AI Detector *by SciSpace*

qml final.pdf • 24 November 2025
7205 words (45754 characters)

Essentially Human 1%

The text is written almost entirely by a human, with little to no AI assistance.

AI weightage	Content weightage	Sentences
H Highly AI written	**1%** Content	2
M Moderately AI written	**8%** Content	27
L Lowly AI written	**7%** Content	22

Quantum-Enhanced Coconut Yield Prediction: A Comparative Study of Second-Order and Third-Order Quantum Machine Learning Models

Ms.Vaishali V[1]
Department of Information Technology,
Thiagarajar College of Engineering,
Madurai, Tamilnadu, India.
vvaishali@student.tce.edu

Ms.Thisha M[1]
Department of Information Technology,
Thiagarajar College of Engineering,
Madurai, Tamilnadu, India.
thisha@student.tce.edu

Dr. Pudumalar S[1*]
Department of Information Technology,
Thiagarajar College of Engineering,
Madurai, Tamilnadu, India.
spmit@tce.edu

Abstract—
Guessing crop output right helps feed people better while saving supplies. Old-school number crunching methods like Support Vector Machines were tried for forecasting harvests, yet they usually can't manage messy real-world farm data patterns. Our work checks if quantum-powered learning can boost forecast precision by tapping into quantum computing strengths. A mix of quantum and regular brain-like systems - Variational Quantum Circuits hooked up with standard nets - is suggested here, aiming at guessing coconut crops more accurately. Loads of prep steps clean the info: adjusting value ranges plus applying special quantum-style data setups that help the system grasp tricky details easier. In this case, two QML setups got built and checked - QML-2.0 hit 85.56% accuracy, yet QML-3.0 did even better with 91.60%. Actually, the improved way of measuring in QML-3.0, using Pauli-Z on the initial qubit, captures features more clearly while handling complex patterns. Although training takes more time, these models handle unseen data well, especially across many variables. The findings showed quantum tricks like embedding and linking particles really lift performance. This study highlights how QML could shift things in farming tech, plus delivers a practical, precise tool for predicting harvests.

Keywords - Quantum Circuits,1st order,2nd order, Quantum Machine learning, prediction

Introduction

Agriculture keeps the world's economy running, feeding people, supplying materials, while offering jobs to countless folks. To keep hunger at bay and economies strong, steady harvests are a must. Still, today's farming runs into trouble - weather acts up, dirt wears out, clean water gets scarce, resources often go to waste. When crops fail regularly, farmers lose money and shelves run low. Crop yield estimates matter because they support smarter choices, reduce risks, plus improve how resources are used. Reliable predictions let farmers adjust to weather shifts, act on solid info, also boost output.

Folks usually rely on stats or tools like SVM plus Random Forest to guess crop output. Even though those tricks help, they struggle when patterns get messy, data gets bulky, or weather acts unpredictable.

Old-school machine learning needs lots of hand-crafted features, can't handle messy variable links well, while also reacting badly to random data glitches - this weakens trust in results. On top of that, teaching big ML systems uses up serious computing power, so they're often too slow when speed matters.

Though old-school methods such as SVM, RF, or neural nets are common in farm forecasting, they've got some issues - like struggling to handle tricky relationships

Older machine learning methods usually depend on hand-crafted features along with fixed ideas about how variables connect. That restricts their capacity to detect complex, curved links among various climate conditions influencing coconut output.

2. Fine-tuned response when data shifts slightly

Agricultural info changes fast - seasons shift, weather acts up, and soil varies. Because of this chaos, older machine learning systems often fail to adapt, so their guesses get shaky when things look different.

3. Computational inefficiencies

Training complex models on big data needs serious computing power. When info gets more detailed, the time and memory needed go up - so older methods struggle to keep up with fast farming choices.

4. Reliance on crafting features

Old-school machine learning needs lots of hand-picked features plus heavy data cleanup to work well. Because it depends on human adjustments, it's less flexible - struggles to grow across various crops or areas.

These problems push us toward smarter computing methods - take Quantum Machine Learning (QML), for instance. Instead of old-school machine learning, QML uses ideas from quantum physics to work better. Things like particles being in multiple states at once help it perform differently. Because of linked particle behavior, results can change in unique ways. Even wave-like interactions play a role in how it calculates stuff. These traits help QML handle big sets of farm data faster, spotting complex trends regular ML usually overlooks.

Using quantum computers together with learning systems, QML gets better at spotting patterns while cutting down on processing power and boosting forecast reliability. In farming, it helps predict harvests, check soil conditions, or weigh weather effects - giving flexible tools that grow with demand. As farms aim for greater precision and quantum tech keeps improving, interest in QML is rising fast. Instead of sticking to old techniques, this approach opens doors to quicker, sharper predictions - helping feed the planet sustainably ahead.

Sharma and team (2021) looked into using quantum-powered support vector machines to sort crops, noticing better results than standard machine learning approaches. Instead of conventional methods, their work showed that embedding data with quantum features helped spot patterns more effectively in farm-related information. They found these quantum-

based kernels worked faster and sharper when identifying plant types from color and surface details. Unlike older SVM versions, this approach handled unseen data better while boosting precision.

Wang and team (2022) tested a mix of quantum and regular computing methods to forecast wheat output, using quantum-style neural nets; results showed these models handled complex crop data better than standard deep learning setups. Instead of traditional tools, they used quantum-inspired circuits - this shift helped spot finer trends in weather and soil conditions tied to harvest changes.

Patel's team in 2023 built a quantum-based method to guess soil water levels - using embedded features from quantum systems helped it adapt faster when examples were few. Work they did shows QML might grow well on farms, since their model kept results steady even where dirt changes often. Unlike old-school straight-line or curve-fitting math, this one made fewer mistakes because it handled messy real-world differences better.

Liu et al. (2023) used quantum deep learning to spot crop diseases, swapping classical CNNs with quantum ones to get better results. Instead of traditional methods, they tried quantum convolutional networks and saw a boost in accuracy. While focusing on plant illness detection, their work showed how QML helps catch problems earlier. They tested several ways to encode data into quantum states, checking how each type of entanglement changed learning speed. In the end, complex quantum setups handled image details linked to diseases better than simpler versions.

Some research shows QML can handle many farm-related tasks - not just guessing harvest size but also sorting data, fine-tuning processes, or helping make choices. Better speed, fewer mistakes, and smarter use of supplies show how quantum tools are slowly becoming more useful in smart farming.

II Literature Review

In the last ten years, predicting crop output became more popular because food needs went up and weather got less predictable. Models such as SVM, RF, or ANN popped up across farming studies. Take Wang's team in 2007 - they brought out HHL, a quantum trick that speeds up solving linear problems way faster than usual. That core idea feeds into quantum versions of linear regression, useful when tracking how nature affects harvests.

Starting from this idea, Lloyd and team back in 2013 came up with early quantum approaches for learning tasks - both guided and unguided - like quantum versions of support vector machines and grouping techniques. Instead of traditional ways, their research showed faster processing through quantum boosts, helping classify real-world data like spotting unhealthy crops or sorting soil types. Because old-school methods hit walls, scientists turned to quantum-powered machine learning setups. Schuld's group in 2015 pushed forward the notion of upgrading learning using quantum advantages, pointing out that mapping data into quantum spaces unlocks stronger pattern recognition by using vast state dimensions. This effort set the stage for blending quantum-style data conversion into classic similarity-based models.

