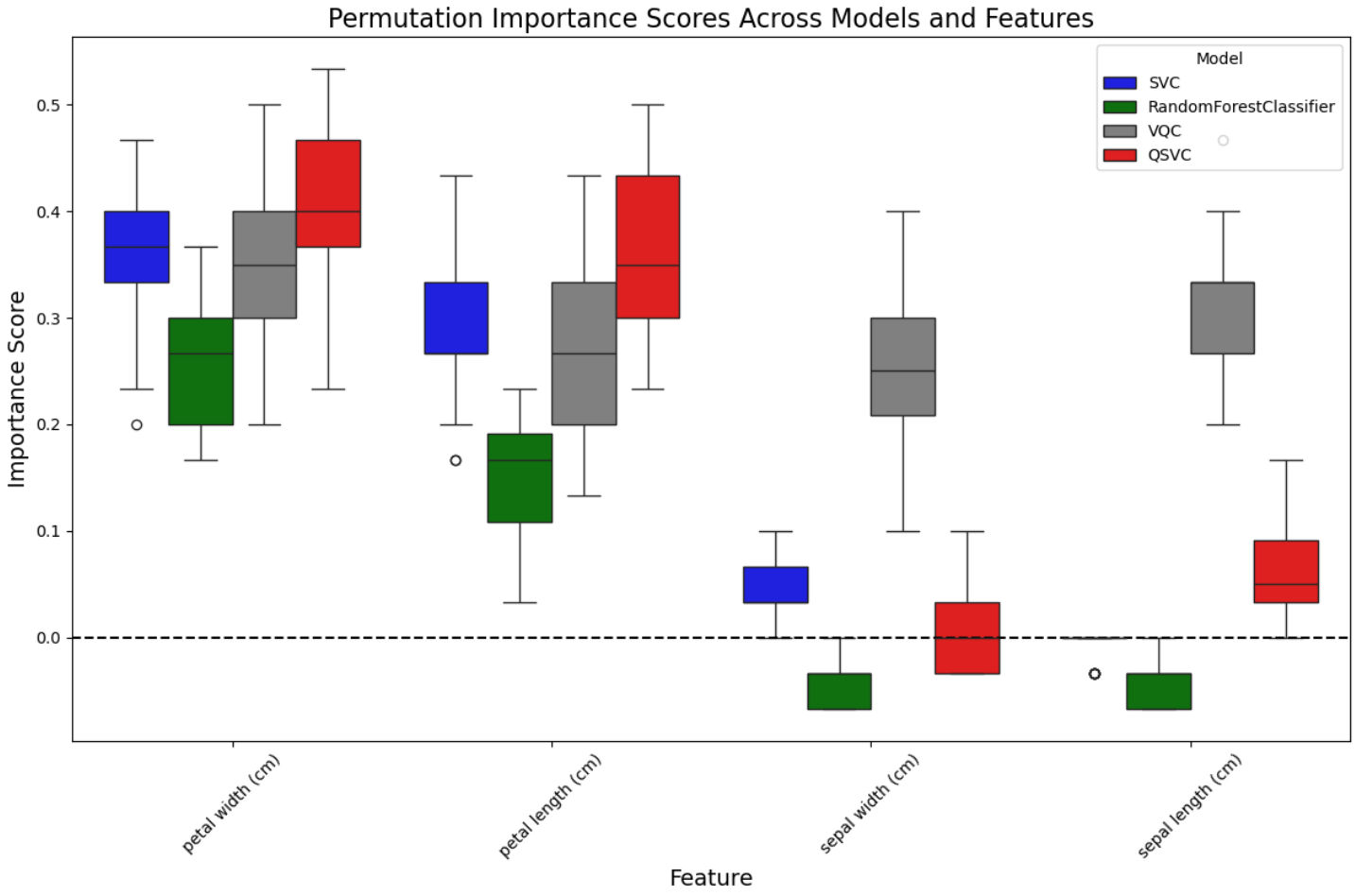

Figure 4.10: Overall Permutation Importance.

4.1.3 | Accumulated Local Effects Feature Importance

ALE computes feature importance differently to the LOO and Permutation Importance methods, rather than using model accuracy, ALE uses probabilities to measure how much the probability of making a classification changes concerning how the value of the feature changes. This can be aggregated and used to get an overall score of a model's feature importance, which will be analysed briefly. Due to the probabilistic nature of ALE, the VQC model is unable to produce a probability vector, as this functionality was deprecated with Qiskit 1.0

(46). Thus the QSVC model was used as the quantum comparison. For a clearer comparison, we will compare the SVC against its quantum counterpart.

