Silesian University of Technology

Faculty of Automatic Control, Electronics and Computer Science

Biologically inspired artificial intelligence

Analysis of stock market trends

Authors	Beberok Patryk Dworaczyk Mateusz

Academic year	2022/2023
Field	Computer Science
Type	SSI
Term	6
Project date	Mondays 13:15-16.15
Teacher	dr inż. Grzegorz Baron

Spis treści

Short introduction presenting the project topic

The stock market has long been a fascinating and dynamic arena, attracting investors and traders from all over the world. The rise and fall of stock prices can have a big impact on economies, industries, and individual investors. Therefore, understanding and predicting stock market trends has become a critical target for financial professionals and researchers.

This project aims to dive into the analysis of stock market trends, employing a combination of statistical techniques, data analysis and machine learning algorithms. By examining historical stock market data, identifying patterns and leveraging different models, we seek to uncover some important values that can help investors make informed decisions and potentially optimize their portfolio performance.

The analysis of stock market trends involves the study of various aspects, including price movements, trading volumes, market volatility, and the influence of external factors such as economic indicators, news events, and company-specific developments. By exploring these dimensions, we aim to gain a comprehensive understanding of the factors driving stock market behaviour and identify potential opportunities for profit.

To accomplish our goals, we will take advantage of the huge repository of historical stock market data, encompassing a wide range of companies, industries, and markets. By examining this rich dataset, we can uncover historical trends, correlations, and anomalies that can serve as valuable indicators for future market behaviour.

Additionally, we will harness the power of machine learning algorithms to develop predictive models capable of forecasting stock market trends. These models will be trained on historical data and will aim to capture complex patterns and relationships within the stock market. By leveraging these models, we hope to provide investors with actionable insights and forecasts that can guide their investment decisions.

It is essential to note that while this project seeks to analyse stock market trends and provide valuable insights, it does not guarantee accurate predictions or financial success. The stock market is inherently unpredictable, influenced by a myriad of factors, and subject to unforeseen events. However, by employing rigorous analysis techniques and leveraging advanced algorithms, we aim to increase our understanding of market dynamics and provide a foundation for more informed decision-making.

In summary, the analysis of stock market trends is a complex and exciting project, aiming to unravel the intricacies of market behaviour and provide investors with valuable insights. By combining historical data analysis, statistical techniques, and machine learning algorithms, we hope to contribute to the growing body of knowledge surrounding stock market dynamics and empower investors to make more informed investment decisions.

Analysis of the task

Possible approaches to solve the problem

Technical Analysis: Analysing historical price and volume data to identify patterns and indicators for predicting future price movements.

Fundamental Analysis: Assessing financial statements, economic indicators, and company-specific factors to evaluate the intrinsic value and long-term potential of a stock.

Quantitative Analysis: Applying statistical models and machine learning algorithms to historical stock market data for developing predictive models and making data-driven investment decisions.

Sentiment Analysis: Analysing news articles and social media data to gauge market sentiment and investor emotions, providing insights into potential market movements.

Event-Driven Analysis: Monitoring corporate earnings, economic reports, policy changes, and geopolitical events to understand their impact on stock prices.

Data Visualization and Pattern Recognition: Using charts, graphs, and visualizations to identify patterns, trends, and anomalies in stock market data for informed decision-making.

Combining these approaches can enhance the understanding of stock market trends, considering market conditions and individual preferences. Continual refinement and adaptation of methodologies based on analysis results are crucial for effective outcomes.

Pros/cons

Technical Analysis

Pros:

- Focuses on patterns and indicators derived from historical price and volume data.
- Can be used for short-term trading decisions.
- Provides visual representation of trends through charts and graphs.

Cons:

- Relies heavily on historical data, which may not accurately predict future market movements.
- Subjective interpretation of patterns may lead to different conclusions.
- Does not consider fundamental factors or external events that can impact stock prices.

Fundamental Analysis

Pros:

- Assesses the intrinsic value and long-term potential of a stock.
- Considers financial statements, economic indicators, and company-specific factors.
- Provides insights into the underlying factors driving a company's performance.

Cons:

- Requires in-depth analysis and understanding of financial statements.
- Can be time-consuming and complex.
- Market reactions may not always align with fundamental analysis predictions.

Quantitative Analysis

Pros:

- Leverages statistical models and machine learning algorithms for data-driven insights.
- Considers multiple factors simultaneously, including price movements, trading volumes, and external variables.

- Can handle large datasets and capture complex relationships.

