Duke
UNIVERSITY

FORECASTING STOCK MARKET TRENDS USING LEARNING-BASED TIME-SERIES MODELING TECHNIQUES

Authors:

Mohammad Afrazi

Patrick Lo

Problem Description

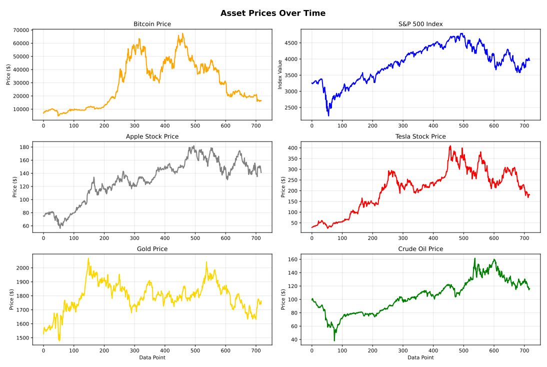
Predicting stock market movements is highly challenging due to the market's volatility, non-linearity, and chaotic dynamics [1]. Traditional time-series models such as ARIMA struggle to capture complex temporal dependencies in financial data, limiting forecasting accuracy and decision-making in finance [3].

We argue and hypothesize that learning-based time-series models - specifically Long Short-Term Memory (LSTM) networks, Temporal Convolutional Networks (TCNs), and Transformer- based models - can better model long-term dependencies in stock price data. This project will develop and compare these models for forecasting the daily closing price of the **S&P 500** index using historical price. Via evaluating LSTM, TCN, and Transformer approaches on a recent dataset of U.S. market data, we aim to advance financial forecasting methods and provide more reliable tools for investment and risk management [4].

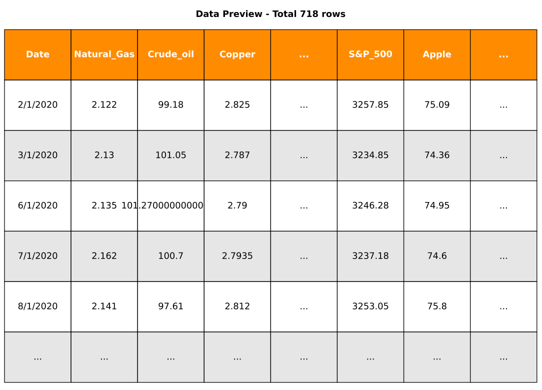
Data Description

To address our research question, we utilize the "US Stock Market Dataset" from Kaggle [2]. This dataset contains over five years of daily data, comprising 36 features. These features include price and volume data for major stock indices (S&P 500, Nasdaq 100), key individual stocks (e.g., Apple, Tesla, Nvidia), major commodities (e.g., Crude Oil, Gold, Silver), and leading cryptocurrencies (e.g., Bitcoin, Ethereum). Our primary objective is to forecast the S&P 500 closing price, treating the remaining asset data as predictive features.

To mitigate the effects of major scale differences across time periods, we restrict our analysis to data from January 2, 2020, to November 29, 2022, yielding a total of 718 daily observations. Figures 1a and 1b provide visual summaries of the dataset, illustrating overall asset price dynamics and a sample of the input features used in our analysis.



(a) Asset trends: Bitcoin, S&P 500, etc.



Data Preview - Total 718 rows

Date	Natural_Gas	Crude_oil	Copper	...	S&P_500	Apple	...
2/1/2020	2.122	99.18	2.825	...	3257.85	75.09	...
3/1/2020	2.13	101.05	2.787	...	3234.85	74.36	...
6/1/2020	2.135	101.27000000000	2.79	...	3246.28	74.95	...
7/1/2020	2.162	100.7	2.7935	...	3237.18	74.6	...
8/1/2020	2.141	97.61	2.812	...	3253.05	75.8	...
...	...	...	...	...	...	...	...

(b) Sample of raw dataset features.

Figure 1: Visualization of the dataset showing asset price dynamics and a sample of the input data.