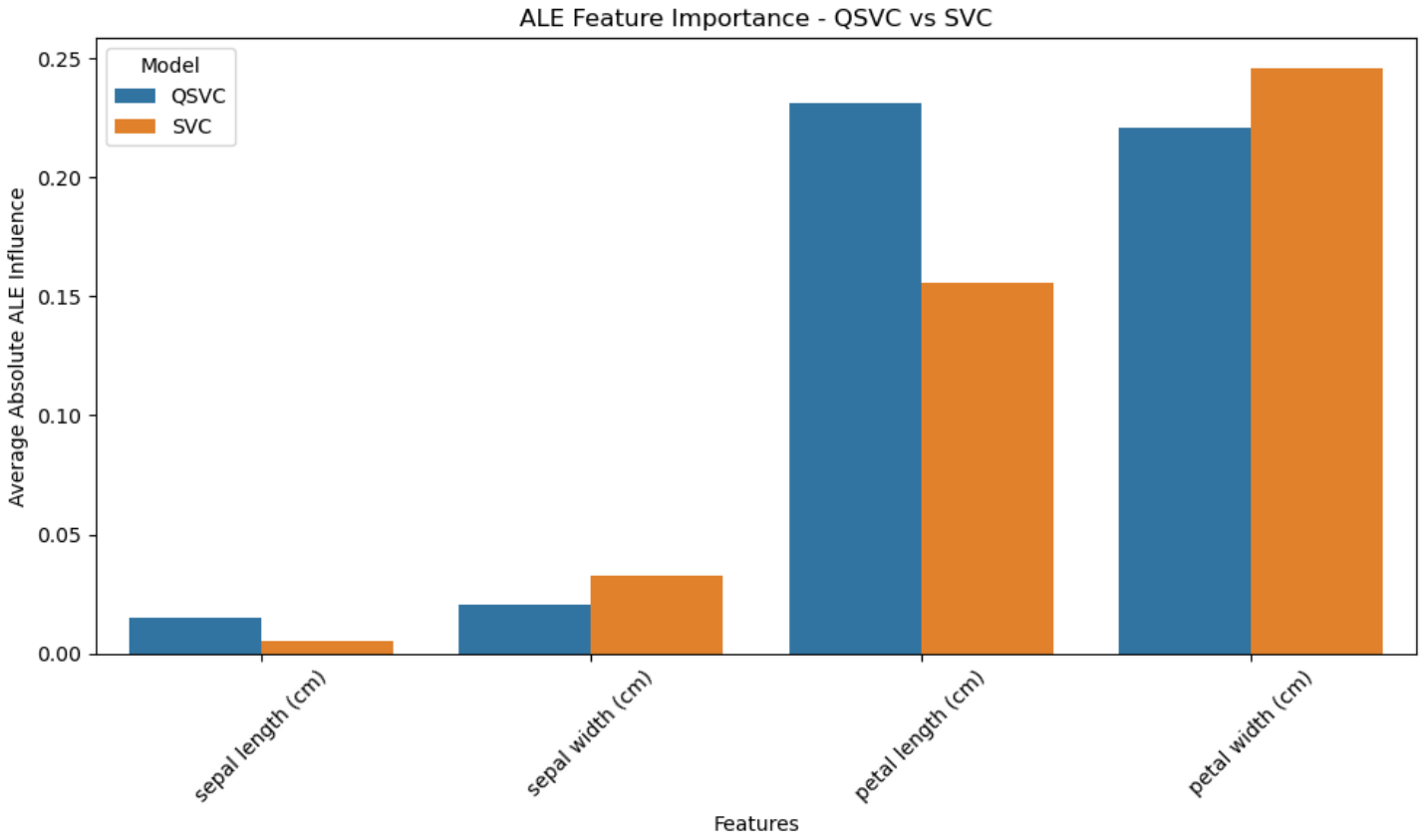

Figure 4.11: ALE Importance.

Plot 4.11 shows that both models follow a similar distribution of importance, however the QSVC results with higher ALE influence in three out of the four features, with the largest increase in the importance of petal length. But overall the ALE values of both the SVC and QSVC are highly similar.
A better way to utilise and understand ALE values is in plots 4.12 and 4.13. These plots show the ALE values for SVC and QSVC respectively. We can interpret the values as the increase or decrease in the probability of predicting the class of interest. We can see that petal length and width are important features due to how much the ALE values change for each of the values. For example,

we can see that the probability of predicting Virginica in both SVC and QSVC increases as petal length increases. We can also see in the QSVC that sepal width has more of an impact in the model prediction as values increase, while the same has very almost no effect in the SVC model.

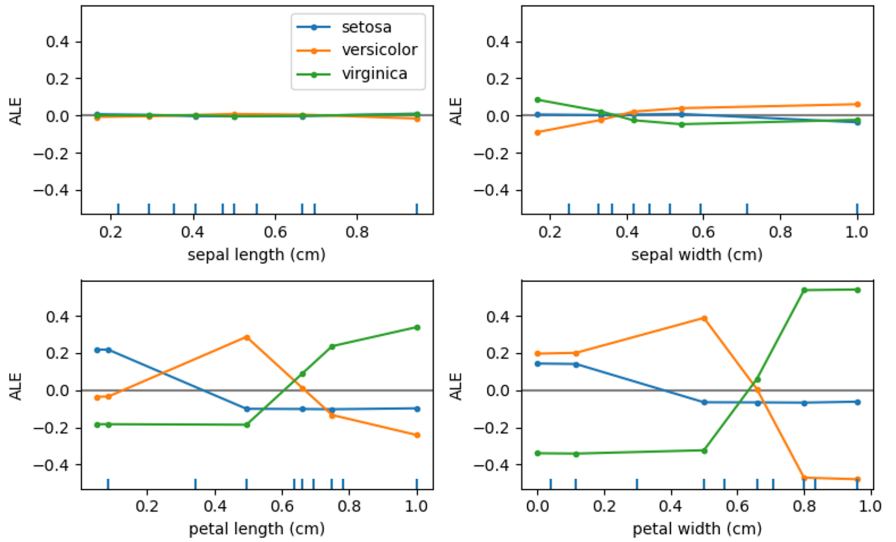

Figure 4.12: SVC ALE.

4.2 | Explainability

While feature importance can be useful for a general overview of what are the most important features in a model, more often than not we would like to see more specific explanations for a particular classification or individual result. The two methods used for this are Accumulated Local Effects and SHapley Additive exPlanations

4.2.1 | ALE For Class Explanation

We have shown in section 4.1.3 how ALE can be used to show how features affect a model's prediction, this can be inverted so that we can see how the different models differ in predicting a class. Given how we can see that the Versicolor

