FPT UNIVERSITY

FINAL REPORT

LSTM and xLSTM in Machine Learning for Stock Price Prediction

July 17, 2024

1 Introduction

1.1 Overview

1.1.1 Project Information

- Project title : LSTM and x-LSTM in Machine Learning for Stock Price Prediction.
- Group name: Group 07.

1.1.2 Project Team

- Supervisor

Full Name	Email	Mobile	Title
Nguyen Quoc Trung	trungnq46@fe.edu.vn	0979350707	Lecturer

- Team Members

Full Name	Email	Mobile	Title
Nguyen Hoang Anh	anhnhse171292@fpt.edu.vn	0377713873	Leader
Dinh Thi Kim Mai	maidtkse171703@fpt.edu.vn	0932631810	Member
Nguyen Hoang Viet	vietnhse171125@fpt.edu.vn	0925024404	Member

2 Background and Context

2.1 Objection

The primary objective of this project is to explore the application of Long Short-Term Memory (LSTM) networks in predicting stock prices. This involves developing a robust model that can analyze historical price data and other relevant financial indicators to forecast future stock prices with high accuracy. The goal is to provide a tool that assists investors in making informed decisions based on predicted market trends.

2.2 Motivation

The motivation behind using LSTM networks for stock price prediction stems from the inherent complexity and volatility of financial markets. Traditional linear models often fall short due to their inability to capture the sequential dependencies and non-linear patterns present in stock price movements. LSTMs, with their capacity to remember long-term dependencies, offer a promising alternative. By leveraging the strength of LSTMs in handling time-series data, this project aims to significantly enhance the predictive accuracy compared to conventional models.

2.3 Background information

Stock price prediction has been a focal area of research within financial economics, with numerous methodologies being tested over the years. Traditional techniques such as ARIMA and exponential smoothing were once prevalent but often lacked the flexibility to model the dynamic changes in the market effectively. The introduction of machine learning has shifted the paradigm, allowing for more complex representations of market behaviors.

LSTMs, a special kind of Recurrent Neural Network (RNN), have gained popularity in sequence prediction due to their ability to capture temporal dependencies over different time intervals. First introduced by Hochreiter & Schmidhuber (1997) [1], LSTMs have been successfully applied in various domains, including natural language processing, speech recognition, and more recently, financial forecasting. The relevance of LSTMs in financial applications is highlighted in works like Fischer and Krauss (2018) [2], who demonstrated that LSTMs could outperform traditional models in stock prediction tasks.

This project is inspired by the success of LSTMs in these domains and aim to use various types of LSTM (at least one), while also making comparisons between traditional LSTM and its variants to gain a more objective perspective.

3 Literature review

3.1 Overview of LSTM Networks

Long Short-Term Memory (LSTM) networks are a specialized kind of recurrent neural network (RNN) capable of learning long-term dependencies. They were introduced by Hochreiter and Schmidhuber in 1997 and are particularly useful in sequence prediction problems due to their ability to remember information for extended periods.

Base on based on Analytics Vidhya's article titled: "What is LSTM? Introduction to Long Short-Term Memory" [3], unlike traditional neural networks, LSTM incorporates feedback connections, allowing it to process entire sequences of data, not just individual data points. This makes it highly effective in understanding and predicting patterns in sequential data like time series, text, and speech. LSTM has become a powerful tool in artificial intelligence and deep learning, enabling breakthroughs in various fields by uncovering valuable insights from sequential data.

3.2 Emerging Trends in LSTM Architectures

A number of modifications to the original LSTM architecture have been suggested over the years such as: Peephole Connections, Gated Recurrent Unit, Multiplicative LSTM and most recently, the introduction of xLSTM, which was announced on May 7, 2024, in a scientific study titled: 'xLSTM:

Extended Long Short-Term Memory' [4].

The Extended Long Short-Term Memory (xLSTM) advances traditional LSTM networks by integrating exponential gating and innovative memory structures, such as scalar and matrix updates. These enhancements, combined into residual block backbones, enable xLSTM to compete with, modern Transformers in performance and scalability, marking a significant evolution in deep learning technologies.

In this project, we are planning to employ both the classic LSTM model and the newly introduced xLSTM for the purpose of predicting stock prices. We aim to conduct a detailed comparison and evaluation of these two methodologies to ascertain their effectiveness and efficiency. This analysis will be performed on a consistent dataset that we have meticulously collected specifically for this study. Our objective is to determine which model offers the most accurate predictions under varying market conditions.