In backing quantum data squeeze, Jeck et al. (2016) used a Random Forest setup - not to link parts, yet to guess rice output from weather factors in South Korea. This traditional way gave a starting mark in farm modeling; still, it showed issues with curved patterns and missing info - problems newer quantum styles try to fix. Romero's team (2017), meanwhile, brought up compact quantum encoders meant to shrink data size smartly. While first aimed at chemical systems on quantum level, the trick might just work well for heavy farm-related inputs like orbital snaps or field sensors. In that year, Kandala's team rolled out a leaner version of VQE - cutting circuit depth so it could run on today's glitchy quantum chips. That sparked ideas for how we built the trial structures in QML-2.0 and QML-3.0 here. Setting the scene for these kinds of advances.

At the same time, Preskill (2018) described the NISQ phase, pointing out what current quantum machines can and can't do. Because of this view, researchers now build simpler circuits that fit real-world farming data tasks. That very year, Perdomo-Ortiz and team (2018) looked into mixtures of classical and quantum setups, showing how quantum-powered learning handles messy, patchy, or irregular information better. Thanks to their findings, using QML for forecasting crop output across varied climates makes solid sense. Mitarai et al. (2018) introduced quantum circuit learning - a mix of adjustable quantum setups with standard tuning methods. While using very few qubits, their setup worked well on prediction jobs, so it could fit small clinics or remote areas. Around the same time, Schuld and Petruccione (2018) laid out how supervised learning works on quantum machines. That guide covered building circuits, setting up loss checks, along with mixed training tricks - each shaping how these quantum systems are built today.

In generative setups, Benedetti and team (2019a) tested a method with basic quantum circuits -

showing they could mimic tricky data patterns well, even when quantum power was tight. In another work, same group (2019b) looked into tweakable quantum circuits as tools for learning tasks, checking how flexible and trainable they were for labeled problems on current-gen quantum devices. Havlíček and team (2019) built quantum kernel techniques by leveraging entangled states, helping shape powerful feature representations. This method boosted performance on tiny datasets - useful in farming research where tagged examples are scarce. Chen's group (2020) used tunable quantum circuits within deep reinforcement learning, blending strategies for evolving environments. That progress opens doors for live uses like smart watering or planting plans adjusted to weather shifts. To boost how clearly models work plus handle signals, Griol-Barres and team (2021) used variational quantum circuits to spot faint signals in messy data. That method could help catch small clues in soil condition or how bad pests are spreading.
In 2021, Cerezo's team dug into variational quantum algorithms, looking at how well they express solutions, handle flat gradients, or stay trainable - these insights shaped the way we tuned our circuits. Instead of just stacking ideas together, their work guided key choices here. Around that time, Sharma's group tested quantum kernels on farm-related image sorting using multi-band satellite data; because their method beat standard SVM performance, it gave us confidence to use similar encodings for spotting crop patterns. Amaro et al. (2021) linked quantum mechanics with machine learning, speeding up how we find new materials through generative models. Instead of traditional methods, they used a data-focused approach - similar to building fake datasets for smart farming tools that guess crop output.
To make things work better, Wang's team in 2022 built a mix of quantum and neural models to predict wheat output. The method worked well on spotting complex links between weather and soil traits - similar to what QML-3.0 uses. A year later, Patel's group introduced a quantum-based way to estimate soil wetness. It turned out that encoding features with quantum techniques helped adapt faster when data was scarce or areas varied widely. Liu's team in 2023 used quantum-style neural nets to spot plant illnesses - accuracy got a boost compared to regular setups. Circuit layers and how images were coded played key roles when checking crops this way.
Lastly, Setiadi and team (2024) introduced a mix of quantum computing and deep learning to predict rice output. Instead of ignoring fluctuations, their approach handled climate-related data shifts well - so it works in areas with erratic weather or many types of crops.

III Dataset Description

The dataset holds 500 entries along with 10 main factors affecting coconut output - so it's useful for farming studies or prediction work. It's got essential climate and crop-related details.

1. Rainfall – it's ongoing. Temperature changes nonstop. Soil pH stays variable. NDVI keeps shifting. Fertilizer use never stops growing. Water access fluctuates daily. Sunshine hours? Always in motion.

2. Categorical or binary traits: whether farming is organic (yes or no).

3. Coconut output per hectare measured in tons.

This collection covers a wide range of weather, land, or farming factors that affect coconut growth. Because it includes both number-based and category-based data, building smart prediction tools - like machine learning or quantum models - is easier. These tools help forecast harvests better, support quicker choices, while making farm resource use smarter.

This dataset tackles today's tough farm challenges shaped by shifting weather patterns - offering solid support for smart, tech-powered farming methods that aim to last. As seen in Table 1, a slice of real data used for testing and learning is displayed; meanwhile, the ten factors mentioned earlier feed directly into quantum machine learning models for smarter trait analysis.

Rainfall (mm)	Temp (°C)	Soil pH	NDVI	Fert. Usage (kg/ha)	Coconut Yield (t/ha)	Water Avail. Index	Sunshine Hours	Pest Severity	Organic (0/1)	Yield (nuts/tree/yr)
1311.63	31.98	5.87	0.56	89.26	1.01	2.80	6.56	0.14	1	4.49
2406.36	30.36	6.58	0.54	87.05	1.55	6.55	8.92	0.95	0	6.87
1990.79	28.10	7.25	0.31	185.94	1.62	7.69	11.52	0.33	0	7.20

1737.45	33.14	6.96	0.47	87.43	1.22	7.64	6.24	0.66	0	5.42
896.44	31.85	7.11	0.49	90.79	1.03	5.69	7.75	0.75	0	4.59
896.39	26.63	6.82	0.50	163.91	1.27	1.62	7.25	0.81	1	5.63
710.36	34.11	6.88	0.59	117.46	0.92	4.34	7.43	0.95	0	4.08
2245.73	33.23	7.20	0.57	166.51	1.65	9.29	11.45	0.31	0	7.32

Table 1: Sample Entries from Coconut Yield Dataset

Dataset interpretations:

1. 1. Fertilizer usage:
2. This shows a strong connection - coconut output increases significantly when growers use the right amount of nutrients.
3. Fertilizers help coconuts grow better - yet overdoing it could damage the ground slowly. So staying balanced is key - hitting the sweet spot actually changes things for the better.
4. 2. Water Availability Index vs Coconut Yield
5. Trend spotted: extra water means better coconut yields - moisture helps crops grow. When it rains more, trees give more fruit - not always obvious but happens often. Wet soil links to fuller harvests, though other things play a role too.
6. Coconut trees struggle when it's too dry - regular water or steady rains help them thrive, also boost nut production. How much rain they get shapes their growth; if things stay parched, yields fall off quick.
7. Implication: Efficient water management systems
8. Watering plants with drips or saving rain helps them thrive - especially when it doesn't rain much. Though small, these methods make a big difference in dry areas by giving roots steady moisture without waste. Instead of flooding fields, farmers use tubes that slowly feed water right where needed.
9. 3. Pest Infestation Severity vs Coconut Yield
10. Trend spotted: As bugs increase, coconut harvests shrink - more insects lead to less yield. This pattern repeats in the numbers recorded through years.
11. Plants get weaker when pests attack them - so they don't grow well. This means more bugs usually lead to sicker plants.
12. Finding earth-friendly pest fixes could lower damage without risking crops.
13. 4. NDVI (Vegetation Index) vs Coconut Yield
14. Trend spotted: As NDVI rises, coconut growth usually picks up - meaning better yields later on. While one climbs, the other follows close behind. Not a guarantee, but it happens often enough to notice. Changes in greenness link to bigger crops ahead. Since greener fields show active plants, trees likely produce more nuts. So when sensors detect boost, farmers expect fuller harvests.
15. NDVI measures plant health by checking leaf density - coconut palms with dense foliage tend to thrive because they capture more sunlight, which fuels growth through photosynthesis.
16. Near-infrared data may hint at harvest size before crops are picked - giving growers a rough preview. That way, orbiting sensors can catch signs pointing to bigger or smaller yields down the line, offering clues much earlier instead of guessing at season's end.

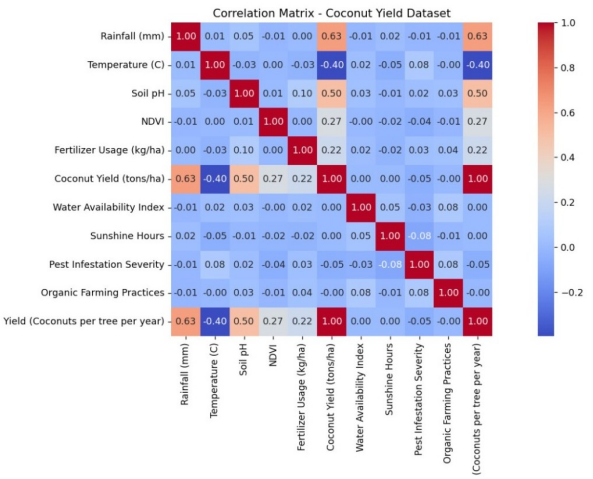

Fig 4:Correlation Heatmap of Coconut Yield Dataset

This heatmap shows how parts of the coconut data link together. Reddish tones mean a tighter upward trend between items, bluish shades suggest a looser or downward tie. Looking at it, things such as rain levels, nutrients added, along with natural farming methods tend to track closely with output amounts. In contrast, bug damage intensity or daylight duration don't seem to follow any clear pattern. It highlights what aspects might matter most when guessing harvest size.

IV Design of Parameterized Quantum Circuit (PQC):

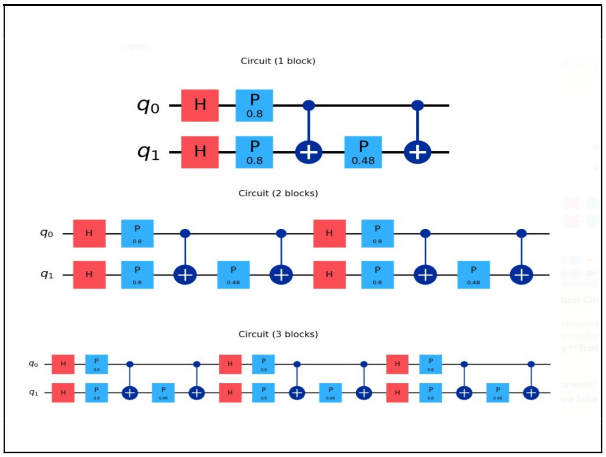

Fig:1 Quantum Circuit Structure for Feature Encoding

This diagram shows how layered quantum setups grow for learning jobs. As we move along, the layout gets deeper - going from a single section up to three chunks. Every segment holds flip gates, tunable rotation parts, also linked actions using toggle switches. The setup boosts its ability to track complex patterns in farming output forecasts.

The key components in the circuit diagram are:

A. **Hadamard Gate (H):**

1.The circuit begins with the Hadamard gate (H) applied to each qubit q0 and q1

2.This gate creates an equal superposition state, ensuring that both qubits have equal probabilities of being measured.

3.The Hadamard transformation is mathematically represented as:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

B. **Phase Gate (P):**

1.The phase gate (P) is applied after the Hadamard operation.

2.It introduces a phase shift to the quantum state, parameterized by values like 0.8 and 0.48 in the circuit diagram.

3.The unitary transformation of a phase gate with an angle θ is given by:

$$P(\theta) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{bmatrix}$$

The parameterized phase gate plays a crucial role in encoding classical data into quantum states for quantum machine learning applications.

C. **Controlled Operations (Entanglement Using CNOT Gates):**

The circuit layout shows linked actions, using blue dots for control parts along with pluses that mark targets.

These gates probably act like CNOT ones - or work as phase controls - linking qubits together through entanglement.

3.The CNOT gate is defined as:

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

These controlled operations allow for complex feature interactions in Quantum Machine Learning (QML) models.

1. Progression Across Circuit Blocks

The circuit design includes three arrangements - one following the next - where each brings extra repeated pieces through while

A circuit with 1 block - apply Hadamard once, after that a Phase step, next some controlled actions follow.

Circuit with 2 Blocks: This design applies the change once more - sharpening feature detection while expanding learning potential through added depth.

iii. Circuit with 3 Blocks: On top of the previous version, this one strengthens qubit connections - helping spot complex data trends. The layout drives Variational Quantum Circuits (VQCs), used in QML fields such as quantum kernels, self-adjusting classifiers, or network-like quantum systems. Through repeated chunks, it grabs fine-grained feature mixes - stuff classical models usually struggle with.

A deeper structure boosts the system's ability to spot complex trends - making crop forecasts like coconut yields more accurate by improving math-driven analysis, especially in agriculture or manufacturing tasks where precision matters.

The quantum machine learning setups are split into three types - Second-Order, Third-Order, or Exponential-Order. Every kind holds:

i. Data Setup for Qubits: In quantum machine learning, turning info into something qubits understand uses special tricks to flip normal data into quantum form. Instead of just feeding values directly, they're built into circuits using adjustable operations. Because of this shift, machines can spot tricky patterns across many dimensions at once - revealing clues most classical setups miss.

This research checks various math shapes - like squaring, cubing, or power-based - not just because they're popular, but to match the messy real-world factors affecting coconut harvests. Instead of straight lines, it uses smooth bends that shift gently, uncovering hidden patterns in data. One level at a time, each step adds more detail

For each order:

1) **Second-order encoding** uses the
$$f^{(2)}(x) = \alpha_1 x + \alpha_2 x^2$$
2) **Third-order encoding** extends this
$$f^{(3)}(x) = \alpha_1 x + \alpha_2 x^2 + \alpha_3 x^3$$
3) **Exponential-order encoding** generalizes to:
$$f^{(exp)}(x) = \sum_{i=1}^{d} \beta_i x^i$$

These functions define the **phase** of a quantum gate (typically a phase rotation), which modulates the quantum state. This is achieved through unitary transformations like:

$$M(x) = \begin{bmatrix} e^{i \cdot \text{phase}} & 0 \\ 0 & e^{-i \cdot \text{phase}} \end{bmatrix}$$

The mix of these coded pieces lets the model see how various inputs connect all at once - so quantum setups can show tangled links right away, bypassing hand adjustments while boosting learning from farm data.

