Università Commerciale Luigi Bocconi

Bachelor of Science in Mathematical and Computing Sciences for Artificial Intelligence

Enhancing Asset Value Prediction Using Long Short-Term Memory Networks

Supervisor:
Prof. Chen Liu

Bachelor of Science Thesis by:
Cesare Bergossi

Academic Year 2023 – 2024

Abstract

The financial sector is undergoing a radical transformation driven by the rapid advancements in machine learning. With the ability to capture complex and non-linear patterns, neural networks, specifically Long Short-Term Memory (LSTM) networks, present a promising tool for predicting stock prices, a task traditionally dominated by linear models. Developed specifically for sequential tasks, LSTMs excel at retaining long-term dependencies through their memory cells. This thesis investigates the application of LSTM networks to forecast stock prices within the S&P 500 index. Traditional linear regression models often fall short in capturing the dynamic nature of financial markets due to their limited representation power. By implementing both univariate and multivariate LSTM models, this study aims to demonstrate the superior predictive capabilities of these advanced techniques. Our findings reveal that univariate LSTM models, using only historical stock prices, significantly outperform linear regression in trend prediction and error reduction. Enhancements such as gradient clipping and early stopping were employed to stabilize the models and improve convergence. Multivariate LSTM models, incorporating additional variables like trading volume and financial ratios, showed initial promise but introduced additional noise, highlighting a clear trade-off between model complexity and stability. Furthermore, varying model depth and hidden layer sizes revealed that while deeper models and larger hidden layers can capture intricate patterns, they are also more susceptible to overfitting and instability, requiring careful hyperparameter tuning. Robustness tests on stocks with extreme price and volume characteristics demonstrated the adaptability of LSTM models to diverse market conditions. Overall, LSTM models, particularly with proper tuning and multivariate inputs, dramatically enhance the accuracy of stock price prediction over traditional methods.

"Если у вас есть реальная цель, которой вы хотите добиться всей душой, то у вас обязательно получится. Сомнений быть не может. А если они есть, и вам кажется, что вы хотите этого не на 100 процентов, а на 99, тогда ищите свою настоящую цель, ради которой вы будете готовы на всё."

"If you have a real goal that you wholeheartedly want to achieve, then you will definitely succeed. There can be no doubt about it. But if there are doubts, and you feel that you want this not 100 percent, but only 99 percent, then look for your true goal—one for which you would be willing to do anything."

— Aliya Mustafina

Contents

Chapter 1

Introduction

1.1 Background and Significance

Over the last decade, the application of machine learning in financial markets has gained significant attention due to its potential to improve investment strategies and decision-making processes. Traditional financial models, while robust, often fail to capture the intricate and dynamic patterns present in financial data. This limitation has led to the exploration of advanced computational techniques, particularly neural networks, to enhance the accuracy and reliability of asset price predictions. Neural networks, in fact, possess remarkable expressive power to learn the features of the data. This enables modern machine learning systems to model complex and time-dependent relationships in an end-to-end manner.

This thesis is part of a larger research project conducted in collaboration with Professors Chen Liu and Yuan Zhang, whom I had the privilege of working with during my exchange program at City University of Hong Kong. The complete project aims to enhance asset value prediction through advanced machine learning methodologies, specifically Long Short-Term Memory (LSTM) Networks on historical data and Natural Language Processing (NLP) on financial news, and subsequently develops a trading strategy using reinforcement learning, with the aim of maximizing expected returns.

1.2 Objectives of the Study

The primary objective of this thesis is to focus on the effectiveness of Long Short-Term Memory (LSTM) networks in predicting asset prices. LSTM networks, a type of recurrent neural network (RNN), are particularly suited for time series data due to their ability to capture long-term dependencies. The goal of the research is thus to determine whether LSTM models can outperform traditional regression techniques in forecasting stock prices.

A key innovation in this thesis is the use of multivariate LSTM networks for financial time series data. While these architectures have been widely used in various fields, their specific application in financial markets for predicting asset prices remains relatively underexplored. Moreover, although LSTM has already been implemented for this task, the model has not been thoroughly explored, for instance by adjusting hyperparameters, including several variables in the training, or testing on various less stable assets. This study aims to fill this gap by systematically evaluating the performance of multivariate LSTMs, testing different model configurations and comparing them with traditional methods.

1.3 Structure of the Thesis

The thesis is structured as follows:

- Data Acquisition and Preprocessing: Collection and initial preparation of financial data from the CRSP and Compustat databases. This includes cleaning the data, handling missing values and transforming features to prepare them for further analysis.

- Exploratory Data Analysis: Preliminary data analysis to find trends, patterns or anomalies in the financial data. It includes analysis of distribution of stock prices, visualization of securities by exchange and industry, and identification of outliers.

- Baseline Methods: Implementation and evaluation of traditional regression models, specifically linear regression and Lasso regression. This provides a benchmark against which the performance of LSTM models can be measured.

- Long Short-Term Memory (LSTM) Networks: Design, training and validation of LSTM models using both single-variable and multi-variable approaches. It also explores enhancements like gradient clipping and early stopping, and analyzes the effect of varying model depth and hidden layer sizes, concluding with robustness tests on assets with extreme price and volume characteristics.

- Conclusion and Future Research: Summarizing the results and discussing potential model improvements and extensions for future studies.

Chapter 2

Literature Review

2.1 Overview of Machine Learning in Finance

The application of machine learning in finance has had an enormous impact on decision-making in finance. Machine learning techniques excel in processing vast amounts of data and identifying subtle patterns that traditional methods might miss.

The financial industry has always heavily relied on data. Traditional methods focused on quantitative analysis for tasks like risk assessment [29], asset valuation and investment strategy development. The introduction of machine learning has brought more depth through more sophisticated data analysis and predictive modeling. Early applications of ML in finance focused on credit scoring [13] and fraud detection [3], using algorithms like logistic regression and decision trees to classify and predict outcomes based on historical data.

Several machine learning techniques have now become standard in finance. The main ones include:

1. Supervised Learning: Techniques like linear regression, support vector machines (SVM) and neural networks are commonly used for prediction. These methods have been applied to forecast stock prices, estimate risk premiums and identify trading opportunities [21].

2. Unsupervised Learning: Methods such as clustering and principal component analysis (PCA) [1] are used for market segmentation [39], anomaly detection [35] and reducing the dimensionality of financial data. These techniques help in identifying hidden structures and relationships within datasets.

3. Reinforcement Learning: This approach has been particularly useful in automating trading strategies [28]. These models learn optimal trading policies through trial and error, adapting to new market conditions to maximize returns.

4. Deep Learning: Convolutional neural networks (CNNs) and recurrent neural networks (RNNs), including Long Short-Term Memory (LSTM) networks, have been used to analyze complex temporal and spatial patterns in financial data. These models excel in tasks such as sentiment analysis [12], high-frequency trading [2] and time series forecasting [25].

These machine learning techniques have had a significant impact on various aspects of financial markets.

In risk management, machine learning models improve risk assessment by analyzing historical data and identifying potential risk factors. They can improve the accuracy of credit scoring, portfolio risk evaluation and stress testing. In credit scoring, ML models can evaluate an individual's or an entity's creditworthiness with greater accuracy by considering a larger number of variables and their interactions., resulting in more reliable credit scores and better lending decisions [13]. For portfolio risk evaluation, these models analyze historical performance and asset correlations to identify vulnerabilities and optimize asset allocations, minimizing risk while maximizing returns [9]. Furthermore, in stress testing, ML techniques can simulate adverse economic scenarios to evaluate the robustness of financial institutions, helping them prepare for potential crises and regulatory requirements by incorporating more complex and non-linear relationships between variables [17].

Algorithmic trading has been revolutionized by ML algorithms, especially those based on reinforcement learning. These models continuously learn and adapt to market conditions, allowing traders to execute orders at optimal prices by predicting short-term price movements with high accuracy [43]. By analyzing historical trading data and real-time market information, these algorithms can identify the best times to buy or sell assets, minimizing market impact and slippage. In addition, reinforcement learning algorithms are excellent at discovering arbitrage opportunities, by detecting slight price discrepancies between different markets or related financial instruments [45].

Fraud detection has also benefited significantly from machine learning techniques, as they are crucial in detecting fraudulent activities by identifying abnormal transaction patterns and behaviors. They analyze vast datasets to find hidden patterns and anomalies that may indicate fraud. ML models keep learning from new data, thus improving their ability to spot advanced fraud tactics [3].

In portfolio management, machine learning-based strategies optimize asset allocation and diversification. They analyze vast amounts of data, including historical market trends, asset performance and correlations between different assets. By identifying patterns and relationships within the data, ML models can make more informed predictions about future market movements. Moreover, by taking into account investor preferences, such as risk tolerance and investment horizon, they allow to build portfolios tailored to the needs of each investor, while maximizing returns and minimizing risk [6].

2.2 Previous Work on Asset Value Prediction

Let us review previous research on asset value prediction, to examine how different models, from traditional statistics to advanced machine learning, have been applied to forecast asset prices.

Gu, Kelly and Xiu (2020) [18] compared machine learning methods for measuring asset risk premiums, discovering that trees and neural networks were particularly effective due to their ability to handle nonlinear interactions among predictors. Their study suggests that machine learning can highly increase economic gains, with neural networks performing exceptionally well, often even doubling the performance of traditional methods. They also interestingly found that shallow learning models outperform deeper networks, possibly due to the intrinsic noise and complexity of financial data. Finally, they emphasize the importance of minimizing MSE and using regularization to improve model robustness and manage overfitting, particularly in financial models overloaded with noisy data.

Taking another path, Derek Snow (2019) [42] addressed the challenge of predicting financial events, which is a significant issue when forecasting asset prices. He focused on the application of machine learning and contrasted it with past approaches that mainly relied on linear models, limited by their capacity to handle variables and data diversity. Snow showed how machine learning, with its ability to use non-linear models and a wide range of variables and data types, can significantly improve predictions of financial events like earnings surprises, bankruptcy and facility closures.

Ding et al. (2019) [14] also focused on the prediction of extreme events in time series data, but not limited to finance. They identified that the weakness of deep learning methods in this context lies in the conventional form of quadratic loss. To tackle this, they introduced a new form of loss called Extreme Value Loss (EVL) specifically for detecting future occurrences of extreme events. Additionally, they proposed the use of a Memory Network to memorize extreme events in historical records. Their approach, which incorporated EVL with an adapted memory network module, provided an end-to-end solution for time series prediction that included extreme events.

Lachaab and Omri (2023) [24] investigated the predictive performance of machine and deep learning methods during the COVID-19 pandemic, focusing on the CAC 40 index of the French stock market. They applied KNN (K-Nearest Neighbors) and LSTM methods, comparing their performance to the traditional ARIMA time series model. The study was conducted under various scenarios, including before, during, and after the COVID-19 pandemic and vaccination periods. The findings revealed that the KNN method outperformed both LSTM and ARIMA in forecasting the CAC 40 index. As the previous authors, they yet again showed how machine learning methods can adapt to and predict market behavior under extraordinary global events like a pandemic.

Similarly, Moghar and Hamiche (2020) [27] explored the use of LSTM (Long Short-Term Memory) networks for predicting future stock market values, once more demonstrating that simple models often fall short due to the complex nature of financial predictions.

Their paper primarily explored the precision level achievable by a machine learning algorithm in predicting market trends and how the number of training epochs can improve the model. The use of LSTM, the type of recurrent neural network which will be used in this thesis, was particularly significant given its ability to remember information over extended periods, which is crucial for capturing and learning from historical patterns and trends.

While previous studies have shown the effectiveness of machine learning methods and especially LSTM networks for predicting stock prices, this thesis aims to further prove this effectiveness and extend these models by incorporating a multi-variable approach, which utilizes not only price data but also additional market indicators, such as trading volume and book-to-market ratios. Furthermore, it will explore the impact of changing the parameters of the LSTM architecture, such as the number of layers and hidden units, to optimize the model's performance.