Preprocessing

Log Return

To make the data more stationary and suitable for modelling, we first compute the **log returns** of each asset. This is to normalize price movements and stabilize variance across different scales. The log return for time k is calculated as

$$r_k = \ln\left(\frac{P_k}{P_{k-1}}\right),$$

where P_k represents the asset price at time k. It also helps mitigate the effects of large price discrepancies between assets, ensuring that model training focuses on relative changes rather than absolute price levels. Compared to Figure 1a, below shows the log-return transformed of all asset prices.

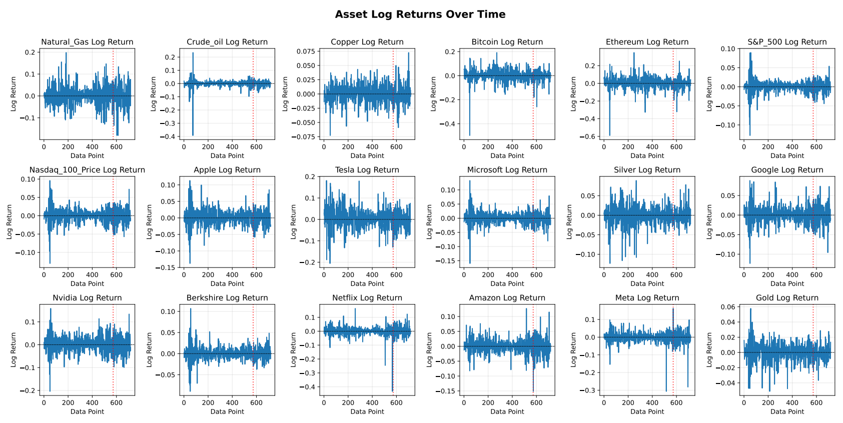


Figure 2: Log-return transformed asset price trends.

Dimension Reduction

Here we perform **lag-principal component analysis (Lag-PCA)** to identify the most informative features in the dataset and reduce dimensionality. Our core hypothesis is that financial time-series data is **non-stationary**; the importance of a given feature (e.g., 'Crude_oil') may change significantly over time. A single, "global" PCA performed on the entire dataset would average out these local temporal dynamics.

To capture these time-varying relationships, we create data chunks by sliding a window of size $l = 70$ over the time series. For each 70-day chunk, we perform PCA independently. This local PCA allows us to rank the importance of the *original features* based on their contribution (i.e., factor loadings) to the principal components *within that specific time window.*

We then apply a **majority voting** scheme to aggregate these local rankings and identify the features that are *consistently* important across all chunks. We maintain a global score vector V, where V_i is the total score for the i^{th} original feature. For each window, we update this score using the following heuristic:

$$V_i \leftarrow V_i + n * \frac{1}{\text{rank}_i}.$$

Here, rank_i denotes the importance rank of the i^{th} feature (from 1 to 36) within the current chunk, and n is the number of principal components we consider to determine this rank (which we set to $n = 10$). This heuristic strongly rewards features that are frequently ranked as most important (i.e., have a low rank_i).

After sliding the window across the entire dataset, the 10 *original features* with the highest aggregated scores (V_i) are selected as the final, most informative features for model training. Figure 3 shows the final aggregated scores from this Lag-PCA process.

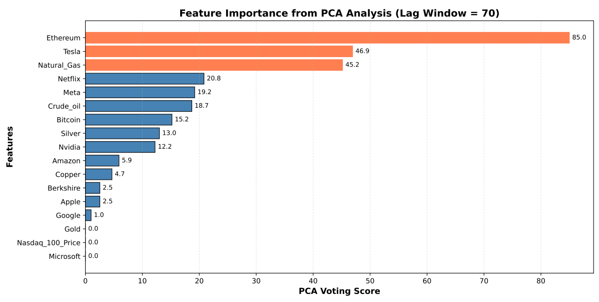


Figure 3: Lag-PCA feature importance.

Methods

In this project report, we implement and compare three learning-based time-series models: Long Short-Term Memory (LSTM) networks, Temporal Convolutional Networks (TCNs), and Transformer-based models. All models are trained to take a sequence of past data to predict the S&P 500 log return for the following day.