4 Data Description

4.1 Source

The NASDAQ, short for the National Association of Securities Dealers Automated Quotations, is a leading U.S. stock exchange famous for its technology and biotech company listings, such as Apple, Microsoft, and Amazon [5]. Founded in 1971, it was the first electronic stock market in the world. It serves as a global digital platform for trading securities.

NASDAQ provides stock data covering up to 25 years, making it an excellent fit for addressing requirements in time series analysis. The extensive dataset includes detailed stock performance metrics over the past decade, ensuring it meets the needs of various time series modeling and forecasting problems effectively.

4.2 Size And Format

NASDAQ offers daily stock prices for all listed stocks, providing up to 25 years of historical data. This data is organized in a table format, which can be downloaded for detailed analysis. Each stock's data includes comprehensive metrics such as the adjusted closing price, making it highly suitable for time series analysis and financial modeling.

4.3 Features

NASDAQ provides comprehensive historical stock data with the following features:

- **Date**: The date of stock transaction.
- **Close/Last**: The last trading price of a stock when the market closes for the day.
- **Volume**: The total number of stocks in the market.
- **Open**: The stock trading price at the beginning of the date.
- **High**: The highest trading price of the stock in the date.
- **Low**: The lowest trading price of the stock in the date.

	Open Price	Close Price	Lowest Price	Highest Price	Volume
count	6199.000000	6199.000000	6199.000000	6199.000000	6.199000e+03
mean	173.564756	173.572126	171.542004	175.488235	3.286727e+07
std	148.003103	147.938678	146.582502	149.223792	3.045830e+07
min	12.990000	13.120000	12.720000	13.190000	7.025000e+05
25%	75.115000	75.260000	74.260000	76.250000	1.189690e+07
50%	136.815000	136.870000	135.020000	138.789000	2.424802e+07
75%	197.160000	197.000000	195.040000	199.590000	4.381867e+07
max	702.410000	702.100000	699.570000	705.070000	3.326072e+08

Figure 1: Describe table of Dataset

5 Exploratory Data Analysis (EDA)

5.1 Data Cleaning and Preprocessing

Overview of data, there are no missing values in the dataset. In our project, we use Apple Stock to demonstrate. The dataset provides Apple stock price from 1999 to 2024. The describe table is in Figure 1. The first step in the exploratory data analysis involves cleaning and preprocessing the data to ensure it is suitable for building an LSTM model for stock price prediction:

- **Handling Missing Values**: The dataset was examined for missing values. In our case, the dataset did not have any missing values, which simplified the cleaning process. However, typical methods for handling missing values include imputation (using mean, median, or mode) and forward or backward filling.
- **Data Normalization**: To improve the performance and convergence of the LSTM model, the stock price data was normalized. Normalization techniques such as Z-Score were applied to scale the data within a specific range ensuring that all features contribute equally to the model training.
- **Data Splitting**: The data was split into training, validation, and test sets to ensure the model is evaluated on unseen data. Typically, 80% of the data was used for training and 20 % for testing.

5.2 Descriptive Statistics

According to Figure 1, following are the components that will be further clarified to further visualize the input data:

- **Count:** The number of observations in the dataset (10,468).
- **Mean:** The average value for each column.
- **Standard Deviation:** The measure of the amount of variation or dispersion of a set of values.
- **Min/ Max:** The minimum and maxium value in the dataset.
- **25%/50%/75%:** The 25th, 50th and 75th percentile value.

5.3 Visualization

1. **Comparison Between Open Price an Close Price 5 Years Recent**

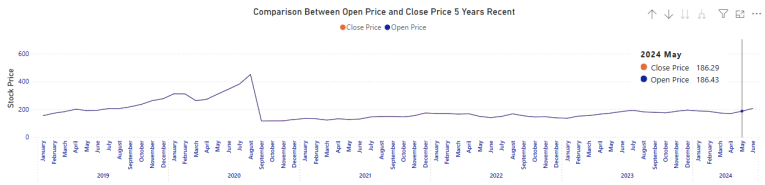

Figure 2: Daily stock price prediction

Insights: The price did not change significantly by 2021 - 2024, making a stable trend for Apple's stock price. However, the LSTM model might face some problems in learning small changes.
Solution: Included data with stable and unstable trends to help the xLSTM model predict well also in both general and detail cases.

2. **Daily Return on Stock**

Figure 3: Daily return on stock

Metric: Daily Return = (Price(t) - Price(t-1)) / 100, t = day

Insights: Insights: Shows the percentage change of the current and previous value. Daily returns represent the gains or losses on an investment over a specific length of time, especially day in here.