Chapter 3

Data Acquisition and Preprocessing

3.1 Data Sources

This research uses data sourced from Wharton Research Data Services (WRDS) [47], a gateway to several financial, economical and accounting databases. WRDS is well-known for its broad range of data that supports financial and economic research, making it extremely valuable for academic and institutional analysts. Through WRDS, we access detailed stock market data from two pivotal databases: CRSP for stock pricing and trading volume, and Compustat for detailed corporate financial records.

Our analysis specifically focuses on stocks listed in the S&P 500, one of the main market indices that represents the stock performance of 500 large companies listed on stock exchanges in the United States [37].

The study covers a period from January 1, 2010, to December 31, 2022. This is a large enough timeframe which includes significant economic events, such as the COVID-19 pandemic [40]. This provides a dynamic context for analyzing the impacts on stock prices and market behavior.

3.2 CRSP Database

The Center for Research in Security Prices (CRSP) provides detailed data on the U.S. stock market, including historical stock prices, returns and trading volume [11]. It covers all listed NYSE, AMEX and NASDAQ stocks, making it particularly useful for long-term studies and detailed market analysis. For our thesis, CRSP is essential as it provides daily historical pricing and volume data for S&P 500 stocks, as well as identifiers for the specific company, type of shares, exchange where the stock is traded and more.

3.2.1 CRSP Data Overview

In this initial phase, we connect to the WRDS platform to query the CRSP database, extracting the main financial metrics for our analysis. To include all relevant trading data, even in periods when stocks were not actively trading, the data query includes specific exchange codes. This ensures that non-trading months are not excluded, which is particularly important for cumulative return calculation in momentum strategies. The dataset contains over 1.48 million records, each carrying a wide array of financial features. The main fields included in our dataset are listed in Table 3.1.

Table 3.1: Description of CRSP data columns.

Column	**Description**
`permno`, `permco`	Unique CRSP identifiers for stocks and companies.
`ncusip`	CUSIP number, a 9-character code identifying North American securities.
`shrcd`, `exchcd`	Type of shares and exchange on which they are traded.
`siccd`	Standard Industrial Classification Code, industry sector of the company.
`ret`	Stock return, typically percentage change in price.
`vol`	Trading volume, number of shares traded.
`shrout`	Shares outstanding, total number of shares issued by the company.
`prc`	Stock price, usually closing price.
`cfacpr`, `cfacshr`	Factors for adjusting prices and shares to analyze historical data consistently.

Additionally, we adjust financial metrics to account for corporate actions that could distort the stock's apparent performance. By calculating adjusted prices (`p`) and total shares outstanding (`tso`), we can derive market capitalization (`me`) as their product.

$$\texttt{me} = \texttt{p} \times \texttt{tso}$$

Market capitalization is a useful measure of company size which reflects the total market value of the company's equity. Then, the company-level market capitalization (`me_comp`) is just the sum of market capitalizations of all stocks belonging to the same company on a given date. This variable helps getting a more consolidated view of the company's total market value across all its traded stocks for each specific date.

3.3 Compustat Database

Compustat offers a wide range of financial and market information on both active and inactive global companies [38]. It is essential for obtaining detailed annual and quarterly financial statements, market data and other derived metrics critical for financial analysis and modeling. In this research, Compustat supplies the corporate financial metrics that complement the pricing and volume data from CRSP. This combination allows for a more complete analysis of market trends and corporate performance.

3.3.1 Compustat Data Overview

To acquire detailed financial metrics for S&P 500 companies, we query the Compustat database. In this way, we want to obtain a multi-dimensional view of each company's financial health and market performance. Note that, unlike the daily CRSP records, this dataset is collected on a monthly basis, resulting in around 97,000 rows. Key features of this dataset are displayed in Table 3.2.

Table 3.2: Description of Compustat data columns.

Column	**Description**
`gvkey`	Global Company Key, unique identifier for companies.
`cusip`	CUSIP number, a 9-character code identifying North American securities.
`sich`	Standard Industrial Classification Historical, denotes the primary industry.
`seq`	Stockholders' equity, total equity held by shareholders.
`pstkrv`, `pstkl`, `pstk`	Preferred Stock: redemption value, liquidation value and total carrying amount.
`txdb`	Deferred taxes on the balance sheet.
`itcb`	Investment Tax Credit Balance, credit against tax liabilities.
`total_assets`	Total value of all assets held by the company.
`common_equity`	Equity attributable to common shareholders.
`total_liability`	Total amount owed by the company.
`sales`	Net amount of sales or turnover.
`common_shares_outstanding`	Number of common shares currently outstanding.
`price_close_annual`	Closing stock price at the fiscal year end.

Furthermore, we compute the preferred stock value (`pref`). Preferred stock represents a class of ownership in a corporation that has a higher claim on its assets and earnings than common stock. Preferred shares typically provide dividends that must be paid out before dividends to common shareholders and have priority over common stock in the event of liquidation. Its evaluation is fundamental for an accurate analysis of a company's financial health. We employ a hierarchical method to determine its value, prioritizing the most accurate measures and resorting to less preferred ones only when necessary:

1. Redemption Value (`pstkrv`): The primary choice, representing the price to redeem the preferred stock, reflecting its market value closely.
2. Liquidating Value (`pstkl`): If redemption value is unavailable, we use the liquidating value, which is the payout amount if the company were liquidated.
3. Carrying Value (`pstk`): Used when neither redemption nor liquidating values are available, this value is recorded on the balance sheet and provides a baseline valuation, though it may not always reflect current market conditions.

Lastly, we calculate the book equity value (`be`), a fundamental financial metric representing the residual interest in a company's assets after deducting all liabilities. It is calculated as the sum of stockholders' equity (`seq`), deferred taxes (`txdb`) and investment tax credits (`itcb`), minus the value of preferred stock (`pref`). This metric provides a precise measure of a company's net worth from an accounting perspective.

$$\texttt{be} = \texttt{seq} + \texttt{txdb} + \texttt{itcb} - \texttt{pref}$$

3.4 Data Merging

To combine financial data from Compustat and trading data from CRSP, we merge the two datasets together, allowing us to analyze financial health and market performance together. To do so, we use the CRSP/Compustat Merged (CCM) link table, mapping the unique identifiers from both datasets, specifically Compustat's gvkey and CRSP's permco and permno [46].

Since CRSP data is on a daily basis and Compustat data is monthly, we want to align the two after merging them together. To do this, we fill missing values for Compustat-exclusive variables by forward-filling (propagating the last known value to the next) and backfilling (replacing initial missing values with the next valid ones) for each variable.

After merging, we can compute another key financial indicator, the book-to-market ratio (`bm`), which compares the book value of a company to its market value. This ratio helps in assessing if a stock is undervalued or overvalued relative to its actual worth. It is calculated by dividing a company's book equity (`be`) by its market capitalization (`me_comp`).

$$\text{bm} = \frac{\texttt{be}}{\texttt{me\_comp}}$$

Chapter 4

Exploratory Data Analysis

4.1 Visualization by Exchange and Industry

We begin this exploratory data analysis by graphically representing the number of securities by exchange and industry.

Before visualizing the data, we transform the listing exchange codes into explicit exchange names for clarity. This transformation, based on Bali, Engle, and Murray (2016) [5], helps in understanding the data without constantly referring back to code definitions. Similarly, we convert the Standard Industrial Classification (SIC) codes into industry descriptions, again following Bali et al. [5].

Visualization by Exchange

Figure 4.1 shows the daily number of securities listed on different exchanges (NASDAQ, NYSE and others) from 2010 to 2022. Clearly, the most common listing exchange among S&P 500 stocks is the New York Stock Exchange (NYSE), which is not surprising since it is the largest stock exchange in the world by market capitalization. However, it exhibits a gradual decline in time, which might suggest delistings, mergers or shifts to other exchanges. The second most prevalent exchange is NASDAQ, a global electronic marketplace for buying and selling securities. Unlike NYSE, NASDAQ shows a steady increase in the number of securities, meaning that more companies are choosing to list on this exchange.

Visualization by Industry

Figure 4.2 displays the daily number of securities by industry sectors such as Agriculture, Construction, Finance, etc. The main industry is manufacturing, which showed a notable decrease since 2010, reflecting industry changes or consolidations. Other industries with a high number of securities are finance and services, which remained relatively stable over time.

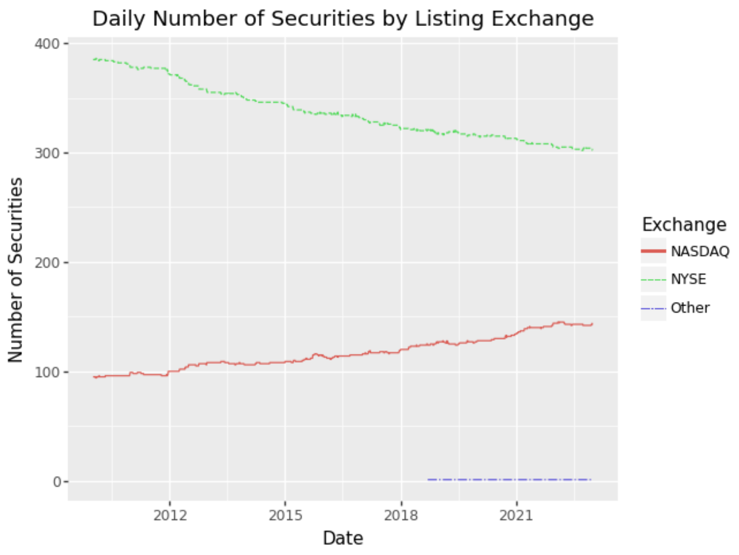


Figure 4.1: Daily number of securities by listing exchange.

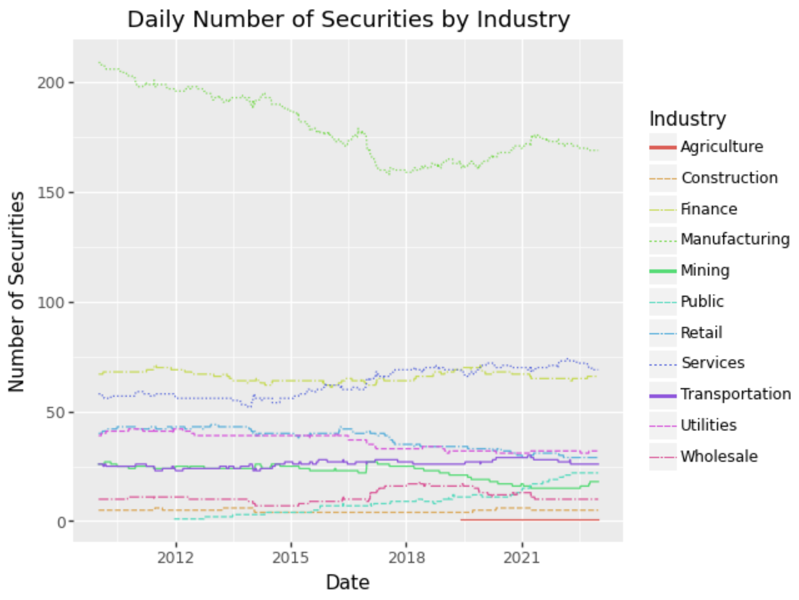


Figure 4.2: Daily number of securities by industry.

4.2 Distribution of Stock Prices

Analyzing the distribution of stock prices helps us understand market conditions, valuation trends and identify potential outliers that may impact investment strategies.

Histogram of Stock Prices

Histograms provide a visual representation of the distribution of stock prices. They highlight the range and frequency of prices within the dataset.

The first histogram (Figure 4.3) shows the distribution of stock prices across the entire range. It can be clearly seen that most stock prices are clustered at the lower end, with only a few high-priced outliers. To get a better understanding, the second histogram (Figure 4.4) focuses on stock prices up to $300, which captures the core of the data.

These histograms indicate that the majority of stock prices in the dataset are below $300, suggesting a right-skewed distribution. This means that there are more low-priced stocks and a smaller number of high-priced outliers.

