Quantum Support Vector Regression for Stock Price Forecast

Melvin Varghese
Department of Electrical and Computer Engineering
University of Southern California
mtvarghe@usc.edu

***Abstract*—Quantum Computing offers an exciting potential for developing more efficient algorithms capable of handling the complexities of financial data. This project explores the feasibility of Quantum Support Vector Regression (QSVR) for stock price prediction, leveraging the potential of quantum algorithms to improve computational efficiency and accuracy over classical methods. Based on a hybrid quantum-classical framework, quantum circuits are used to process and model nonlinear patterns in financial time series data, whereas classical computing components handles data pre-processing and post-processing.**

The methodology for this project involves constructing a quantum feature map to transform classical data into a quantum state and using a variational quantum circuit to implement the regression. The performance of QSVR is benchmarked against historical stock price data.

This project not only demonstrates the potential of quantum algorithms in financial predictions. It gives evidence for the promised quantum advantage in several applications.

***Index Terms*—Quantum Computing, Financial Modelling, QSVR, Quantum Feature Maps.**

I. Introduction

The support vector machine (SVM) is a well-developed and widely used technique for classification and regression, mainly because of its strength and effectiveness in high-dimensional spaces. SVMs work by computing a hyperplane that best separates different classes in feature space, or in the case of regression, by fitting the best line within a threshold error margin. Although SVMs have been successful, there are some deficiencies for SVMs that should be improved.

The advent of quantum computing into the scientific community carried the promise of improving computational capacities over those available with current, classical methodologies. This improvement would be by virtue of superposition and entanglement, enabling quantum computers to function in ways that are fundamentally unavailable to classical systems. In this regard, Quantum Support Vector Machines (QSVMs) become a very very interesting quantum-classical hybrid approach.

QSVMs modify the classical support vector machine algorithm in the quantum computing framework. QSVMs attempt to exploit quantum mechanical properties so as to bring about the realization of the information processing operation with higher efficiency. This is mostly done through quantum feature maps, where classical data is embedded into quantum states and the quantum system, theoretically speaking, can evaluate more complex, high-dimensional feature spaces better than classical algorithms. Quantum Support Vector Regression (QSVR) extends the QSVM through application in regression predictive modeling. The quantum data-to-state is done by a similar quantum feature map used by QSVM; subsequently, the QSVR is modeled and the continuous outcome is predicted by a variational quantum circuit. We believe that this method could deliver breakthrough improvements in predictive accuracy and computational speed on tasks where there are complex nonlinear relationships within large datasets.

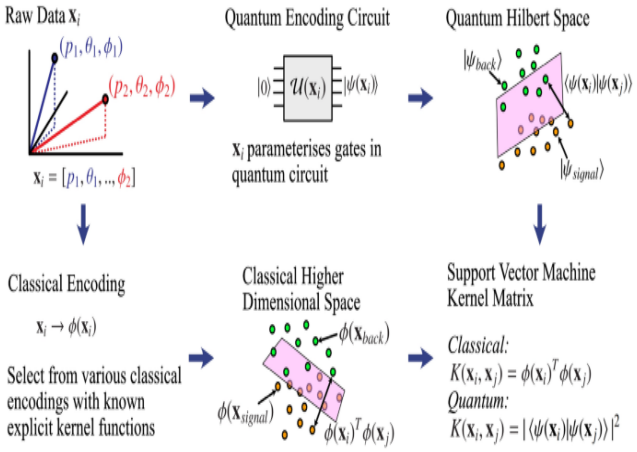

Fig. 1. Pipeline for a QML model

In QSVR, the financial domain is considered to be one of the most suitable application areas because of its nonlinearity and dependence on several interrelated factors, such as economic indicators, political events, and market sentiment. In fact, the nature of financial markets is always characterized by some nonlinear property, under the influence of plenty of factors associated with each other, like economic indicators, political events, and market sentiment. All these complexities make the normal models fail, so their great interest in more capable algorithms. The game is played not only through modeling the complex influences but in the timely and accurate modeling; in this, quantum-enhanced algorithms may bring about a considerable betterment. In this project, QSVR is applied in a challenge regarding the forecasting of stock prices, testing not only the theoretical advantages of quantum computing against classical approaches but also its applicability and effectiveness in practice while performing real-world financial analytics.