3. **Comparison between Highest Price and Lowest Price**

Figure 4: Comparison between Highest Price and Lowest Price

Insights: This chart represent the different between highest and lowest price of each year, reflect the growth of Apple company. The growth started at 2020, due to the Covid-19, and people began to care more about technology products.

4. **Volume of Stock Traded**

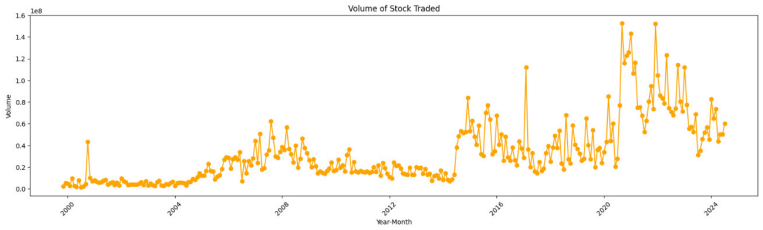

Figure 5: Volume of Stock Traded

Insights: Volume is an important feature in predicting future prices. When a stock has low trading volume, many investors may think that this stock does not have enough potential.

6 Methodology

6.1 Long Short Term Memory (LSTM)

1. **Model Selection**: The selection of the Long Short-Term Memory (LSTM) model for this project was primarily driven by its proven ability to handle sequential and time-series data, which is critical in stock price prediction. "Traditional models like ARIMA often fail to capture the non-linear dependencies and the complex temporal dynamics present in stock market data" [6]. LSTM, a type of recurrent neural network (RNN), is specifically designed to overcome these challenges by remembering long-term dependencies and providing a more accurate prediction over time. This capability of LSTM to effectively manage the vanishing gradient problem, which is common in traditional RNNs, further justifies its selection for our stock price prediction model.

2. **Data Splitting Strategy**: The dataset was split into training and testing sets to evaluate the performance and generalizability of the LSTM model. The historical stock price data from NASDAQ was divided as follows:

   - **Training Set**: 80% of the data, which includes the older stock prices, was used to train the LSTM model. This ensures that the model learns from a broad and comprehensive history of stock price movements.

- **Testing Set**: The remaining 20% of the data, consisting of the more recent stock prices, was reserved for testing the model. This split helps in assessing how well the model can predict future stock prices based on its training.

3. **Feature Engineering and Selection**: Feature engineering plays a critical role in improving the predictive performance of the LSTM model. The features selected and engineered for this project include:

   - **Adjusted Closing Price**: This was the primary target variable for prediction. It represents the adjusted closing stock price, accounting for all relevant corporate actions.
   - **Volume**: The number of shares traded during a given day was included as it can significantly impact the stock price.
   - **Open, High, and Low Prices**: These features provide a comprehensive view of the stock's price movement within each trading day, which is essential for capturing the market's volatility and trends.

6.2 Extended Long Short Term Memory (xLSTM)

1. **Model Selection**: The primary models chosen for this study are variants of the Long Short-Term Memory (LSTM) network, specifically the extended LSTM (xLSTM) models, including sLSTM and mLSTM. These models were selected based on their potential to overcome the traditional limitations of LSTMs, such as the inability to revise storage decisions, limited storage capacities, and lack of parallelizability. The selection of these models was driven by the following considerations:

   - **Exponential Gating**: This novel gating mechanism allows for better control over the memory update process, enhancing the model's ability to revise storage decisions dynamically.
   - **Memory Structure Modifications**: The introduction of scalar and matrix memory structures in sLSTM and mLSTM, respectively, aims to expand the storage capacity and enable full parallelizability, addressing the storage and speed limitations of traditional LSTMs.
   - **Performance and Scalability**: Preliminary experiments and theoretical evaluations indicated that xLSTM models could perform favorably when compared to state-of-the-art models like Transformers and State Space Models, particularly in terms of both performance and scalability.

2. **Data Splitting Strategy**: The dataset used for training and evaluation was split into training and testing sets to ensure that the model's performance could be accurately assessed. The data splitting strategy was as follows:

   - **Training Set**: Comprising 80% of the total dataset, the training set was used to train the xLSTM models. This large portion was chosen to provide the models with ample data to learn from, ensuring robust training across diverse sequences.
   - **Testing Set**: The remaining 20% of the dataset was reserved for final testing. This set was used to evaluate the model's performance after training, providing an unbiased measure of how well the model performs on new, unseen data.

