DECISION 546Q.201D

Team 29

Jojo Dong; Ayush Kumar; Yolanda Li; Colette Ma; Syed Irtiza Mehdi

Predicting Stock Price

Motivation

Our motivation behind building an effective model to address a highly critical challenge within the financial market for decades, predicting stock prices, was the substantial impact it can have in making investments decisions and managing risks, given its real-world applications.

This project develops a robust model utilizing the Long Short-Term Memory (LSTM) neural network to predict stock prices using historical data. This model will address the problem of requiring a tool that leverages sequential patterns in time-series in order to enhance the prediction accuracy. Accordingly, predicting stock prices efficiently has very high practical significance in fields such as: Portfolio Management, algorithmic trading, as well as financial risk assessments that drive major decisions, including stock purchases and mergers and acquisitions.

Data Understanding

The dataset consists of historical stock price data, including features such as open, high, low, close prices, and trading volume. These data points are sourced from publicly available stock market data repositories or APIs. The dataset is used to model time-series patterns and trends, enabling the prediction of future stock prices. Preprocessing steps ensure the data is normalized, cleaned of missing values, and structured into sequential input-output pairs, making it suitable for deep learning applications like LSTMs.

Data Preparation

The data preparation involves several steps to transform raw data into a format suitable for deep learning. These steps ensure the dataset is clean, standardized, and preprocessed for sequential modeling.

- Data Fetching and Storage：Using the data_fetcher.py script, historical stock data is fetched for a list of stock symbols from the Yahoo Finance API and saved as individual CSV files, one for each stock, in a specified directory. This process ensures the availability of comprehensive historical data for each stock symbol while filtering out symbols with insufficient or missing data to prevent errors in downstream tasks. The script uses the yfinance library to retrieve the data and automatically skips stocks with insufficient data, specifically those with a file size of less than 2 KB, ensuring only viable datasets are retained for further analysis.
- Data standardization: This process involves removing invalid rows and standardizing column names to a uniform format (date, adj close, close, high, low, open, volume). Additionally, the index for each dataset is reset to prepare the data for sequential modeling. This standardization eliminates discrepancies in column names and addresses missing values, ensuring data integrity. Furthermore, it prepares the data for downstream tasks, such as splitting into training and testing datasets, facilitating smooth and efficient model training.
- Data Preprocessing: The **data_preprocessor.py** script processes stock data into sequences suitable for deep learning, including normalization, sequence generation (e.g., 30-day time steps), and data splitting into training and testing sets. The **StockEnhanceDataSet**class handles preprocessing for individual stock symbols,

managing missing data and ensuring viable datasets. The preprocessed data, formatted as NumPy arrays, integrates features like historical prices for input sequences and generates labels for prediction tasks, ensuring compatibility with LSTM models for efficient training and evaluation.The output is formatted as NumPy arrays, ensuring compatibility with LSTM models or other neural networks for efficient training and evaluation.

Modeling

We chose two main model architectures: LSTM (Long Short-Term Memory) and Bidirectional LSTM, both of which are able to capture long-term dependencies in time series data. These architectures are well-suited for time-series forecasting as they effectively capture temporal dependencies and patterns in sequential data.

- The input layer accepts sequences of time-series data, each sample consists of num_steps time steps and input_size features. For standard LSTM, the final layer outputs the last hidden state, summarizing the sequence's information, while Bidirectional LSTM processes the data in both forward and backward directions, enabling it to capture dependencies from both past and future contexts.
- To prevent overfitting, Dropout regularization is applied after each LSTM layer, with a dropout rate of 0.2. The model ends with a fully connected Dense output layer, which predicts the stock price for the next time step, making it suitable for regression tasks. Mean Squared Error (MSE) is used as the loss function, while the Adam optimizer ensures efficient training with its adaptive learning rate. For Bidirectional LSTM, a Flatten layer is introduced to reshape the output, making it compatible with the subsequent dense layer for prediction.

- To further improve the model's performance, hyperparameters such as batch_size and max_epoch were optimized using a grid search, and cross-validation with 5 folds was employed to enhance robustness. The best-performing configuration was found with a batch size of 32 and 10 epochs. In each cross-validation, the model is trained on a different training/validation split. The loss curve is recorded during training, and the validation set loss is monitored to evaluate the training progress.Analyze forecast errors using histograms and time series plots, and visually compare trends in predicted and true values.At the same time, comparison with the simple flat prediction baseline model (Naive Baseline) proves that the deep learning model has significant prediction advantages.

Model Architectures:

- **LSTM:**

LSTM models are highly effective at capturing long-term dependencies in sequential data, such as stock prices, and are particularly advantageous as they mitigate the vanishing gradient problem inherent in traditional RNNs. However, they come with certain drawbacks, including being computationally expensive compared to simpler models like ARIMA or basic RNNs. Additionally, they require careful hyperparameter tuning to strike a balance between performance and overfitting.

- **Bidirectional LSTM:**

Bidirectional LSTM models excel at capturing patterns in both forward and backward directions, making them particularly useful for context-dependent sequences where both past and future information are equally important. However, this enhanced capability comes at the cost of higher computational and memory requirements compared to unidirectional LSTMs.