II. Key Concepts

Stock Market: The stock market is a financial system where shares of publicly-held companies are issued, bought, and sold. In today's economy the stock market serves a crucial role, providing companies with access to capital in exchange for giving investors a slice of ownership and potential profits. Several factors such as economic indicators, company performance, and market sentiment influence the prices in the stock market. For investors and analysts, predicting stock prices is a central challenge, driven by the potential for significant financial gains.

Regression Models: Regression models are statistical constructs designed to predict a continuous outcome variable (dependent variable) based on the value of one or more predictor variables (independent variables). The goal is to establish a relationship between the predictors and the target variable, which can be used for forecasting future outcomes. Regression models vary from simple linear regression to more complex models like polynomial and logistic regression.

Support Vector Regression (SVR): SVR is a form of Support Vector Machines (SVM), a popular machine learning algorithm used for classification and regression tasks. In SVR, the objective is to find a function that has the minimum deviation from the actually obtained targets in the training data, which is also as flat as possible. SVR is particularly useful due to its ability to manage high-dimensional data and its robustness against overfitting, especially in cases where the number of dimensions exceeds the number of samples.

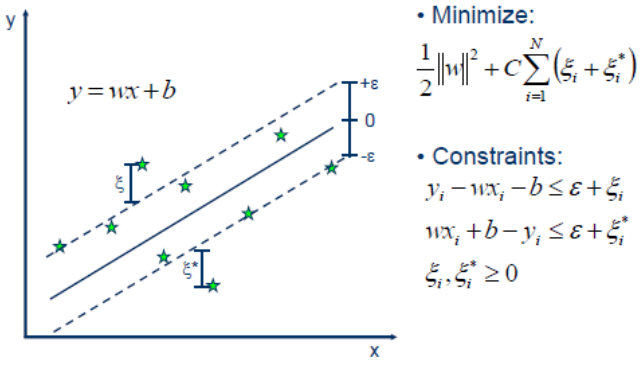

Fig. 2. Support Vector Regression model.

Quantum Support Vector Regression (QSVR): QSVR is a combination of the SVR model with the quantum computing method. By using quantum feature maps, QSVR can potentially model complex, non-linear decision boundaries more effectively than its classical counterparts. The quantum enhanced SVR c

Quantum Feature Maps: Quantum feature maps are the tools that actually provide the quantum advantage, where classical data is encoded into a high-dimensional quantum state. The feature map effectively raises the data into quantum mechanical data, which quantum algorithms can process. The use of quantum feature maps is crucial for implementing machine learning algorithms on quantum computers, as they allow for leveraging quantum computing effects such as superposition and entanglement to explore correlations in data that are intractable to classical approaches.

III. Methodology

As established, forecasting the prices of commodities in the stock market is a very crucial problem in the financial sector. There are numerous factors that affect the daily movement of the prices of these equities. Several models have been developed to evaluate a stock and it's potential trajectory in the future. Additionally, forecasting itself is categorized based on several parameters, with an important one being the timeframe. Different models are used for short-term and long-term forecasts.

In any forecast models, the general paradigm is to use historical data attributed to a stock, such as it's daily closing prices, year high price and so on, to arrive at a correlation between these parameters and the the future price of a stock. Regression models generally perform a significant role in these tasks. If the model created can map the correlations to the data to a high accuracy, then the forecast results can be very precise. Therefore, the choice of the model and the features that are used for training plays a significant role in the success of the model. In a area, where even marginal improvements in forecasting can have monumental results, a more successful model can help an investor take better decisions about selling/buying a stock and strategizing to manage good portfolio of stocks.

In this work, we use the Simple Moving Average(SMA) of a stock, as one of the metrics for predicting the movement of a stock. SMA is a widely used parameter in forecasting models. The SMA is calculated simply by taking the average price of the stock for a specific number of days prior to a particular day. SMA is calculated for any arbitrary timeframe and an expertise in the domain will help to decide what periods are to be chosen for the SMA. For this project, we choose two periods of 5 and 10 days and calculate the SMA for both these intervals.