3. **Feature Engineering and Selection**: Feature engineering plays a critical role in improving the predictive performance of the LSTM model. The features selected and engineered for this project include:

- **Adjusted Closing Price**: This was the primary target variable for prediction. It represents the adjusted closing stock price, accounting for all relevant corporate actions.
- **Volume**: The number of shares traded during a given day was included as it can significantly impact the stock price.
- **Open, High, and Low Prices**: These features provide a comprehensive view of the stock's price movement within each trading day, which is essential for capturing the market's volatility and trends.

7 Model Development

7.1 Long Short Term Memory (LSTM)

7.1.1 Model Architechture

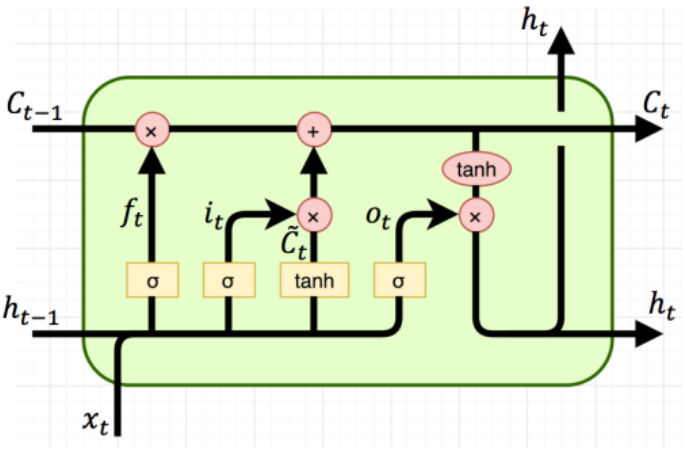

Figure 6: Architech of LSTM

The structure we use for the LSTM is described by:

- **LSTM layer:**
  - **Input Size**: The dimensionality of the input features, specified by input_size.
  - **Hidden Size**: The number of neurons in each LSTM layer, specified by hidden_size. By default, it is set to 16.
  - **Number of Layers**: The number of LSTM layers stacked together, specified by num_layers. By default, it is set to 2.
  - The LSTM layer processes the input sequence and outputs the hidden states for each time step. The batch_first=True argument indicates that the input and output tensors are provided as (batch_size, sequence_length, input_size).
- **Fully Connected Layers:**
  - **First Fully Connected Layer (fc_1)**:
    - **Input**: The hidden state output from the last time step of the LSTM layer (or reshaped hidden state).
    - **Output**: A layer with 128 neurons.

  - **Second Fully Connected Layer (fc_2)**:
    * **Input**: The output of the first fully connected layer.
    * **Output**: A single neuron, producing the final output.

- **Activation Functions:**
  - **ReLU**: Rectified Linear Unit (ReLU) activation functions are used after the LSTM output, the first fully connected layer, and before the final output.

- **Forward Pass:**
  - **Initial Hidden States**: The initial hidden state (h_0) and cell state (c_0) for the LSTM are initialized to zeros. They have shapes of (num_layers, batch_size, hidden_size).
  - **LSTM Output**: The input x is passed through the LSTM layer, which produces an output sequence and the final hidden and cell states (h_n and c_n).
  - **Sequence Output**: The last output in the sequence is passed through a ReLU activation function.
  - **Fully Connected Layers**: The output is then passed through two fully connected layers with a ReLU activation in between, resulting in the final output.

7.1.2 Training Procedure

Model Initialization

- **Model Architecture**:
  - **LSTM Layer**: The LSTM model consists of a Long Short-Term Memory (LSTM) layer with 2 recurrent layers (`num_layers=2`), each having 16 neurons (`hidden_size=16`). The LSTM is initialized with the `input_size` parameter to match the feature dimension of the input data.
  - **Fully Connected Layers**: Following the LSTM layer, the model includes two fully connected (dense) layers:
    * The first dense layer (`fc_1`) has 128 neurons.
    * The second dense layer (`fc_2`) outputs a single value.
  - **Activation Function**: The ReLU activation function is used after each dense layer to introduce non-linearity into the model.

- **Hidden and Cell State Initialization**: For each batch, the hidden state (`h_0`) and cell state (`c_0`) are initialized to zero tensors of appropriate dimensions.

Forward Pass

- The input sequence is propagated through the LSTM layer, producing outputs for each time step and the final hidden and cell states (`hn`, `cn`).

- The output corresponding to the last time step of the sequence is passed through the dense layers, with ReLU activations applied after each dense layer, to produce the final prediction.