LSTM

We implement the LSTM model using PyTorch. The model's architecture is designed to handle sequences of the $N = 10$ features selected by our Lag-PCA.

The model takes an input tensor of shape (W, N), where W is the lookback window (e.g., $W = 10$ previous days) and $N = 10$ is the number of features. This sequence is processed by a single LSTM layer with $H = 32$ hidden units. We use the output from the final time step of the LSTM layer, which represents the encoded state of the entire sequence. This output (of size $H = 32$) is then passed through a dropout layer (with dropout rate 0.3) followed by two fully connected (Linear) layers. The first fully connected layer reduces the dimension from $H = 32$ to $F = 16$, and the final layer maps this to a single output value ($O = 1$), representing the predicted log return for the next time step.

The model is trained using the Adam optimizer with a learning rate of 0.001 and Mean Squared Error (MSE) as the loss function. We employ early stopping, monitoring the validation loss with a patience of 10 epochs, to prevent overfitting and save the best-performing model.

Preliminary Results

Here we include the preliminary results of the LSTM model. The training and validation loss curves are shown in Figure 4. We observe that the training loss decreases steadily,

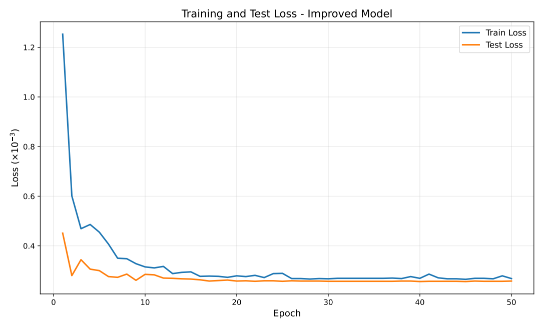


Figure 4: Training and Test Loss for the Improved LSTM model (1 LSTM layer, 32 hidden units, 2 FC layers, and Dropout)

while the validation loss starts to increase after a certain number of epochs, indicating potential overfitting.

Conclusions and Future Work

- The initial **LSTM** model for time-series prediction has been successfully implemented and evaluated.
- We observed that the test loss is notably higher than the training loss, which indicates potential **overfitting** of the model. To mitigate this, our primary plan is to expand the dataset by incorporating more historical data.
- Pending dataset expansion, we intend to implement and benchmark the performance of **Temporal Convolutional Networks (TCN)** and **Transformer-based models** against the established LSTM baseline.
- Additionally, the performance of these deep learning approaches will be contextualized by comparing their results against traditional **regression models**.
- We also plan to investigate the impact of our dimensionality reduction strategy by conducting an analysis on whether reducing the number of **PCA components** significantly degrades model performance.
- Finally, we aim to conduct a comparative study by training a model on the **raw price data** (without the log-return transformation) to validate the efficacy of our preprocessing methodology.

Work Breakdown:

- **Mohammad Afrazi:** Data preprocessing, feature engineering, writing, and model training/evaluation.
- **Patrick Lo:** Data preprocessing, feature engineering, writing, and model training/evaluation.

We, Mohammad Afrazi and Patrick Lo, affirm that we have adhered to the Duke University Honor Code in the completion of this project.

References

[1] Thomas Fischer and Christopher Krauss. Deep learning with long short-term memory networks for financial market predictions. *European journal of operational research*, 270(2):654–669, 2018.

[2] Saket K. “2019–2024 us stock market data: 5-year trends in crypto, stocks, metals, and energy markets”. Kaggle Dataset, 2024. Accessed: 2025-11-11.

[3] Mehdi Khashei and Mehdi Bijari. A novel hybridization of artificial neural networks and arima models for time series forecasting. *Applied Soft Computing*, 11(2):2664–2675, 2011.

[4] David MQ Nelson, Adriano CM Pereira, and Renato A De Oliveira. Stock market’s price movement prediction with lstm neural networks. In *2017 International joint conference on neural networks (IJCNN)*, pages 1419–1426. Ieee, 2017.