Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Project Report

Forecasting Stock Prices with LSTM: Integrating Financial Sentiment & Technical Indicators

submitted by

Jack Jürgens (7563519)
Industrial Engineering

Jannick Drechsler (6844093)
Business Informatics

Simon Papekyan (7524084)
Business Administration

Konstantinos Papanagiotou (7925666)
Business Administration

Submitted on June 30, 2025

Supervisor: Pranav Agrawal

Contents

1 Abstract

This project explores the application of data science techniques, specifically sentiment analysis and deep learning, for forecasting stock prices. We developed a hybrid prediction model that integrates sentiment scores derived from a large-scale financial news dataset with traditional technical indicators, including RSI, MACD, Volume and Bollinger Bands. Historical stock data from high-volatility companies (NVIDIA, Tesla, Inc., Microsoft, and Apple Inc.) were used to train a Long Short-Term Memory (LSTM) neural network capable of learning temporal patterns. The sentiment analysis was based on headlines tagged with relevant stock tickers, enabling targeted sentiment quantification. Our experiments show that augmenting price data with both sentiment metrics and technical indicators improves prediction performance. The LSTM model achieved high accuracy on test data, with R^2 values exceeding 0.85 for two stocks. These findings highlight the potential of combining textual sentiment signals and time-series modeling to enhance predictive insights in financial decision-making.

2 Introduction (Jannick Drechsler)

When we started this project our first goal was to find common ground on our interests to help us better understand what type of project we should decide on. Luckily 3 members of our group have a finance background in their studies while all have a major interest in the global financial markets as well as the topic of data science. Naturally we decided to take the intersection of those fields and focus on the sentiment analysis of text with relevancy for the financial markets. In this essay we will look at the data we chose to run the sentiment analysis on, the financial indicators we chose to train the model on historic data as well as the connected models, methods and our code needed to interconnect everything into a precise prediction tool for future market developments. We will showcase the different experiments we ran to improve our model along the way and the predictors we created from our different data sources.

3 Data (Jannick Drechsler)

Arguably the most important part of our project is the selection and preparation of usable data. In the following section we are going to look at both data sources, firstly our financial data and then the news headlines used in our sentiment analysis.

3.1 Data selection

There are many different text datasets for financial analysis already in existence that supply us with the necessary raw data. The problem we needed to solve first was to compose a complete dataset that we could use for all our training, as many of the datasets available online only contain limited time periods of tweets or newspaper headlines.

Luckly we found a sizable dataset of newspaper headlines on “Huggingface” that ranges from the early 1900s to 2024 from all major news outlets. We used this dataset in two different versions. Our test dataset originally included around 5 million news headlines and was around 5gb in size. The whole 90 Million rows dataset was used for the final analysis and we will see in our experiment section how the size of the data set impacted our final precision scores.

This gives us the option to try different sentiment analysis models, including ones specialized on financial market terminology. This separation is also represented in our code an makes it easier to run different experiments on all our data sources. This will give us the most precise information to see which version of our prediction model works the best.

In regards to the stock market data, we worked with historical price data for several volatile stocks. This live stock market data includs essential information such as the date, opening price, highest and lowest prices of the day, closing price, and trading volume. The specific companies we chose to focus on in our model had to br popular and highly talked about, meaning their stocks would have to be highly traded and highly volatile. Choosing to work with the stock data of:

- NVIDIA Corporation (NVDA)
- Tesla, Inc. (TSLA)
- Microsoft Corporation (MSFT)
- Apple Inc. (AAPL)

ensured that their movement will possibly coincide, or have a correlation to, the data that was selected to derive our sentiment on.

3.2 Data preparation

As not all data is relevant to the analysis that we are trying to run as we have limited the time horizon we are analysing to 2011-2024. This means we end up needing about 15 Million rows of our final dataset to cover that timeframe.

We change the date format of our data frame to yyyy-mm-dd to unify with the other components of our program and clean the "Article title" column of the data frame to only contain usable stings, dropping all other rows.

We prepare the necessary columns needed to store the data that we are creating during the sentiment analysis as our LSTM modell will need require multiple features to maximise the precsison score.

Our stock market data was cleaned to ensure chronological order and to fill missing data points using forward-filling methods.