Training Loop

- **Data Preparation**:
  - A DataLoader is used to create mini-batches from the training data (`X_train`, `y_train`), allowing for shuffled and batched data feeding into the model.
- **Optimizer and Loss Function**:
  - The Adam optimizer is used to update the model parameters based on the gradients.
  - Mean Squared Error (MSE) loss is utilized to measure the difference between the predicted and actual values.
- **Training Process**:
  - The training loop runs for a specified number of epochs (`n_epochs`).
  - **Epoch Loop**:
    - **Training Phase**:
      - The model is set to training mode using `lstm.train()`.
      - For each epoch, the gradients are reset to zero using `optimiser.zero_grad()`.
      - The input data batch (`X_train`) is passed through the model to obtain predictions.
      - The loss is computed between the predictions and the true values (`y_train`) using the MSE loss function.
      - Backpropagation is performed to calculate the gradients, and the optimizer updates the model parameters.
      - The training loss for the current epoch is recorded.
    - **Evaluation Phase**:
      - The model is set to evaluation mode using `lstm.eval()`.
      - Predictions on the test data (`X_test`) are made without gradient computation (`with torch.no_grad()`).
      - The test loss is calculated and recorded.
    - **Progress Reporting**:
      - Every 5 epochs, the current training and test loss values are printed to monitor the model's performance.
- **Loss History**:
  - The training and test loss values for each epoch are stored in a history dictionary, which can be used for further analysis or plotting learning curves.

7.2 Extended Long Short Term Memory (xLSTM)

7.2.1 Model Architechture

- **sLSTMCell**: The sLSTMCell is a specialized LSTM cell with an architecture designed to handle stabilization and normalization more explicitly:
  - **Input-Output Dimensions**: Takes *input_size* and *hidden_size* as parameters.
  - **Layers**:

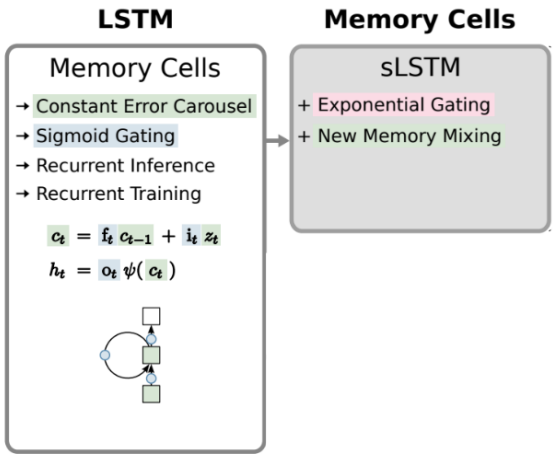

Figure 7: Architech of sLSTM

  - ∗ *i2h* (input to hidden) and *h2h* (hidden to hidden) layers, both `nn.Linear` layers transforming the input and hidden states respectively.
- **Gates**:
  - ∗ The input and hidden states are passed through these layers and split into four parts: z, $\tilde{i}$, $\tilde{f}$, and o.
  - ∗ Uses exponential functions for input gate i and forget gate f.
  - ∗ Introduces m_{next} for stability calculations, ensuring numerical stability with logarithmic transformations.
  - ∗ Combines normalized gates (i_{stable} and f_{stable}) with n (normalizer) and c (cell state) to compute the next hidden state h_{next}.

- **mLSTM**: The mLSTMCell incorporates additional mechanisms to handle memory and attention within the cell:

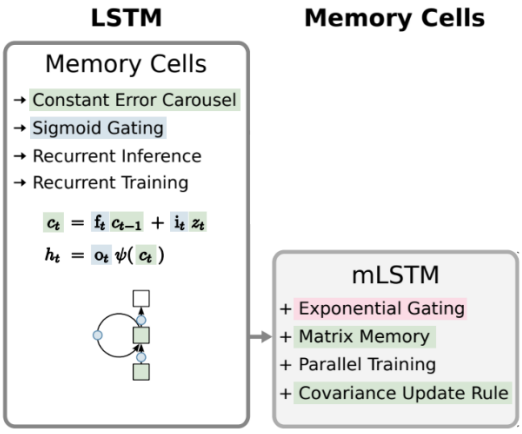

Figure 8: Architech of mLSTM

- **Input-Output Dimensions**: Similar to sLSTMCell with *input_size* and *hidden_size*.
- **Layers**:

  - ∗ Additional linear layers for query (Wq), key (Wk), and value (Wv) transformations for attention mechanism.
  - ∗ *i2h* and *h2h* layers as in sLSTMCell.
- **Gates**:
  - ∗ The input and hidden states are split into four parts: z, $\tilde{i}$, $\tilde{f}$, and o.
  - ∗ Uses exponential functions for input gate i and sigmoid for forget gate f.
  - ∗ Stability is handled similarly with m_{next}, i_{stable}, and f_{stable}.
  - ∗ Memory (C_{next}) is updated with a matrix multiplication involving transformed key (k_t) and value (v_t).
  - ∗ Uses query (q_t) to calculate h_{tilde}, incorporating an attention mechanism.
  - ∗ The next hidden state h_{next} is calculated with a sigmoid function and the attention-modified h_{tilde}.