Box Plot of Stock Prices

Box plots are useful for visualizing the spread and outliers in stock prices. They show the median, quartiles and extreme values in the data. The box plot (Figure 4.5) reveals that the median stock price is much lower than the mean, indicating that most stocks have lower prices. The long whiskers and numerous outliers suggest once again that while many stocks are moderately priced, a few have very high prices.

These high-priced outliers, which can reach up to $6000, may belong to high-growth industries or represent stocks that have experienced significant price increases. This wide range of stock prices shows the diversity in market valuations and warns us not to make assumptions about stock performance or market trends based on average prices alone.

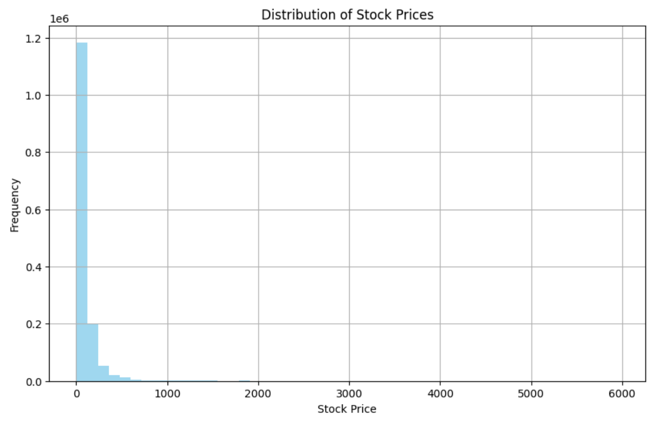


Figure 4.3: Distribution of stock prices.

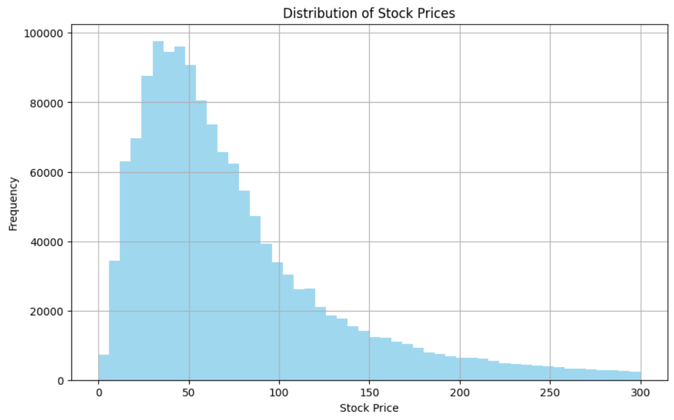


Figure 4.4: Distribution of stock prices (up to $300).

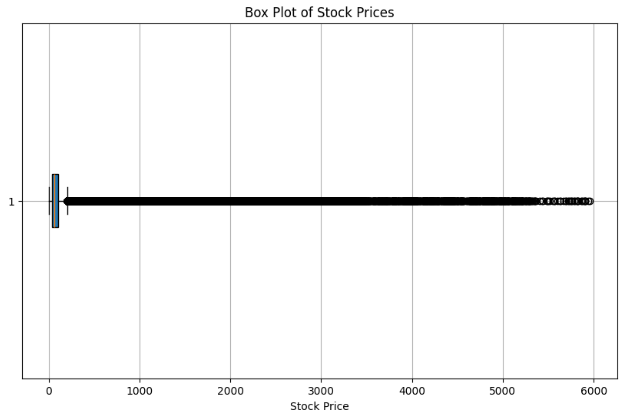


Figure 4.5: Box plot of stock prices.

4.3 Price Trends Over Time

Analyzing the price trends of individual stocks over time can help us understand how different companies have performed historically. Here, we plot the time series of six randomly selected stocks from our dataset, to understand their behavior from 2010 to 2022.

Figure 4.6 showcases the diverse nature of stock price movements within the market. Each stock has unique behavior, which reflects the performance of the company, sector dynamics and market conditions over time. For example, Electronic Arts Inc. (ERTS) shows a steady growth, especially from 2014 onwards, possibly indicating the company's successful product launches and market expansion. In contrast, Kroger Co. (KR) stock price is quite volatile, with peaks around 2016, which could reflect the competitive and dynamic nature of the retail sector.

These variations tell us that it is impractical to assume a common distribution, such as the normal distribution, for stock prices. Prices are influenced by a range of factors such as earning reports, investor sentiment, macroeconomic indicators and geopolitical events. Each stock requires specific analysis to understand its potential and risks, rather than making broad assumptions based on the overall market.

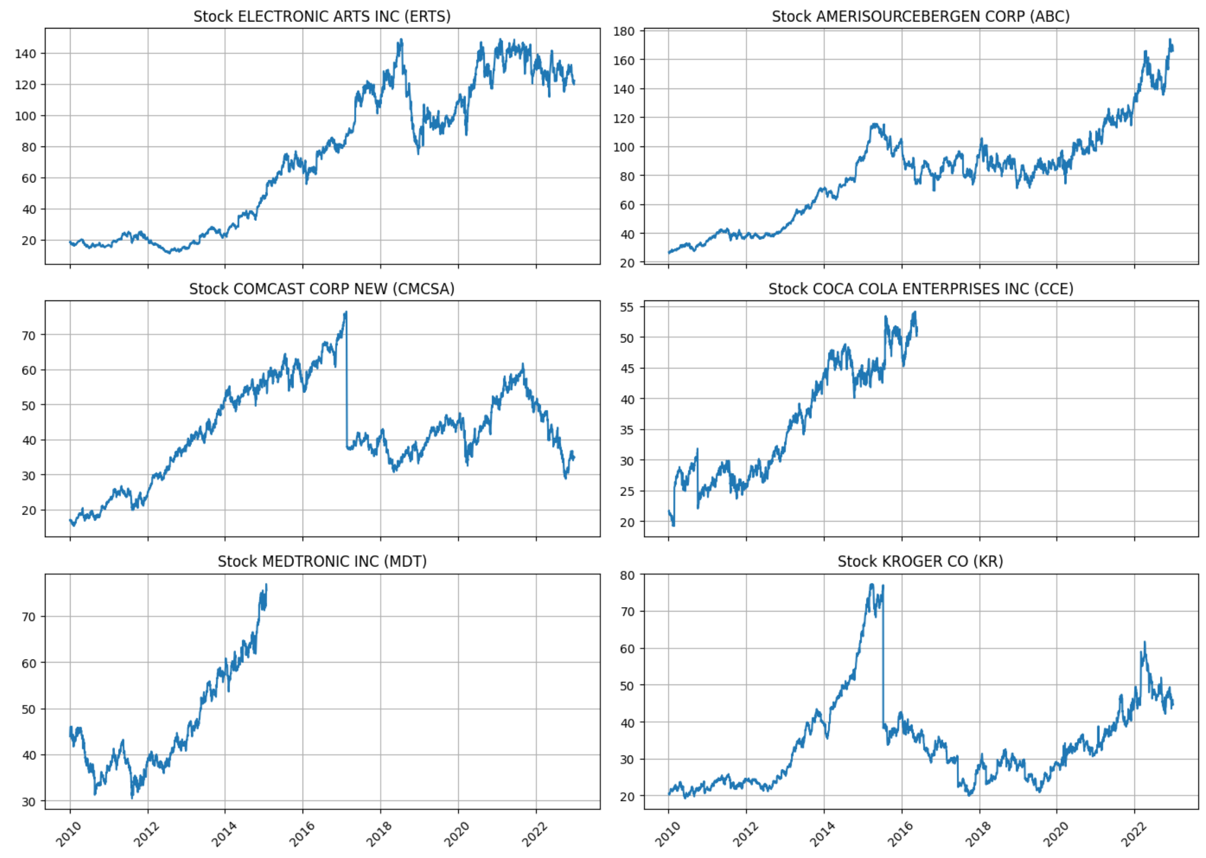


Figure 4.6: Price trends of selected stocks over time.

4.4 Including Outliers

Since our goal is to build a model capable of predicting stock prices independently of their price scale, it is important to include stocks with very high prices in our analysis. These stocks often behave differently from the majority of the market, influenced by unique factors such as company-specific news or broader economic policies, and exhibit distinct patterns and reactions to market events.

Including these outliers is crucial, as it helps the model learn from a wider variety of data, making it more versatile and capable of handling different types of stocks and market conditions. This approach can help the model understand the full spectrum of market dynamics, ensuring it can accurately predict not only average stocks but also high-value ones.

Chapter 5

Baseline Methods

5.1 Linear Regression

Before exploring complex machine learning models like neural networks, we will establish baseline methods. These methods provide a reference point to compare the performance of more advanced techniques. We start with simpler models first to understand the basic relationships in the data and set a benchmark for evaluating the added value of neural networks.

The first method we address is linear regression, one of the most elementary for prediction tasks [7]. Linear regression is a fundamental statistical method used to model the relationship between a dependent variable (price, in our case) and one or more independent variables (for instance, trading volumes, financial ratios, etc.). The linear regression model assumes a linear relationship between the input variables (features) and the output variable (target).

The general form of a linear regression equation is:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \epsilon$$

where:

- y is the dependent variable (the variable to be predicted).
- $x_1, x_2, \ldots, x_n$ are the predictors or the sets of true values (the feature variables).
- β_0 is the intercept term (the value of y when all predictors are 0).
- $\beta_1, \beta_2, \ldots, \beta_n$ are the coefficients for the independent variables $x_1, x_2, \ldots, x_n$, representing the rate at which the variables change.
- ϵ is the error term, the difference between the observed and predicted values.

By fitting a linear regression model, we can estimate the coefficients (β values) that best describe the relationship between features and the stock prices.

Note that linear regression is not formulated to predict time series. To add some kind of time dependency to the model, we add a new numerical feature representing the number of days that passed after the first date we consider for our analysis (January 4, 2010).

Advantages and Limitations of Linear Regression

Linear regression offers several advantages, including simplicity, interpretability and efficiency. It is easy to understand and implement, making it accessible for a wide range of applications. The coefficients in linear regression provide a clear interpretation of the relationship between the features and the target variable, which helps in understanding the underlying data dynamics. Additionally, linear regression can be computed quickly, even on large datasets, making it an efficient choice for many tasks.

However, linear regression also has limitations. It assumes a linear relationship between the features and the target variable, which may not always hold true for stock prices and other complex datasets. This assumption can lead to inaccuracies when the actual relationships are non-linear. Furthermore, linear regression is sensitive to outliers, which can significantly affect the model's performance by distorting the estimated relationships. Another limitation is the potential for overfitting, especially when dealing with a large number of features. Overfitting occurs when the model captures noise in the training data, leading to poor generalization on new, unseen data.

5.1.1 Model Evaluation Metrics

To evaluate the performance of our linear regression models, we use several key metrics to estimate their accuracy and efficiency. These metrics help us understand how well our model predicts stock prices and identify areas for improvement [8].

- **Mean Squared Error (MSE)**: Measures the average squared difference between the estimated values and the actual values. It is a common measure of prediction accuracy, where a lower MSE indicates a better fit. MSE is usefulfor identifying large errors due to the squaring, making it more sensitive to outliers.
- **Root Mean Squared Error (RMSE)**: Represents the square root of MSE. It is interpreted as the prediction error in the same units as the response variable, which provides a better understanding of the magnitude of prediction error. It makes it easier to interpret the error and compare it with actual stock prices.
- **Mean Absolute Error (MAE)**: Calculates the average absolute difference between the predicted values and the actual values. It offers a clear measure of prediction accuracy without overly penalizing large errors as MSE does. It provides a direct interpretation of the average prediction error.
- **Mean Absolute Percentage Error (MAPE)**: Expresses the prediction error as a percentage of the actual values. It is particularly useful to understand the error

magnitude in relative terms, making it easier to compare across different datasets or models. This is helpful when dealing with datasets that have a wide range of values.

- **R-Squared (R^2)**: Indicates the proportion of the variance in the dependent variable that is predictable from the independent variables. An R^2 value close to 1 means that the model explains a large proportion of the variance in the target variable. It helps to understand the overall explanatory power of the model [10].