Cons:

- Models are only as good as the data they are trained on, and historical data may not perfectly represent future market conditions.
- Overfitting or underfitting models can lead to inaccurate predictions.
- Interpretation of model outputs may still require human judgment.

Sentiment Analysis

Pros:

- Provides insights into market sentiment and investor emotions.
- Utilizes news articles and social media data to gauge public opinion.
- Can be used to anticipate market reactions to significant events or news.

Cons:

- Sentiment analysis accuracy heavily relies on the quality and relevance of the data sources.
- Identifying and categorizing sentiment accurately can be challenging.
- Market sentiment does not always align with actual market movements.

Event-Driven Analysis

Pros:

- Considers specific events or news that can impact stock prices.
- Provides the ability to react quickly to market-changing events.
- Can uncover opportunities for short-term trading or long-term investments.

Cons:

- Requires constant monitoring of news and events.
- Market reactions to events can be unpredictable or short-lived.
- Some events may have delayed or indirect impacts on stock prices.

Data Visualization and Pattern Recognition

Pros:

- Enables easy comprehension of complex data through visual representation.
- Identifies patterns, trends, and anomalies in stock market data.
- Facilitates decision-making by presenting information in a clear and concise manner.

Cons:

- Visualization can be subjective and dependent on the chosen visual representation.
- Identifying patterns does not guarantee future accuracy.
- May overlook underlying factors or relationships not visually apparent.

Detailed description of selected methodology

Long Short-Term Memory (LSTM) networks have proven to be effective in analysing sequential data, making them a popular choice for time series analysis, including stock market trends. In this methodology, we will use Python programming language, TensorFlow library, and Jupyter Notebook to implement an LSTM network for stock market trend analysis.

Data Preparation

- Obtain historical stock market data, including price, volume, and any relevant variables.
- Pre-process the data by normalizing or scaling it to a common range to ensure compatibility with the LSTM model.
- Split the data into training and testing sets, ensuring that sequential ordering is maintained.

LSTM Model Architecture

- Import the required libraries, including TensorFlow, Keras (a high-level neural network API), and relevant data processing libraries.
- Set up the LSTM model architecture using the Keras Sequential API.
- Configure the LSTM layers, including the number of LSTM units, activation functions, and input shape.
- Add additional layers like Dense layers for model refinement if desired.
- Compile the model by specifying the loss function, optimizer, and evaluation metrics.

Training the LSTM Model

- Fit the model to the training data using the model.fit() function.
- Specify the batch size and number of epochs for training.
- Monitor the training process to ensure convergence and adjust hyperparameters if needed.

Evaluation and Validation

- Evaluate the trained LSTM model's performance on the testing data.
- Calculate relevant metrics such as accuracy, mean squared error, or any other appropriate metric for the specific problem.
- Validate the model's performance by visualizing the predicted stock market trends against the actual trends.

Fine-tuning and Optimization

- If necessary, fine-tune the model by adjusting hyperparameters such as the number of LSTM layers, units, or the learning rate.
- Experiment with different variations of the model architecture or pre-processing techniques to improve performance.
- Employ techniques such as regularization or dropout to mitigate overfitting.

Deployment and Prediction

- Once the model is trained and validated, deploy it to make future predictions.
- Prepare new input data in the same format as the training data and feed it into the deployed LSTM model.
- Obtain predictions for future stock market trends and analyse the results.

Throughout the implementation process, Jupyter Notebook provides an interactive environment that allows for step-by-step execution and visualizations. It facilitates data exploration, model development, and result analysis in a collaborative and flexible manner.

By leveraging LSTM networks, TensorFlow, and Jupyter Notebook, this methodology offers a robust framework for analysing stock market trends. It combines the power of deep learning algorithms, sequential data analysis, and an interactive development environment to enable insightful predictions and informed decision-making in the dynamic world of stock markets.

Possible/available datasets

Yahoo Finance provides historical data for various stocks and indices. You can access historical stock prices, dividends, and splits. Alpha Vantage offers free access to historical stock market data, including daily, weekly, and monthly prices, adjusted close prices, and trading volume. Quandl provides a vast collection of financial, economic, and alternative data. They offer historical stock prices, corporate fundamentals, futures data, and much more. Intrinio offers a wide range of financial data, including historical stock prices, dividends, splits, and corporate actions. They provide both free and premium datasets. Tiingo offers historical stock market data, including prices, dividends, and splits. They also provide intraday data and access to real-time data via their API. Also on kaagle.com we can find some interesting datasets like:

"Historical Stock Prices" by Kaggle user "borismarjanovic": This dataset provides historical stock prices for a large number of companies, including opening, closing, high, and low prices, as well as trading volumes.