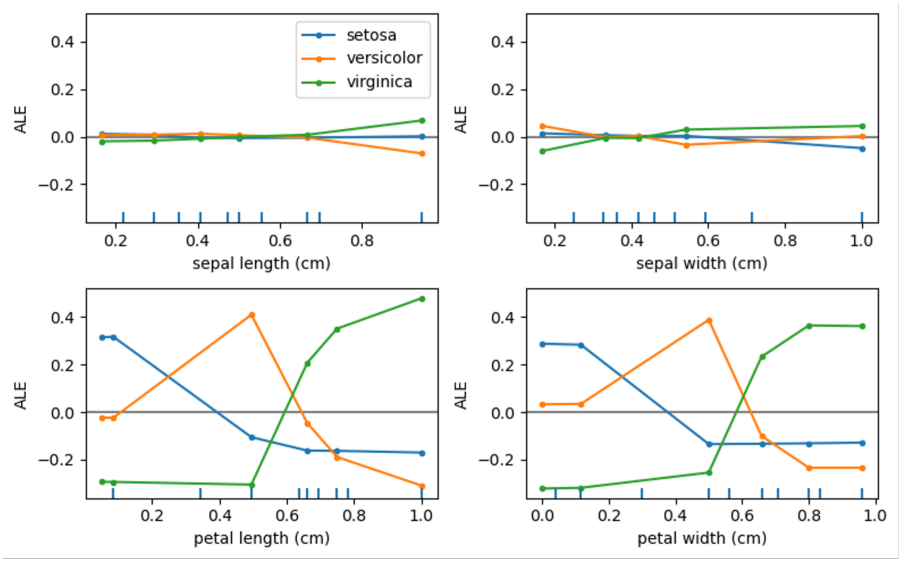

Figure 4.13: QSVC ALE.

and Virginica classes overlap in the pairs plot 3.1, we will investigate how ALE values can explain how the different models differ in predicting these classes.

Versicolor Prediction Model Comparison

In plot 4.14 we can see how the ALE values for each feature differ by model when the class of interest is Versicolor. We can see that all three models follow similar trends for each of the features, however, we can see that a higher value of sepal width resulted in the QSVC increasing its probability of predicting the sample as Versicolor, while the same value had zero impact in the SVC and Random Forest models. A very interesting observation is also made concerning sepal length. While a high value of sepal length indicated a higher probability of being Versicolor, a higher value of sepal length decreased the probability of making that same classification. However, we can see the magnitude of these changes is only a fraction of the decision when compared to the ALE values for petal length and petal width, with the sepal features giving ALE values in the region of 0 to 0.05 compared to the petal features who's effects are in the range of 0 to 0.4, which strongly decrease the probability of predicting a Versicolor class.

Plot 4.14 show that the three models are most likely to predict that a case is Versicolor when the normalised values of petal length and width are approximately between 0.25cm and 0.65cm, with ALE scores decreasing at either side of this interval.

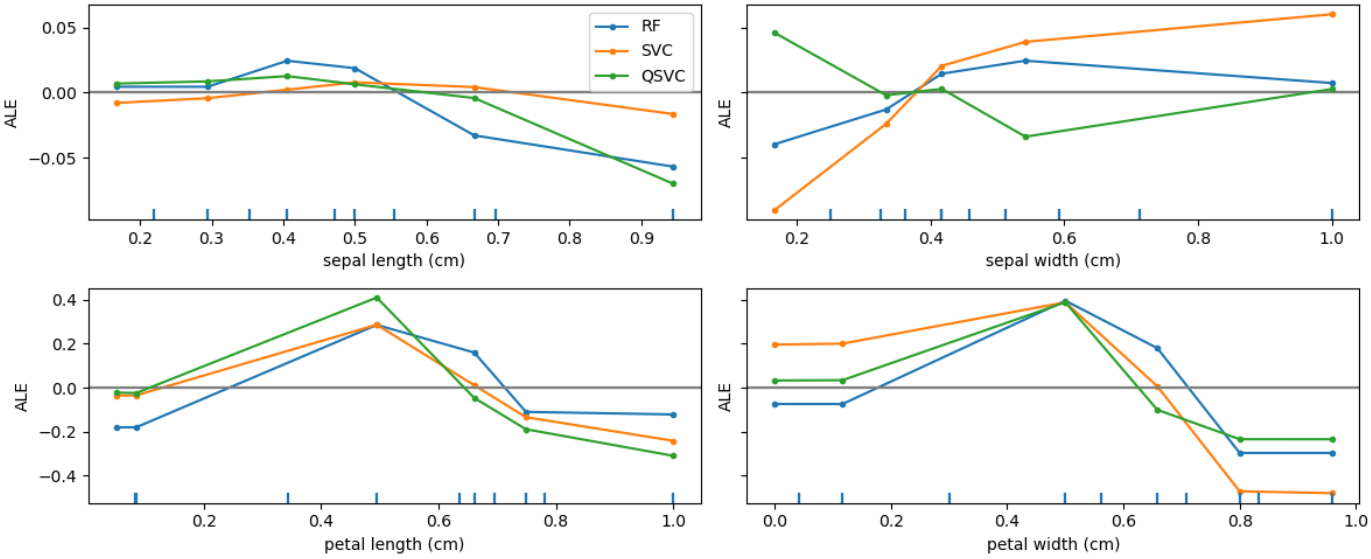

Figure 4.14: Versicolor ALE Values.

Virginica Prediction Model Comparison

How the ALE values fluctuate across the features and models is shown if plot 4.15. Similarly to the Versicolor analysis, we can see that the QSVC's ALE values increase with sepal length and width, but this time the effect is greatly magnified with the same pattern in petal length and width. What we could infer from this visualisation is that for high values of all four features, the QSVC model has a higher probability of predicting Virginica.

When the two plots 4.14 and 4.15 are viewed together we really see how petal width and length differentiate the model's predictions between the two classifications. With all three models significantly decreasing as petal measure-

ments increase in predicting Versicolor and at the same time, we can see the ALE values for Virginica increasing. This could also help us further understand where the model may make mistakes, particularly when an instance of petal length and width is in approximately the 0.6 to 0.7 regions.

