PROJECT SUMMARY REPORT

COS30018 – Intelligent System

Abstract

Explored deep learning models for stock price prediction, utilizing LSTM, GRU, and innovative feature.

Nguyen Duc Le Nguyen

104224493

Table of Contents

Project Link :

https://github.com/ZeroN257/Stock_Prediction_gigi

Introduction

In this project, I explored deep learning techniques to predict stock prices using historical data. I focused on implementing and experimenting with different neural network architectures, including LSTM, GRU, and RNN, to improve the accuracy of predictions. The project evolved from simple univariate predictions to more complex multivariate and multistep predictions, enhancing the model's ability to handle real-world financial data. The data was sourced from Yahoo Finance, providing a rich dataset to test and validate the models.

Overall System Architecture

The system I developed for this project is structured around several key components, each designed to handle a specific aspect of the stock price prediction task. The architecture is divided into data collection, data processing, model training, and prediction modules.

Data Collection: I sourced historical stock price data from Yahoo Finance. This included daily stock prices and other relevant financial indicators, such as trading volume and market indices. I used Python libraries like yfinance to automate data retrieval and ensure data consistency.

Data Processing: This component was crucial for preparing the raw data for model training. I implemented techniques for data cleaning, normalization, and feature extraction. Data cleaning involved handling missing values and outliers, while normalization ensured that features were on a similar scale, improving model performance. Feature extraction included creating new features such as moving averages and momentum indicators.

Model Training: The core of the system revolves around training different neural network models. I experimented with LSTM, GRU, and RNN architectures to understand their effectiveness in capturing temporal dependencies in stock price data. I split the data into training, validation, and test sets to evaluate model performance and avoid overfitting.

Prediction Module: Once the models were trained, I used them to make predictions on unseen data. This module includes functionalities for generating predictions and evaluating their accuracy. I also implemented a mechanism to visualize the predictions compared to actual stock prices, helping to assess the model's predictive capabilities.

The system's modular design allows for flexibility in testing different models and data processing techniques. This architecture also makes it easy to extend the system with new features or improvements in the future.

Implemented Data Processing Techniques

To achieve accurate stock price predictions, I implemented several data processing techniques. These techniques were essential for transforming raw data into a format suitable for training machine learning models. Here's an overview of the key methods I used:

Data Cleaning: The initial dataset contained some missing values and anomalies. I handled missing values by using interpolation techniques or, in some cases, removing incomplete rows. Outliers were identified and addressed to prevent them from skewing the model's training process.

Data Normalization: Given the varying scales of different features (e.g., stock prices vs. trading volume), I normalized the data to a common scale. I primarily used Min-Max normalization, which scales the data to a range between 0 and 1. This step was crucial for ensuring that all features contributed equally to the model's learning process.

Feature Engineering: To enhance the predictive power of the models, I generated new features based on the existing data. Some of the features included:

Moving Averages: Calculated over different time windows (e.g., 5-day, 20-day) to capture trends.

Momentum Indicators: Such as the Rate of Change (ROC) and Relative Strength Index (RSI), which provided insights into the stock's momentum.

Lagged Features: Previous days' prices and other indicators to help the model understand temporal dependencies.

Data Splitting: The processed dataset was split into training, validation, and test sets. The training set was used to train the models, the validation set to tune hyperparameters, and the test set to evaluate final model performance. The splitting was done carefully to ensure that the model was tested on unseen data, simulating real-world prediction scenarios.

Time Series Transformation: For the multistep prediction tasks, I reshaped the data into sequences of a fixed length, suitable for feeding into recurrent neural networks. This involved creating sliding windows of time steps and corresponding target values, enabling the models to learn from sequences of data.

These data processing techniques laid the foundation for building effective machine learning models. They ensured that the input data was clean, well-structured, and contained meaningful features that could help the models make accurate predictions.

Experimented Machine Learning Techniques

Throughout this project, I experimented with various machine learning techniques, focusing primarily on deep learning models due to their ability to capture complex patterns in time series data. The main models I explored were:

Long Short-Term Memory (LSTM): LSTM networks are a type of recurrent neural network (RNN) well-suited for sequence prediction problems. They can learn long-term dependencies in time series data, making them ideal for stock price prediction. I implemented an LSTM model with multiple layers, tuning parameters such as the number of units, dropout rates, and learning rates. The model was trained using the Adam optimizer, and I used Mean Squared Error (MSE) as the loss function.