"S&P 500 stock data" by Kaggle user "Yan Sun": This dataset contains historical stock data for companies in the S&P 500 index, including stock prices, trading volumes, and other financial indicators.

"NYSE Stock Market" by Kaggle user "Samarth Agrawal": This dataset provides historical stock data for companies listed on the New York Stock Exchange (NYSE). It includes information on stock prices, volumes, and other financial details.

Detailed description of the chosen ones

After many discussions we choose the dataset from yahoo finance. The Yahoo Finance dataset for Apple (AAPL) and Microsoft (MSFT) provides historical stock market data for these two companies. Here is a description of the dataset:

Dataset Source: Yahoo Finance is a widely used financial website that offers historical price data, financial news, and other financial information.

Data Coverage: The dataset covers the historical stock market data for Apple (AAPL) and Microsoft (MSFT). It includes various attributes such as date, open price, high price, low price, closing price, adjusted closing price, and trading volume.

Historical Time Range: The dataset typically includes daily data spanning several years, allowing for analysis and trend identification over an extended period.

Stock Price Attributes:

- Date: The specific date for each data point.
- Open Price: The price at which the stock opens for trading at the beginning of the day.
- High Price: The highest price the stock reaches during the trading day.
- Low Price: The lowest price the stock reaches during the trading day.
- Closing Price: The price of the stock at the end of the trading day.
- Adjusted Closing Price: The closing price adjusted for factors such as dividends, stock splits, or other corporate actions.
- Trading Volume: The total number of shares traded during the day.

Dataset Format: The dataset is typically available in a tabular format, such as a CSV file, where each row represents a specific date, and each column corresponds to a specific attribute of the stock.

Usage: The dataset is commonly used for analysing the historical performance of Apple and Microsoft stocks, identifying trends, conducting technical analysis, developing trading strategies, and conducting research on the stock market.

Accessing the Yahoo Finance dataset for Apple (AAPL) and Microsoft (MSFT) can provide valuable insights into the historical stock market behaviour of these companies, facilitating quantitative analysis, trend identification, and the development of trading or investment strategies.

Analysis of available tools/frameworks/libraries suitable for task solution

Python

Python is a versatile programming language with extensive data analysis and machine learning functions. It offers several libraries and programs suitable for analysing Yahoo financial data, such as Panda for data transformation, NumPy for statistical functions, scikit-learn for machine learning algorithms, Matplotlib and Seaborn for data visualization, and TensorFlow or Keras for learning deep LSTM and other models.

TensorFlow

TensorFlow is an open-source library widely used for machine learning and deep learning tasks. It provides a flexible framework for building and training neural networks, making it suitable for implementing LSTM models to analyse stock market trends using the Yahoo Finance dataset.

Keras

Keras is a high-level neural network API that runs on top of TensorFlow. It offers a user-friendly interface for building and training deep learning models, including LSTM networks. Keras simplifies the process of constructing the LSTM architecture and provides convenient functions for data pre-processing and model evaluation.

Jupyter Notebook

Jupyter Notebook is an interactive development environment that allows for data exploration, model development, and result analysis in a collaborative and flexible manner. It supports Python and provides a web-based interface for writing and executing code, visualizing data, and documenting the analysis process. Jupyter Notebook is well-suited for working with the Yahoo Finance dataset, allowing for step-by-step execution and the inclusion of visualizations and explanations.

Pandas

Pandas is a powerful library for data manipulation and analysis in Python. It provides data structures like DataFrames that facilitate handling and processing tabular data. With Pandas, you can efficiently load and pre-process the Yahoo Finance dataset, perform time series analysis, handle missing data, and create meaningful features for LSTM model input.

Matplotlib and Seaborn

Matplotlib and Seaborn are Python libraries used for data visualization. They offer a wide range of plotting functionalities to create informative visualizations of the Yahoo Finance dataset. These libraries enable the generation of line plots, bar charts, scatter plots, and other types of visual representations that help in understanding and presenting stock market trends.

Scikit-learn

Scikit-learn is a popular machine learning library that provides a comprehensive set of tools for various tasks, including regression, classification, and clustering. It offers a range of algorithms and evaluation metrics that can be useful for analysing the Yahoo Finance dataset. Scikit-learn can be used to develop alternative models, perform feature selection, or assess the performance of the LSTM model.