We verified the continuity of the data for completeness across the selected date range. In cases of missing market data (e.g., due to holidays), we employed forward fill to maintain continuity. Additionally, we ensured synchronization of data frequency (daily) across both sentiment and financial datasets to allow for accurate window-based LSTM input generation.

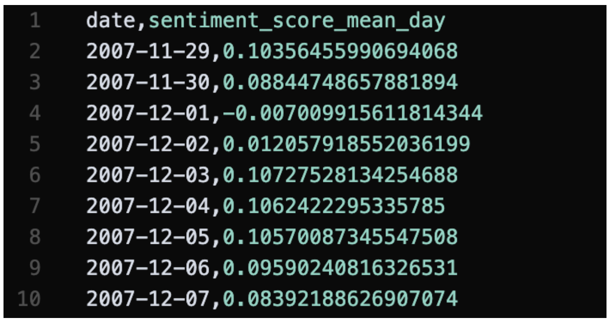
Finally, the merged dataset was split into training and testing sets using an 80/20 chronological split to simulate real-world forecasting conditions. This partitioning approach avoids information leakage and sets the foundation for the model evaluation phase discussed later in the paper.

4 | Sentiment Analysis (Jannick Drechsler)

Our initial idea was to pass our sentiment analysis model tweets or other short text data that would give us an indication about stock market movements. As described in the data selection section, we've landed on the FNSPID dataset [DFP24] from "Hugging Face" that we pass into the model.

Our first step was to prepare the different datasets used and normalize them to the relevant information. The Information needed are the Dates of the news, the headline / text itself as well as the information which companies are mentioned within the text. To recognize the companies names that we selected we are analysing each headline for either the company name or stock ticker to recoginise if a headline is targeted at one of our stocks. This enables us to store this data separately and weigh it differently in our model compared to the overall markets entiment that we are otherwise storing as a value between -1 and 1. The lower number representing a negative sentiment and a high number a positive one.

All headlines are then run through the sentiment analysis model and the resulting sentiment scores are then calculated into a general mean sentiment value for that day. This gives us the data we need to pass on to the machine learning model complete with the other indicators collected for the same periods of time. The data frame that is returned from the sentiment analysis model has the following form:



```
date,sentiment_score_mean_day
2007-11-29,0.10356455990694068
2007-11-30,0.08844748657881894
2007-12-01,-0.007009915611814344
2007-12-02,0.012057918552036199
2007-12-03,0.10727528134254688
2007-12-04,0.1062422295335785
2007-12-05,0.10570087345547508
2007-12-06,0.09590240816326531
2007-12-07,0.08392188626907074
```

Figure 4.1: Sentiment analysis returned data

5 | Financial Analysis (Konstantinos Papanagiotou)

5.1 Financial data preparation and workflow

The method used was technical analysis. Our core motivation behind using technical analysis, and in doing so, calculating financial indicators, is that they have been used by traders and analysts for decades. In accordance to our resources, as in, the historical prices dataset, technical analysis focuses solely on price action and chart patterns, offering time-tested signals about approximating future stock behavior. The financial analysis was performed using Python within the Cursor development environment, which provided real-time AI assistance while coding. The combination of Cursor and Python libraries such as pandas and matplotlib allowed for efficient calculation and visualization of financial indicators despite limited prior coding experience for the member of the team who overtook this segment of the project. Past that point, any needed adjustments were made by reading documentation and editing the code inside the Google Colab workspace. By then enriching these established metrics with sentiment analysis, we hope to assess whether combining both approaches can yield better predictive performance than using either one alone.

5.2 Financial data preparation and workflow

This analysis was mainly focused on widely-used technical indicators that are essential in helping to understand financial markets. Below, we outline each of these indicators, what they measure, why they are important, and how they were calculated.

Daily Returns

Daily returns measure the percentage change in the stock's closing price from one day to the next. They provide insight into the day-to-day performance and volatility of a stock. The daily return was computed as the percentage change in closing price: Daily Return =

$$(Close_t - Close_{t-1})/Close_{t-1}$$

Moving Averages (MACD)

Moving averages are trend-following indicators that smooth out short-term fluctuations to highlight longer-term price trends. MACD is a technical indicator to help investors identify entry points for buying or selling.

$$MACD = 12 - PeriodEMA26 - PeriodEMA$$

MACD is calculated by subtracting the long-term EMA (26 periods) from the short-term EMA (12 periods). An EMA is a moving average (MA) that places a greater weight and significance on the most recent data points. These were calculated using rolling window methods in Python.