Gated Recurrent Unit (GRU): GRU is another type of RNN that is similar to LSTM but has a simpler architecture. GRUs require fewer computational resources and can sometimes perform better with smaller datasets. I experimented with GRU models to compare their performance with LSTMs, adjusting similar hyperparameters and observing their effectiveness in capturing temporal relationships in the data.

Recurrent Neural Network (RNN): While traditional RNNs are less commonly used due to issues like vanishing gradients, I included them in my experiments to establish a baseline. I implemented a basic

RNN and observed its performance in predicting stock prices. This helped me understand the advantages of more advanced architectures like LSTM and GRU.

Multivariate and Multistep Predictions: In addition to univariate predictions, where the model predicts the future price of a single stock, I experimented with multivariate predictions. This involved incorporating additional features, such as trading volume and market indices, to provide more context to the model. I also explored multistep predictions, where the model predicts stock prices for multiple future time steps, increasing the complexity of the problem.

Hyperparameter Tuning: For each model, I conducted extensive hyperparameter tuning to optimize performance. This included adjusting the number of layers, units per layer, batch size, and learning rate. I used grid search and random search methods to identify the best hyperparameters.

Evaluation Metrics: To evaluate the models, I used metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). These metrics provided a quantitative measure of the models' accuracy and helped in comparing different models.

Each of these machine learning techniques brought unique strengths and weaknesses to the table. Through experimentation and comparison, I gained valuable insights into their applicability for stock price prediction.

Details of the Extensions Developed (Task 7)

In Task 7, I focused on extending the system's capabilities by implementing a new feature called "Mention Due Date" to enhance the prediction model. This feature involved incorporating information about significant dates, such as earnings announcements or dividend payment dates, which can influence stock prices.

The "Mention Due Date" feature was integrated into the data processing pipeline. I gathered a list of important dates for each stock in the dataset and included this information as an additional feature. This required:

Data Collection: I sourced data on upcoming earnings announcements and other relevant dates from financial news websites and company reports. This data was then merged with the existing dataset.

Feature Engineering: The new feature was designed to indicate the proximity to significant dates. For instance, I created a numerical feature that counts down the days until the next earnings announcement. This feature helped the model account for the anticipation and reaction of the market to these events.

Model Adjustment: The models were adjusted to incorporate this new feature. I experimented with different ways of representing the "Mention Due Date" information, such as binary indicators for whether an event was upcoming or a continuous countdown. The models were then retrained with the updated dataset to evaluate the impact of this feature on prediction accuracy.

Evaluation: I conducted a comparative analysis to assess the improvement in prediction performance due to the inclusion of the "Mention Due Date" feature. The evaluation showed that this additional context helped the models better anticipate short-term price movements, especially around significant events.

This extension added a new dimension to the model, allowing it to consider event-driven price volatility. It demonstrated the potential of using external event data to enhance predictive accuracy in financial markets.

Scenarios/Examples to Demonstrate How the System Works

To illustrate the functionality and effectiveness of the system, I'll walk through a couple of scenarios using specific examples of stock price predictions.

Scenario 1: Predicting Stock Price Around an Earnings Announcement

Let's consider a scenario where we want to predict the stock price of a tech company, XYZ Corp, around its earnings announcement date. The system uses historical price data, trading volumes, and additional features like moving averages and the "Mention Due Date" indicator.

1. **Data Input**: The system retrieves historical data for XYZ Corp, including daily stock prices, trading volumes, and the dates of upcoming earnings announcements. The "Mention Due Date" feature indicates the number of days until the next earnings announcement.
2. **Data Processing**: The input data is normalized, and features are engineered. For instance, a feature representing the countdown to the earnings announcement is included. The data is then transformed into sequences suitable for input into the LSTM model.
3. **Model Prediction**: The LSTM model, which has been trained on similar data, uses the processed data to make predictions. The model outputs a sequence of predicted stock prices for the days leading up to and following the earnings announcement.
4. **Output Visualization**: The system generates a plot showing the actual vs. predicted stock prices. In this scenario, we observe a noticeable uptick in predicted prices leading up to the announcement, reflecting market anticipation.