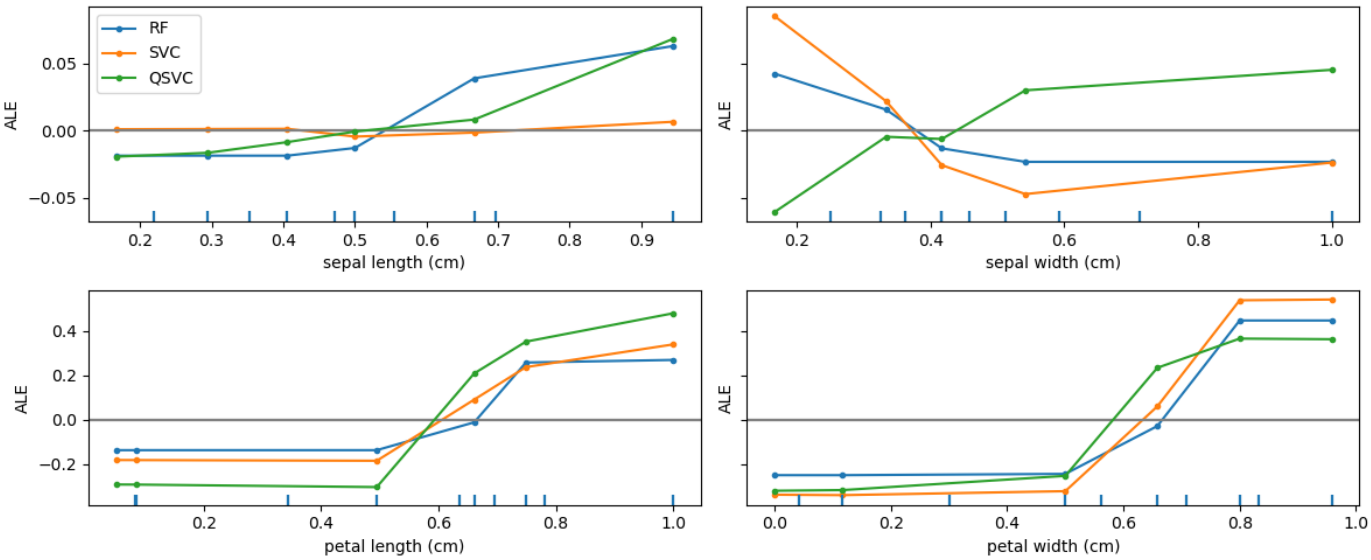

Figure 4.15: Virginica ALE Values.

We can see in this analysis how much more information can be inferred from the data by ALE by both classical and quantum classification models when analysis goes beyond the generality of feature importance analysis. Such analysis could also be made using SHAP values, which expands again on the ALE values by proving clear breakdowns of individual results.

4.2.2 | SHAP For Case-Specific Predictions

SHapley Additive exPlanations uses game theory coalition values to estimate the marginal contribution from a feature to a model's prediction. This is a method that can be used to quantify how features in a model contribute to individual prediction, rather than focusing on a feature importance method like Per-

mutation Importance, which has its merits when we want to get an idea of how a model would typically work, however, these generalisations cannot quantify the effects in individual cases.

The table 3.5, and the plot 3.4 show us that five out of the thirty test points are misclassified. We can see that point 4, which has a true label of Virginica was misclassified as Versicolor (label 1) by all models except the QSVC, this makes this an ideal candidate to use for further investigation. The feature values for point 4 can be found in table 4.5.

Feature	**Value**
0 (Sepal Length)	0.5
1 (Sepal Width)	0.25
2 (Petal Length)	0.78
3 (Petal Width)	0.54

Table 4.5: Features of the test point 4

SVC SHAP Values

Figure 4.16 is a waterfall plot of the SHAP values for sample 4. We can see that the expected output, or base value is 1.1, which would fall into the Versicolor class, we can see that Feature 3 (petal width) causes the largest contribution to this prediction, however the push is almost completely offset by feature 2 (petal width), this confirms what has been discussed previously, being petal width and length being the most important features in the SVC. The SHAP values quantify their importance in making this particular prediction. We can see the effects of sepal length and width are minimal in comparison. The contrasting effects by features two and three that the model uses could be explained that the model could be using the effect of petal width and length to cancel each other out in order to stay, on the base classification and align with the classification of 1, Versicolor.

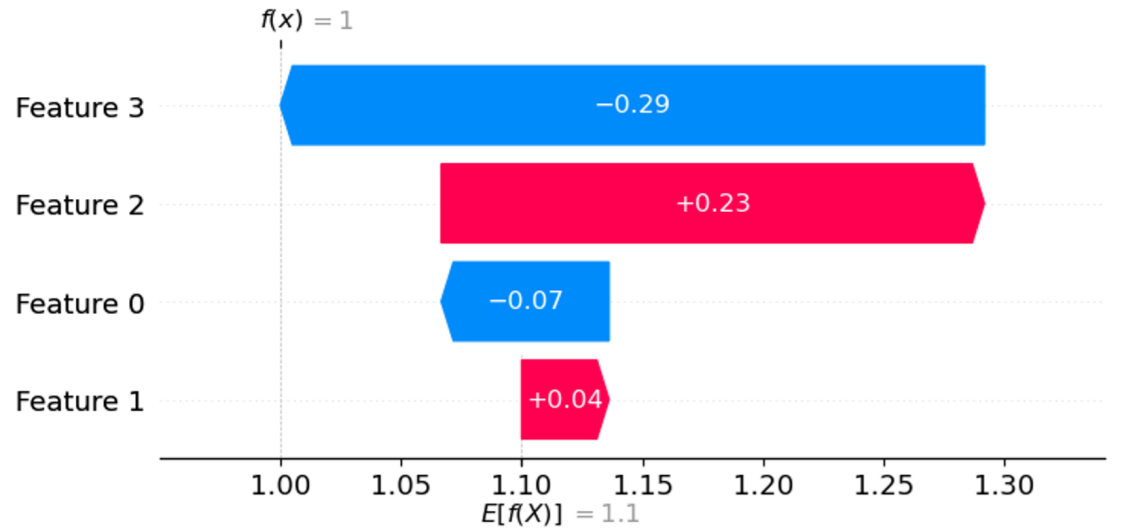

Figure 4.16: SVC SHAP

Random Forest SHAP Values

With respect to the Random Forest classifier, we can see that it tended to go towards a setosa classification, again we can see that petal length strongly pushes to the right, however in this model we can see that it is sepal width is a highly important feature for this case, which differs if we look at an aggregation of all the SHAP values in plot 4.18, where petal sepal width almost has no importance overall.

Incorrect classification aside, this shows how much SHAP values can demonstrate how important it is to be able to understand a models functionality on a case-by-case basis rather than using a general overview of a features importance.