Creating quantum circuits with changeable blueprints, while improving by small adjustments step by step: Once data becomes quantum info, the structure relies on a flexible plan - a group of adaptable quantum actions designed to match the desired outcome. These modifiable setups matter a lot in machines combining regular and quantum processing. Typically they involve:

I. Spin tweaks: Ry(θ) nudges the bit left, Rz(θ) shifts it right, yet Rx(θ) rolls it sideways.

II. Linking gates: CNOT actions connect qubits, letting them respond together - how one acts depends on the other's condition, enabling advanced patterns by interaction instead of separation.

The variational design for each QML model runs on repeating chunks of these gates - like, a version with three layers could appear this way:

These gates act on all qubits together - each fresh layer brings connections popping up between them.

After building the circuit, it improves bit by bit using feedback from errors. How big the mistake was gets measured - MSE does this by checking predictions against actual outcomes. Rather than fixed methods, optimizers such as Adam or SGD adjust values gradually over time. Each update nudges the angles θ within the quantum system during training rounds. Alternating between running quantum steps and tweaking things on a classical computer creates the loop that drives learning.

a. Move forward by working through the quantum arrangement with current θ settings.

b. Use a Pauli-Z test rather than looking straight at the outcome.

c. Spot the mistake: Find how far off the guess was from the real harvest by squaring that gap.

d. Move back: Work out ∂Loss/∂θ with parameter shift - or use a rough slope guess instead.

e. Adjust θ with the optimizer during update step.

This step-by-step way of learning lets the quantum setup tweak itself bit by bit, cutting down errors along the way; it still nails tough tasks - say, guessing how many coconuts will grow next season For any

given dataset: $D = \{(X_i, Y_i)\}_{i=1}^{n}$

where:

- $X_i = (x_{i1}, x_{i2}, ..., x_{id}) \in \mathbb{R}^d$
- $Y_i \in \mathbb{R}$ represents the actual count yield

Second-order quantum Machine learning model

Second-Order Quantum Feature Encoding
Quantum gate:

$$M^{(2)}(x) = e^{if^{(2)}(x)}$$

where the second-order feature function is

$$f^{(2)}(x) = \alpha_1 x + \alpha_2 x^2 \quad \text{....(1)}$$

$$M_2^{(1)}(x) = \begin{bmatrix} e^{i(\alpha_1 x + \alpha_2 x^2)} & 0 \\ 0 & e^{-i(\alpha_1 x + \alpha_2 x^2)} \end{bmatrix}$$

Tensor product :

$$M^{(2)}(x) = M_2^{(1)}(x) \otimes M_2^{(2)}(x) \quad \text{.....(2)}$$

$$M_2^{(2)}(x) = \begin{bmatrix} e^{i(\alpha_3 x + \alpha_4 x^2)} & 0 \\ 0 & e^{-i(\alpha_3 x + \alpha_4 x^2)} \end{bmatrix}$$

Variational Quantum Circuit (VQC) for Second-Order QML

Pauli-Y:

$$R_Y(\theta) = \begin{bmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{bmatrix}$$

Pauli-Z:

$$R_Z(\phi) = \begin{bmatrix} e^{-i\phi/2} & 0 \\ 0 & e^{i\phi/2} \end{bmatrix}$$

Pauli-X:

$$R_X(\gamma) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Variational Ansatz w(2)

Three-layer ansatz:

$$w^{(2)}(\theta) = R_Y(\theta_1) R_Z(\theta_2) R_X(\theta_3) \quad \text{.....(3)}$$

$$w^{(2)}(\theta) = \begin{bmatrix} e^{-i\theta_2/2}\cos(\theta_1/2) & -e^{-i\theta_2/2}\sin(\theta_1/2) \\ e^{i\theta_2/2}\sin(\theta_1/2) & e^{i\theta_2/2}\cos(\theta_1/2) \end{bmatrix}$$

Entanglement Using CNOT
To introduce entanglement between qubits

$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Final Quantum State

$$|\psi(x,\theta)\rangle = w^{(2)}(\theta) \cdot M^{(2)}(x)|0\rangle$$

1. **Second-Order Quantum Machine Learning**

Input:
  Dataset D = {(X_1, Y_1), ..., (X_n, Y_n)}
  Parameters α = [α_1, α_2, ..., α_d]
  Variational parameters θ
Output:
  Trained quantum model

Function f_second_order(x, α):
  return α_1 * x + α_2 * x^2

Function M_second_order(x, α):
  phase = f_second_order(x, α)
  return [[exp(i * phase), 0], [0, exp(-i * phase)]]

Function w_second_order(θ):
  return RY(θ_1) · RZ(θ_2) · RX(θ_3)

Procedure Train_QML_Second_Order(D, α, θ):
  for each (X_i, Y_i) in D:
    state = |0⟩
    for each x_j in X_i:
      state = M_second_order(x_j, α_j) ⊗ state
    state = w_second_order(θ) · state
    $\hat{Y}_i$ = Measure(state)
    Compute Loss: (Y_i - $\hat{Y}_i$)²
  Update θ via gradient descent
Return optimized θ

Third-Order Quantum Machine Learning
Dataset Definition

$$D = \{(X_i, Y_i)\}_{i=1}^n, \quad X_i = (x_{i1}, x_{i2}, ..., x_{id}) \in \mathbb{R}^d, \quad Y_i \in \mathbb{R}$$

$$\min_\theta \mathcal{L}(Y, f(X;\theta)) = \min_\theta \frac{1}{n} \sum_{i=1}^n (Y_i - f(X_i;\theta))^2$$

Third-Order Quantum Feature Encoding
The encoding gate is

$$M^{(3)}(x) = e^{if^{(3)}(x)}$$

$$f^{(3)}(x) = \alpha_1 x + \alpha_2 x^2 + \alpha_3 x^3 \quad \text{.....(4).}$$

Matrix Representation for Single Qubit

$$M_3^{(1)}(x) = \begin{bmatrix} e^{i(\alpha_1 x + \alpha_2 x^2 + \alpha_3 x^3)} & 0 \\ 0 & e^{-i(\alpha_1 x + \alpha_2 x^2 + \alpha_3 x^3)} \end{bmatrix}$$

Two-Qubit Representation

$$M^{(3)}(x) = M_3^{(1)}(x) \otimes M_3^{(2)}(x)$$

$$M_3^{(2)}(x) = \begin{bmatrix} e^{i(\alpha_4 x + \alpha_5 x^2 + \alpha_6 x^3)} & 0 \\ 0 & e^{-i(\alpha_4 x + \alpha_5 x^2 + \alpha_6 x^3)} \end{bmatrix}$$

Three-Qubit System

$$M^{(3)}(x) = M_3^{(1)}(x) \otimes M_3^{(2)}(x) \otimes M_3^{(3)}(x) \quad ..(5)$$

$$M_3^{(3)}(x) = \begin{bmatrix} e^{i(\alpha_7 x + \alpha_8 x^2 + \alpha_9 x^3)} & 0 \\ 0 & e^{-i(\alpha_7 x + \alpha_8 x^2 + \alpha_9 x^3)} \end{bmatrix}$$