By leveraging these tools, frameworks, and libraries, analysts and researchers can effectively work with the Yahoo Finance dataset. Python provides a robust ecosystem for data analysis and machine learning, and TensorFlow, Keras, Jupyter Notebook, Pandas, Matplotlib, Seaborn, and Scikit-learn are key components that facilitate data pre-processing, model development, visualization, and evaluation.

Internal and external specification of the software solution

Project solution

We did our project in Jupyter Notebook, because of several main advantages:

1. Interactive Environment – it provides an interactive environment that allows users to run code in real-time, making it ideal for exploratory analysis and iterative development. It facilitates a seamless combination of code, visualizations, and explanatory text within a single document.

2. Clear code intensions– it allows for the integration of code, markdown, and rich media elements (such as images, videos, and LaTeX equations). This enables clear and concise explanations of the project, making it more readable and accessible to collaborators and stakeholders.

3. Step-by-Step Execution – it enables executing code cells individually, allowing for easy debugging and troubleshooting. This feature is particularly useful when working with complex algorithms or when investigating specific sections of the project.

4. Data Visualization – it supports the integration of various data visualization libraries, such as Matplotlib, Seaborn, and Plotly. These libraries provide powerful tools for creating interactive and visually appealing charts, plots, and graphs, enhancing the project's ability to communicate insights effectively.

5. Collaboration and Sharing – it facilitates collaboration by allowing multiple users to work on the same notebook simultaneously. This promotes teamwork and knowledge sharing among project members. Additionally, notebooks can be easily shared in various formats, including HTML, PDF, and Markdown, making it convenient to distribute project findings and reports.

6. Reproducibility – it promotes reproducibility by enabling users to document and share their code, data, and results in a single document. This ensures that others can replicate the project's findings and verify the analysis, enhancing transparency and credibility.

7. Extensive Library Ecosystem – it leverages the vast Python ecosystem, providing access to a wide range of libraries for data manipulation, analysis, and machine learning. This allows for efficient integration of pre-existing tools and algorithms into the project, accelerating development and enhancing functionality.

8. Support for Different Kernels – it supports multiple programming languages through its kernel system. This means that besides Python, users can also work with kernels for R, Julia, and other languages. This flexibility allows project developers to leverage the strengths of different languages and libraries for specific tasks.

9. Education and Documentation – it is widely used in educational settings due to its interactive nature and ability to present concepts visually. It is often employed for teaching data science, machine learning, and programming due to its ability to combine code execution, explanations, and visualizations in one platform.

10. Integration with Version Control – it integrates well with version control systems like Git, allowing users to track changes made to the notebook over time. This enables collaboration, facilitates project management, and provides a historical record of the project's evolution.

The most interesting code fragments are presented on figures 1-4.

```
dataset = pd.read_csv(
    filepath_or_buffer='https://raw.githubusercontent.com/Patryk0990/biai-stock-market/main/dataset/historical/AAPL.csv',
    index_col='Date')

close_price_values = dataset.filter(['Close']).values
scaler = MinMaxScaler(feature_range=(0, 1))
train_data = scaler.fit_transform(close_price_values)

x_train = []
y_train = []

for index in range(10, len(train_data)):
    x_train.append(train_data[index-10:index, 0])
    y_train.append(train_data[index, 0])

# Convert trainset to array and reshape x_train data
x_train, y_train = np.array(x_train), np.array(y_train)
x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))
```

Figure 1 Training dataset preparation

```
gpus = tf.config.experimental.list_physical_devices('GPU')
if gpus:
    try:
        for gpu in gpus:
            tf.config.experimental.set_memory_growth(gpu, True)
    except RuntimeError as e:
        print(e)

# Default LSTM model
lstm = Sequential(
    layers=[
        LSTM(512, return_sequences=True, input_shape=(x_train.shape[1], 1)),
        LSTM(256),
        Dense(1)
    ]
)
lstm.compile(optimizer='adam', loss='mean_squared_error')
lstm.fit(x_train, y_train, batch_size=256, epochs=100)
```

Figure 2 Creating and teaching our first model

```
test_dataset = pd.read_csv(
    filepath_or_buffer='https://raw.githubusercontent.com/Patryk0990/biai-stock-market/main/dataset/three_years/MSFT.csv',
    index_col='Date')
test_dates_str = [dt.datetime.strptime(d, '%Y-%m-%d').date() for d in test_dataset.index.values]

test_price = test_dataset.filter(['Close'])
scaler = MinMaxScaler(feature_range=(0, 1))
test_data = scaler.fit_transform(test_price.values)

x_test = []
y_test = test_price.values[10:len(test_data), :]

for index in range(10, len(test_data)):
    x_test.append(test_data[index-10:index, 0])

x_test = np.array(x_test)
x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1))
```

Figure 3 Test dataset preparation

```
predictions = lstm.predict(x_test)
predictions = scaler.inverse_transform(predictions)
rmse=np.sqrt(np.mean((predictions - y_test) ** 2))
print(f'Default model RMSE: {rmse}')
```

Figure 4 Validation of our model efficiency

The rest of our code is available under the link in the 'Link to the GitHub' section.