When the short-term moving average crosses above the long-term moving average, it often signals a bullish trend. When the reverse happens, it may signal a bearish trend. Moving averages are essential for identifying market direction and potential trend reversals.

Bollinger Bands

Bollinger Bands are volatility indicators that consist of a SMA, simple moving average (middle band) and two bands that are typically two standard deviations away from the moving average. Calculation:

- Middle Band: 20-day SMA
- Upper Band: Middle Band + 2 × (20-day rolling standard deviation)
- Lower Band: Middle Band – 2 × (20-day rolling standard deviation)

When stock prices touch or move outside the upper or lower bands, it can signal overbought or oversold conditions, which traders often use to predict price corrections or breakout opportunities.

Relative Strength Index (RSI)

RSI is a momentum oscillator that measures the speed and change of price movements on a scale of 0 to 100.

RSI was calculated using a 14-day rolling period based on the magnitude of recent gains and losses. An RSI above 70 is generally considered overbought, while an RSI below 30 is considered oversold. These thresholds help identify when a stock might be due for a price reversal. Visualization To better understand the stock trends and validate the indicator calculations, we created several visualizations:

- Price Charts with Moving Averages: To show trend-following signals and potential crossovers.
- Bollinger Bands Overlay: To visualize periods of high or low volatility.
- RSI Time Series: To detect overbought or oversold zones.

5.2.1 Integration with Sentiment Analysis

The financial indicators computed in this section were saved in a structured dataset with dates aligned to match the sentiment analysis outputs derived from tweets and news headlines. This merged dataset forms the basis of our predictive modeling phase, where we aim to assess whether the combination of traditional financial analysis and sentiment signals offers superior predictive power over models based on only one of these sources.

6 Algorithm Selection (Jack Jürgens)

To predict stock prices based on complex and sequential data, it is important to select appropriate algorithms that can effectively process and learn from the given information. Not every model is suitable for this task, as the data's temporal nature and multimodal features require specialized handling to achieve reliable predictions. To forecast continuous stock prices reliably, we require an algorithm with the following characteristics:

6.1 Requirements

6.1.1 Time-Series Handling

Stock prices change over time, what happens today depends on what happened before. A good model must understand these time patterns and learn how trends develop. LSTM (Long Short-Term Memory) networks are made to remember what happened earlier using memory cells and gates. They work well for time series problems where past values affect the future [HS97b; BJ76].

6.1.2 Multimodal Data Integration

Our data includes:

- Daily stock values (Open, High, Low, Close, Volume) from yFinance
- Technical indicators like RSI, MACD, and Bollinger Bands
- Sentiment Analysis from financial news

A strong model should combine price data with extra information. LSTM models can handle many types of features at the same time and learn from them together [FK18; Tsa05].

6.1.3 Data Quality & Volume

Good predictions need clean and complete data over many years. Our system makes sure:

- All data files are clean, with no missing values
- We use long-term stock data (TSLA, NVDA, MSFT, AAPL)
- Features (indicators, sentiment scores...) are well prepared and stable over time

This helps the model learn from good clean data [Tsa05].

6.1.4 Regression Objective

We want to predict a number, not a category. The goal is to find the next day's closing price. LSTM models give real values as output and work well with MSE loss, so they are a good fit for this kind of task.

6.1.5 Conclusion

For continuous, multi-feature, time-dependent stock prediction using rich financial and sentiment data, LSTM seems to be the best choice. It meets all four requirements: sequence awareness, multimodal fusion, robust data handling, and regression capability. [HS97b; FK18; Tsa05].