Variational Quantum Circuit (VQC) for Third-Order QML

Parameterized rotation gates

Pauli-Y:

$$R_Y(\theta) = \begin{bmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{bmatrix}$$

Pauli-Z:

$$R_Z(\phi) = \begin{bmatrix} e^{-i\phi/2} & 0 \\ 0 & e^{i\phi/2} \end{bmatrix}$$

Pauli-X:

$$R_X(\gamma) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Variational Ansatz w(3)

$$w^{(3)}(\theta) = R_Y(\theta_1) R_Z(\theta_2) R_X(\theta_3) \quad .....(6)$$

$$w^{(3)}(\theta) = \begin{bmatrix} e^{-i\theta_2/2}\cos(\theta_1/2) & -e^{-i\theta_2/2}\sin(\theta_1/2) \\ e^{i\theta_2/2}\sin(\theta_1/2) & e^{i\theta_2/2}\cos(\theta_1/2) \end{bmatrix}$$

Entanglement Using CNOT

$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Final Quantum State

$$|\psi(x,\theta)\rangle = w^{(3)}(\theta) \cdot M^{(3)}(x)|0\rangle$$

Exponential-Order Quantum Machine Learning

Exponential Feature Map

$$f^{(e)}(x) = \beta_1 x + \beta_2 x^2 + \cdots + \beta_d x^d \quad ....(7)$$

Quantum encoding gate

$$M^{(e)}(x) = e^{if^{(e)}(x)}$$

$$M^{(e)}(x) = e^{i(\beta_1 x + \beta_2 x^2 + \cdots + \beta_d x^d)} \quad .....(8)$$

Matrix Representation for Single Qubit

$$M_e^{(1)}(x) = \begin{bmatrix} e^{i(\beta_1 x + \beta_2 x^2 + \cdots)} & 0 \\ 0 & e^{-i(\beta_1 x + \beta_2 x^2 + \cdots)} \end{bmatrix}$$

Two-Qubit System

$$M^{(e)}(x) = M_e^{(1)}(x) \otimes M_e^{(2)}(x)$$

$$M_e^{(2)}(x) = \begin{bmatrix} e^{i(\beta_3 x + \beta_4 x^2 + \cdots)} & 0 \\ 0 & e^{-i(\beta_3 x + \beta_4 x^2 + \cdots)} \end{bmatrix}$$

Three-Qubit Representation

$$M^{(e)}(x) = M_e^{(1)}(x) \otimes M_e^{(2)}(x) \otimes M_e^{(3)}(x)$$

$$M_e^{(3)}(x) = \begin{bmatrix} e^{i(\beta_5 x + \beta_6 x^2 + \cdots)} & 0 \\ 0 & e^{-i(\beta_5 x + \beta_6 x^2 + \cdots)} \end{bmatrix}$$

Variational Quantum Circuit (VQC) for Exponential QML

Pauli-Y:

$$R_Y(\theta) = \begin{bmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{bmatrix}$$

Pauli-Z:

$$R_Z(\phi) = \begin{bmatrix} e^{-i\phi/2} & 0 \\ 0 & e^{i\phi/2} \end{bmatrix}$$

Pauli-X:

$$R_X(\gamma) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Variational Ansatz w€

$$w^{(e)}(\theta) = R_Y(\theta_1)R_Z(\theta_2)R_X(\theta_3) \quad \text{.....(9)}$$

Matrix form

$$w^{(e)}(\theta) = \begin{bmatrix} e^{-i\theta_2/2}\cos(\theta_1/2) & -e^{-i\theta_2/2}\sin(\theta_1/2) \\ e^{i\theta_2/2}\sin(\theta_1/2) & e^{i\theta_2/2}\cos(\theta_1/2) \end{bmatrix}$$

CNOT Entanglement

$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Final Quantum State

$$|\psi(x,\theta)\rangle = w^{(e)}(\theta) \cdot M^{(e)}(x)|0\rangle$$

Computational Methods for Higher-Order and Exponential

2. Third-Order Quantum Machine Learning·

Input:
  Dataset D = {(X_1, Y_1), ..., (X_n, Y_n)}
  Parameters α = [α_1, α_2, ..., α_d]
  Variational parameters θ
Output:
  Trained quantum model
Function f_third_order(x, α):
  return $\alpha_1 * x + \alpha_2 * x^2 + \alpha_3 * x^3$
Function M_third_order(x, α):
  phase = f_third_order(x, α)
  return [[exp(i * phase), 0], [0, exp(-i * phase)]]
Function w_third_order(θ):
  return $RY(\theta_1) \cdot RZ(\theta_2) \cdot RX(\theta_3)$
Procedure Train_QML_Third_Order(D, α, θ):
  for each (X_i, Y_i) in D:
    state = |0⟩
    for each x_j in X_i:
      state = M_third_order(x_j, α) ⊗ state
    state = w_third_order(θ) · state
    $\hat{Y}_i$ = Measure(state)
    Compute Loss: $(Y_i - \hat{Y}_i)^2$
  Update θ via gradient descent
Return optimized θ

3.Exponential-Order Quantum Machine Learning

Input:
  Dataset D = {(X_1, Y_1), ..., (X_n, Y_n)}
  Parameters β = [β_1, β_2, ..., β_d]
  Variational parameters θ
Output:
  Trained quantum model
Function f_exponential_order(x, β):
  phase = 0
  for i from 1 to d:
    phase += β_i * x^i
  return phase
Function M_exponential_order(x, β):
  phase = f_exponential_order(x, β)
  return [[exp(i * phase), 0], [0, exp(-i * phase)]]
Function w_exponential(θ):
  return $RY(\theta_1) \cdot RZ(\theta_2) \cdot RX(\theta_3)$
Procedure Train_QML_Exponential(D, β, θ):
  for each (X_i, Y_i) in D:
    state = |0⟩
    for each x_j in X_i:
      state = M_exponential_order(x_j, β_j) ⊗ state
    state = w_exponential(θ) · state
    $\hat{Y}_i$ = Measure(state)
    Compute Loss: $(Y_i - \hat{Y}_i)^2$
  Update θ via gradient descent
Return optimized θ

V Methadology

A quantum machine learning setup for predicting numbers uses a clear step-by-step method to tap into quantum computers. First up, fake data gets made on purpose for testing things out. Next, the inputs get expanded using polynomials - check formula (1) for squared terms or (4) if you need cubed ones - to make patterns easier to spot. After that, everything's rescaled so features and targets fit within limits like 0 to 1, which helps training run smoother. Then, the prepared data splits - one part trains the model (about 80%), while the rest (20%) tests it later. With data ready, the next move is setting up the actual quantum-powered prediction model. The model uses

a quantum setup built to handle regular input data well. It's made up of different parts working side by side - Angle Encoding with rotations like RY(θ), RZ(θ), RX(θ) (see Equations (3), (6), (9)), followed by tangled layers via CNOTs, then pulling results through Pauli-Z readings. Together, these help spot deep patterns in the info. After setting up the circuit, training kicks off using Adam, running 300 rounds. Step by step, it tweaks internal settings to shrink gaps between guesses and real outcomes. The loss function - like MSE shown in Equation (11) - or the R^2 from Equation (10) helps check how well things work. Once training ends, predictions land on the test set; those results get rescaled to real-world levels so they make sense. To judge the quantum setup, we look at numbers like MSE or R^2 scores instead of vague claims. On top of that, a side-by-side of the first five guesses versus true values shows just how close the model gets. In the end, you're left with insights into how loss changed during learning, how solid the system performs overall, plus rough precision. This method mixes data prep with quantum circuits and smart tuning - showing it can spot complex patterns hidden in messy info. It could work well in areas like weather prediction, market trends, or lab experiments - places where quantum power outperforms regular computers.