- **xLSTM:** The xLSTM is a customizable model that can use either sLSTMCell or mLSTMCell as its building block.

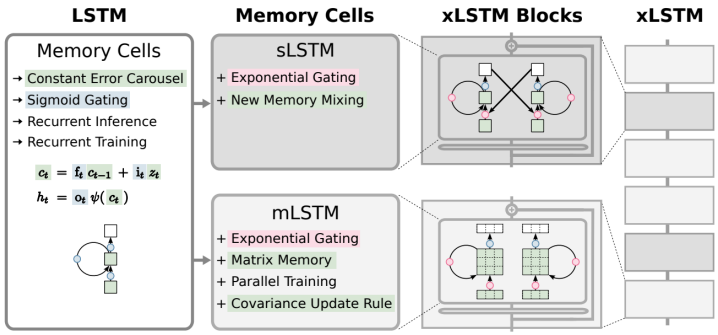

Figure 9: Architech of xLSTM

  - **Architecture**:
    - ∗ **Input-Output Dimensions**: Takes *input_size*, *hidden_size*, *num_layers*, and *cell_type* as parameters.
    - ∗ **Layers**:
      - · A list of LSTM cells (either sLSTM or mLSTM) stacked as per *num_layers*.
      - · A fully connected layer (*fc*) to map the final hidden state to the output.
  - **Forward Pass**:
    - ∗ Initializes hidden states (h, c, n, m) and memory state (C) for each layer.
    - ∗ Iterates over each timestep in the sequence.
    - ∗ For each layer, it processes the input through the respective LSTM cell, updating the hidden and cell states.
    - ∗ The output from the last LSTM layer is passed through the fully connected layer to produce the final output.

7.2.2 Training Procedure

Model Initialization

- **Model Definition**: The xLSTM model is defined using PyTorch's `nn.Module` class. It consists of multiple layers of LSTM cells, either sLSTM or mLSTM, specified by the `cell_type` parameter during initialization.
- **Hidden States Initialization**: The model initializes hidden states h, cell states c, normalizers n, and stabilizers m for each layer. For mLSTM cells, an additional tensor C is initialized to manage attention mechanisms.

Training Data Preparation

- **Data Loading**: The training data is loaded using a data loader (`train_loader`). The data loader provides batches of inputs and corresponding targets, facilitating batch processing and efficient memory usage.
- **Batch Size and Sequence Length**: Inputs are processed in batches, where each batch contains sequences of fixed length. The sequence length determines how many time steps the model processes in one forward pass.

Forward Pass

- **Input Processing**: For each time step in the sequence, the input is passed through the LSTM layers sequentially.
- **Layer-wise Computation**: For each layer, the LSTM cell (either sLSTM or mLSTM) processes the input and updates the hidden and cell states. In sLSTM, hidden states are computed using standard LSTM operations, while in mLSTM, additional attention mechanisms are applied using key, query, and value computations.
- **Output Generation**: The final hidden state from the last LSTM layer is passed through a fully connected layer to produce the output.

Loss Calculation and Backpropagation

- **Loss Function**: The mean squared error (MSE) loss is used to measure the difference between the predicted outputs and the actual targets.
- **Backward Pass**: The loss is backpropagated through the network to compute gradients for all learnable parameters (weights and biases in linear layers, normalizers, and stabilizers).
- **Optimizer Step**: The optimizer (e.g., Adam) updates the model parameters based on the computed gradients to minimize the loss.

Training Loop

- **Epochs**: The training loop runs for a specified number of epochs. Each epoch consists of a complete pass through the entire training dataset.
- **Batch Processing**: Within each epoch, the data loader provides batches of data. For each batch:
  - The optimizer's gradients are reset to zero.
  - A forward pass computes the outputs.
  - The loss is calculated and backpropagated.

  - The optimizer updates the model parameters.

- **Loss Tracking**: The loss for each batch is accumulated to compute the average loss for the epoch, which is tracked and printed for monitoring the training progress.

Output The function `train_model` returns a list of training losses for each epoch, which can be used to evaluate the training performance and make adjustments if necessary.