6.2 Overview of Alternative Algorithms

Algorithm	**Typical Use Cases**	**Why Not Suitable for Our Project**
Linear Regression	Simple numeric prediction with linear relationships	Cannot capture nonlinear trends or temporal dependencies in stock prices. Assumes independence between data points.
K-Nearest Neighbors (KNN)	Pattern recognition, classification or regression with low-dimensional static data	No built-in concept of time or sequences. High dimensionality (multiple indicators over time) weakens distance-based logic.
Decision Trees	Interpretable decision logic for classification or regression	Makes decisions based on current input only. Cannot handle temporal patterns or sequences without major modifications.
Random Forest	Ensemble of decision trees for more stable predictions	More robust than a single tree, but still ignores time-dependencies. Predictions are based on snapshot data, not sequences.
XGBoost	Gradient boosting method used for structured/tabular data tasks	Very powerful for tabular data, but not designed for sequences. Requires manual feature engineering to capture temporal structure.
Convolutional Neural Networks (CNN)	Image recognition, spatial pattern detection	Good at extracting local features, but lacks memory. Cannot track how signals change over time unless heavily adapted.
Reinforcement Learning (RL)	Interactive agents in dynamic environments (e.g., games, trading bots)	RL is goal-driven and needs reward feedback. Overkill for one step regression prediction like nextday prices. Complex to train and evaluate reliably.

Table 6.1: Comparison of alternative algorithms and their limitations for the proect

6.3 Algorithm Limitations

While many classical machine learning algorithms offer powerful capabilities, they are not well-suited for time-series-based stock price prediction.

6.3.1 Linear Regression, Decision Trees, Random Forest and XGBoost

Models like Linear Regression, Decision Trees, Random Forests, and XGBoost work best when the data points are independent and follow the same distribution. This means they assume each example is unrelated to the others. However, this is not true for time series data like stock prices, where each value depends on what happened before. Because of this, these models struggle to learn important time-related patterns such as ongoing trends, periods of high or low volatility, and momentum changes. While you can try to fix this by manually adding past values or technical indicators as extra features, this process is time-consuming, fragile, and often still fails to capture long-term dependencies in the data effectively.

K-Nearest-Neighbors

K-Nearest Neighbors (KNN) suffers from similar limitations. It operates in a fixed feature space and uses distance metrics that do not account for sequential order. In high-dimensional financial datasets with multiple overlapping signals, the performance degrades due to the curse of dimensionality and the loss of temporal data.

6.3.2 Convolutional Neural Networks

Convolutional Neural Networks are really good at finding local patterns in data like images, where spatial relationships matter. When used for time series, 1D CNNs can pick up on short-term patterns or features. However, CNNs don't have a built-in way to remember information over longer periods, so they can miss important connections that happen across time. Because of this, CNNs alone aren't the best choice for tasks where understanding longer sequences and their dependencies is important—unless they're combined with models that have memory, like LSTMs.

6.3.3 Reinforcement Learning

Reinforcement Learning is usually used for problems where an agent has to make a series of decisions over time and gets feedback or rewards, such as playing games or controlling trading bots. But in our case, we don't need to make multiple decisions or learn from rewards over time. We only want to predict the next day's stock price based on past data. Using RL here would make the problem much more complicated without adding any real advantage, so it's not the best fit for our task.

6.3.4 Long Short-Term Memory

Long Short-Term Memory (LSTM) networks are made to work with sequences of data. They are a special kind of Recurrent Neural Network and have a built-in memory that helps them remember important information from earlier steps while ignoring things that don't matter. This is really useful for financial time series, where something that happened days or even weeks ago can still affect today's stock price. Unlike models that treat each data point separately, LSTMs look at the full sequence and learn time-based patterns without the need to manually add time-shifted features. They also produce real numbers as output and work well with loss functions like Mean Squared Error (MSE), which makes them a strong choice for tasks like predicting tomorrow's stock price.

6.4 Final Algorithm Selection

Choosing LSTM for our stock price prediction task makes sense because it's great at handling data that changes over time and spotting complex patterns. It can combine different types of information, like technical indicators and sentiment data, to make better predictions. LSTMs also work well with large sets of historical data, learning trends without needing a lot of manual preparation. They're perfect for predicting numbers, like the next day's closing price, as shown by successful uses in financial markets [FK18] and backed by reviews of deep learning in finance [SGO21]. LSTMs are flexible, scalable, and accurate, making them ideal for our project's needs.

7 Results (Simon Papekyan)

The predictive model is based on a Long Short-Term Memory (LSTM) neural network, which is well-suited for sequence modeling and can capture long-term dependencies in time series data. Our goal was to forecast daily stock closing prices by learning from historical trends and sentiments. The model design integrates both technical indicators and sentiment analysis, aiming to improve predictive accuracy by providing the network with broader market context beyond raw prices.