5.1.2 Random Train-Test Split

The first linear regression model we try involves splitting our dataset into training and testing sets using a random train-test split. We use the `train_test_split` function from the `scikit-learn` library [34] to randomly divide the data, specifically 80% used as training set and the remaining 20% is reserved for testing.

Once the data is split, we create a pipeline that includes a feature scaler called `RobustScaler`, also from `scikit-learn`, and the linear regression. We decided to use the `RobustScaler` to preprocess our dataset since the data is not normally distributed, as we previously saw. This scaler is extremely useful in such scenarios because it employs robust statistics (median and interquartile range) to center and scale the data. Its resilience to outliers ensures that our model is not disproportionately affected by extreme values.

The pipeline is trained on the training data and then predictions are made on the test set. We finally evaluate the model using the aforementioned metrics to understand its performance.

Table 5.1: Model evaluation metrics for random train-test split.

Metric	**Value**
Mean Squared Error (MSE)	26123.87
Root Mean Squared Error (RMSE)	161.63
Mean Absolute Error (MAE)	62.24
Mean Absolute Percentage Error (MAPE)	118.79%
R-Squared (R^2)	0.29

Directly interpreting the metrics in Table 5.1 can be challenging due to the variability in stock prices, as seen in our initial analysis of price distributions. However, after adjusting with the `RobustScaler`, the price scale is similar to typical financial metrics and ranges seen in the stock market, so the magnitudes of these errors are quite significant (as underlined by an astonishing 118.79% MAPE). We expected this result because when dealing with financial data the relationship between stock prices and the features used in the model is often not linear. Financial markets are influenced by a huge amount of factors, many of which interact in complex and non-linear ways. This makes it difficult for a simple linear regression model to capture all the dynamics that influence stock prices.

5.1.3 Time-Based Train-Test Split

Next, we train our model on data from 2010 to 2018 and test it on data from 2018 to 2022. Splitting the dataset based on time helps us understand how well the model can predict future stock prices using past data. This method mimics real-world scenarios where models are used to forecast future outcomes based on historical data. After evaluating the model, we get the results shown in Table 5.2.

Table 5.2: Model evaluation metrics for time-based train-test split.

Metric	**Value**
Mean Squared Error (MSE)	72603.43
Root Mean Squared Error (RMSE)	269.45
Mean Absolute Error (MAE)	97.03
Mean Absolute Percentage Error (MAPE)	93.98%
R-Squared (R^2)	0.19

After splitting the data based on time, we observe a noticeable decline in model performance, as shown by higher error metrics and a lower R-squared. This drop in performance can be due to several reasons. First, as mentioned previously, stock prices are affected by many factors, like economic indicators, company performance and market sentiment. These relationships are usually non-linear and complex, while linear regression assumes a linear relationship between input variables and target [22]. Then, events such as financial crises, changes in government policies or shifts in technology can cause market volatility. These events can change the behavior of stock prices, so patterns in the training set might be less relevant in the testing set [41]. The model might also have overfitted to specific trends in the training data that do not apply well to future data. This is particularly true for long-term data: past trends might not be relevant to the current market. One example of this could be the COVID-19 pandemic, which had a distinct and significant impact on financial markets, leading to unexpected volatility and economic challenges. Lastly, financial time series data often show non-stationary behaviors: their mean, variance and covariance change over time. Models built on non-stationary data will perform poorly because the statistical properties of the target variable change in the test set.

5.1.4 Analysis of High and Low Priced Stocks

To understand the effect of stock price levels on the performance of linear regression models, we segment stocks into high-priced and low-priced groups based on their price quantiles. We then train separate models for each group to compare their performance.

First, we calculate the quantile thresholds to define high-priced and low-priced stocks. Stocks above the 75th percentile are considered high-priced, while those below the 25th percentile are considered low-priced. Running a linear regression on each segment separately yielded the performance metrics shown in Table 5.3.

Table 5.3: Comparison of model performance for high-priced and low-priced stocks.

Metric	**High-Priced Stocks**	**Low-Priced Stocks**
MSE	48939.32	54.15
RMSE	221.22	7.36
MAE	101.86	6.11
MAPE	44.88%	38.74%
R^2	0.56	0.26

The model for high-priced stocks shows an R-squared value of 0.56, indicating it explains approximately 56% of the variability in the stock prices. The RMSE and MAE values are quite high, but reflect the higher range of stock prices. The MAPE of 44.88% indicates that there is still significant room for improvement in predicting high-priced stocks, given the typical market volatility.

For low-priced stocks, the model yields an even lower R-squared value of 0.26, meaning it only explains about 26% of the variance in stock prices. The lower RMSE and MAE are expected and reflect the lower absolute scale of stock prices in this segment. However, the higher MAPE of 38.74% suggests that linear regression is performing poorly even for this segment. The low R-squared value indicates that the model is not capturing much of the variability in the prices and tells that other factors not included in the model might be influencing these stocks more heavily, or that their price movements are more random.

The R-squared values indicate that the model fits high-priced stocks better than low-priced stocks. The percentage errors (MAPE) for both segments are similar, but the higher R-squared value for high-priced stocks means the model explains more of the variance for this group. The inconsistency in model fit for the two segments can be due to the different behaviors and dynamics of high-priced and low-priced stocks. High-priced stocks are often more established and might have more predictable patterns influenced by fewer, more affecting factors that are well captured by the model. In contrast, low-priced stocks could be influenced by many more market factors, making them more volatile and harder to predict with the given model features. Additionally, the dataset contains a way larger number of low-priced stocks, so this increased variability might be another cause of the lower R-squared value. While the overall prediction errors are somewhat comparable for the two segments, the model's explanatory power is significantly better for high-priced stocks.

5.2 Lasso Regression

To try to improve linear regression, we employ Lasso regression, short for Least Absolute Shrinkage and Selection Operator [16, pp. 130–131]. It is an extension of linear regression that includes regularization to prevent overfitting. It adds a penalty equal to the absolute value of the magnitude of the coefficients to the loss function, which can shrink some coefficients to zero. This makes lasso regression useful not only for prediction but also for identifying and selecting important features in the model, while discarding irrelevant ones. This can reduce complexity and lead to better generalization and more robust predictions.

The lasso regression equation is the same as linear regression, but adds a regularization term to the loss:

$$\text{Loss} = \text{RSS} + \lambda \sum_{j=1}^{n} |\beta_j|$$

where RSS is the residual sum of squares and λ is the regularization parameter that controls the strength of the penalty.

In our case, we use the `LassoCV` model to identify and select the most important features from the dataset. This results in the exclusion of only a few features, meaning that most of the features are considered relevant by the algorithm. Then, we train a standard linear regression model using only the selected features, with the resulting performance metrics presented in Table 5.4.

Table 5.4: Lasso regression performance metrics.

Metric	**Value**
Mean Squared Error (MSE)	72334.12
Root Mean Squared Error (RMSE)	268.95
Mean Absolute Error (MAE)	97.37
Mean Absolute Percentage Error (MAPE)	96.58%
R-Squared (R^2)	0.2

The model performs almost identically to linear regression trained on all the features. This means that, in this case, removing less relevant features does not really impact the model's predictive ability, most likely because the original features were already optimal for prediction.

Chapter 6

Long Short-Term Memory (LSTM) Networks

6.1 Neural Networks

After analyzing linear regression, we noted how these models are straightforward and easy to employ, but they are also unable to capture non-linear relationships and are extremely sensitive to market volatility and non-stationary data. This is why we turn to more advanced models: neural networks [16, pp. 253–263].

Neural networks are a class of machine learning models designed to simulate the behavior of the human brain. This allows them to recognize patterns and model complex, non-linear relationships, which perfectly adapts to the dynamism of financial time series. A neural network consists of layers of interconnected nodes (or neurons), which process input data and learn patterns through training. They generally include an input layer, which receives the initial data, one or more hidden layers, performing computations through their activation functions, and an output layer, which produces the final output of the model.

6.1.1 Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) are a special type of neural network designed for sequential data tasks, such as time series forecasting, language modeling and speech recognition. Unlike traditional neural networks, RNNs have connections that loop back on themselves (as shown in Figure 6.1), which allows them to maintain a memory of previous inputs. This helps RNNs capture temporal dependencies over time [16, pp. 379–383].

However, standard RNNs face challenges in learning long-term dependencies due to issues like vanishing and exploding gradients. These might occur during backpropagation, an algorithm which trains neural networks by propagating the error from the output layer back through the network, using gradient descent to minimize the error. Vanishing and exploding gradients happen when gradients either become too small, causing the model to stop learning, or too large, resulting in unstable updates and numerical overflow. These problems make it tough for RNNs to effectively learn and retain information over long sequences.

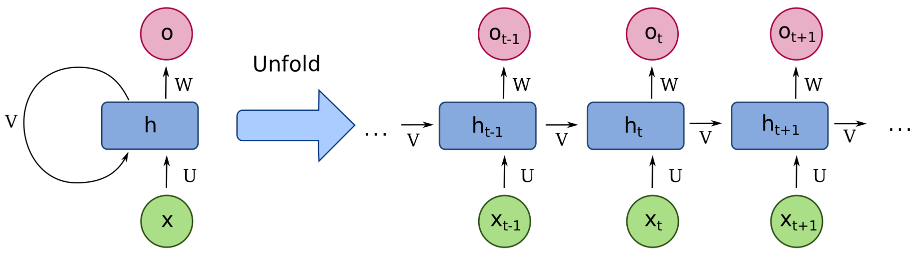


Figure 6.1: Structure of a Recurrent Neural Network (RNN). x represents the input vector, h the hidden state (the "memory" of the network), o the output vector and U, V, W are weight matrices. The right part shows the network unfolded over time.

6.2 Introduction to LSTM Networks

Long Short-Term Memory (LSTM) networks were developed to address these limitations [20]. LSTMs are a type of RNN designed to overcome the vanishing gradient problem and effectively capture long-term dependencies. The name "Long Short-Term Memory" reflects the ability of the network to remember information over long sequences while also effectively managing short-term memory. They achieve this through a sophisticated mechanism which retains and updates memory dynamically. This allows LSTMs to remember and forget information selectively, preserving important data and discarding irrelevant data.

Given this explanation, the choice of LSTM networks is clearly justified. We expect LSTMs to excel in the prediction of stock prices, as they can learn from historical price data and capture intricate patterns that influence future prices. Using these architectures we aim to improve the predictive performance drastically compared to linear regression models.

6.2.1 Structure of LSTMs

LSTMs achieve their effectiveness through a more complex structure, that includes memory cells and gating mechanisms.

Memory cells are a key component of LSTM networks. They store relevant information for extended periods, allowing the network to keep a "memory" of previous inputs over time. Gating mechanisms regulate the flow of information into and out of the memory cell, thus allowing the network to remember or forget information. LSTMs use three types of gates:

1. Input Gate: Determines which new information is added to the memory cell.
2. Forget Gate: Decides which information in the memory cell should be forgotten.
3. Output Gate: Controls what information is read from the memory cell and used for the current output.

These gates are controlled by learned weights and biases, which allow the network to decide the importance of information dynamically as it processes the sequence.

Figure 6.2 illustrates the structure of an LSTM cell. The diagram shows the flow of information through an LSTM cell, including the input gate, forget gate and output gate. The input gate is represented by the first sigmoid (σ) function on the left, the forget gate by the second sigmoid (σ) function, and the output gate by the sigmoid (σ) function on the right. These gates control the flow of information into and out of the cell state (c_t) and hidden state (h_t), enabling the LSTM to maintain and modify the memory over time.

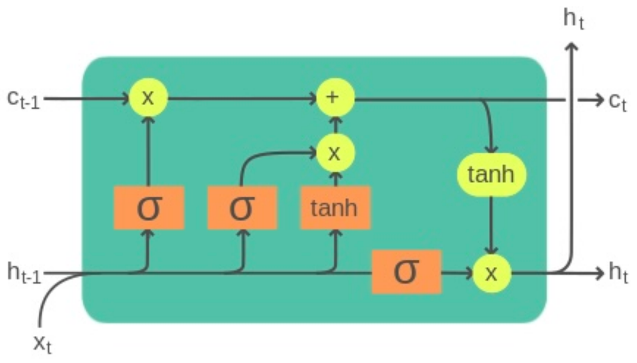


Figure 6.2: Structure of an LSTM cell.