Data structures

The visualization of our dataset that we used in the project are on the figure 5 and 6.

Date	Open	High	Low	Close	Adj Close	Volume
2020-01-02	74.059998	75.150002	73.797501	75.087502	73.449402	135480400
2020-01-03	74.287498	75.144997	74.125000	74.357498	72.735313	146322800
2020-01-06	73.447502	74.989998	73.187500	74.949997	73.314880	118387200
2020-01-07	74.959999	75.224998	74.370003	74.597504	72.970093	108872000
2020-01-08	74.290001	76.110001	74.290001	75.797501	74.143890	132079200

Figure 5 The sample data from dataset used to train our project

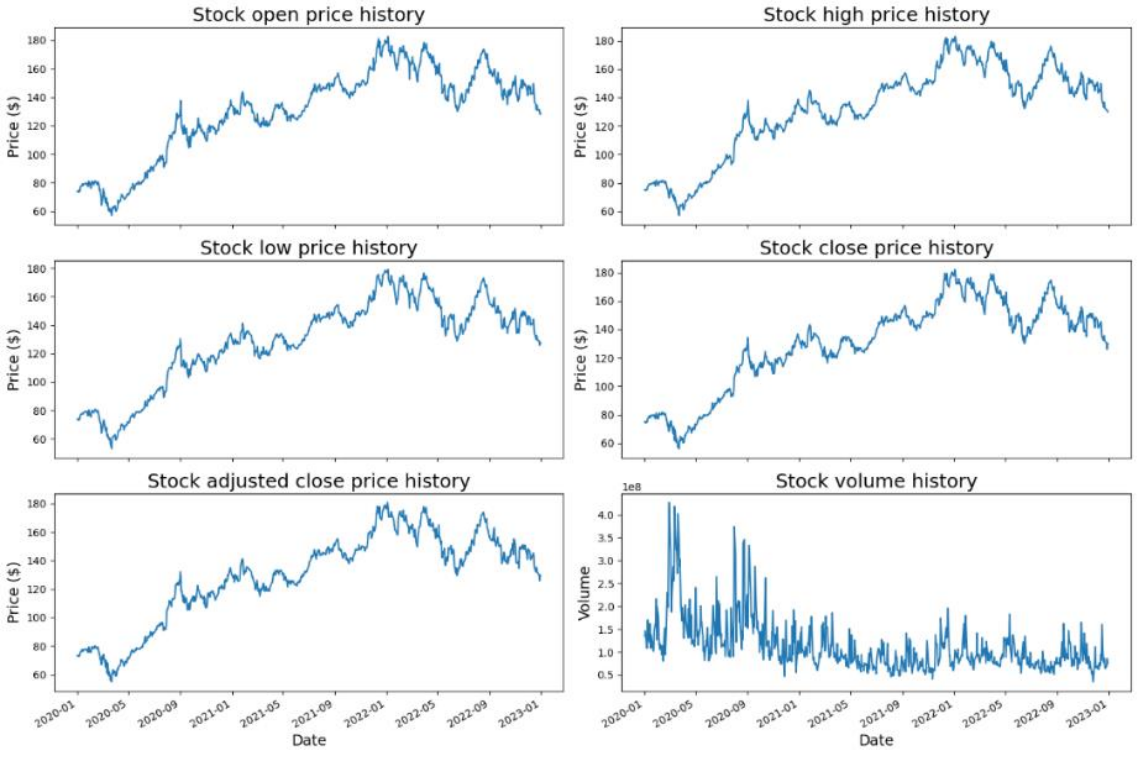

Figure 6 The graphical representation of our dataset

In the further steps of our project we extracted closing prices from the dataset given above, then for the purpose of training our model we made our training dataset. Each sample of the set was a list of ten previous closing prices.

User interface / GUI/console

When using Jupyter Notebook, one of the notable advantages is the ability to avoid repeatedly training models and easily explore data while creating and analysing plots.

It allows for the execution of code cells in a non-sequential manner. This means that once a model has been trained and its parameters have been saved, there is no need to retrain it from scratch every time the notebook is run. This saves time and computational resources, especially when working with large datasets or complex models.