7.1 Data & Features

We collected daily stock data for multiple technology companies (Tesla, NVIDIA, Apple, Microsoft) from 2010 to early 2024, and we engineered various features. Technical analysis features were calculated from historical price series, and we incorporated a sentiment score per day derived from financial news or social media. Notably, the model was trained on *multiple stocks simultaneously*: a one-hot encoding of the stock ticker was included as part of the input features, allowing a single LSTM model to learn patterns across different stocks while distinguishing each stock's identity. The full feature set per day included both price-derived indicators and the sentiment metric, for a total of 12 features.

Key features comprised:

- Price-based Technical Indicators: e.g. Relative Strength Index (14-day RSI), Moving Average Convergence Divergence (MACD) with signal line, Bollinger Bands (20-day window), trading Volume, and simple moving averages . These indicators summarize momentum, trend, volatility, and liquidity aspects of the stock's recent performance.
- Sentiment Score: a daily sentiment indicator (e.g. average sentiment of news or tweets) aligned with the same date range . This reflects market sentiment or investor mood, which has been shown to correlate with stock price movements . By including sentiment, we intend for the model to account for exogenous information, such as optimistic or pessimistic news, that might affect stock prices.
- Ticker Encoding: categorical identifiers for each stock (e.g. TSLA, NVDA) encoded as additional binary features. This enabled the multi-stock model to implicitly learn a separate representation for each company's behavior, while still training on all data combined.

All features were normalized via Min-Max scaling. We used a sliding window approach to construct sequences: for each day t, the input $\mathbf{X}_t$ consisted of the previous 60 trading days of features (sequence length = 60), and the target y_t was the closing price at day t. Using 60-day sequences allows the LSTM to consider roughly three months of historical context for each prediction. Such sequence modeling leverages the LSTM's ability to maintain a memory of prior time steps, capturing both short-term patterns and longer trends in the data.

7.2 LSTM Model Architecture

The LSTM network was implemented in TensorFlow Keras as a sequential model. It contains two stacked LSTM layers followed by dense layers. The first LSTM layer has 64 units and returns sequences (so that the next LSTM layer can receive the sequence output), and the second LSTM layer has 32 units, returning a single output at the final time step. We applied a dropout regularization of 20% after each LSTM layer to prevent overfitting. Finally, a small fully-connected section with a Dense layer of 16 units and an output Dense layer of 1 unit produces the predicted price. The architecture and key hyperparameters are summarized in Listing7.1.

```
model = Sequential([
LSTM(64, return_sequences=True, input_shape=(60, 12)),
Dropout(0.2),
LSTM(32, return_sequences=False),
Dropout(0.2),
Dense(16, activation=   relu   ),
Dense(1)   # linear output
])
model.compile(optimizer=   adam   ,
    loss=   mean_squared_error   )
```

Listing 7.1: LSTM model architecture and compilation in Keras

We trained the model using the Adam optimizer with a mean squared error loss function, which is standard for regression. The training was performed for 30 epochs with a batch size of 32, using an 80/20 train-test split [HS97a]. The model saw roughly 14 years of data for training and was tested on the most recent years. We also set aside a portion of the training data for validation each epoch to monitor the model's performance on unseen data and mitigate overfitting. The final trained model encapsulates both the price dynamics and the influence of sentiment.

7.3 Results and Evaluation

We evaluated the LSTM model's predictive performance on the test dataset for each stock. Key evaluation metrics included Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and the coefficient of determination (R^2). The R^2 score is particularly informative as it reflects how well the model's predictions explain the variance in the actual stock prices. An R^2 of 1 indicates a perfect fit, while 0 indicates no better than predicting the mean [Bre+17]. In our multi-stock scenario, we computed these metrics separately for each stock's predictions to diagnose which stocks the model predicts more accurately.

7.3.1 Overall Performance

The model achieved solid accuracy for most stocks, though with notable variation across different companies. Table 7.1 summarizes the results for the four stocks. Tesla (TSLA) and Apple (AAPL) were predicted with the highest accuracy, each showing an R^2 above 0.94 (indicating that over 94% of the variance in the actual prices was captured by the predictions). Microsoft

(MSFT) also had a high R^2 around 0.88. In contrast, NVIDIA (NVDA) proved significantly more challenging. The model's R^2 on NVDA was only about 0.41, substantially lower than for the other stocks. The error metrics reinforce this pattern. For instance, NVDA's RMSE was around 8.7, $compared to only about$ 4.5 for Apple, which had the lowest error. The average R^2 across all four stocks was roughly 0.80, indicating overall strong performance with the exception of the NVDA case.

