Financial Instrument Prediction Using Machine Learning: LSTM Model Report

# 1. Introduction

This report outlines the use of a Long Short-Term Memory (LSTM) model to predict future prices of financial instruments based on historical data. The algorithm leverages deep learning techniques to model the temporal dependencies in asset prices and make forward-looking predictions.

# 2. Model Overview

## 2.1 Long Short-Term Memory (LSTM) Model

LSTM is a type of recurrent neural network (RNN) designed to capture long-term dependencies in sequential data. Unlike traditional RNNs, LSTMs can learn from both short-term and long-term patterns, making them particularly effective for time series forecasting, such as predicting stock prices.

## 2.2 Model Architecture

The LSTM model used in this prediction algorithm has the following architecture:  
- Input Layer: Accepts a sequence of historical stock prices.  
- LSTM Layers: Two LSTM layers with 50 units each, the first returning sequences and the second not.  
- Dropout Layers: Two dropout layers with a 20% dropout rate to prevent overfitting.  
- Dense Layers: Two dense layers, the final layer outputs a single value predicting the next price.

## 2.3 Data Preprocessing

The input data undergoes several preprocessing steps:  
- Normalization: Stock prices are normalized to a range between 0 and 1 using `MinMaxScaler`.  
- Sequence Creation: Historical data is segmented into sequences (typically 60 time steps) to train the model.  
- Scaling: The LSTM model requires scaled input data to ensure consistent performance.

# 3. Prediction Process

## 3.1 Training the Model

The LSTM model is trained using historical data, where it learns the patterns in the stock prices. The model is validated using a portion of the data not seen during training, allowing the evaluation of its performance.

## 3.2 Making Predictions

After training, the model can predict future prices:  
- Future Days: The `future\_days` parameter allows the user to specify how many days ahead they want to predict. By default, this is set to 30 days.  
- Iterative Prediction: The model makes predictions iteratively, updating the input sequence with each new predicted price to predict the subsequent day's price.

## 3.3 Handling Non-Business Days

The algorithm incorporates a calendar to exclude weekends and holidays, ensuring that predictions are made only for business days. This is critical for accuracy, as stock markets do not operate on these days.

# 4. Visualization and Output

## 4.1 Actual vs. Predicted Prices

The model generates graphs comparing the actual historical prices with the predicted prices. These graphs are saved as PNG files and provide a visual representation of the model's accuracy.

## 4.2 Future Price Predictions

Predicted future prices are saved in CSV files, containing the date and predicted price for each business day in the specified future period.

## 4.3 Output Directory

Both the graphs and the CSV files are saved in the specified directory (`/Users/romingandhi/Desktop/Portfolio Optimization/data/Predicted Prices`), ensuring organized output storage.

# 5. Performance Considerations

## 5.1 Runtime

The time required to generate predictions depends on several factors:  
- Model Complexity: The number of layers and units in the LSTM.  
- Data Size: The length and volume of the historical data.  
- Hardware: The use of CPU or GPU significantly impacts performance. Running on a GPU can greatly reduce computation time.  
- Future Days Count: More days to predict increases the time required due to iterative processing.

## 5.2 Optimization Recommendations

To optimize performance:  
- Simplify the Model: Reduce the number of layers or units if runtime is an issue.  
- Use GPU Acceleration: If available, utilize GPU support for faster computations.  
- Reduce Future Days: Start with a smaller number of future days for quick evaluations and then scale up as needed.

# 6. Conclusion

This LSTM-based algorithm provides a powerful tool for predicting future stock prices based on historical data. By incorporating deep learning techniques, it captures complex temporal patterns and offers a flexible approach to forecasting. While effective, the model's performance can be impacted by runtime considerations, which can be mitigated through optimization strategies like model simplification and hardware acceleration.  
  
The results, including actual vs. predicted prices and future price predictions, are visualized and saved in a structured manner, providing valuable insights for financial analysis and decision-making.