QSVC SHAP Values

In our QSVC we can see that the model decided that all the feature values indicated a Virginica classification, with petal length and width being the two most important features by far, with sepal length only being assigned a SHAP value

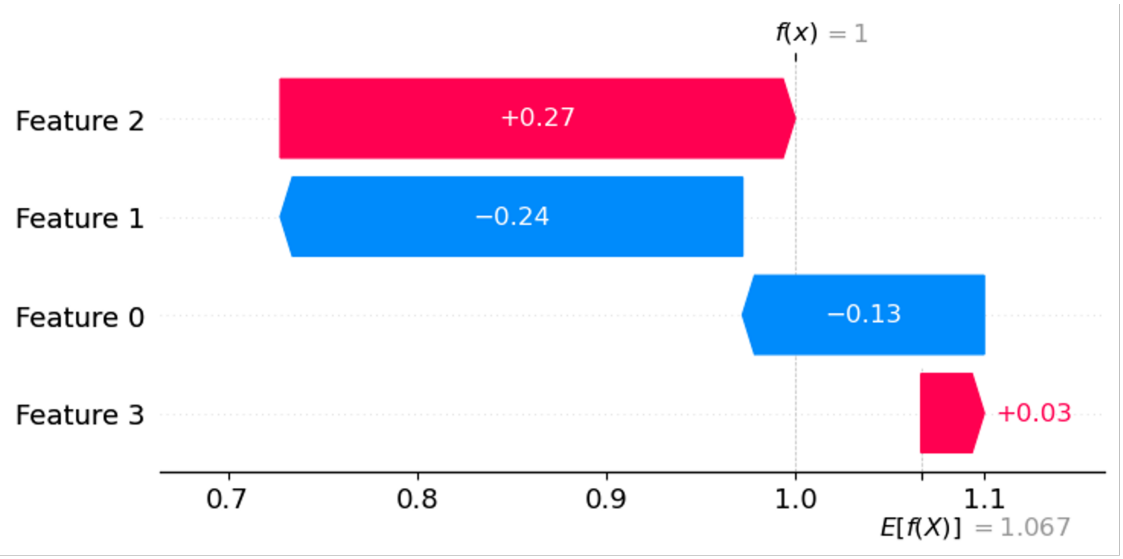

Figure 4.17: Random Forest SHAP

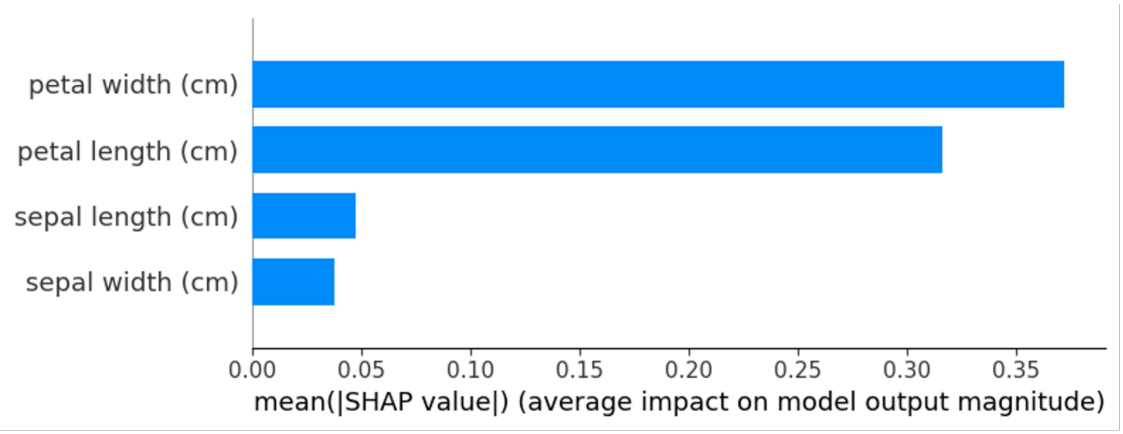

Figure 4.18: Random Forest SHAP Feature Importance

of 0.02 and sepal width not contributing at all in this classification. It is also an interesting thing to note that none of the feature values in the QSVC suggested that the prediction should change in the other direction.

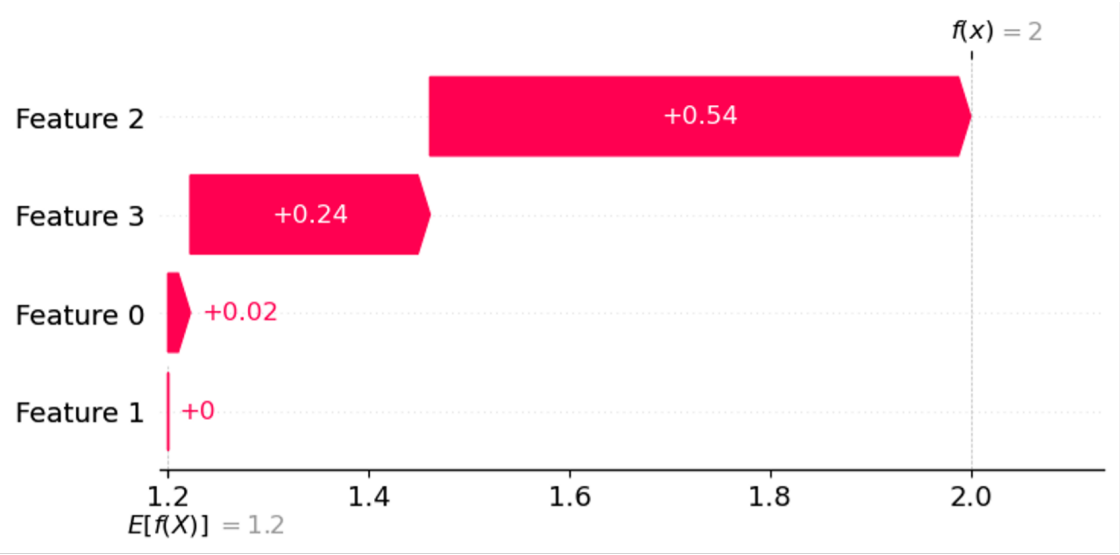

Figure 4.19: QSVC SHAP

4.2.3 | Summary of Analysis

Feature Importance

We have explored methods of feature importance across different models using three distinct methodologies: Leave-One-Out, Permutation Importance, and Accumulated Local Effects. Each method offered an insight into how features affect model performance and accuracy. We have then applied these methods to QML models.

