Project Roadmap: Quantum-Enhanced Stock Price Prediction

Our strategy is to build a **Hybrid Quantum-Classical Recurrent Neural Network**. This model will leverage the strengths of classical Long Short-Term Memory (LSTM) networks for processing time-series data and integrate a trainable Variational Quantum Circuit (VQC) to enhance the model's feature-learning capabilities. This approach is powerful, meets the competition's requirements, and is at the forefront of applied quantum machine learning research.

Here are the phases of our project:

Phase 1: Data Reconnaissance and Preprocessing

- **Objective:** Prepare the dataset for our hybrid model, ensuring we extract the most valuable temporal patterns.
- **Key Actions:**
    1. **Data Loading and Inspection:** Load X_train.csv and X_test.csv and perform an initial analysis to understand the data's structure, and check for any missing values or anomalies in the training set.
    2. **Feature Engineering:** We will create additional features from the existing data to better capture market dynamics. This includes:
        - **Moving Averages:** Calculate 5-day and 10-day moving averages for the 'Close' price.
        - **Lagged Features:** Use the 'Close' prices from previous days as input features.
    3. **Data Scaling:** Normalize all features using a MinMaxScaler. Crucially, we will fit this scaler **only** on the training data (X_train.csv) to prevent data leakage and apply the same scaling to the test data.
    4. **Windowing:** Restructure the data into "windows" or sequences. For example, we will use a window of the last 10 days of data ([Open, High, Low, Close, Volume, MA5, MA10]) to predict the 'Close' price of the 11th day. This transforms our time-series problem into a supervised learning format suitable for our model.

Phase 2: Architecting the Hybrid Quantum-Classical Model

- **Objective:** Design a seamless integration of a classical LSTM network with a quantum circuit.
- **Key Actions:**
    1. **Classical Backbone (LSTM):** We'll use a classical LSTM layer as the primary component. LSTMs are exceptionally good at remembering long-term dependencies, making them ideal for financial time-series. This layer will process the input data windows and learn their temporal structure.
    2. **Quantum Enhancement Layer (VQC):** The core of our quantum advantage.
        - The output from the LSTM layer will be passed to a Variational Quantum Circuit.
        - **Data Encoding:** We will use **Angle Encoding** (AngleEmbedding in PennyLane) to load the classical features from the LSTM into the quantum state of our qubits.
        - **Quantum Circuit Design:** The VQC will consist of a series of parameterized quantum gates (e.g., CNOTs and rotational gates). These parameters are trainable, just like weights in a classical neural network. The circuit will process the encoded data, creating complex correlations and features in a high-dimensional Hilbert space that are difficult for classical models to capture.
    3. **Output Layer:** The measurement outcome from the quantum circuit will be fed into a final classical dense layer, which will produce the final 'Close' price prediction.

Phase 3: Training and Optimization

- **Objective:** Efficiently train the entire hybrid model end-to-end to minimize prediction error.
- **Key Actions:**
    1. **Frameworks:** We will use **PennyLane** for the quantum components and **PyTorch** for the classical components. These libraries are designed to integrate seamlessly.
    2. **Loss Function:** We'll use **Mean Squared Error (MSE)** to quantify the model's prediction error, as required by the competition rules.
    3. **Optimizer:** The **Adam optimizer** will be used to update the parameters of both the classical LSTM and the quantum circuit simultaneously. The magic of frameworks like PennyLane is that they can automatically calculate the gradients through the quantum circuit (using the parameter-shift rule), allowing for standard backpropagation.
    4. **Training Protocol:** We will train the model on the windowed data from X_train.csv, iterating for a set number of epochs until the loss on a validation set converges.

Phase 4: Prediction and Evaluation

- **Objective:** Use our trained model to predict the missing 'Close' values in X_test.csv and format the submission.
- **Key Actions:**
    1. **Iterative Prediction:** Since we need to predict a sequence of 10 days, we'll use an iterative (or auto-regressive) approach:
        - Take the last 10 days of data from the training set to predict the **first** day in the test set.
        - Append this new prediction to our sequence.
        - Use the updated 10-day window to predict the **second** day.

    - Repeat this process until all 10 'Close' values for X_test.csv are generated.

2. **Inverse Scaling:** The model's predictions will be in the normalized scale. We must use our saved MinMaxScaler to transform these predictions back to their original price scale.
3. **Submission:** The final predictions will be formatted into a predictions.csv file with 'Date' and 'Close' columns, as per the example provided.
4. **Evaluation:** We will assess our model using the specified metrics: **Mean Squared Error (MSE)** and the **R^2 Score**. A lower MSE and an R^2 score closer to 1 will indicate a successful model.