6.2.2 Mechanism of LSTMs

At each time step in the sequence, the LSTM network processes an input and combines it with the previous hidden state to decide what information to update, forget and output. This is achieved through a series of gating mechanisms. The input gate updates the memory cell with the new information, while the forget gate removes irrelevant information. Then, the output gate determines the current output based on the updated memory cell. This process allows LSTMs to maintain and update their memory, effectively learning long-term dependencies.

6.3 Single Layer LSTM with Single Asset Price

We begin the neural network approach with the base LSTM case: predicting the price of a single asset using only its historical price data. To keep the model simple and to understand its basic capabilities, we will use a single LSTM layer.

6.3.1 Choice of Single Asset

For this analysis, we specifically select the Johnson & Johnson (JNJ) asset from our dataset. Johnson & Johnson is a stable and well-established company within the S&P 500, known for its consistent performance and significant market presence. This makes JNJ an ideal candidate for our initial model analysis, the stability of this asset gives a clear benchmark for the evaluation of the predictive capabilities of our model.

We begin by visualizing the price trend of Johnson & Johnson, to understand its historical performance and identify any noticeable patterns or anomalies.

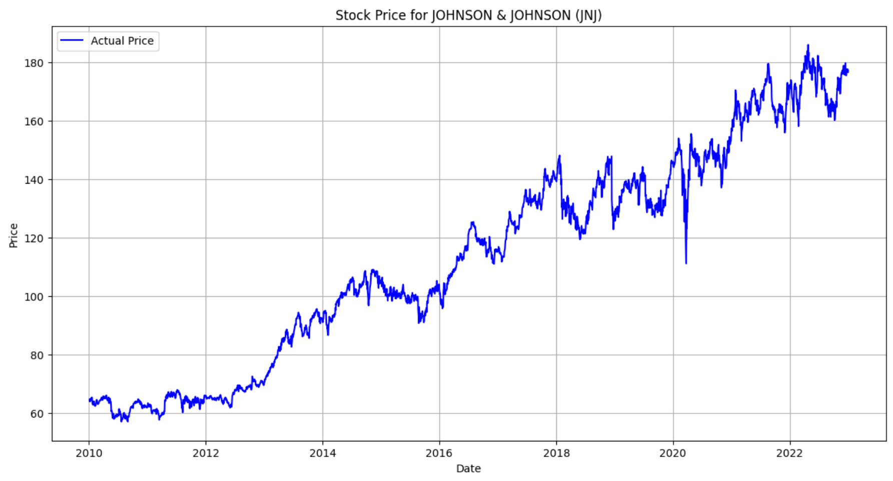


Figure 6.3: Historical stock price of Johnson & Johnson (JNJ) from 2010 to 2022.

The plot shows an upward trend indicating a general increase in stock price, which reflects the company's growth and stability in the market. There are however notable fluctuations, such as the sharp decline around 2020, most likely corresponding to market-wide impacts like the COVID-19 pandemic. For this reason, despite JNJ's stability, its intrinsic financial volatility makes price predictions not straightforward.

6.3.2 Data Preprocessing

Training an LSTM model requires careful preparation of time-series data. This involves several steps. To begin, as usual, we split the dataset into training and test sets (we use 67% of the data for training and the remaining 33% for testing). Then, we need to perform normalization, as LSTM models are particularly sensitive to the scale of input data. Here, we use `MinMaxScaler` which standardizes stock prices to a scale of 0 to 1. Lastly, since LSTM models expect inputs in the form of sequences, we convert the series into overlapping sequences (windows) of fixed size. Each sequence serves as input to the LSTM and the model learns to predict the value at the next time step. The `lookback` parameter defines the number of previous time steps used to predict the next time step. In our case, we chose a lookback of 10. To build our model, we will use PyTorch, a commonly used open-source machine learning library particularly used for deep learning tasks [33]. PyTorch requires input data in the form of Torch tensors, thus the final step is converting the sequences into tensors.

6.3.3 Building the Model

To build our LSTM model, we define a class that contains a simple LSTM network for time series prediction. This class initializes with a single LSTM layer followed by a linear layer:

- LSTM Layer: This layer takes sequences with a single feature (`input_size=1`) and processes through 50 hidden units (`hidden_size=50`).
- Linear Layer: After the LSTM processes the input data, the output passes through a linear (fully connected) layer, which maps the hidden state at the last timestep to a single output value, representing the predicted value at the next timestep.
- Forward Pass: In the forward method, the input data is passed through the LSTM layer, followed by the linear layer. Then, only the output from the linear layer at the last timestep is selected for each sequence in the batch.

6.3.4 Training and Testing

At this point, we are ready to train the LSTM model and subsequently test it. The model is trained over 100 epochs with a batch size of 4. As optimizer, we chose Adam (Adaptive Momentum) [23], which combines the advantages of two other popular optimizers: AdaGrad [15] and RMSProp [19]. It adjusts the learning rate for each parameter dynamically based on estimates of first and second moments of the gradients. This helps with faster convergence and improved performance, especially for large datasets and complex models. As loss function, the standard Mean Squared Error (MSE) is employed to measure the difference between predicted and actual prices.

The training and validation process works as follows. Data is loaded in minibatches using `DataLoader`, which shuffles the training data to ensure random sampling. Then, the data enters a training loop, consisting of:

- Forward Pass: Making predictions for the current batch of input data.
- Loss Calculation: Computing the difference between predictions and actual data with MSE.
- Backpropagation: Calculating gradients based on the loss and adjusting model weights to minimize the loss using the Adam optimizer.

To monitor the performance of the model, at the end of each epoch the model switches to evaluation mode, to calculate the RMSE (Root Mean Squared Error) for both the training and testing datasets. Additionally, we will plot the RMSE values over epochs to visualize the error time evolution and better understand the training process. This training and validation process yielded the results shown in Table 6.1.

Table 6.1: Evolution of RMSE during training.

Epoch	**Train RMSE**	**Test RMSE**
0	0.0205	0.0976
20	0.0102	0.0265
40	0.0131	0.0267
60	0.0105	0.0283
80	0.0100	0.0252
100	0.0097	0.0253

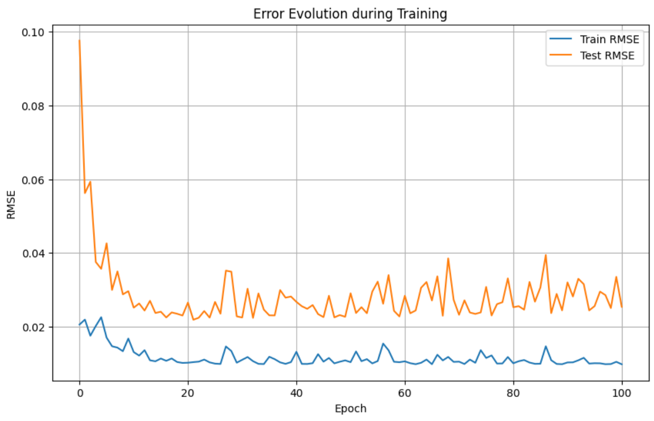


Figure 6.4: Error evolution during training.

The graph in Figure 6.4 shows that both training and test errors undergo an initial decline, with test RMSE having a significant drop. After epoch 20, the errors seem to stabilize at low enough values, meaning that the model begins to capture the underlying patterns in the data. However, as training progresses, there still are noticeable fluctuations, especially in the test RMSE, due to variability in the performance of the model on different batches of data. The final RMSE values, respectively 0.0097 (train) and 0.0253 (test) after normalization between 0 and 1, still show the model's high accuracy. These values are relatively small, suggesting that the model predictions are very close to the true values.

6.3.5 Visualization of Predictions

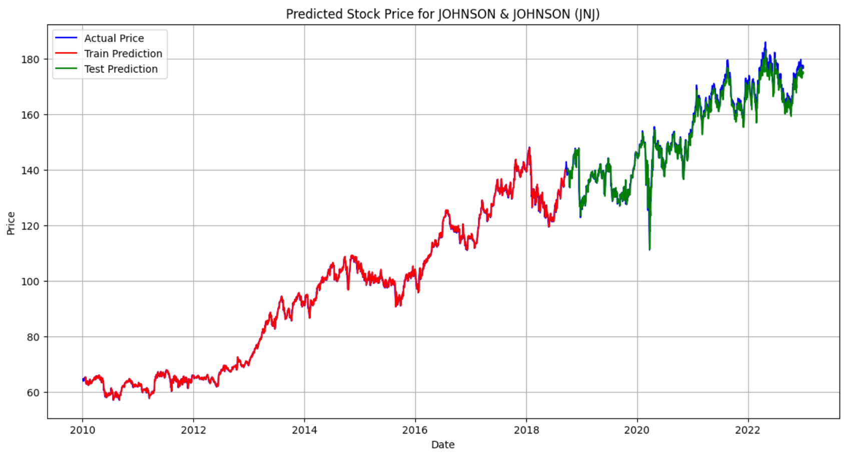


Figure 6.5: Predicted stock price for Johnson & Johnson (JNJ) using LSTM model. The blue line represents the actual stock prices, the red line shows the train predictions, and the green line depicts the test predictions.

Figure 6.5 illustrates the performance of the LSTM model in predicting the stock price of Johnson & Johnson (JNJ). The actual stock prices are represented by the blue line, the train predictions by the red line, and the test predictions by the green line. The plot shows the incredible predictive capabilities of the LSTM model. The red line (train prediction) closely follows the actual stock prices (blue line), which indicates that the model has learned effectively from the training data. In the test phase, the green line closely mirrors the actual price trends, even though the model has not seen this data before. Despite some minor mismatches towards the end, the test predictions track the overall trend of the actual prices well, meaning the model can generalize well to unseen data.

6.4 Enhancements to LSTM Models

To improve the performance and stability of our LSTM models, we introduce two techniques aimed at solving common issues encountered during the training of deep neural networks, in particular exploding gradients and overfitting.

6.4.1 Gradient Clipping

Gradient clipping is a technique used to address the issue of exploding gradients in training deep neural networks, particularly LSTMs [32]. When training involves processing through numerous layers, gradients can sometimes grow exponentially during backpropagation, causing weight updates to be excessively large. This can result in numerical instability and slow down convergence.

We introduce gradient clipping to maintain computational stability and improve the convergence rate during training. This technique caps the gradients at a threshold (1.0 in this case), to prevent them from becoming too large. This, combined with an increased batch size of 32, allows us to increase the number of training epochs from 100 to 500.

6.4.2 Early Stopping

Early stopping is a regularization technique used to prevent overfitting in deep learning models [36]. It monitors the performance of the model on a validation dataset during training and halts the training process if the model's performance stops improving or begins to deteriorate. This not only saves computational resources but also ensures that the model does not learn noise or overly complex patterns from the training data that do not generalize well to unseen data.

Here, we use early stopping as a safeguard against overfitting, especially given the complexity of LSTM architectures. By setting a patience parameter, the training process is stopped after a specified number of epochs if there is no improvement in the prediction error on the validation set. This balance between learning from the data and maintaining generalization ability improves the model's performance on unseen data. Here, the patience parameter is set to 40 epochs.

6.4.3 Performance Evaluation

We will now again analyze the RMSE evolution in time and predictions after applying gradient descent, increasing the batch size to 32 and including early stopping.

Table 6.2: Evolution of RMSE during training with gradient clipping, early stopping and increased batch size.