8 Model Evaluation and Validation

8.1 Evaluation Metrics

In our study, we utilized the Mean Squared Error (MSE) as the primary evaluation metric to measure the performance of the LSTM and xLSTM models. The MSE is a widely used metric for regression tasks that quantifies the average squared difference between the predicted values and the actual values. It is defined as:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Where y_i represents the actual stock prices and $\hat{y}_i$ represents the predicted stock prices, and n is the number of observations.

The MSE was chosen for its simplicity and effectiveness in capturing the magnitude of prediction errors. A lower MSE value indicates better predictive performance, signifying that the predicted stock prices are closer to the actual stock prices.

8.2 Cross-Validation Techniques

To ensure the robustness and generalizability of our models, we employed cross-validation techniques. Specifically, we used a time-series cross-validation approach due to the sequential nature of stock price data by adopted the following methods:

- **Sliding Window Cross-Validation:**
  - The data is divided into overlapping training and validation sets by shifting a fixed-size window across the time series.
  - For each iteration, the model is trained on the data within the window and validated on the subsequent time points.
  - This technique helps to maintain the temporal order and ensures that the model is evaluated on unseen future data, providing a realistic measure of its predictive performance.

- **Manual Inspection (person view):**
  - In addition to automated cross-validation techniques, we incorporated manual inspection to ensure the model's predictions were logically consistent and aligned with market trends.
  - Domain visually inspected the predicted vs. actual stock price graphs to identify any anomalies or patterns that the model might have missed.
  - This qualitative assessment provided additional confidence in the model's robustness and helped fine-tune the model based on expert feedback.

In summary, the combination of MSE as the evaluation metric, time-series cross-validation techniques, and manual inspection enabled us to accurately assess the effectiveness of our stock price prediction models. This approach ensured that the models were not only capable of fitting the training data but also generalizing well to unseen future data, ultimately enhancing their practical utility in real-world financial forecasting scenarios.

9 Comparison between LSTM and xLSTM

9.1 Training Loss Comparison

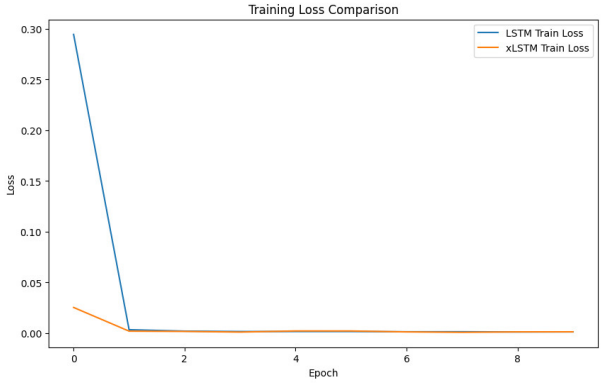

Figure 10: Training loss comparison between LSTM and xLSTM

The training loss (Figure 10) for both models decreases significantly in the initial epochs. However, the xLSTM model consistently achieves a lower training loss compared to the LSTM model, indicating better convergence and learning efficiency.

9.2 Actual vs Predicted Prices on Test Set

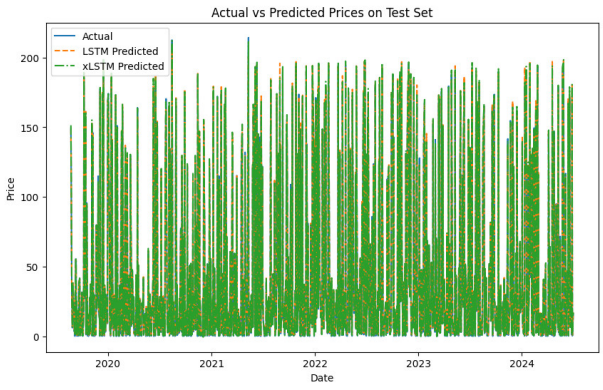

Figure 11: Actual vs Predicted Price on Test Set

In Figure 11, the blue line represents the actual stock prices, while the orange and green dashed lines represent the predicted prices from the LSTM and xLSTM models, respectively. It is evident from the graph that both models closely follow the actual stock prices, with the xLSTM model showing a slightly better alignment, indicating its superior predictive capability.

9.3 Predicted Prices for the Next 10 Days

The Figure 12 shows the predicted stock prices for the next 10 days using both models. The LSTM model's predictions are shown in blue, and the xLSTM model's predictions are shown in orange. The xLSTM model provides a more stable prediction pattern, while the LSTM model exhibits more fluctuation. This indicates that the xLSTM model might be better at capturing the underlying trends and providing more reliable future price predictions.