LOO analysis highlighted the differential impact of features across models. For instance, SVC and Random Forest models showed an increase in accuracy upon the removal of petal features, suggesting a possible over-reliance or redundancy. In contrast, the VQC model exhibited significant performance drops, especially with the removal of petal width, indicating a high dependency on this feature. This disparity underscores the different ways models prioritize or handle feature dependencies.

Permutation importance further emphasized the significance of petal features in classical models, consistent with LOO results but provided additional details

on feature interactions and their impact on model accuracy. The QSVC model's feature importance distribution appeared more aligned with classical models, however, we can observe some variations in sepal features' impact.
ALE uses probabilities to gauge a feature's impact on the model, however, due to constraints in the implementation of the VQC, this model had to be omitted from this method. The ALE plots demonstrated how the QSVC follows the trends seen in the classical models, but displays a higher sensitivity to the changes across all the features with almost no flat lines in any of the ALE plots 4.14 4.15.

Model Explainability

Model explainability was addressed through ALE and SHAP analyses, focusing on how specific features influence individual predictions. ALE analysis demonstrated the variable influence of features across different classes, providing insights into how models differentiate between the classes Versicolor and Virginica based on feature values. This analysis allows us to uncover a better understanding of how the decision-making processes in both classical and quantum models work.

SHAP values offered deeper insights into the contribution of individual features to specific predictions, highlighting the interplay of features in models like QSVC, which successfully identified the Virginica class in test point 4, which was misclassified by the other three models.

5

Conclusions

In this project we investigated different methods of feature importance (LOO, Permutation Importance, ALE) and explainability (ALE, SHAP) and applied these methods to quantum machine learning models, comparing the results to their classical counterpart. In this comparison, we have set a small milestone into this field of explainable quantum machine learning. This project displays that these methods can indeed by applied to QML models, and so are potential candidates as tools in the unexplored field of explainable quantum machine learning, or Explainable Quantum Artificial Intelligence (XQAI). It was also a highly interesting takeaway in that we haven't necessarily seen a reliable example of the quantum advantage in our QML models, probably due to the small scale of the data, but there are still applications to be explored through QML, for example, researching gradients and making quantum software ready for machine learning applications (47).

It is also important to remember that these are methods that were designed, tested, implemented and are used with almost exclusively classical machine learning tools in mind, meaning that there is likely a need for not only amending these tools but perhaps a need to come up with completely new quantum-specific methods of explainability. (40)

5.1 | Reflection

Before the undertaking of this project, my knowledge of anything quantum was simply the result of some curiosity in the subject, that would summarise any aspects of the field of quantum physics into a notion of tiny particles doing funny things. I have read some articles and seen some videos on the strides taken in quantum computing in recent years, as well as the explosion of machine learning and AI. Due to my interest in both these fields the combination of these in a single project made the selection of this project almost a no-brainer.

Over the course of working on this project, I was able to learn so much about how quantum phenomena were able to be applied in computing. Working through IBM's summer school in Quantum Machine Learning, was a vital learning experience which allowed me to build some sense of intuition behind the workings of quantum circuitry. The process of learning Quantum Machine Learning with no real background in quantum fields, but seeing how it connected to the machine learning techniques and mathematical concepts that were covered in my course in Data Science and Analytics was extremely satisfying. The IBM course provided guides on not only the theoretical backgrounds and the matrix calculations behind some of the quantum gates, but also detailed tutorials where I got to build quantum algorithms, and an opportunity to run the code on an actual quantum computer. After the completion of this course reading the scientific literature on the subject matter was a much more fluid experience, and a less daunting task to try to grasp the algorithms discussed and the explanations of the functionality behind quantum processes.

One significant realization during this project was the recognition of some personal areas that could be improved upon for future projects, particularly in project management and time allocation. The ambitious scope of the project occasionally led to challenges in both academic and technological implementations, including the loss of five days of running code resulting in my computer crashing, losing the progress. Upon this reflection, I feel I also spend a disproportionate amount of time on the research and learning of the theory, time that may have been better spent in experimenting and actually implementing code. I also learnt the importance of code dependencies and version control,

with a Qiskit update resulting in a number of packages in use being deprecated. Shortly after this, I began utilising git and GitHub for almost every aspect of this project, and will be doing the same for all my future projects.

In conclusion, this project was not just about learning quantum machine learning but also about personal growth and applying the skills I have learned over the past four years. It has prepared me for future challenges, and I look forward to expanding on this project, beginning more of my own and applying the important lessons that were learned over the course of this project.

5.2 | Project Work Summary

The Gannt chart in figure 5.1 provides a high-level overview of how the project was completed over the course of the year. After selecting and being assigned the project, which was originally titled 'ML to QML for Critical Applications', which aimed at implementing QML and identifying what new inferences could be made when compared to a classical model. The first task was to research and study quantum machine learning, which took up most of the first semester, given the scope of the topic. Coming into the Christmas season I had the opportunity to implement my first quantum algorithms including running them on actual quantum hardware. At the beginning of the new year, I was able to begin implementing my first quantum classifiers. However, my initial quantum classifiers were giving results of 40% accuracy which of course then required heavy revision of the implementation and optimisation of the model and improving the selection of feature maps and ansatzes.

Around this same time in February, the initial plan was to implement QML on more complex, real-world data, in particular, omics data for looking at specific protein levels. However given the scale and complexity of this data, successfully implementing an operational quantum model was infeasible. After some reconsideration of the objective of the project, I had taken the notion of making inferences about the data and interpreted that as applying methods of feature importance in quantum machine learning, which subsequently expanded into the application of explainable methods, resulting in the final project 'Feature

Importance and Explainability in Quantum Machine Learning'.