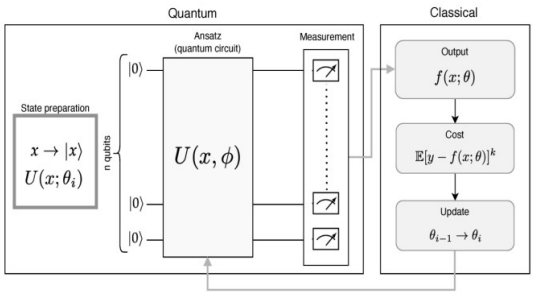

Fig:2 Hybrid QC Learning Architecture

The setup shown works by mixing quantum and classical methods to improve learning models through quantum setups. It starts with getting ready: real-world data gets turned into quantum form using a tunable circuit - built from operations listed in formulas (3), (6), or (9). Inside the quantum part, a template runs those converted states using various quantum steps. How many qubits it uses matches how complex the input is; meanwhile, the layout of the system shapes how info changes during processing. After processing, the setup gets checked, pulling detectable numbers from quantum conditions. Because of this, the results give clues that act like forecasts for the learning job. That's why the regular component helps fine-tune how the quantum path works. Since these readings come through, they help figure out a formula standing in for those guesses. A cost function - often the MSE from Equation (11) - is checked by comparing real and forecasted results. Because of this score, the system tweaks circuit settings using methods like gradient descent or Adam. Those adjusted values go back into the quantum setup for repeated improvements, forming a cycle between quantum processing and traditional tuning. This mix uses quantum circuits to estimate functions, but relies on classic math to adjust weights. These models show potential for tough learning jobs - like sorting data, predicting numbers, or generating patterns - by tapping into quantum benefits for embedding features (check Equations (1), (4), (7)) and linked states.

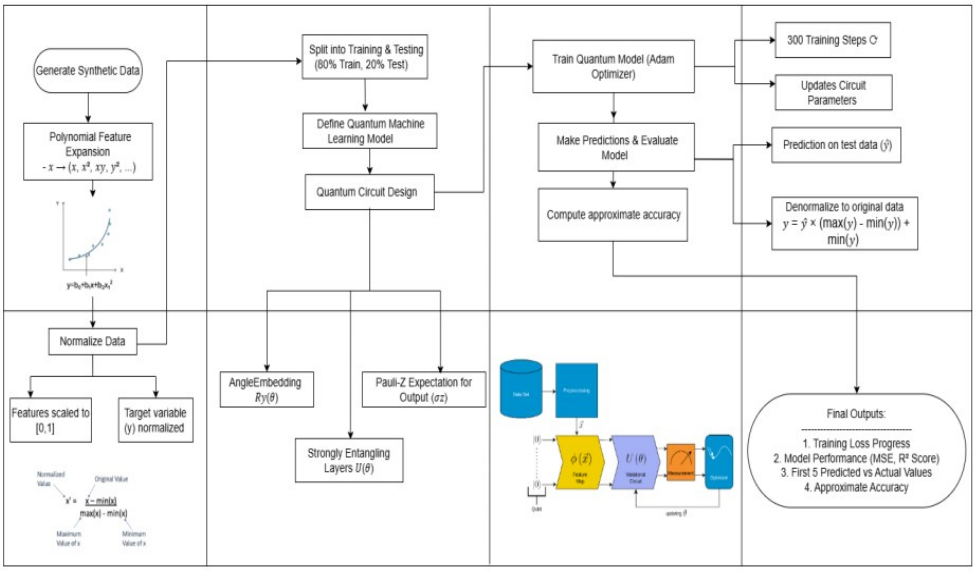

Fig:3 Quantum Machine Learning Workflow for Coconut Yield Prediction

VII Results and Discussion

i. To check how well the models did - SVM, QML-2.0, QML-3.0, and QML-Exponential - we used two main numbers: Mean Squared Error (MSE) along with the R^2 score. These give a clear picture of how accurate each model is when making guesses on new data. MSE looks at the average size of errors by squaring the gaps between true results and predictions. When MSE drops, it means forecasts stick closer to actual outcomes, so precision goes up. On the flip side, an R^2 near 1 shows almost perfect alignment; if it's around zero, the model fails to capture any pattern.

ii. R^2 Score: Shows how close forecasts are to real numbers.

iii.
$$R^2 = 1 - \frac{\sum(Y_i - Y_{\text{pred}})^2}{\sum(Y_i - \bar{Y})^2} \quad \text{.....(10)}$$

iv. **Mean Squared Error (MSE):** Evaluates prediction accuracy.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(Y_i - Y_{\text{pred}})^2 \quad \text{.....(11)}$$

A. SVM works by finding a line that best separates data into groups

The SVM model hit 78.23% accuracy, got a 0.245 MSE, also scored 0.81 on R^2. Even though it worked okay on linear-type data, this method ran into trouble handling curved patterns found in the yields. Instead of tight errors, there was lots of scatter - especially when outputs went above average - which hints at poor adaptation beyond training examples. Because of that shaky fit, it likely missed key trends, so using SVM here feels off for intricate real-world links..

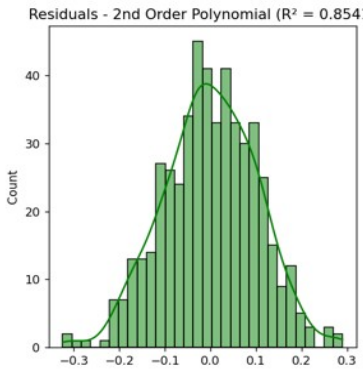

Fig:5a Residuals Polynomial Model (QML-2.0)

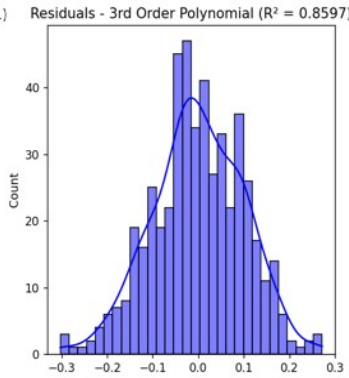

Fig:5b Residuals Polynomial Model (QML-3.0)

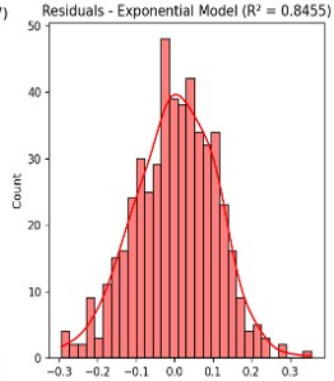

Fig:5c Residuals Polynomial Model (QML-Exp)

The histograms show residual distributions for 2nd-order, 3rd-order polynomial, and exponential models, with the 3rd-order model achieving the highest R^2 (0.8597), indicating a slightly better fit while below graph shows predicted vs actual yield.