Stock	MAE ($)	RMSE ($)	R^2
TSLA	9.34	12.42	0.9513
NVDA	5.31	8.70	0.2891
AAPL	3.56	4.54	0.8629
MSFT	10.17	13.81	0.7557

Table 7.1: Prediction performance by stock

The most striking result is the disparity between Tesla and NVIDIA, representing the best and worst predictions, respectively. We focus on these two cases to analyze the possible reasons behind the model's performance.

7.3.2 TESLA

Tesla's stock predictions were very accurate, with $R^2 \approx 0.951$ on the test data (Figure 7.1). This means the LSTM captured Tesla's price trajectory extremely well. Several factors may explain this strong result. First, Tesla's stock exhibited a long upward trend over much of the 2010–2024 period, especially with substantial growth in recent years. The LSTM likely learned this predominant trend, and the 60-day window may have been sufficient to gauge the ongoing momentum at each point. Moreover, Tesla's stock is known to be heavily influenced by public sentiment and news (for example, tweets by CEO Elon Musk often sway the stock price). By incorporating the sentiment feature, the model had an advantage in predicting Tesla's moves. Positive shifts in sentiment often coincided with rallies in Tesla's price, so the model could use sentiment upticks as a signal for upward price movement. This aligns with findings in prior research that social media sentiment is strongly correlated with Tesla's stock fluctuations [BB22].

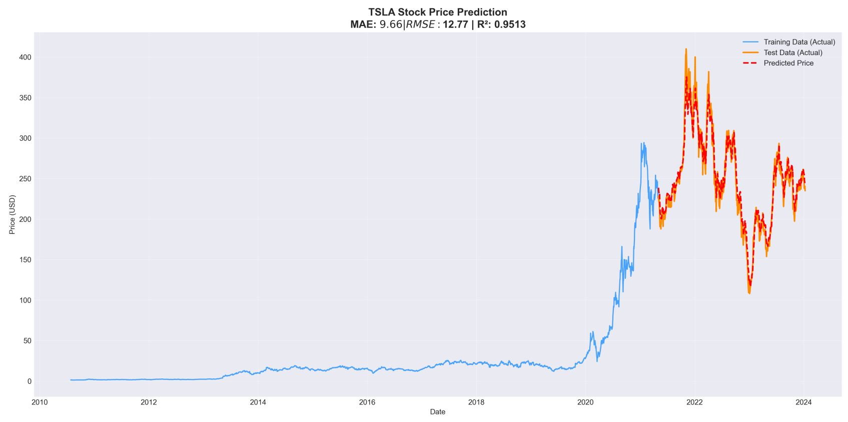


Figure 7.1: prediction TSLA

7.3.3 NVIDIA

NVIDIA's stock proved difficult for the model to predict, yielding the lowest R^2 (around 0.29) among the stocks (Figure 7.2). In practical terms, the model often failed to capture the timing or magnitude of NVDA's price swings, explaining less than half of the variance. This can be attributed to the nature of NVIDIA's stock behavior. NVDA is a highly volatile stock; over the last decade it experienced periods of rapid growth (for instance, explosive gains during 2016–2017 and again in 2020–2021) as well as sharp drawdowns. Such volatility, especially when caused by sudden market events, is inherently hard for a sequential model to learn. The LSTM might struggle to foresee abrupt trend reversals or surges that were not preceded by similar patterns in the prior 60 days [XYX24].