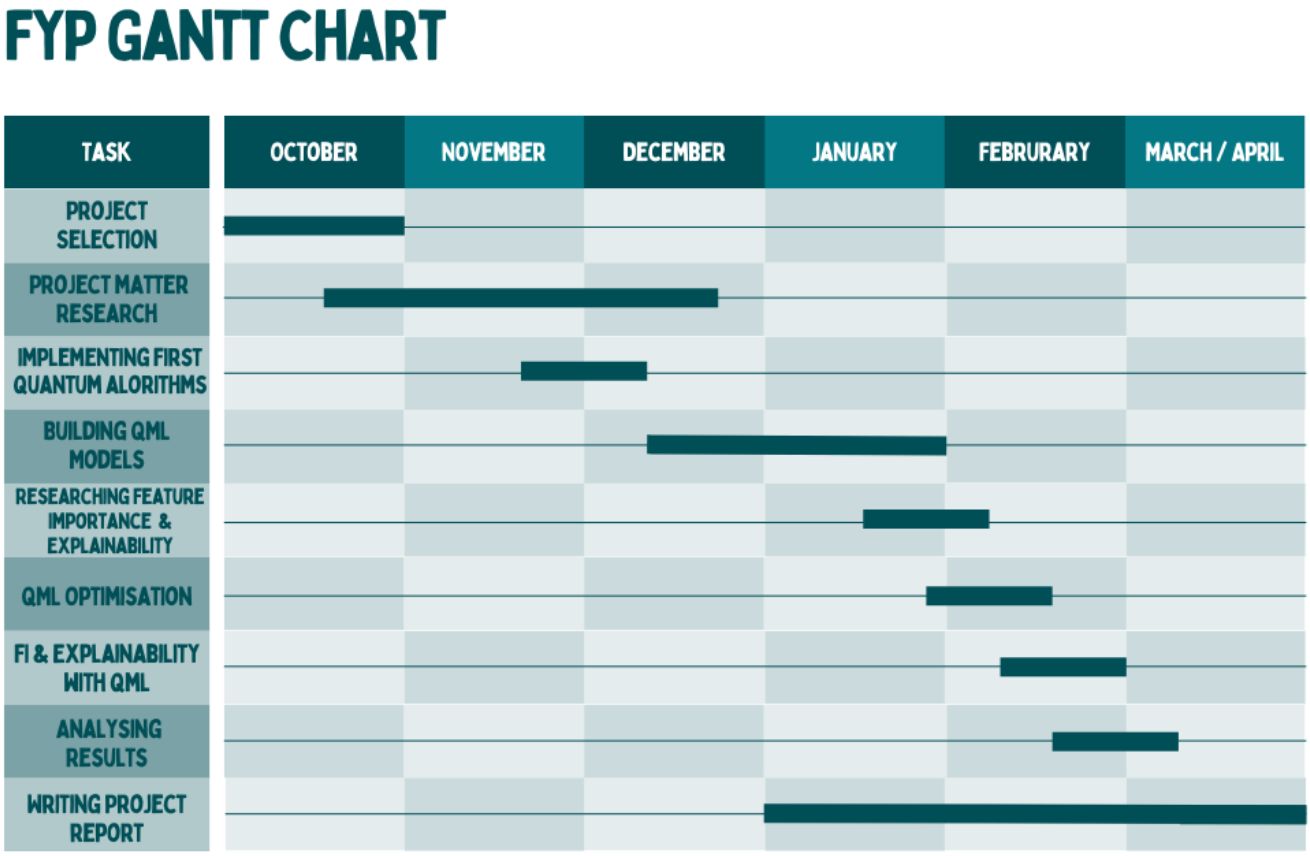

Figure 5.1: FYP Gannt chart

5.3 | Future Plans for the Project

The future plans for this project are to take the QML models and train them with more complicated and larger, real-world data. This would further enhance

the necessity of applying explainability techniques. To do this we would need to further optimise the quantum models, as well as design them optimally for running on a real quantum computer. The lack of models trained on an actual quantum computer was one of the greatest limitations of this project, as a simulator was used to avoid the extra time and complexity required, which was able to work due to the small scale of the data used. Due to the necessary time and resources this would take, attempting this on a larger dataset was beyond what was believed to be a realistic task for this project's time-frame. It would also be required to explore more methods of quantum machine learning and algorithms such as Quantum Boltzmann Machines, and perhaps explore the possibility of quantum-specific explainability techniques.

The notebooks containing the code, and the resources for this project can be found on the repository accessible at: https://github.com/LukePower01/ml-to-qml

References

[1] Andertoons, "Shakespeare's cat: "to be or not to be" - schrödinger's cat," https://andertoons.com/cat/cartoon/8046/shakespeares-cat-to-be-or-not-to-be-schrodingers-cat-both, 2024.

[2] A. Burkov, *The Hundred-Page Machine Learning Book*, 2019.

[3] M. Schuld, I. Sinayskiy, and F. Petruccione, "An introduction to quantum machine learning," *Contemporary Physics*, vol. 56, no. 2, Oct 2014.

[4] D. J. Wineland, "Nobel lecture: Superposition, entanglement, and raising schrödinger's cat," *Reviews of Modern Physics*, vol. 85, no. 3, p. 1103–1114, Jul 2013.

[5] V. Belle and I. Papantonis, "Principles and practice of explainable machine learning," *Frontiers in Big Data*, vol. 4, Jul 2021.

[6] E. Horel, "Towards explainable ai: feature significance and importance for machine learning models," Ph.D. dissertation, Stanford University, 2020.

[7] [Online]. Available: https://scikit-learn.org/stable/modules/permutation_importance.html

[8] S. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," 2017.

[9] [Online]. Available: https://shap-lrjball.readthedocs.io/en/latest/

[10] C. Molnar, "Interpretable machine learning," Aug 2023. [Online]. Available: https://christophm.github.io/interpretable-ml-book/ale.html

[11] "IBM Quantum and Qiskit," https://www.ibm.com/quantum/qiskit, 2024.