Epoch	**Train RMSE**	**Test RMSE**
0	0.0634	0.1236
20	0.0168	0.0392
40	0.0235	0.0362
...	...	...
120	0.0101	0.0219
140	0.0125	0.0336

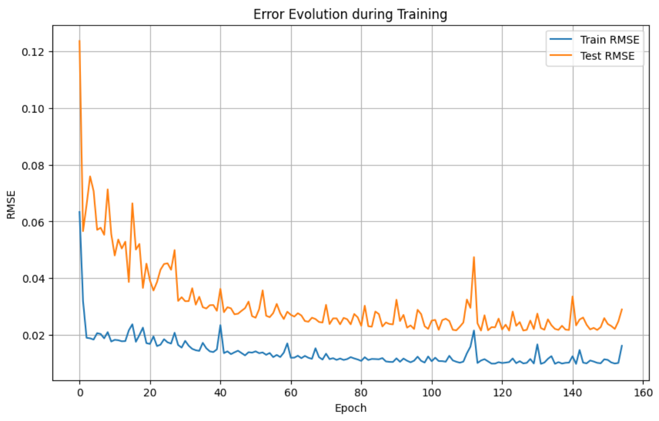


Figure 6.6: Error evolution during training with gradient clipping, early stopping and increased batch size.

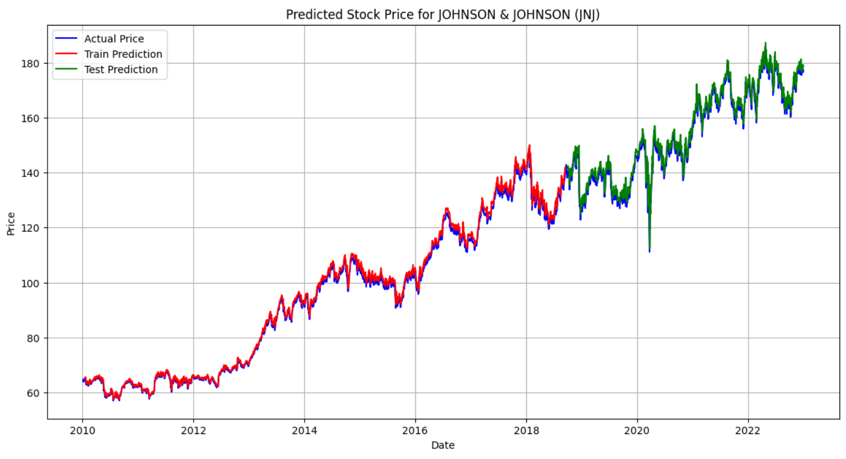


Figure 6.7: Predicted stock price for Johnson & Johnson (JNJ) using LSTM model with gradient clipping, early stopping and increased batch size.

The enhancements techniques we applied proved to be effective. The performance of the LSTM model shows notable improvements in terms of stability and convergence speed. The RMSE values at various epochs are similar but more stable compared to the previous results. Even though the initial test RMSE was higher (0.1236), it quickly dropped to more stable and lower values as training progressed. By epoch 140, the test RMSE was 0.0336, and early stopping was triggered at epoch 154 with a test RMSE of 0.0290, indicating that the model had reached its optimal performance without overfitting. Overall, the RMSE value was similar for training data but slightly higher for testing data. This discrepancy could be due to the intrinsic noise in financial data, as well as the early stopping mechanism, which does not account for error fluctuations within the set patience epochs. Still, the computational and stability benefits of these techniques make it more convenient to include them in the model.

6.5 Incorporating Multiple Variables

Predicting stock prices using a single variable (historical prices) yielded impressive results. We now want to analyze the impact of using multiple variables on the predictive accuracy of our LSTM model. By integrating additional features, such as trading volume, book-to-market ratios and macroeconomic indicators, we would like to capture a wider range of market dynamics and identify more complex patterns in the data.

6.5.1 Building the Multivariate LSTM Model

To incorporate multiple variables, however, we need to adapt some of the procedures. First of all, the sequence creation process should be modified to work with multiple features. Specifically, the sequence creation function will now slice the input features and the target variable separately. Then, we need to build a Multivariate LSTM model. We perform a slight change in our LSTM class to accommodate multiple input features, by modifying the input size parameter of the LSTM layer to match the number of features being used, 18 in our case.

6.5.2 Performance Evaluation

By training and testing the multivariate LSTM model, we get interesting results. For brevity, we will include only the final evaluation metrics at the epoch where early stopping was triggered.

Table 6.3: Final evaluation metrics (epoch 113) of the multivariate LSTM model.

Metric	**RMSE**
Train RMSE	0.0103
Test RMSE	0.0290

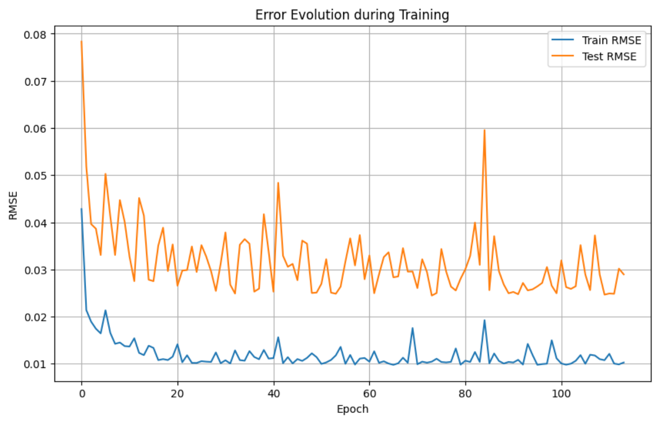


Figure 6.8: Error evolution during training of multivariate model.

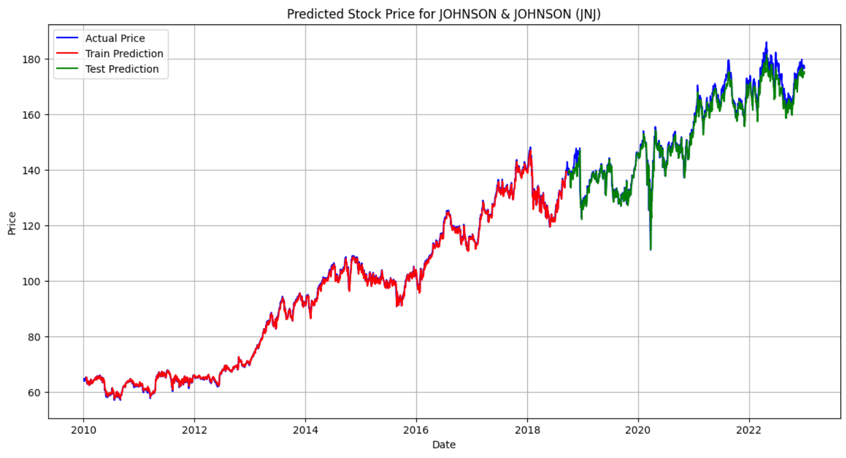


Figure 6.9: Predicted stock price for Johnson & Johnson (JNJ) using multivariate LSTM model.

The multivariate model shows some improvement in RMSE during the initial epochs compared to the univariate model. The inclusion of additional variables probably helps the model capture more complex relationships in the data, leading to faster convergence. Despite these improvements, the test RMSE shows more fluctuation in the multivariate model, possibly due to the increased complexity and noise introduced by additional features. The final test RMSE is, however, exactly the same as the univariate model. By analyzing the predictions plots in Figure 6.9, we can notice the different behavior. The predictions of the model with only price as variable is slightly misaligned compared to the actual values, but the (small) error is consistent in time, both for training and testing. On the contrary, the multivariate model predictions show great train predictions, but less accurate test predictions, supporting the idea that multiple variables might introduce additional variability.

6.6 Exploring Model Depth and Hidden Layer Size

In the previous sections, we analyzed the capabilities of a single-layer LSTM model in predicting stock prices using historical data. While the results were promising, we found that merely incorporating more features without adjusting the model architecture may not yield improved predictions. To address this, we now aim to investigate whether increasing the model's complexity can improve its predictive power. We will focus on exploring two fundamental aspects of this model's architecture: depth and hidden layer size. The detailed visualizations of error evolution and predictions for different model depths and hidden layer sizes are provided in the Appendix.

6.6.1 Varying the Model Depth

Model depth refers to the number of LSTM layers stacked in the network. Increasing the depth usually allows the model to capture more complex patterns and dependencies in the data by learning hierarchical representations. However, deeper models are also more prone to overfitting and require more computational resources. We will experiment with 2 and 3 hidden layers, while keeping 50 hidden units.

Two LSTM Layers

The two-layer LSTM model performed very similarly to the single-layer model in terms of final RMSE. The predictions produced by the two models are in fact nearly identical. However, when analyzing the error evolution during training, we observe that the convergence of the RMSE is less stable in the two-layer model. The test RMSE struggles to converge in the initial epochs and shows greater fluctuations compared to the single-layer model. This instability might be due to the increased complexity of the two layers, which introduce more parameters and can make the optimization process more challenging.

Three LSTM Layers

When implementing three layers, the differences become more evident. As expected, the added complexity helped maintain a very low train RMSE, being able to capture more patterns in the data. Yet, this also led to overfitting, yielding a significantly higher initial test RMSE. Although the test RMSE decreased rapidly with training epochs, the larger error persisted, causing less accurate predictions on the test set, as seen in the plot. While adding more layers has the potential to capture more intricate patterns in the data, it requires more careful tuning to ensure stable convergence and prevent overfitting.

6.6.2 Varying the Hidden Layer Size

The hidden layer size in an LSTM network refers to the number of units within each layer. Adjusting the size of these hidden layers directly influences the model's capacity to learn and represent intricate patterns in the data. Larger hidden layers should technically capture more detailed and complex relationships, potentially improving the predictive performance of the model. However, as with increasing model depth, it also raises the risk of overfitting and demands more computational resources. We will compare single-layer models with hidden layer sizes of 100 and 150 units and evaluate their impact on prediction performance.

100 Hidden Units

The model with a hidden layer size of 100 performs similarly to the one with 50 hidden units, with comparable error scales. However, there are some differences. The training RMSE is noisier and shows greater difficulty in converging, while the test RMSE is significantly more stable compared to the 50 hidden units model. This stability is reflected in a lower test error and more accurate prediction of stock prices on the test set. Increasing the number of hidden units to 100 can improve generalization and provide more reliable predictions, but it might also introduce more noise during the training process.

150 Hidden Units

While increasing the number of layers to 100 seems to perform better on the testing data, raising the hidden units to 150 introduces much higher variability in the error. Both the training and testing RMSE show extreme fluctuations throughout the entire training process and struggle to reach an equilibrium value. Despite this instability, the predicted prices are reasonably accurate. This can be attributed to early stopping, which fortunately halted the training at a point where the evaluation metrics were sufficiently small, in between peaks. Choosing the right number of hidden units can enhance the model's ability to capture complex patterns, but excessively high numbers can lead to instability and reduced performance.

6.7 Testing on Other Assets

In order to further evaluate the robustness and generalizability of our LSTM models, it is essential to test their performance on a variety of different assets, focusing on those representing extreme cases. Specifically, we will use assets with the highest and lowest prices as well as those with the highest and lowest trading volumes. These assets might indeed exhibit unique market behaviors and patterns, so testing on these extremes can help us assess how well the model adapts to different market conditions and whether it can handle the variability inherent in financial data. This ensures that the LSTM model is not only working well on average and stable cases but is also robust enough to provide reliable predictions in different scenarios. We aim to mimic real-world situations, where the model will need to predict asset prices in dynamic and often unpredictable market environments. It is important to note that we will choose the extreme assets based on average prices and average trading volumes, to avoid considering stock splits or similar corporate actions. A stock split increases the number of shares outstanding while decreasing the price per share, making the stock more affordable for investors without changing the company's overall market value. This often results in a significant drop in the historical price chart and a sudden exorbitant increase in trading volume, which is however artificial and not reflective of the usual performance of the stock.