Jupyter facilitates efficient data exploration by providing an interactive environment. With the ability to execute code cells on-demand, analysts can easily inspect and manipulate data in real-time. This allows for quick data inspection, summarization, and filtering, providing valuable insights into the dataset without the need for separate scripts or commands.

Notebook integrates well with data visualization libraries such as Matplotlib, Seaborn, and Plotly. These libraries provide a wide range of functions and methods to create various types of plots and charts. With Jupyter Notebook, creating, customizing, and analysing plots becomes straightforward. The inline plotting feature also enables the immediate display of visualizations within the notebook, making it easier to observe trends, patterns, and corelations in the data.

By combining these features, Jupyter Notebook offers a seamless workflow for data analysis and model development. It allows for an iterative and interactive approach, enabling users to explore, manipulate, and visualize data efficiently while easily reusing pre-trained models. This accelerates the development process and enhances productivity, especially in projects that involve repetitive model training or extensive data exploration.

Experiments

What parameters/conditions were the subject of change and why

During the experiments we changed parameters such as layer types, layer numbers, epoch numbers, and input data format in a neural network model is justified to improve performance. Modifying layer types helps leverage their specific strengths for the task. Adjusting layer numbers balances complexity and generalization. Optimizing epoch numbers prevents underfitting or overfitting. Adapting input data format ensures the model receives appropriate representations. We made many different variations of given above parameters, because of our willingness to understand how the LSTM neural network works.

What results were observed and why

While testing the variations of our model structures and input data, we observed that extending number of layers or number of units inside each one, mostly not ever mean that efficiency of our model was higher than the last observed one. The main reason for observing such results is that extending the complexity of our model causes it to be less sensitive to data fluctuations. In addition to that, we had to balance between the complexity of our model and number of epochs used to train it to avoid the underfitting and overfitting.

Method of results presentation – description of diagrams/axes/values

When presenting the results, we will focus on showcasing the raw values of the Root Mean Square Error (RMSE) metric. RMSE is a widely used measure for evaluating the accuracy of predictions. In addition to that we created the diagram showing the predicted and actual prices in given timestamp.

Raw results, calculations (methodology, equations etc.)

We used the RMSE indicator presented on figure 7 for validating our model accuracy.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}}$$

Figure 7 The equation used to calculate RMSE

On the figures 8 and 9 are presented the graphs with the results.