[12] GeeksforGeeks, "Supervised ml process," https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.geeksforgeeks.org%2Fsupervised-unsupervised-learning%2F&psig=AOvVaw0oefQ1g2qt9Sp55SE-R6VV&ust=1711472059130000&source=images&cd=vfe&opi=89978449&ved=0CBIQjRxqFwoTCNCh-cPwj4UDFQAAAAAdAAAAABAD.

[13] "What is a support vector machine?" https://datatron.com/what-is-a-support-vector-machine/, 2024.

[14] Mazen Ahmed, "Non-linear support vector machines explained," https://linguisticmaz.medium.com/support-vector-machines-explained-ii-f2688fbf02ae, 2021.

[15] E. Wolsztynski, "Support vector machines."

[16] ——, "Random forest."

[17] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, vol. 549, no. 7671, 2017.

[18] H.-Y. Huang, M. Broughton, J. C. S. Chen, J. Li, M. Mohseni, H. Neven, R. Babbush, R. Kueng, J. Preskill, and J. R. McClean, "Quantum advantage in learning from experiments," *Science*, vol. 376, no. 6598, 2022.

[19] IBM, "Quantum advantage," https://www.ibm.com/thought-leadership/institute-business-value/public/static/images/Quantum-Report/Figure3.svg.

[20] D. Diego, M. P. da Silva, C. A. Ryan, A. W. Cross, A. D. Córcoles, J. A. Smolin, J. M. Gambetta, M. J. Chow, and B. R. Johnson, "Demonstration of

quantum advantage in machine learning," *npj Quantum Information*, vol. 3, no. 1, 2017.

[21] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, "Supervised learning with quantum-enhanced feature spaces," *Nature*, vol. 567, no. 7747, 2019.

[22] Qiskit, "Platypus/notebooks/summer-school/2021 at main · qiskit/platypus." [Online]. Available: https://github.com/Qiskit/platypus/tree/main/notebooks/summer-school/2021#

[23] S. Pattanayak, *Quantum Machine Learning with python: Using CIRQ from Google Research and IBM qiskit*. Apress, 2021.

[24] I. Glendinning, "The bloch sphere," in *QIA meeting. Vienna*, 2005.

[25] D. Qiu and S. Zheng, "Generalized deutsch-jozsa problem and the optimal quantum algorithm," *Physical Review A*, vol. 97, no. 6, p. 062331, 2018.

[26] L. K. Grover, "A fast quantum mechanical algorithm for database search," *Proceedings of the 28th annual ACM symposium on Theory of computing*, 1996.

[27] Qiskit, "Building a quantum variational classifier using real-world data," 2021, https://medium.com/qiskit/building-a-quantum-variational-classifier-using-real-world-data-809c59eb17c2.

[28] A. Elbeltagi, "Vqc schematic," https://www.researchgate.net/figure/Schematic-representation-of-a-variational-quantum-circuit-VQC-VQC-is-created-by$_fig1_358006357$.

[29] "Quantum kernel machine learning - qiskit machine learning 0.7.2." [Online]. Available: https://qiskit-community.github.io/qiskit-machine-learning/tutorials/03_quantum_kernel.html

[30] M. Rath and H. Date, "Quantum data encoding: A comparative analysis of classical-to-quantum mapping techniques and their impact on machine learning accuracy," 2023.

[31] [Online]. Available: http://ndl.ethernet.edu.et/bitstream/123456789/73371/1/320.pdf

[32] Qiskit, "Ibm quantum documentation," 2021, https://docs.quantum.ibm.com/api/qiskit/qiskit.circuit.library.ZZFeatureMap.

[33] A. Das and P. Rad, "Opportunities and challenges in explainable artificial intelligence (xai): A survey," *arXiv preprint arXiv:2006.11371*, 2020.

[34] "Permutation importance," https://www.google.com/url?sa=iurl=https

[35] D. W. Apley and J. Zhu, "Visualizing the effects of predictor variables in black box supervised learning models," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 82, no. 4, pp. 1059–1086, 2020.

[36] I. Piatrenka and M. Rusek, "Quantum variational multi-class classifier for the iris data set," in *International Conference on Computational Science*. Springer, 2022, pp. 247–260.

[37] A. Baughman, K. Yogaraj, R. Hebbar, S. Ghosh, R. U. Haq, and Y. Chhabra, "Study of feature importance for quantum machine learning models," *arXiv preprint arXiv:2202.11204*, 2022.

[38] P. Steinmüller, T. Schulz, F. Graf, and D. Herr, "explainable ai for quantum machine learning," *arXiv preprint arXiv:2211.01441*, 2022.

[39] G. Abdulsalam, S. Meshoul, and H. Shaiba, "Explainable heart disease prediction using ensemble-quantum machine learning approach," *Intell. Autom. Soft Comput*, vol. 36, no. 1, pp. 761–779, 2023.

[40] R. Heese, T. Gerlach, S. Mücke, S. Müller, M. Jakobs, and N. Piatkowski, "Explaining quantum circuits with shapley values: Towards explainable quantum machine learning," *arXiv preprint arXiv:2301.09138*, 2023.

[41] [Online]. Available: https://scikit-learn.org/stable/datasets/toy_dataset.html

[42] [Online]. Available: https://docs.quantum.ibm.com/api/qiskit/qiskit.circuit.library.RealAmplitudes

[43] [Online]. Available: https://docs.quantum.ibm.com/api/qiskit/qiskit.circuit.library.EfficientSU2

[44] [Online]. Available: https://docs.quantum.ibm.com/api/qiskit/0.44/qiskit.algorithms.optimizers.COBYLA

[45] [Online]. Available: https://docs.quantum.ibm.com/api/qiskit/0.19/qiskit.aqua.components.optimizers.SLSQP

[46] [Online]. Available: https://docs.quantum.ibm.com/api/qiskit/0.19/qiskit.aqua.algorithms.VQC

[47] M. Schuld and N. Killoran, "Is quantum advantage the right goal for quantum machine learning?" *PRX Quantum*, vol. 3, no. 3, Jul. 2022. [Online]. Available: http://dx.doi.org/10.1103/PRXQuantum.3.030101