CS240 Project Presentation

Mradul Sonkar (23B0980)
Vijaya Raghavendra (23B1042)

May 3, 2025

Input

A time series dataset of stock market features including open prices, close prices, and trading volume, along with their respective lags.

Output

A real-valued prediction representing the closing price of the stock in the next interval.

Example

Input:

Time	Open Price	Close Price	Volume	Lagged Features
$t-2$	101.25	102.10	1205000	...
$t-1$	102.30	101.80	1150000	...
t	101.90	102.50	1234000	...

Output:

Predicted Close Price at $t+1$ = **102.85**

Description

This project aims to build a predictive model for stock prices using sequential deep learning techniques. Specifically, it uses a Long Short-Term Memory (LSTM) network to capture temporal dependencies in historical price and volume data. The model is trained and evaluated using a streaming setup enabled by the **RollingRegressor** from the **deep-river** library, making it suitable for real-time or online prediction scenarios.

Why did we choose this problem?

The idea for this project originated from a previously suggested stock market prediction task in the CS240 course, which was based on old and static historical data. The project was useful but it lacked real-time applicability. To address this gap, we aimed to develop a more relevant and practical system—one capable of making stock market predictions on up-to-date data in a streaming setting. This enhances both the realism and the impact of the problem, making it more aligned with real-world financial systems where timely decisions are crucial.

What makes it interesting, impactful, or challenging?

Financial markets are influenced by various dynamic factors which makes them interesting. Accurate predictions can help investors and financial planners making it impactful. Stock market prediction is challenging as it requires capturing temporal dependencies and reacting to sudden market changes.

Is there a real-world application or gap this project addresses?

Yes, this project bridges the gap between traditional statistical models and deep learning-based sequence models for stock forecasting. Most retail investors lack access to robust forecasting tools—this app/program brings predictive insights to their fingertips.

Example

Input:

Time	Open Price	Close Price	Volume	Lagged Features
$t-2$	101.25	102.10	1205000	...
$t-1$	102.30	101.80	1150000	...
t	101.90	102.50	1234000	...

Output:

Predicted Close Price at $t+1$ = **102.85**

Source

The data is collected using the `yfinance` Python API from Yahoo Finance.

Citation: Yahoo Finance API

Model Architecture

The model is a two-layer bidirectional LSTM followed by two fully connected layers:

- **Input:** Sequence of vectors (each with n features), representing past timesteps
- **LSTM:** 2 layers, 32 hidden units per direction, dropout = 0.2, bidirectional
- **Dense Layers:** A ReLU-activated hidden layer of size 32, followed by a final linear layer outputting a single real value (the predicted close price)

Methodology

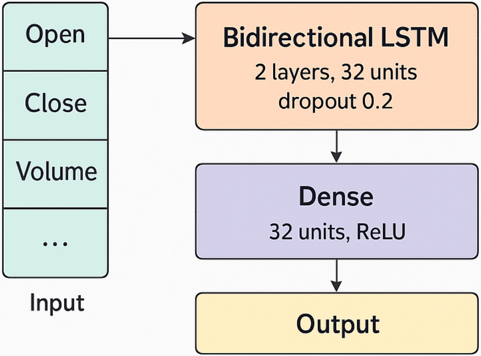


Figure 1: Model Architecture

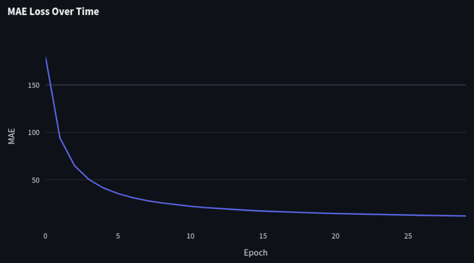
Intuition

The bidirectional LSTM captures both forward and backward dependencies across the input sequence, improving pattern recognition in short-term trends. The dense layers enhance learning capacity and non-linearity, enabling better generalization over noisy stock price data.

Implementation

Framework:	PyTorch
Training Strategy:	Online learning using **RollingRegressor** from **deep-river**
Loss Function:	Mean Squared Error (MSE) and Mean Absolute Error (MAE)
Optimizer:	Adam (learning rate = 0.001)
Features:	Lagged Open, Close, and Volume over 3 timesteps (t, $t-1$, $t-2$)

Performance Metrics



Metric	Value
MSE	162.04
MAE	11.96

Figure 2: Error vs Epochs

Predictions

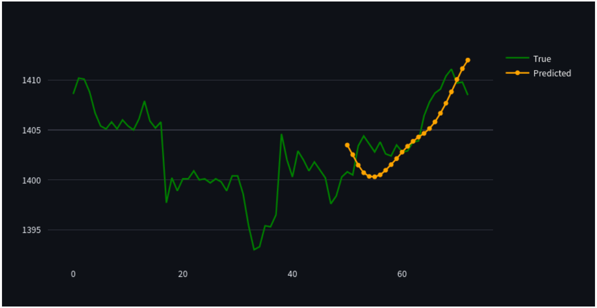


Figure 3: Actual vs. Predicted Values

Technical Learnings

Through this project, we gained in-depth experience with time series forecasting, particularly using LSTM (Long Short-Term Memory) networks. We learned how to preprocess financial data, integrate multiple features such as open, close, and volume prices, and handle issues related to missing data and noise. We also learnt using libraries like **PyTorch** for implementing neural networks and **deep-river** rolling regressors for real-time prediction.

Problem-Solving

We learned how to approach real-world problems where data is dynamic, unstructured, and noisy. Addressing issues like overfitting, underfitting, and hyperparameter tuning taught us the importance of balancing model complexity with performance. Experimenting with different architectures (e.g., bidirectional LSTMs) enhanced my problem-solving skills in a dynamic and unpredictable environment like the stock market.

Experimentation

The project involved continuous experimentation with model parameters, training strategies, and input features to improve prediction accuracy. We learned the importance of tracking results, analyzing errors, and refining hypotheses to iterate toward the best solution. This experimental mindset is essential for refining machine learning models and optimizing them for deployment.

Thank You!