Hyperparameter Choices:

Batch sizes of 32, 64, and 128 were tested during grid search to achieve a trade-off between learning stability and training speed. The number of epochs ranged from 10 to 40, aiming to identify an optimal training duration that avoids overfitting. Additionally, the model architecture was experimented with up to 2 layers and 128 LSTM units, striking a balance between capturing complex patterns and maintaining generalization.

Alternative approaches include simpler models like ARIMA or Prophet, which are well-suited for time-series forecasting but may struggle with capturing non-linear dependencies. On the other hand, transformer-based architectures such as Temporal Fusion Transformers can offer superior performance but come with significantly higher computational complexity.

Implementation

The models were built from scratch using TensorFlow/Keras to ensure flexibility and customization, avoiding reliance on high-level APIs. A class-based implementation (LstmRNNEh) was developed to encapsulate key functionalities such as model building, training, and evaluation. The architecture utilized Sequential models with stacked LSTM or BiLSTM layers, combined with dropout for regularization and dense output layers for prediction. A robust training pipeline was implemented with 5-fold cross-validation to evaluate model performance comprehensively. Additionally, mechanisms for saving and loading trained models were integrated, enabling efficient reuse and deployment of the best-performing models.

Results and Evaluation

The Bidirectional LSTM model trained and tested on the stock price dataset achieved promising performance metrics, indicating its ability to capture the sequential dependencies inherent in stock price data, the key evaluation metrics for the stock symbol A is:

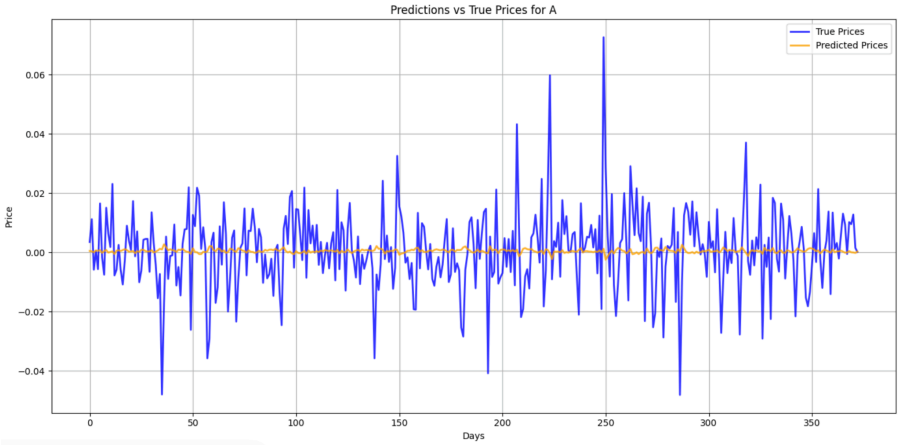
Mean Squared Error (MSE): 0.0002

Root Mean Squared Error (RMSE): 0.0136

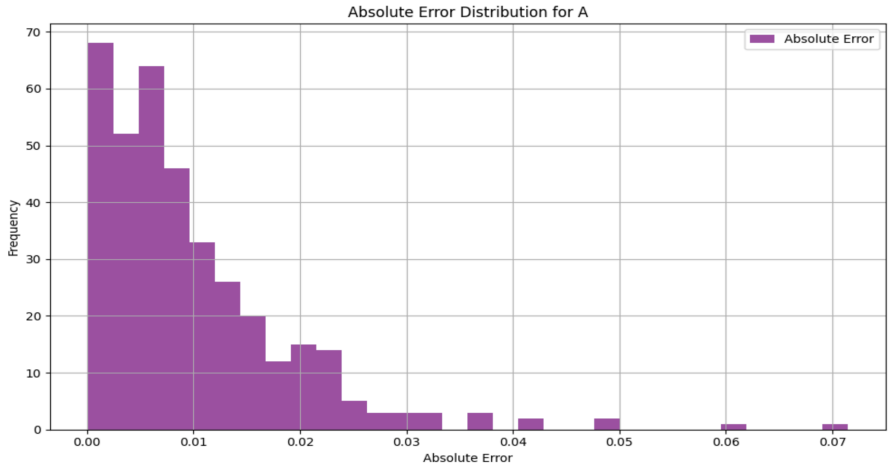
Mean Absolute Error (MAE): 0.0100

Mean Absolute Percentage Error (MAPE): Infinite (caused by zero values in ground truth)