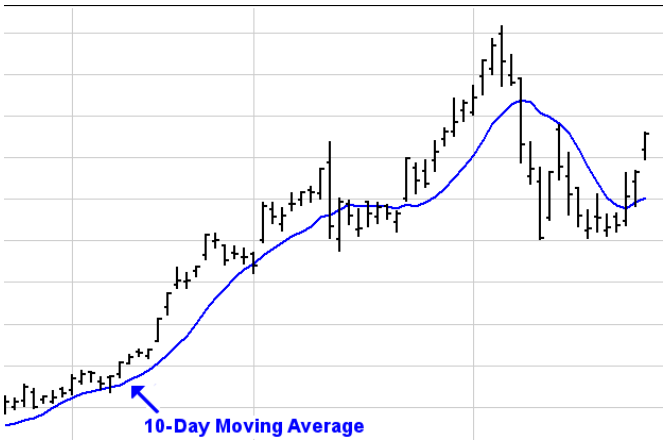

Fig. 3. SMA graph with 10 day interval.

This, along with the Closing value of the stock on each day will be taken as the features that are used to train the QSVR model. In this work, we take 1 year of the stock data of the blue chip firm Apple(NASDAQ: 'AAPL') from January 2020 to 2020 December. The daily stock data of any stock can be obtained through various methods. For this work, the YFinance python package has been used to retrieve the data from Yahoo Finance server. The next step in this project is the processing of the data to a form that can be handled using Machine Learning(ML) techniques, which is at the core of this work. We use methods from the Scikit-learn package to pre-process the data into relative values less than or equal to 1, so that the ML models can train efficiently.

At this stage, the classical part of this hybrid quantum-classical model is over. The data is ready to be fed for training the model. A quantum kernel which is used to map the data into higher dimensions is instantiated at this point. This is the step that provides the quantum step to the model. As opposed to classical models, the feature map that is used here is the ZZFeatureMap. The ZZFeatureMap is specifically structured to capture interactions between different features using quantum gates. It consists of layers of single-qubit rotations and two-qubit entangling gates. Typically, the single-qubit rotations are based on the data being encoded, and the entangling gates (ZZ gates) are used to create correlations between different qubits. Once the quantum kernel is instantiated

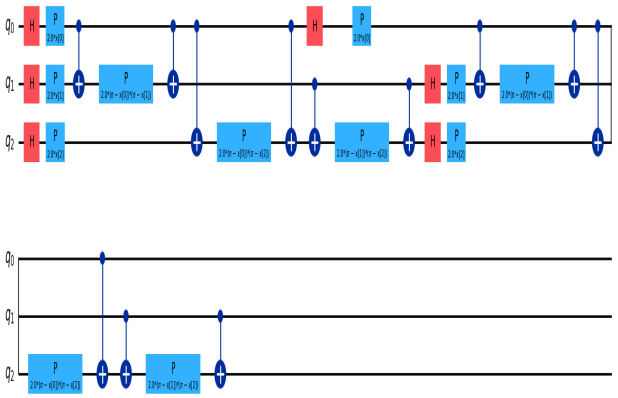

Fig. 4. ZZFeatureMap for this QSVR

with the above mentioned feature map, the model is now fitted with the training data that was prepared. The training data is split for the original stock data for 1 year, and a test data set is also created from that, which is used to test the accuracy of the model. Once the model is trained with the training data, it is then tested against the test data to compute the accuracy.

IV. Results

Once the model is trained and fitted, we use the the score function available as an attribute to the QSVR method to compute the accuracy of the model. The model is run on a QasmSimulator that can simulate the noise on an actual quantum device. One measure of the accuracy of a regression model is the R-square value, which is a metric that denotes the measure of the correlation between the features that were used to predict the price and the the actual price. An R-square value of 1 means that the model can perfectly map all the variable correlations. In this model, the R-square value was computed to be 0.31. This is a promising value of R-square, since the model only uses minimal features. For data that is impacted by human behaviour, it is generally determined that regression models would typically have a maximum value of 0.5, since human behaviour is unpredictable, especially when it comes to financial decisions. The graph obtained from the

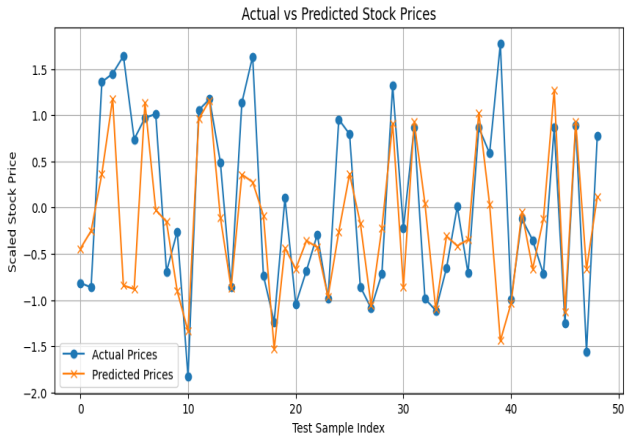

Fig. 5. Experiment on classical QSVR

actual data shows that QSVR model can very well capture the trend of the stock price movement. In many cases, it is accurately predicting the correct prices, even in cases it is not, it is very close to the correct values. The Y axis in the graph gives the scaled stock price with the 0.0 being the scaled average price of the stock.