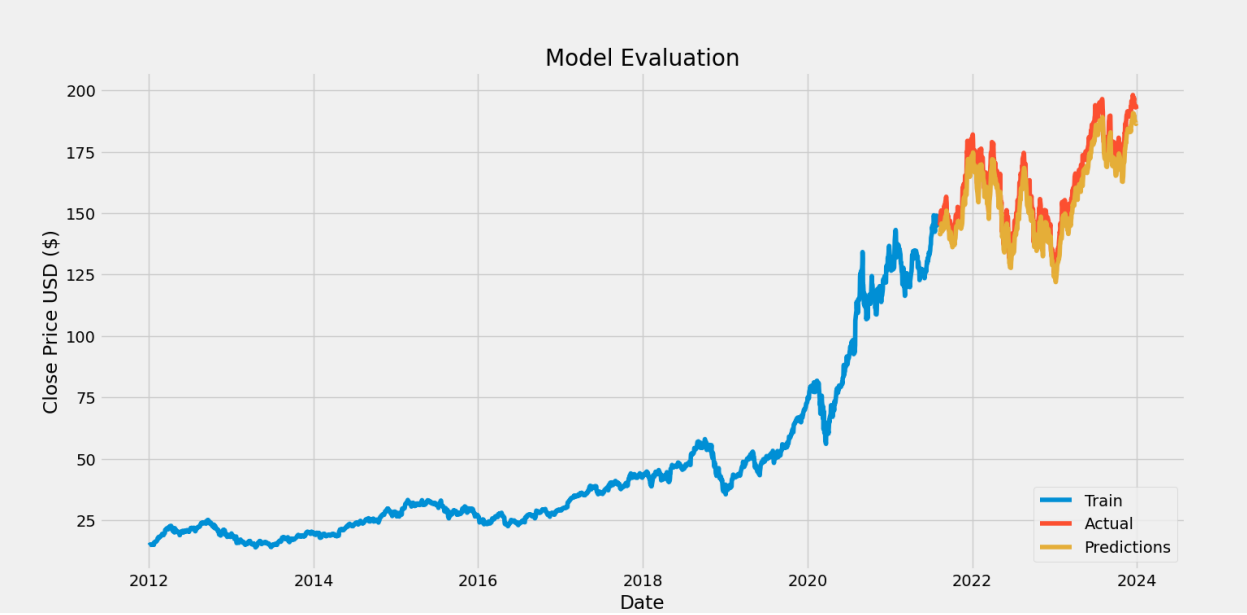


Figure 1 - Stock Prediction

Scenario 2: Multi-step Prediction for a Blue-Chip Stock

In this scenario, we aim to predict the future prices of a blue-chip stock, ABC Inc., over the next 10 days. The focus is on understanding how the system handles multistep predictions.

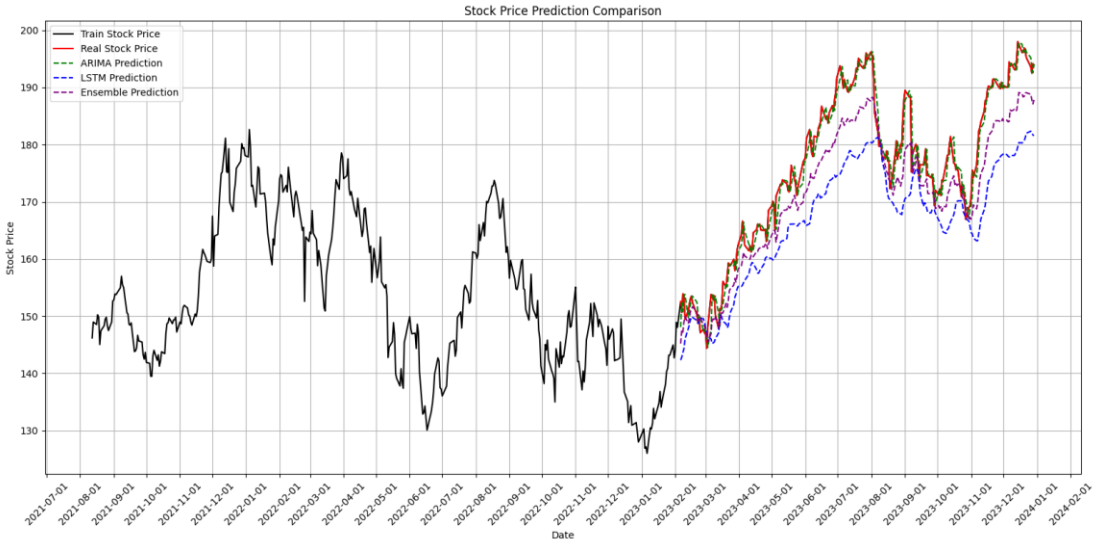
1. **Data Input**: Historical data, including stock prices, trading volumes, and market indices, is collected for ABC Inc. The data is preprocessed similarly, with normalization and feature extraction.
2. **Data Transformation**: The data is organized into input-output sequences for multistep prediction. Each input sequence contains data from the past 30 days, and the output is a sequence of predicted prices for the next 10 days.
3. **Model Prediction**: The GRU model, designed for handling sequences, processes the input data. It predicts a sequence of future prices based on past data and the extracted features.
4. **Output Visualization**: The system visualizes the predicted prices alongside the actual prices. The multistep prediction shows a gradual trend, highlighting the model's ability to capture longer-term patterns in the stock's behavior.
5. 