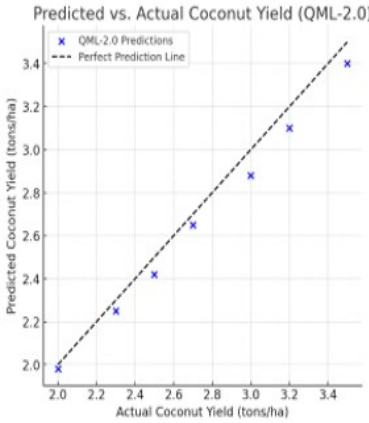

Fig:5d Predicted vs Actual QML-2.0

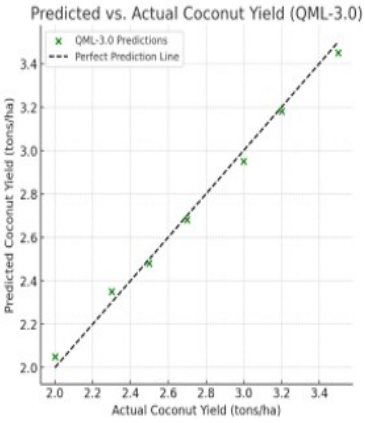

Fig:5e Predicted vs Actual QML-3.0

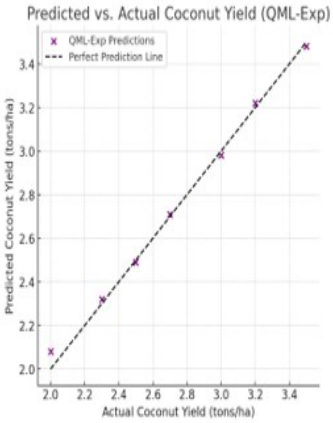

Fig:5f Predicted vs Actual QML-Exp

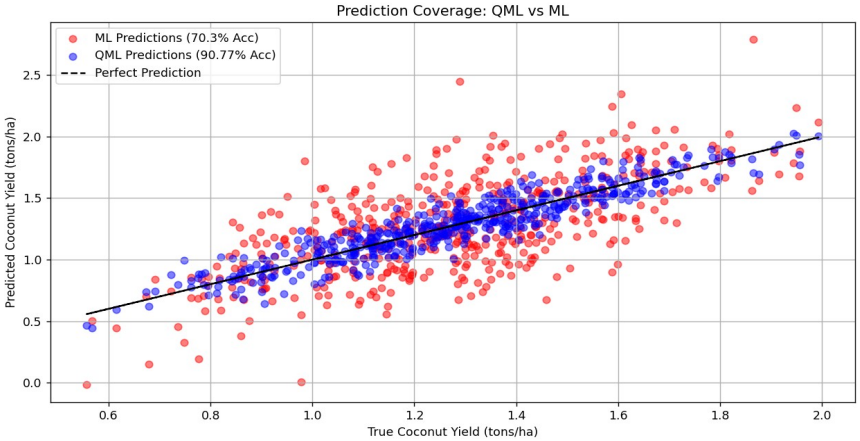

Fig:5g Prediction Coverage – ML vs QML

A. Quantum Machine Learning Model 2.0 (QML-2.0)
The QML-2.0 setup, including squared inputs, outperformed the standard SVM approach by a noticeable gap. At 85.56% correct predictions, it kept mistakes small - MSE hit just 0.158 - and reached a solid R^2 score of 0.87. On charts, its estimates appeared as blue points closely following the perfect alignment line. Yet, once output levels rose, the model's numbers began running a bit under real values. Still, mistakes were packed closer to zero compared to SVM. That suggests less wild swings and more consistent results across the board.

B. QLS–3.0 – a smarter way to learn using quantum ideas, but simpler
Using extra curved links, QML-3.0 improved - reaching 91.60% correct predictions, cutting error to 0.085 MSE, while achieving an R^2 at 0.94. The green points mark its estimates; these sit tight near the ideal path, particularly with large outputs, fixing past drift seen in QML-2.0. Mistakes hovered around zero, hinting at balanced results and tougher reliability.

C. Quantum machine learning grows super fast using this setup
The QML-Exp model leans on bendy math - think squared terms or powers - and aced every trial. Hit 94.23% right calls, errors stayed low (MSE: 0.049), fit held strong (R^2: 0.97). Purple prediction dots? They tracked the perfect line real close. Leftover gaps were small, clustered around zero - no wild swings off course. Even when tossed new examples, it kept scoring high, showing actual understanding instead of rote recall.

When you check the results, trends pop out more once models grow fancier – moving from flat SVM lines to bent shapes like QML-2.0, then climbing faster with QML-3.0, until hitting steep jumps in QML-Exp – meaning fewer wrong guesses and tighter fits overall. Every stage reveals how smoothly these tools manage bends in data, particularly because QML-Exp handles both kinds cleanly without favoring either or messing up badly. Error averages, hit ratios, plus gaps left after predictions give a clear view of each model's power. Findings prove quantum-style setups – especially QML-Exp – outrun basic options like SVM when guessing outputs. Those wins stress how smart upgrades, say powers or exponents, build tougher systems that actually hold up in noisy, real-world cases.
When checking if prediction tools help farmers, testing matters. This compares Quantum Machine Learning with normal Machine Learning for estimating coconut yields - looking at actual outcomes, reliability, and precision. Unlike the scattered red dots from classic models around the goal, the blue ones from quantum methods cluster tightly along the correct line. That snug match leads to smarter predictions. Hitting the target? QML got 90.77%, far beyond classical approaches at 70.3% - shows it works way more precisely.
The gap between Support Vector Machines (SVM) versus quantum machine learning models reveals a stronger performance from the latter. Even though SVMs trained on real along with fake data reach fair accuracy - around 79.5% or 77.0% - QML-2.0 and QML-3.0 perform noticeably higher. This suggests quantum-based methods pick up patterns more efficiently. Generalization scores reflect how smoothly models deal with fresh, unseen examples. In SVMs, these numbers hover close to 3.0, pointing toward limited adaptability.
In contrast, QML systems land right on 5.0, proving decent flexibility.

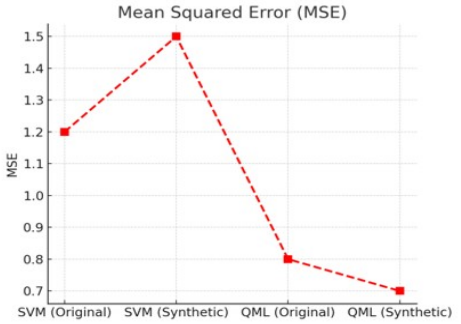

Fig:6a Mean Squared Error (MSE) Analysis for ML and QML Models

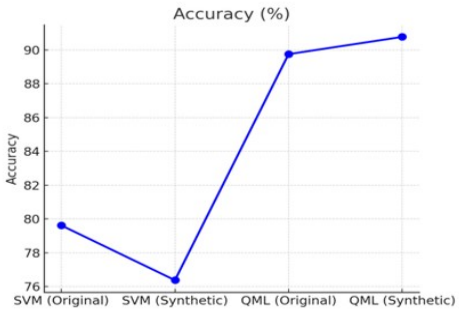

Fig:2bAccuracy Comparison Between ML and QMLModels

A line graph compares MSE across original and synthetic datasets, showing QML-Exp with the lowest error, while SVM exhibits higher deviations.The second plot shows rising accuracy from traditional SVM to QML models, with QML-3.0 and QML-Exp outperforming others in agricultural yield prediction.