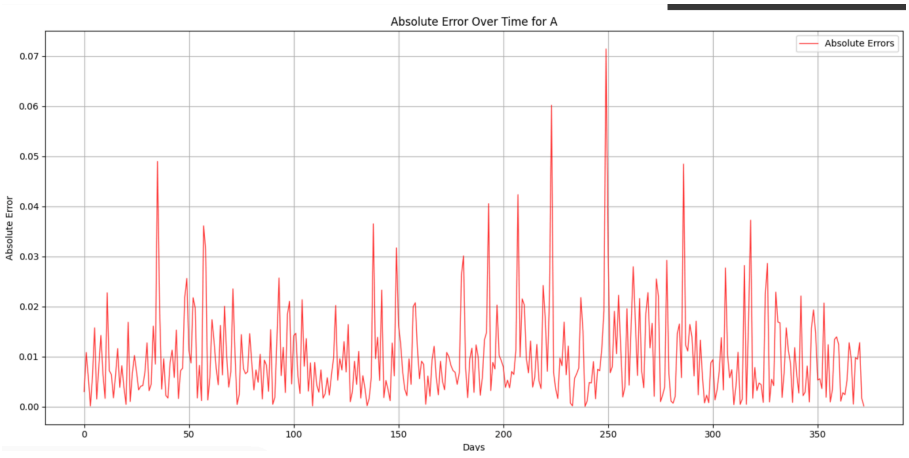
Mean Absolute Scaled Error (MASE): 0.7183



The plot *Predictions vs. True Prices* shows the alignment between the predicted and true stock prices for the test set. Despite some deviations during high volatility periods, the model demonstrates a strong ability to predict overall trends.



The histogram of absolute errors illustrates the majority of errors are concentrated near zero, reflecting high accuracy. Only a small portion of predictions exhibit higher error values.



The temporal distribution of absolute errors highlights periods where the model's predictions are less accurate. These deviations are likely associated with sudden market movements or other anomalies in the data.

The Bidirectional LSTM model effectively handles complex sequential data like stock prices by capturing both forward and backward temporal patterns, which significantly improves predictive accuracy. Additionally, it provides a scalable framework for modeling more complex,

multi-variable datasets. However, these advantages come at the cost of higher computational demands and greater sensitivity to hyperparameter tuning, making the model more resource-intensive compared to simpler alternatives like ARIMA or Prophet.

This predictive framework can assist stock traders with short-term price forecasts and enhance portfolio management by anticipating market trends. Future improvements could include integrating more market indicators and exploring transformer-based architectures for enhanced performance.

Deployment

The results from this LSTM-based stock price prediction model can be deployed using a cloud-based application, or integration into a web platform, with membership-access. This setup will allow the stakeholders to input data regarding the stock, assess the predictions made by the model in real-time, and make better-informed investment decisions. The model can also be deployed with existing financial dashboards or be offered as an API to brokers and portfolio managers to integrate into their operations. The deployment must be included with automated updates of the model using new data, including the economic and market conditions, to ensure increased relevance and accuracy.

There are certain challenges/issues regarding deployment that we need to be mindful of:

- **Overfitting:** Considering that the model is trained on historical data, it may perform on training data but fail to generalize given the dynamic market conditions. Regular retraining with updated data is highly crucial so that the model is able to integrate market and other such factors while making its predictions.

- **Data Security and Privacy Concerns:** Data relating to stock and predictions is highly sensitive. It is imperative that we implement robust cybersecurity measures to prevent any unauthorised access and other such data breaches.
- **Scalability:** The model must be able to handle large volumes of data and user queries efficiently, especially in the periods where the trading volume is high.
- **Interpretability:** Users, who might want to better understand certain predictions, will demand transparency regarding the model's output. We can enhance explainability of the model by integrating additional modules, such as SHAP (SHapley Additive exPlanations) or LIME (Local Interpretable Model-agnostic Explanations), allowing insights into which features significantly influence the predictions. This additional transparency will also build trust into the model and allow for users to make better decisions.

There are certain ethical considerations in application to this model, such as Market Manipulation, where the predictions can be misused by certain organizations in manipulating the stock markets and influencing the prices. This risk can be mitigated by deploying strict access control and such user agreements. The model can also inadvertently favor certain stocks or industries, producing biases in over/under-estimating the prices for these stocks. Diverse and unbiased dataset must be used to avoid such unfair advantages in the price predictions. The dependence on automation, relying on the model may also undermine human judgement in investment decisions. The users must be made aware of the model's limitations, and best practices on how they can utilize the model to enhance their decisions.

Risks are inherent when we are utilizing models to make predictions:

- **Inaccurate Predictions:** The predictions may fail to reflect certain unprecedented market conditions in its predictions. Incorporating such external market signals, such as news sentiments, economic conditions can improve the robustness of the model.
- **Limited Dataset:** Due to the limited dataset, the model may not be able to capture long-term trends. It will be beneficial to expand data to include factors such as macroeconomic conditions, and longer time horizons.
- **Operational Risks:** Deployment can be disrupted due to technical failures or system downtime, which might significantly impact the daily operations of the firm. Regular maintenance, redundant systems and failover mechanisms are highly essential to be added into the model.
- **Compliance Risks:** The deployment must fully comply with financial regulations, and should be updated accordingly as well, to avoid any legal consequences. Partnering with compliance experts will ensure that such relevant laws are adhered to.

It is essential to be proactive in addressing these risks, and ensuring that the model deployment is effective, reliable and ethically sound in making its stock price predictions.

Appendix

Yolanda Li - modeling, data understanding, evaluation, project proposal

Ayush Kumar - status report, presentation deck

Syed Irtiza Mehdi - motivation, deployment

Colette Ma - data preparation, status report, presentation deck

Jojo Dong - presentation