.

Lab Manual: Comparing Classical ML and Quantum ML

1. Theoretical Background

1.1 Classical Machine Learning

Classical machine learning refers to traditional algorithms that learn patterns from data, such as regression, classification, clustering, etc. Logistic Regression, for instance, is a popular statistical method used for binary classification tasks. It estimates the probability that an instance belongs to a particular category.

1.2 Quantum Machine Learning

Quantum machine learning leverages quantum computing principles to improve learning tasks. Quantum circuits can model complex relationships in data through superposition and entanglement, potentially outperforming classical counterparts in certain scenarios.

2. Data Preparation

2.1 Dataset

We will use the **Breast Cancer Wisconsin (Diagnostic)** dataset from `scikit-learn`. This dataset contains features derived from digitized images of breast cancer tumors, classified as malignant or benign.

2.2 Steps for Data Preparation

1. Load the dataset.
2. Split the dataset into training and testing sets.

3. Pipeline

1. Load the Breast Cancer dataset.
2. Preprocess the data (train-test split).
3. Train a classical model (Logistic Regression).
4. Implement a quantum model (quantum circuit).
5. Get predictions from both models.

6. Combine predictions.
7. Evaluate and compare the results.
8. Visualize the results.

4. Pseudocode

```
1. Load the Breast Cancer dataset
2. Split the dataset into training and testing sets
3. Train a classical model using Logistic Regression
4. Evaluate the classical model and print accuracy
5. Create a quantum circuit
6. Simulate predictions using the quantum circuit
7. Evaluate the quantum model and print accuracy
8. Combine predictions from classical and quantum models
9. Evaluate the combined model and print accuracy
10. Print classification reports for all models
11. Plot the accuracies of all models
```

5. Implementation

5.1 Python Code

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
from qiskit import QuantumCircuit
from qiskit_aer import AerSimulator

# Step 1: Load the Breast Cancer dataset
data = load_breast_cancer()
X = data.data
y = data.target

# Step 2: Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=42)

# Step 3: Train a classical model (Logistic Regression)
classical_model = LogisticRegression(max_iter=1000)
classical_model.fit(X_train, y_train)

# Step 4: Predictions and evaluation for classical model
y_pred_classical = classical_model.predict(X_test)
classical_accuracy = accuracy_score(y_test, y_pred_classical)
print("Classical Model Accuracy:", classical_accuracy)

# Step 5: Create a quantum circuit for quantum ML
def create_quantum_circuit(num_qubits):
    circuit = QuantumCircuit(num_qubits, num_qubits)
```

```
    for i in range(num_qubits):
        circuit.h(i)  # Apply Hadamard to each qubit
    circuit.measure(range(num_qubits), range(num_qubits))  # Measure each
qubit
    return circuit

# Step 6: Run the quantum circuit and evaluate on the test set
def run_circuit(qc):
    simulator = AerSimulator()
    result = simulator.run(qc).result()  # Directly run the circuit
    counts = result.get_counts()
    return counts

# Simulate predictions based on the quantum circuit
def quantum_predictions(X_test):
    predictions = []
    for i in range(len(X_test)):
        num_qubits = 2  # Use 2 qubits for binary classification
        qc = create_quantum_circuit(num_qubits)
        counts = run_circuit(qc)
        result = max(counts, key=counts.get)  # Get the most frequent
measurement result
        label = int(result, 2) % 2  # Simplified prediction
        predictions.append(label)
    return predictions

# Step 7: Get quantum predictions
y_pred_quantum = quantum_predictions(X_test)

# Step 8: Evaluate quantum model accuracy
quantum_accuracy = accuracy_score(y_test, y_pred_quantum)
print("Quantum Model Accuracy:", quantum_accuracy)

# Step 9: Combine classical and quantum predictions
def combine_predictions(classical_preds, quantum_preds):
    combined_preds = []
    for classical, quantum in zip(classical_preds, quantum_preds):
        # Use classical predictions as default
        combined_preds.append(classical)
    return np.array(combined_preds)

# Step 10: Get combined predictions
y_pred_combined = combine_predictions(y_pred_classical, y_pred_quantum)

# Step 11: Evaluate combined model accuracy
combined_accuracy = accuracy_score(y_test, y_pred_combined)
print("Combined Model Accuracy:", combined_accuracy)

# Step 12: Print classification reports for all models
print("\nClassification Report for Classical Model:")
print(classification_report(y_test, y_pred_classical))

print("\nClassification Report for Quantum Model:")
print(classification_report(y_test, y_pred_quantum))

print("\nClassification Report for Combined Model:")
print(classification_report(y_test, y_pred_combined))
```

```
# Step 13: Plotting results
def plot_results(classical_accuracy, quantum_accuracy, combined_accuracy):
    models = ['Classical ML', 'Quantum ML', 'Combined Model']
    accuracies = [classical_accuracy, quantum_accuracy, combined_accuracy]

    plt.bar(models, accuracies, color=['blue', 'orange', 'green'])
    plt.title('Model Accuracies Comparison')
    plt.ylabel('Accuracy')
    plt.ylim(0, 1)  # Set y-axis limits to [0,1]
    plt.show()

# Plot the results
plot_results(classical_accuracy, quantum_accuracy, combined_accuracy)
```

6. Comparisons

6.1 Results

- **Classical Model Accuracy**: Generally high (e.g., 96.5%).
- **Quantum Model Accuracy**: Typically lower (e.g., ~55.2%).
- **Combined Model Accuracy**: May improve upon the quantum model alone but depends on the combination logic.

6.2 Pros and Cons

Classical ML

- **Pros**:
    - Well-established techniques with high accuracy on many datasets.
    - Faster training and inference times.
    - Extensive libraries and community support.
- **Cons**:
    - May struggle with complex datasets (e.g., high-dimensional data).
    - Limited by classical computation constraints.

Quantum ML

- **Pros**:
    - Potential for better performance on certain tasks due to quantum properties.
    - Ability to handle complex data structures.
- **Cons**:
    - Currently, limited by noise and error rates in quantum hardware.
    - Requires understanding of quantum circuits and algorithms.
    - Still experimental with fewer libraries and tools available.

7. Conclusion

This lab provides a framework for comparing classical and quantum machine learning approaches using the Breast Cancer dataset. While classical models currently outperform quantum models in many scenarios, the field of quantum machine learning is rapidly evolving. Future improvements in quantum algorithms and hardware may bridge this gap.

```
(quantum) PS C:\Users\user\Desktop\Quantum_ML_Lab> python
quantum_machine_learning_basics.py

Classification Report for Classical Model:
              precision    recall  f1-score   support
           0       0.96      0.94      0.95        54
           1       0.97      0.98      0.97        89
    accuracy                           0.97       143
   macro avg       0.96      0.96      0.96       143
weighted avg       0.97      0.97      0.96       143


Classification Report for Quantum Model:
              precision    recall  f1-score   support
           0       0.44      0.59      0.50        54
           1       0.69      0.54      0.60        89
    accuracy                           0.56       143
   macro avg       0.56      0.57      0.55       143
weighted avg       0.59      0.56      0.57       143


Classification Report for Combined Model:
              precision    recall  f1-score   support
           0       0.96      0.94      0.95        54
           1       0.97      0.98      0.97        89
    accuracy                           0.97       143
   macro avg       0.96      0.96      0.96       143
weighted avg       0.97      0.97      0.96       143
```