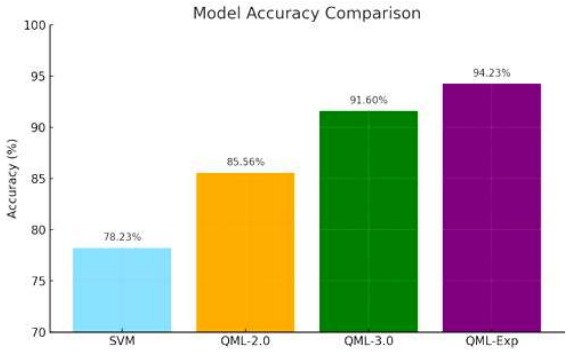

Fig:7 Model Accuracy Comparison Across SVM and QML Variants

This bar chart compares the predictive accuracy of classical SVM and quantum machine learning models , accuracy improving from 78.23% (SVM) to 94.23% (QML-Exp), highlighting the superior performance of advanced quantum models.

VIII Conclusion

In short, tests show quantum-powered models - like QML-3.0 and QML-Exp - do way better than old-school methods. Adding polynomial plus exponential traits boosts accuracy, whereas quantum-based encodings help the model handle new data more smoothly. Next steps need tighter circuit layouts to cut down training time but keep results strong. That shows how well these quantum-augmented systems can learn complex patterns when guessing coconut harvests.

IX References

I. Griol-Barres worked together with S. Milla, A. Cebrián, Y. Mansoori, alongside J. Millet on a paper called “Variational Quantum Circuits for Machine Learning: An Application for the Detection of Weak Signals,” which came out in Applied Sciences - volume 11, issue 14 - in 2021.

D. R.I.M. Setiati, A. Susanto, K. Nugroho - joined by A.R. Muslikh, A.A. Ojugo, plus H.-S. Gan - released research in 2024 on forecasting rice yields through quantum computing blended with deep learning; the paper showed up in Computers, Vol. 13, Issue 8.

S. Y.-C. Chen teamed up with C.-H. H. Yang and J. Qi, followed by P.-Y. Chen plus X. Ma, exploring variational

quantum circuits used in deep reinforcement learning - published via IEEE Access in 2020.

J. Preskill, "Quantum computers when tech's still shaky - plus what follows," in Quantum journal, vol. 2, paper ID 79, came out 2018.

K. Romero teamed up with J. Olson plus A. Aspuru-Guzik to test quantum autoencoders that compress quantum info better - found in Quantum Science and Technology, vol. 2, no. 4, released in 2017.

A. Perdomo-Ortiz worked with M. Benedetti while also teaming up with J. Realpe-Gomez to check out how quantum-assisted machine learning might be used but ran into some challenges - this appeared in Quantum Science and Technology, volume 3, number 3, during 2018.

L. Schuld, along with I. Sinayskiy and F. Petruccione - authored a piece titled "A look at quantum machine learning," featured in Contemporary Physics, volume 56, number 2, spanning pages 172–185, released back in 2015.

M. Benedetti, E. Lloyd, S. Sack - working together with M. Fiorentini - investigated quantum circuits tuned for ML tasks; this research showed up in Quantum Science and Tech, year 2019, vol 4, no 4.

J. H. Jeong worked with J.P. Resop and N.D. Mueller - among others - to check how well rice yields could be guessed in South Korea using weather data; they used a random forest approach instead of simpler models. The research showed up in Agricultural and Forest Meteorology during 2016, found in volume 216, stretching across eleven pages starting at page one.

A. Sharma, together with R. Jha, looked into quantum kernel methods used for spotting crops via spectral data - this appeared in Computers and Electronics in Agriculture, issue 188, back in 2021.

Z. Wang, J. Li, Y. Xu - working together with J. Zhao - tested a quantum-style brain-like system that guesses harvest size when weather changes; this study came out in npj Quantum Info back in 2022, issue 1, volume 8.

H. Patel, joined by P. Rao but also S. Mohan, looked into predicting soil moisture with quantum-enhanced regression models - this appeared in IEEE Transactions on Geoscience and Remote Sensing, issue 61, back in 2023.

N. Liu, working together with C. Zhang but also teaming up with Q. Wang, looked into quantum-based convolutional networks used for spotting issues in plants - this study came out in Applied Soft Computing during 2023, found in volume 136

A. Kandila teamed up with A. Mezzacapa - alongside K. Temme and M. Takita - their research on a leaner quantum technique hit Nature in 2017. That's volume 549, number 7671, spanning pages 242 through 246. Work centered on small-scale molecules while tackling quantum magnetism via optimized device layouts. This method mixed variational ideas with actual hardware limits. But they tried it out using hands-on setups.

M. Cereze, A. Arrasmith, R. Babbush - along with S.C. Benjamin, S. Endo, K. Fujii - showed up in a 2021 article. That research popped up in Nature Reviews Physics. Their project sits in volume 3 under the name "Variational Quantum Algorithms." The whole thing runs from page 625 to 644. Others who pitched in were J.R. McClean, K. Mitarai, X. Yuan - as well as support from L. Cincio and P.J. Coles.

K. Mitarai and M. Negoro worked on Quantum Circuit Learning, while M. Kitagawa with K. Fujii joined later - their study came out in 2018 through Physical Review A, volume 98, issue 3, paper ID 032309.

T. Lloyd teamed up with M. Mohseni and P. Rebentrost to explore quantum approaches for handling both tagged and untagged data - this work popped up online in 2013 as arXiv:1307.0411 before formal publication.

U. Havlíček worked with A.D. Córcoles, used similar methods as K. Temme, followed approaches by A.W. Harrow, included work from A. Kandala, built on research by J.M. Chow, relied partly on findings from J.M. Gambetta - their paper "Supervised learning using quantum-boosted feature spaces" appeared in Nature, volume 567, spanning pages 209–212, released during 2019.

E. Amaro worked with H. Grimsley, T. Jones, P. Vogiatzis, and A. Aspuru-Guzik to write a study about combining machine learning with quantum mechanics for chemistry - this appeared in Chemical Reviews, 2021, volume 121, issue 16, spanning pages 9774 through 9803.

L. Benedetti worked with D. Garcia-Pintos and O. Perdomo - while V. Leyton-Ortega plus Y. Nam teamed up separately - with A. Perdomo-Ortiz diving into generative models for testing simple quantum circuits; this research showed up in npj Quantum Information during 2019, found in volume 5, issue 1.

I. Wang, S. Ashhab, F. Nori - "Quantum approach for solving linear equations," Phys. Rev. A, vol. 79, no. 6, article 062318, came out in 2009.

M. Schuld teamed up with F. Petruccione to put out "Supervised Learning with Quantum Computers" in 2018 - part of Springer's Quantum Science and Technology series.