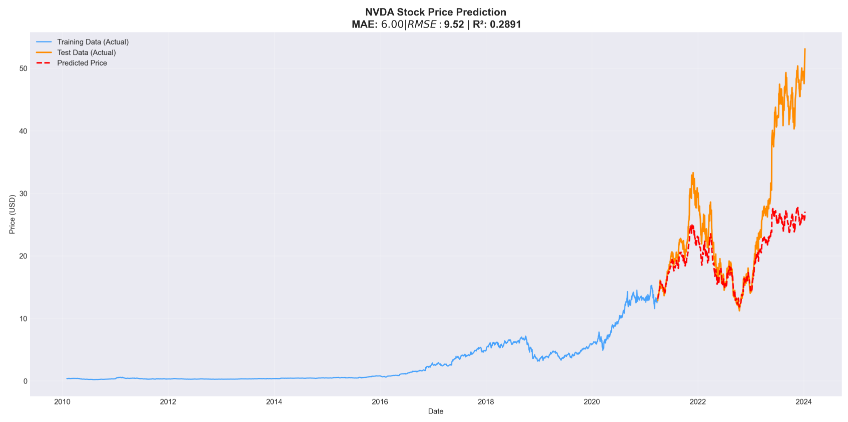


Figure 7.2: prediction NVDA

For completeness, the model's performance on Apple and Microsoft should be mentioned. Apple (AAPL) had an R^2 of about 0.86 with very low error, which is on par with Tesla's performance. Apple's stock is relatively stable with long-term growth, so the LSTM could model it quite well. Microsoft (MSFT) had an intermediate $R^2 \approx 0.756$, slightly lower than Apple's but still indicating a very good fit. Microsoft's larger absolute price scale and perhaps some idiosyncratic events like product launches or CEO changes, might have contributed to higher error than Apple. Nonetheless, the model handled both of these large-cap stocks effectively. The inclusion of technical indicators and sentiment likely benefited these predictions too, though their stocks are less sentiment-driven than Tesla. Overall, the multi-stock LSTM demonstrated strong generalization for stable growth stocks, while its weaknesses emerged with the more erratic pattern of NVIDIA.

7.4 Conclusion

In summary, we developed an LSTM-based model incorporating technical indicators and sentiment analysis to predict stock prices for multiple companies. The model architecture and the use of a 60-day historical window were effective in capturing long-term trends, as evidenced by excellent prediction accuracy for stocks like Tesla and Apple.

Overall, the project achieved its aim of stock price prediction with a meaningful degree of accuracy. By analyzing the best and worst cases, we have gained insights into the model's behavior. It excels when patterns are consistent and sentiment cues align with price movements, and it falters when faced with sudden market shifts that are hard to learn. These findings are consistent with broader research in stock prediction and highlight the importance of aligning model design to the characteristics of the target asset. Our LSTM model, with its multi-feature input, provides a strong foundation that could be further refined for improved robustness across different market conditions.

Bibliography

[BB22] Amey Bhadkamar and Sonali Bhattacharya. *Tesla Inc. Stock Prediction using Sentiment Analysis*. In: *Australasian Accounting, Business and Finance journal* 16.5 (2022).

[BJ76] George E. P. Box and Gwilym M. Jenkins. *Time Series Analysis: Forecasting and Control*. Holden-Day. 1976.

[Bre+17] Claus Brell et al. *Statistik in zwei Dimensionen*. In: *Statistik von Null auf Hundert: Mit Kochrezepten schnell zum Statistik-Grundwissen* (2017), pp. 57–68.

[DFP24] Zihan Dong, Xinyu Fan, and Zhiyuan Peng. *FNSPID: A Comprehensive Financial News Dataset in Time Series*. 2024. arXiv: 2402.06698 `[q-fin.ST]`.

[FK18] Thomas Fischer and Christopher Krauss. *Deep learning with long short-term memory networks for financial market predictions*. European Journal of Operational Research, 270(2), 654–669. 2018.

[HS97a] Sepp Hochreiter and Jürgen Schmidhuber. *Long Short-Term Memory*. In: *Neural Computation* 9.8 (1997), pp. 1735–1780.

[HS97b] Sepp Hochreiter and Jürgen Schmidhuber. *Long short-term memory*. Neural Computation, 9(8), 1735–1780. 1997.

[SGO21] Omer Berat Sezer, M. Ugur Gudelek, and Ahmet Murat Ozbayoglu. *Financial time series forecasting with deep learning: A systematic literature review: 2005–2019*. Applied Soft Computing, 102, 107181. 2021.

[Tsa05] Ruey S. Tsay. *Analysis of Financial Time Series*. John Wiley & Sons, 2nd Edition. 2005.

[XYX24] Yiluan Xing, Chao Yan, and Cathy Chang Xie. *Predicting NVIDIA's Next-Day Stock Price: A Comparative Analysis of LSTM, MLP, ARIMA, and ARIMA-GARCH Models*. 2024. DOI: 10.48550/arXiv.2405.08284. arXiv: 2405.08284 `[econ.EM]`.