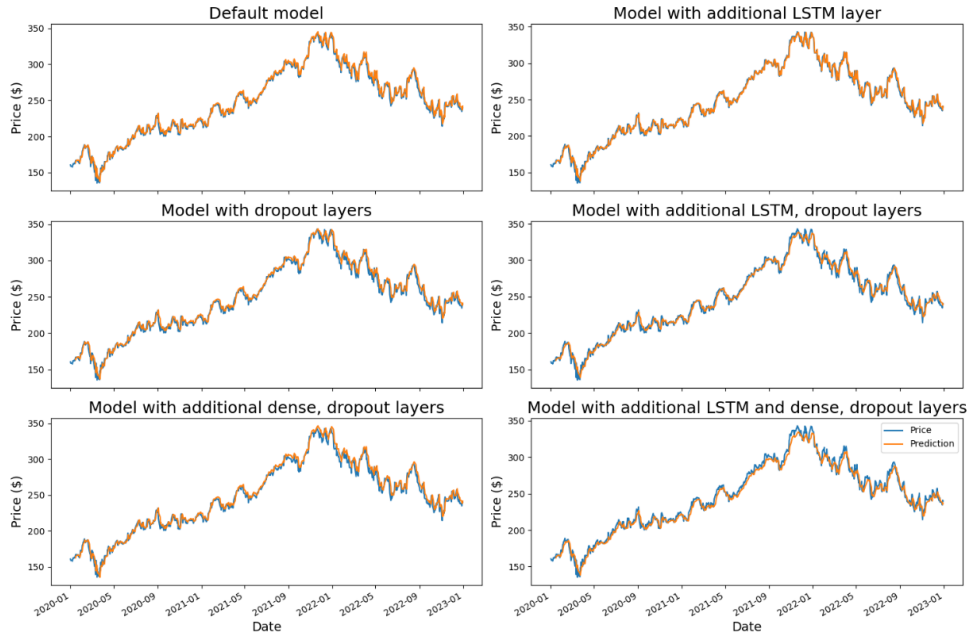

Figure 8 Graphical representation of our results

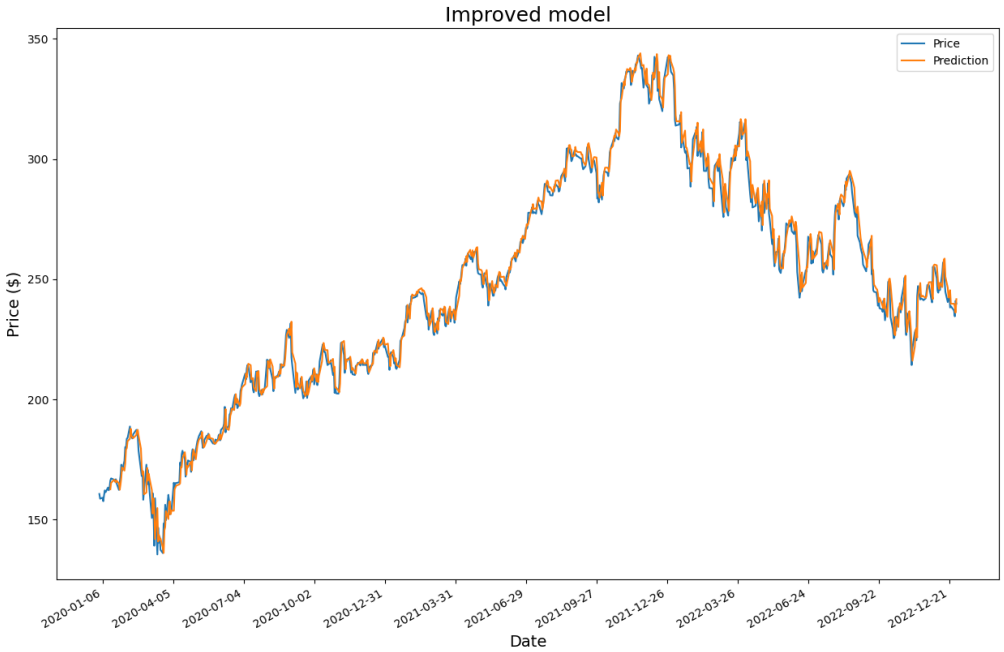

Figure 9 Graphical representation of our best model

Presentation of experiments

Adding an extra LSTM layer has proven to be more accurate in our analysis. This additional layer allows the model to capture more complex patterns and dependencies in the data, resulting in improved prediction accuracy. Introducing a dropout layer adds regularization to the model, making it less sensitive to changes in the input data. This helps prevent overfitting and enhances the model's ability to generalize to unseen data. By randomly dropping out certain connections during training, the model becomes more robust and less prone to memorizing noise or irrelevant patterns.

On the other hand, incorporating a dense layer allows for faster responsiveness to changes in the input data. The dense layer, also known as a fully connected layer, connects every neuron from the previous layer to the current layer. This facilitates efficient information flow and enables the model to quickly adapt to varying input patterns, resulting in faster reactions to changes.

Regarding the increase in the quantity and complexity of input data in our configuration, we observed that it did not positively impact the effectiveness of our model. Despite the larger and more intricate dataset, the model's performance did not improve significantly. This suggests that the additional data did not contain significant additional information or patterns that could enhance the model's predictive capabilities. It is crucial to consider the diminishing returns of adding more data and evaluate if the increased complexity is justified by the potential improvement in performance.

5. Summary

Overall conclusions

In conclusion, this project focused on the analysis of stock market trends using a Python-based LSTM model implemented in Jupyter Notebook. Throughout the project, we made several key observations and drew important conclusions:

1. Jupyter Notebook proved to be a valuable tool for this project, as it allowed us to avoid repetitive model training and provided an interactive environment for data exploration. The ability to easily visualize data and create plots within the notebook greatly facilitated the analysis process.
2. Experimentation with model parameters, such as layer types, layer numbers, and input data format, was essential. However, we found that increasing the complexity of the model, such as adding more layers or units, did not always result in improved efficiency. Careful consideration and striking the right balance between model complexity and generalization were crucial.
3. Additional LSTM layers in the model enhanced accuracy by capturing more complex patterns and dependencies in the data. Dropout layers provided regularization and made the model less sensitive to changes, while dense layers enabled faster reactions to input changes.
4. Increasing the quantity and complexity of input data did not necessarily lead to improved model performance. The diminishing returns of adding more data and the absence of significant additional information or patterns in the extra data highlighted the need for careful evaluation of the trade-off between data complexity and performance improvement.