V. Challenges

In this work, we have used three features for training the model and for predicting the stock value for the future. There are numerous features of a wide variety which convey different information about the state of the stock. However, it is not possible to train the model using all those available features at the moment, since the QSVR utility offered by Qiskit is a relatively novel implementation. The choice to execute this model on actual quantum devices have not yet been developed for this utility. The lack of access to real hardware means that the actual quantum advantage is not yet realized in this method. The quantum kernel that was instantiated using the ZZFeatureMap is run on a simulator. This also means that only a model with very few parameters can be run, as classical computers don't have the computing power to simulate quantum systems with more than a few qubits. Additionally, as is the case with many quantum computing applications that can be already done classically, there is the requirement to show the quantum advantage over classical model. For a

substantially large enough dataset, with a significant number of features, QSVR has the potential to give more accurate and robust models than classical regression models, because the classical ones will not be able to learn all the different feature correlations.

VI. Future Scope

In this project, we have employed the Quantum Support Vector Regression(QSVR) model for predicting the movement of stocks and their prices in the future. Once the QSVR utility is developed with more functionality, there can be a significant improvement in the modelling capabilities of the model. One other caveat with the model is that, although few parameters other than the kernel are also are defined for the model, such as 'gamma', 'epsilon', there is no option available to alter these parameters, as the QSVR method only takes in quantum kernel as parameter. When such options do come in the future, along with access to real quantum hardware, QSVR will be a very exciting candidate for stock modelling. Similarly, the current generation of NISQ devices are also limited in their capacity, and once devices with more qubit connectivity, better error mitigation techniques come along, these models hold much promise.

VII. Python Code Snippet

This is a condensed version of the Python code that was used to execute the model. Only the relevant parts of the code have been attached here to give a sense of how the model was implemented.

```
import pandas as pd
import numpy as np
import yfinance as yf
import matplotlib.pyplot as plt
from qiskit_aer import QasmSimulator
from qiskit.circuit.library import ZZFeatureMap
from qiskit_machine_learning.kernels import
    FidelityQuantumKernel
from qiskit_machine_learning.algorithms import QSVR
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

# Download data
data = yf.download('AAPL', start='2020-01-01', end='
    2021-01-01')
prices = data['Close'].values
dates = data.index

# Preprocess data: Use the closing prices and
    generate some simple moving averages as features
data['SMA_5'] = data['Close'].rolling(window=5).mean
    ()
data['SMA_10'] = data['Close'].rolling(window=10).
    mean()
data.dropna(inplace=True)

# Features and labels
X = data[['Close', 'SMA_5', 'SMA_10']].values
y = data['Close'].shift(-1).values[:-1]  # Next day'
    s price
X = X[:-1]  # Remove the last element to match the
    label size

# Scale features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
y_scaled = scaler.fit_transform(y.reshape(-1, 1)).
    flatten()

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y_scaled, test_size=0.2, random_state
    =42)

# Quantum machine learning setup
backend = QasmSimulator()
feature_map = ZZFeatureMap(feature_dimension=3, reps
    =2, entanglement='full')
quantum_kernel = FidelityQuantumKernel(feature_map=
    feature_map)
qsvr = QSVR(quantum_kernel=quantum_kernel)
qsvr.fit(X_train, y_train)

# Prediction using the QSVR model
y_pred = qsvr.predict(X_test)

# Accuracy of the QSVR model
score = qsvr.score(X_test, y_test)
print("Accuracy of QSVR model:", score)

# Plotting the results
plt.figure(figsize=(10, 5))
plt.plot(y_test, label='Actual Prices', marker='o')
plt.plot(y_pred, label='Predicted Prices', marker='x
    ')
plt.title('Actual vs Predicted Stock Prices')
plt.xlabel('Test Sample Index')
plt.ylabel('Scaled Stock Price')
plt.legend()
plt.grid(True)
plt.show()
```