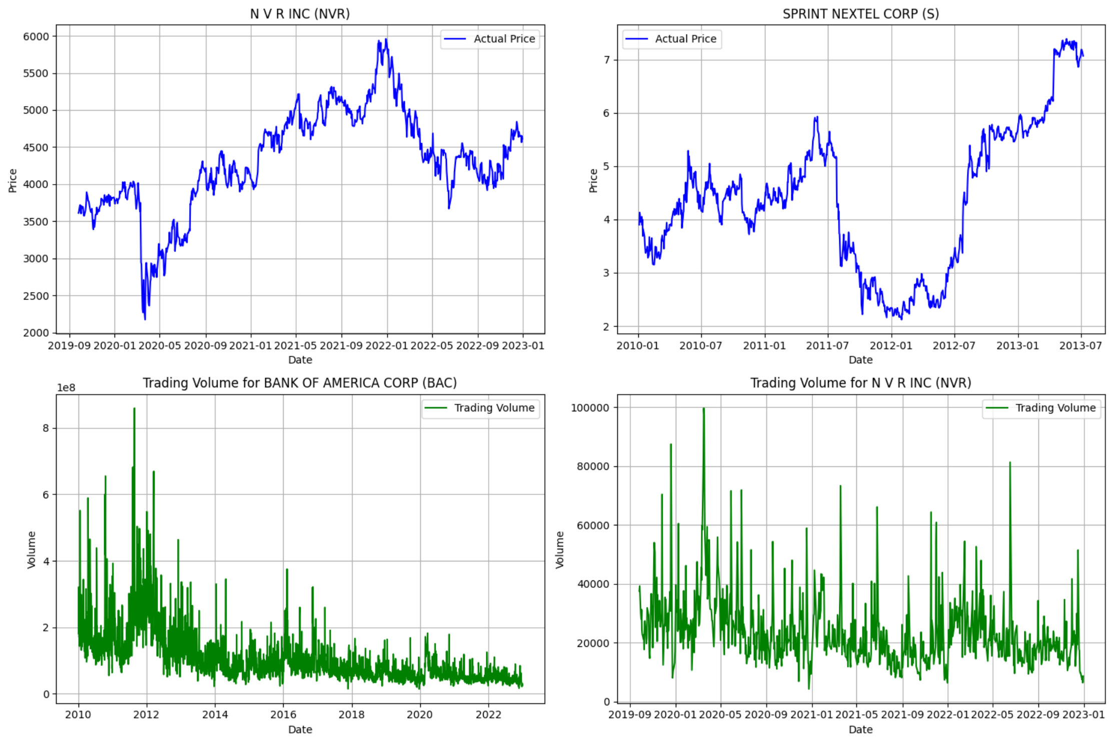


Figure 6.10: Price and trading volume of selected stocks with extreme values.

- **Stock with the highest price and the lowest trading volume: NVR Inc. (NVR)**

  It is not surprising that the stock with the highest average share price, NVR Inc., also has the lowest trading volume. Due to the high price, indeed, fewer investors might be able to afford to trade such expensive shares frequently. NVR Inc. has consistently high stock prices, reaching a staggering price of almost $6000 per share. This stock has however significant fluctuations throughout the whole time period, which is critical given the large scale of its price. Lastly, it should be noted how NVR Inc. was only added to the S&P 500 index in 2019, reducing the temporal landscape available for analysis [31]. Therefore, we do not expect extremely accurate predictions of this asset.

- **Stock with the lowest price: Sprint Nextel Corp. (S)**

  Sprint Nextel Corp. shows a relatively lower stock price during its entire permanence in the S&P 500, with prices fluctuating between $2 and $7. The company was however delisted from the S&P 500 on July 8, 2013, due to a significant acquisition by SoftBank Corp. [30]. This acquisition resulted in Sprint's public float falling substantially below the 50% public float criteria required for inclusion in the S&P 500 index. Similarly to NVR Inc., the reduced time period will probably result in less accurate price predictions.

- **Stock with the highest volume: Bank of America Corp. (BAC)**

  Bank of America Corp. has the highest average trading volume, with peaks around the early 2010s of almost a billion shares traded in a day. This likely corresponds to the period following the financial crisis when there was heightened activity in the financial markets. Over time, the volume shows a decreasing trend but remains relatively high, suggesting that Bank of America continues to be a popular stock among investors. The consistent trading volume has a significant impact on prices, as it allows for easier buying and selling of the stock without causing remarkable price changes.

6.7.1 Highest Priced and Lowest Trading Volume Stock (NVR Inc.)

The NVR stock exhibited extremely close training and testing RMSE values throughout the epochs (Figure 6.11a). Both metrics started relatively high but rapidly decreased, stabilizing around 0.03 to 0.04, higher than the values observed for Johnson & Johnson. Consequently, the model's predictions for NVR stock were fairly accurate, capturing the overall trend effectively despite not being perfectly precise (Figure 6.11b). The slight mispredictions were expected, given the analysis period extends a bit over three years.

6.7.2 Lowest Priced Stock (Sprint Nextel Corp.)

In this case, the test RMSE was quite higher than the train RMSE, stabilizing around 0.1 after quickly decreasing from the initial 0.5 (Figure 6.11c). The model's predictions for Sprint Nextel Corp., while not flawless, consistently captured the overall trend (Figure 6.11d). Especially in the test set, the model had a slight tendency to undervalue the asset, but still the predictions accurately mirrored the actual price behavior, capturing both upward and downward trends. It is also important to note that, given the much lower scale of the prices, these inaccuracies have a relatively minor impact. The lower absolute prices mean that percentage errors result in smaller absolute deviations, making the model's performance more acceptable in practical terms.

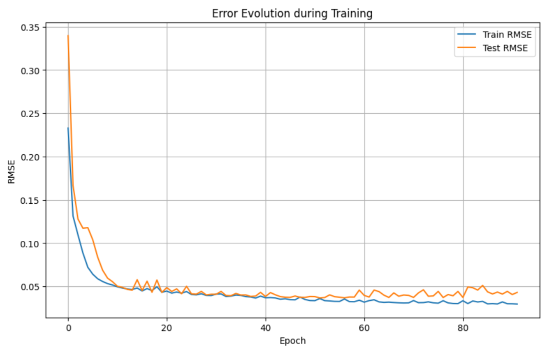
6.7.3 Highest Trading Volume Stock (Bank of America Corp.)

Bank of America shows an outstanding performance, comparable to that of Johnson & Johnson. Initially, the test RMSE increases during the early epochs, but it subsequently decreases and reaches a relative stability (Figure 6.11e). Although the test RMSE remains higher than the train RMSE, it is still quite low, indicating a high level of prediction accuracy, as evident from the plot (Figure 6.11f). This strong performance can be attributed to the well-established and stable nature of Bank of America, as reflected in its consistently high trading volumes. The stability and robustness of the company make its stock price movements more predictable, allowing the LSTM model to capture the underlying patterns effectively. Furthermore, Bank of America was continuously present in the S&P 500 index from 2010 to 2022, providing a larger time frame for training the model and exposing it to additional market events and economic conditions.

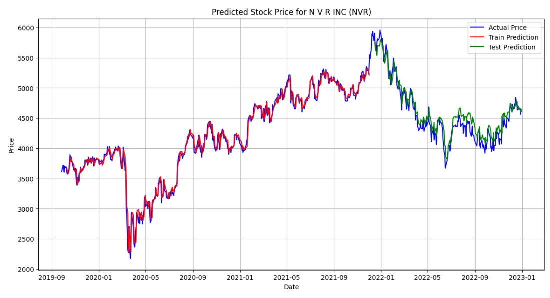
6.7.4 Summary of Extreme Asset Cases

Overall, the LSTM model demonstrated robust performance across different market conditions. The model managed to predict trends accurately for high-priced stocks like NVR Inc., despite some prediction inaccuracies due to the limited data period available. Similarly, for low-priced stocks such as Sprint Nextel Corp., the model captured overall price trends effectively, even though there was an apparent pattern of underconfidence. The analysis on Bank of America showed that the model can indeed generalize well to different market conditions, but also underscored the importance of having data spanning a longer period of time for more accurate predictions.

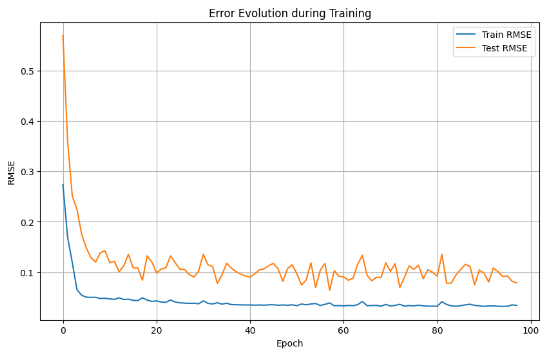
These results illustrate that the LSTM model can learn intrinsic characteristics of the data and adapt to various market scenarios, whether dealing with time periods right before delisting, stocks with incredibly low or high prices, or those with significant trading volumes. However, the performance is still influenced by the duration of available data, with shorter time periods presenting challenges for the model's predictive accuracy.



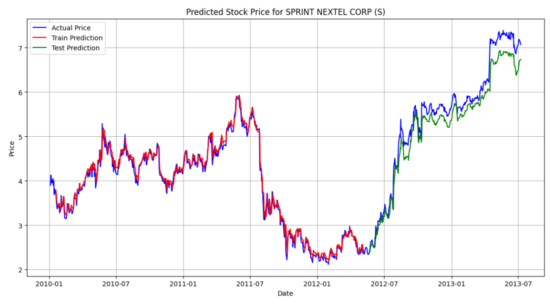
(a) Train and test error for NVR (high price, low trading volume).



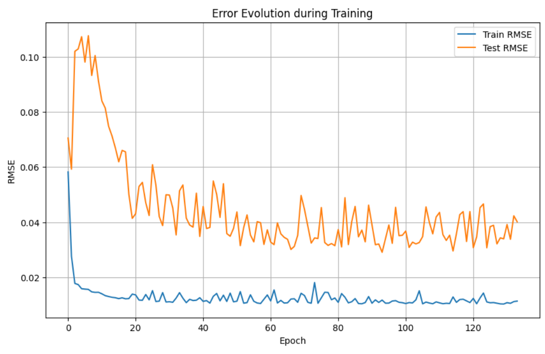
(b) Predictions for NVR (high price, low trading volume).



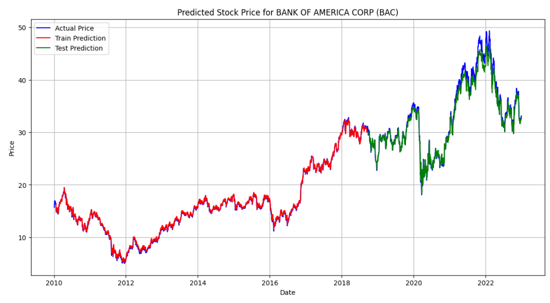
(c) Train and test error for Sprint Nextel (low price).



(d) Predictions for Sprint Nextel (low price).



(e) Train and test error for Bank of America (high trading volume).



(f) Predictions for Bank of America (high trading volume).

Figure 6.11: Comparison of RMSE and predictions over time for different stocks.

Chapter 7

Conclusion and Future Research

7.1 Summary of Findings

Throughout this thesis, we investigated the application of Long Short-Term Memory (LSTM) networks for stock price prediction, focusing on both univariate and multivariate approaches, while also exploring the impact of varying model depth, hidden layer sizes and market conditions. The primary objective was to determine whether LSTM models could outperform traditional linear regression techniques in forecasting stock prices and to understand the complexities of optimizing LSTM architectures for financial time series data.

We first implemented a univariate LSTM model, using only historical stock prices to predict future prices. The initial model with a single LSTM layer showed significant improvements over linear regression, with lower RMSE values and better trend prediction. Moreover, the model's performance showed a rapid decrease in error during the initial epochs and stabilization at low values, indicating an effective learning of temporal dependencies.

To address common training issues such as exploding gradients and overfitting, we incorporated gradient clipping, an increased batch size and early stopping. These enhancement techniques stabilized the training process and improved convergence. The final univariate model showed robust performance with stable and low RMSE values, especially benefiting from the early stopping mechanism, which prevented overfitting.

We extended the univariate approach by incorporating additional variables such as trading volume and financial ratios into a multivariate LSTM model. The inclusion of these variables aimed to capture more complex market dynamics. While the multivariate model showed initial improvements in RMSE, it also introduced more noise, leading to greater fluctuations in test RMSE. Despite this, the model provided a clear understanding of market behaviour, demonstrating a trade-off between model complexity and stability.