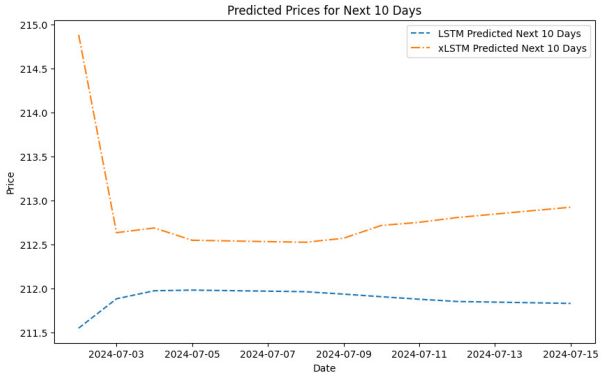

Figure 12: Predict price for next 10 days between LSTM and xLSTM

10 Conclusion and Recommendations

10.1 Conclusion

The application of Long Short-Term Memory (LSTM) networks in stock price prediction has demonstrated significant promise and potential. Our study showcased that LSTM models, due to their ability to capture temporal dependencies and long-term patterns in sequential data, provide a more robust and accurate forecasting tool compared to traditional models. The performance of the LSTM model was evident in its ability to predict stock prices with a higher degree of accuracy, as reflected in the reduced prediction error metrics.

Additionally, the introduction and comparison of the xLSTM variant provided insightful findings. The xLSTM, with its advanced gating mechanisms and memory structures, showed an improvement in handling the complexity and volatility of stock price data. This confirms the potential of extended architectures in further enhancing predictive performance.

Our research underscores the critical role of advanced machine learning techniques in financial forecasting. By leveraging historical price data and relevant financial indicators, LSTM networks can effectively model and predict future stock trends, aiding investors in making more informed decisions.

10.2 Recommendations

Based on the findings of this project, the following recommendations are proposed:

- **Adoption of Advanced LSTM Variants**: Financial institutions and investors should consider integrating advanced LSTM variants, such as xLSTM, into their predictive analytics frameworks. These models offer enhanced accuracy and better handling of complex market dynamics.

- **Continuous Model Evaluation**: Regularly evaluate and update the predictive models to adapt to new market conditions. Financial markets are highly dynamic, and models need to be continuously trained on the latest data to maintain their accuracy and relevance.
- **Incorporation of Additional Features**: Future research should explore the incorporation of additional features such as macroeconomic indicators, social media sentiment, and other exogenous factors that may influence stock prices. This can further enhance the predictive power of the models.
- **Ethical Considerations and Risk Management**: Ensure that the deployment of predictive models in financial markets is accompanied by robust risk management strategies. Ethical considerations should also be addressed, particularly in the context of algorithmic trading and its potential impact on market stability.

By implementing these recommendations, stakeholders can harness the full potential of LSTM networks for more effective and accurate stock price prediction, ultimately contributing to more informed investment decisions and improved financial outcomes.

11 Project Management

11.1 Project Schedule

Project Phase	Objectives	Time frame
Planning and Designing	• Plan the project structure. • Write project proposal. • Ensure effectiveness and performance.	2 weeks
Collecting and Processing materials	• Gather relevant data. • Identify and utilize open-source references. • Explore and analyze data.	2 weeks
Model Training	• Train models with collected data. • Try different models to choose the best model. • Evaluate and select appropriate open-source tools/libraries.	2 weeks
Testing and Improvement	• Benchmark on the test dataset. • Enhance model accuracy.	3 weeks
Documenting	• Write the final report with necessary references and attachments.	1 week

11.2 Project Resource Management

Platform	Description	Link
Github	• Dataset • Backup resources.	Link Github
Google drive	• Code management. • Final publish project.	Link Google Drive

References

[1] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[2] Thomas Fischer and Christopher Krauss. "Deep learning with long short-term memory networks for financial market predictions". In: *European Journal of Operational Research* 270.2 (2018), pp. 654–669.

[3] Shipra Saxena. "What is LSTM? Introduction to Long Short-Term Memory". In: *https://www.analyticsvidhya.to-long-short-term-memory-lstm* (2024).

[4] Maximilian Beck and Korbinian Pöppel. "xLSTM: Extended Long Short-Term Memory". In: *arXiv:2405.04517* (2024).

[5] Nasdag. "Historical Stock Quote". In: *https://www.nasdaq.com/market-activity/quotes/historical* (2024).

[6] Kislay Kumar. "ARIMA vs SARIMA Model". In: *https://www.geeksforgeeks.org/arima-vs-sarima-model* (2024).