Overall, this project demonstrated the iterative nature of model development and optimization. It emphasized the importance of fine-tuning parameters, selecting appropriate model architectures, and critically evaluating the impact of data complexity on model efficiency. These insights are valuable for future projects in the field of AI and can inform decision-making to develop more accurate and efficient models for analysing stock market trends.

Possible improvements

Here are some possible improvements for the project:

1. Feature Engineering: Explore additional features or transformations that can capture more meaningful information from the stock market data. Consider technical indicators, sentiment analysis, or external factors that may influence stock prices.
2. Hyperparameter Tuning: Conduct a more extensive search for optimal hyperparameters, including learning rate, batch size, regularization techniques, and optimizer choices. Utilize techniques like grid search or random search to systematically explore the parameter space.
3. Ensemble Methods: Implement ensemble methods, such as model averaging or stacking, to combine predictions from multiple LSTM models. This can help improve the model's robustness and generalization.
4. Model Architecture: Investigate alternative model architectures beyond LSTM, such as hybrid models combining CNN and LSTM, or attention-based models. Explore the potential benefits of using pre-trained embeddings or transfer learning techniques.
5. Regularization Techniques: Experiment with different regularization techniques, such as L1 or L2 regularization, to prevent overfitting and improve the model's ability to generalize.
6. Data Augmentation: Consider applying data augmentation techniques to increase the diversity and size of the training data. This can help the model learn more robust representations and improve performance on unseen data.
7. Time Series Analysis: Explore advanced time series analysis techniques, such as autoregressive models (AR), moving average models (MA), or autoregressive integrated moving average models (ARIMA), to complement the LSTM model and capture specific temporal patterns.
8. Cross-Validation: Implement robust cross-validation strategies, such as k-fold cross-validation, to obtain more reliable performance estimates and assess the model's stability across different subsets of the data.
9. Regular Monitoring and Updating: Regularly monitor the performance of the model and retrain it with new data to adapt to changing market conditions. Stay updated with the latest research and advancements in the field to incorporate any relevant improvements or techniques.
10. Incorporate External Data: Consider incorporating relevant external data sources, such as financial news sentiment, economic indicators, or market trends, to enhance the model's predictive capabilities and capture broader market dynamics.

These improvements can help refine the model, enhance its accuracy and generalization, and enable better analysis of stock market trends. It is important to evaluate each improvement carefully and assess its impact on the specific problem and dataset at hand.

Future work

Here are some concise suggestions for future work on the project:

1. Explore advanced deep learning techniques like transformer models or graph neural networks.
2. Investigate transfer learning to leverage pre-trained models for improved performance.
3. Consider incorporating reinforcement learning for building trading agents and adaptive strategies.
4. Extend the project to real-time analysis by incorporating live data feeds.
5. Combine the predictive model with portfolio optimization techniques for intelligent investment strategies.
6. Enhance interpretability and explainability of the model's predictions.
7. Develop a user-friendly interface for deployment and integration into trading systems.

8. Conduct thorough performance evaluations and comparisons with alternative models.
9. Explore integration of alternative data sources to capture additional signals.
10. Collaborate with domain experts, share findings, and engage with the financial community.

These future work suggestions aim to further enhance the model's capabilities, improve performance, and address emerging challenges in stock market analysis and prediction.

6. References – list of sources used during the work on the project

https://media.geeksforgeeks.org/wp-content/uploads/20200622171741/RMSE1.jpg

https://towardsdatascience.com/what-does-rmse-really-mean-806b65f2e48e

https://www.kaggle.com/general/182134

https://finance.yahoo.com/quote/CSV/history/

https://www.investopedia.com/terms/e/ema.asp

https://www.investopedia.com/ask/answers/difference-between-simple-exponential-moving-average/

https://www.investopedia.com/terms/m/movingaverage.asp

https://towardsdatascience.com/lstm-recurrent-neural-networks-how-to-teach-a-network-to-remember-the-past-55e54c2ff22e

https://towardsdatascience.com/an-introduction-to-long-short-term-memory-networks-lstm-27af36dde85d

https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/

https://keras.io/api/layers/recurrent_layers/lstm/

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

https://analyticsindiamag.com/a-complete-understanding-of-dense-layers-in-neural-networks/

https://towardsdatascience.com/introduction-to-convolutional-neural-network-cnn-de73f69c5b83

https://towardsdatascience.com/dropout-in-neural-networks-47a162d621d9

https://machinelearningmastery.com/a-gentle-introduction-to-sigmoid-function/

https://paperswithcode.com/method/hard-sigmoid

https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

7. Link to the GitHub

https://github.com/Patryk0990/biai-stock-market