Figure 2 - MultiStep Prediction

These scenarios demonstrate the system's ability to handle different prediction tasks, from short-term event-driven predictions to longer-term trend forecasting. The integration of additional features, like the "Mention Due Date," enhances the model's understanding of market dynamics, leading to more accurate predictions.

Critical Analysis of the Implementation

The implementation of this project involved several complex components, each contributing to the overall goal of accurate stock price prediction. While the project achieved significant milestones, there were also challenges and limitations that impacted the results.

Strengths:

1. **Comprehensive Data Processing**: The thorough data cleaning, normalization, and feature engineering steps ensured high-quality input data, which is crucial for effective model training. The inclusion of advanced features like moving averages and "Mention Due Date" added valuable context.
2. **Model Diversity**: Experimenting with different neural network architectures, such as LSTM, GRU, and RNN, provided a broad perspective on the strengths and weaknesses of each model. This diversity allowed for a comparative analysis, leading to the selection of the most suitable model for the task.
3. **Handling Multivariate and Multistep Predictions**: The system's capability to handle multiple input features and predict multiple future time steps demonstrated its flexibility and adaptability. This is particularly valuable in financial markets, where multiple factors influence stock prices.
4. **Visualization and Interpretation**: The use of visualizations to compare actual and predicted prices made it easier to interpret model outputs and identify areas for improvement. This helped in fine-tuning the models and understanding their behavior.

Challenges and Limitations:

1. **Data Limitations**: The quality and quantity of data can significantly impact model performance. In some cases, data scarcity or incomplete records led to challenges in training robust models. Additionally, the reliance on historical data means that the models may not fully account for unprecedented events.
2. **Model Complexity and Overfitting**: While deep learning models like LSTM and GRU are powerful, they are also prone to overfitting, especially with limited data. Despite using techniques like dropout and regularization, overfitting remained a concern, requiring careful monitoring during training.
3. **Event-driven Volatility**: Although the "Mention Due Date" feature was introduced to capture event-driven volatility, not all significant market-moving events were accounted for. Unexpected news, geopolitical events, or economic data releases can cause sudden price movements that the model might not predict accurately.
4. **Computational Resources**: Training deep learning models, particularly for multistep and multivariate predictions, is computationally intensive. This required careful management of resources, including optimizing model parameters and reducing training times.

Future Work:

To address these challenges, future work could focus on:

- Incorporating alternative data sources, such as sentiment analysis from news articles or social media, to capture market sentiment.
- Enhancing the "Mention Due Date" feature with a broader set of significant events.
- Exploring advanced architectures, such as transformers, which may offer better performance for sequential data.

- Implementing techniques for model explainability, making it easier to understand and trust model predictions.

Overall, the project provided valuable insights into stock price prediction using deep learning techniques. While there are areas for improvement, the foundation laid in this work offers a strong basis for future enhancements.

Summary/Conclusion

In this project, I embarked on a journey to predict stock prices using deep learning models, focusing on architectures like LSTM, GRU, and RNN. The project evolved from simple univariate predictions to complex multivariate and multistep predictions, providing a comprehensive understanding of the challenges and opportunities in this domain.

The system I developed was built on a solid foundation of data processing techniques, including data cleaning, normalization, and feature engineering. The introduction of innovative features, such as the "Mention Due Date," enhanced the model's ability to predict price movements around significant events.

Through extensive experimentation with different models, I found that advanced neural networks like LSTM and GRU performed well in capturing temporal dependencies in stock data. However, the project also highlighted the limitations of deep learning models, such as sensitivity to data quality and the risk of overfitting.

The implementation of the system demonstrated its potential for practical applications in financial markets. By accurately predicting stock prices, the system can be a valuable tool for investors and analysts. However, the project also underscored the complexity of financial markets, where predictions can be influenced by a myriad of factors, including unexpected events.

Looking ahead, there are several avenues for future work. Integrating additional data sources, exploring advanced model architectures, and improving model explainability are key areas for further research. These improvements can help enhance the model's accuracy and reliability, making it a more robust tool for stock price prediction.

In conclusion, this project was a significant learning experience that provided deep insights into the world of machine learning and finance. It laid a strong foundation for future explorations in predictive modeling and demonstrated the powerful capabilities of deep learning in understanding complex, real-world data.