We then systematically explored the impact of varying the depth (number of LSTM layers) and hidden sizes on model performance. Increasing the number of LSTM layers from one to two and three revealed that deeper models could capture more intricate patterns but were more prone to overfitting and instability. The two-layer model showed similar final RMSE values to the single-layer model but had more fluctuations during training. The three-layer model demonstrated significant overfitting, with high initial test RMSE that decreased over time but remained higher than simpler models. Adjusting the hidden layer size to 100 and 150 units showed that larger hidden layers could capture more relationships in the data. The model with 100 hidden units performed similarly to the 50-unit model but with more stability in the test RMSE. However, the 150-unit model introduced significant variability, indicating the challenges of balancing complexity and stability.

To evaluate the robustness of the LSTM models, we tested them on assets with extreme characteristics, such as highest and lowest prices and volumes. For NVR, company with the highest prices but also the lowest trading volume, the model performed well in capturing the overall trend but had some prediction inaccuracies due to the limited data period. In the case of Sprint Nextel, which had the lowest prices, the model accurately captured price trends despite slightly undervaluing the asset in the test set. Lastly, regarding Bank of America, the model showed outstanding performance with stable and low RMSE, reflecting the stability and high trading volume of the stock. These tests demonstrated the model's ability to adapt to different market conditions, though they also highlighted the impact of shorter data periods on prediction accuracy.

Overall, this thesis showed that LSTM models, particularly with the right enhancements and multivariate inputs, significantly improve stock price predictions over traditional methods. The findings show that fine-tuning models and including multiple variables are crucial for capturing complex market behaviors, but also come with challenges related to overfitting and data variability in financial forecasting. These results provide a solid foundation for further research and practical applications in financial modeling and investment strategies.

7.2 Implications for Investors and Policymakers

The findings of this study have important implications for investors. Using LSTM models can significantly improve the accuracy of stock price predictions. As seen, these models can capture complex, non-linear patterns in historical price data, offering a powerful tool for developing more effective investment strategies. This can result in better returns and more efficient risk management, especially in volatile markets. Including additional variables like trading volume and financial ratios can further enhance the model's predictive abilities, allowing investors to make more informed decisions based on additional market indicators.

For policymakers, understanding the effectiveness of advanced machine learning models like LSTM networks in financial markets is crucial. This knowledge can help in creating regulatory frameworks that encourage the responsible use of these technologies. Policymakers can use insights from LSTM models to monitor market trends and detect early signs of financial instability, improve market surveillance and ensure more robust financial systems. Encouraging the use of such advanced models can also promote innovation in financial analysis and risk assessment, contributing to the overall health and stability of financial markets.

7.3 Limitations of the Study

Despite the promising results, this study still has several limitations.

First of all, the analysis was restricted to stocks listed in the S&P 500, during a period from January 1, 2010, to December 31, 2022. While this time frame is extensive, it might not be fully capturing all market conditions and rare events that could affect stock prices (for instance, the dataset does not include the financial crisis of 2008). Additionally, focusing on S&P 500 stocks limits the generalizability of the findings to other indices and international markets.

Furthermore, while increasing the complexity of LSTM models showed potential benefits, it also introduced challenges such as overfitting and instability. Balancing model complexity and generalizability still remains a significant challenge, since more complex models require careful tuning and validation to avoid overfitting and ensuring stable performance.

Then, there is the problem of model interpretability. LSTM models, like many deep learning models, are often considered black boxes due to their complex internal structures. This lack of interpretability can make it difficult to understand how the model is making its predictions. In financial applications, this is a significant drawback because stakeholders, including investors and regulatory bodies, often require clear explanations for why certain predictions are made.

Lastly, training LSTM models, especially with multiple layers and large datasets, is computationally intensive. The limited computational resources available for this study restricted the number of experiments and the depth of hyperparameter tuning. More powerful computational resources would allow for more complete experimentation and potentially better model performance.

7.4 Suggestions for Future Research

While this thesis provides a solid foundation for the application of LSTM networks in financial time forecasting, there are several areas where future research could expand and improve these findings.

In the first place, as previously stated, future research could benefit from expanding the dataset to include assets beyond the S&P 500, including international stocks and different market indices. This would provide a more complete evaluation of LSTM models and their ability to generalize across various market conditions and regions. Additionally, data from time periods before 2010 should be incorporated, to train the model on different and potentially extreme market conditions, thereby improving its performance on unseen future data.

Moreover, other advanced architectures beyond LSTM should be explored, such as Transformers [44] or hybrid models that combine LSTM with convolutional neural networks (CNNs). Transformers in particular have shown promising results in capturing long-term dependencies in sequential data, because of their self-attention mechanism. Unlike traditional recurrent neural networks (RNNs) and LSTMs, which process data sequentially, transformers can process all elements of the sequence simultaneously, allowing them to capture dependencies between distant elements more effectively. This could make them well-suited for financial time series forecasting.

In addition, future work could aim at developing real-time trading strategies based on LSTM predictions and evaluating their performance in live market conditions. This would be done by integrating LSTM models into trading platforms and continuously updating them with new data to ensure their predictions remain relevant and accurate.

It was mentioned in the previous section how deep learning models like LSTM are typically seen as black boxes. Some investigation should be done on improving the interpretability of LSTM models, to better understand how predictions are made. This could be performed through techniques such as attention mechanisms [4], which highlight the most important parts of the input data for each prediction, or model-agnostic methods like SHAP (SHapley Additive exPlanations) [26]. The latter explains the output of any machine learning models via a game theoretic approach that measures each player's contribution to the final outcome. Improving interpretability is crucial for building trust with stakeholders and ensuring compliance with regulatory standards.

Finally, future research should analyze methods to further enhance the robustness and stability of LSTM models. This could include regularization techniques, ensemble methods or adversarial training, which could help mitigate issues related to overfitting and model instability. Ensuring that models remain robust under various market conditions and data anomalies is essential for their practical application in real financial scenarios.

Bibliography

[1] Hervé Abdi and Lynne J. Williams. “Principal component analysis”. In: *WIREs Computational Statistics* 2.4 (2010), pp. 433–459. DOI: 10.1002/wics.101. URL: https://doi.org/10.1002/wics.101.

[2] Andrés Arévalo et al. “High-Frequency Trading Strategy Based on Deep Neural Networks”. In: *Intelligent Computing Methodologies. ICIC 2016. Lecture Notes in Computer Science.* Ed. by De-Shuang Huang, Kang Han, and Amir Hussain. Vol. 9773. Springer, Cham, 2016, pp. 424–436. DOI: 10.1007/978-3-319-42297-8_40. URL: https://doi.org/10.1007/978-3-319-42297-8_40.

[3] John O. Awoyemi, Adebayo O. Adetunmbi, and Samuel A. Oluwadare. “Credit card fraud detection using machine learning techniques: A comparative analysis”. In: *2017 International Conference on Computing Networking and Informatics (ICCNI).* 2017, pp. 1–9. DOI: 10.1109/ICCNI.2017.8123782.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. “Neural machine translation by jointly learning to align and translate”. In: *arXiv preprint arXiv:1409.0473* (2014).

[5] Turan G Bali, Robert F Engle, and Scott Murray. *Empirical asset pricing: The cross section of stock returns.* John Wiley & Sons, 2016.

[6] Söhnke M Bartram et al. “Machine learning for active portfolio management”. In: *The Journal of Financial Data Science* 3.3 (2021), pp. 9–30.

[7] Fetsje Bijma, Marianne Jonker, and Aad van der Vaart. *An Introduction to Mathematical Statistics.* Amsterdam: Amsterdam University Press, 2018, pp. 259–269.

[8] Alexei Botchkarev. “Performance metrics (error measures) in machine learning regression, forecasting and prognostics: Properties and typology”. In: *arXiv preprint arXiv:1809.03006* (2018).

[9] Wei Chen et al. “Mean–variance portfolio optimization using machine learning-based stock price prediction”. In: *Applied Soft Computing* 100 (2021), p. 106943. ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2020.106943. URL: https://www.sciencedirect.com/science/article/pii/S1568494620308814.

[10] Davide Chicco, Matthijs J Warrens, and Giuseppe Jurman. “The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation”. In: *Peerj computer science* 7 (2021), e623.

[11] CRSP. *CRSP US Stock Databases*. URL: https://www.crsp.org/.

[12] Nhan Cach Dang, María N. Moreno-García, and Fernando De la Prieta. “Sentiment Analysis Based on Deep Learning: A Comparative Study”. In: *Electronics* 9.3 (2020). ISSN: 2079-9292. DOI: 10.3390/electronics9030483. URL: https://www.mdpi.com/2079-9292/9/3/483.

[13] Xolani Dastile, Turgay Celik, and Moshe Potsane. “Statistical and machine learning models in credit scoring: A systematic literature survey”. In: *Applied Soft Computing* 91 (2020), p. 106263. ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2020.106263. URL: https://www.sciencedirect.com/science/article/pii/S1568494620302039.

[14] Daizong Ding et al. “Modeling extreme events in time series prediction”. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2019, pp. 1114–1122.

[15] John Duchi, Elad Hazan, and Yoram Singer. “Adaptive Subgradient Methods for Online Learning and Stochastic Optimization”. In: *Journal of Machine Learning Research* 12.7 (2011), pp. 2121–2159.

[16] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. Beijing, Boston, Farnham, Sebastopol, Tokyo: O’Reilly Media, 2019.

[17] Periklis Gogas, Theophilos Papadimitriou, and Anna Agrapetidou. “Forecasting bank failures and stress testing: A machine learning approach”. In: *International journal of forecasting* 34.3 (2018), pp. 440–455.

[18] Shihao Gu, Bryan Kelly, and Dacheng Xiu. “Empirical asset pricing via machine learning”. In: *The Review of Financial Studies* 33.5 (2020), pp. 2223–2273.

[19] Geoffrey Hinton. *Neural Networks for Machine Learning*. Coursera lecture 6e. 2012.

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

[21] Klaudia Kaczmarczyk and Marcin Hernes. “Financial decisions support using the supervised learning method based on random forests”. In: *Procedia Computer Science* 176 (2020). Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 24th International Conference KES2020, pp. 2802–2811. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2020.09.276. URL: https://www.sciencedirect.com/science/article/pii/S1877050920321803.

[22] Angelos Kanas. “Nonlinearity in the stock price–dividend relation”. In: *Journal of International Money and Finance* 24.4 (2005), pp. 583–606.

[23] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[24] Mohamed Lachaab and Abdelwahed Omri. “Machine and deep learning-based stock price prediction during the COVID-19 pandemic: the case of CAC 40 index”. In: *EuroMed Journal of Business* ahead-of-print (2023).

[25] Bryan Lim and Stefan Zohren. “Time-series forecasting with deep learning: a survey”. In: *Philosophical Transactions of the Royal Society A* 379.2194 (2021), 2020 0209.

[26] Scott M. Lundberg and Su-In Lee. “A Unified Approach to Interpreting Model Predictions”. In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017, pp. 4765–4774.

[27] Adil Moghar and Mhamed Hamiche. “Stock market prediction using LSTM recurrent neural network”. In: *Procedia computer science* 170 (2020), pp. 1168–1173.

[28] John Moody et al. “Performance functions and reinforcement learning for trading systems and portfolios”. In: *Journal of forecasting* 17.5-6 (1998), pp. 441–470.

[29] Johnathan Mun, M Glantz, and R Kissell. “A Primer on Quantitative Risk Analysis”. In: *Multi-asset risk modeling* (2014), pp. 63–118.

[30] *Nielsen Holdings N.V. to Join S&P 500; Sprint Nextel Corp. to be Removed from S&P 500.* 2013. URL: https://www.spice-indices.com/idpfiles/spice-assets/resources/public/documents/12558_sprintniel.pdf.

[31] *NVR to join S&P 500 index.* 2019. URL: https://www.spglobal.com/marketintelligence/en/news-insights/trending/QxXc3KkY1-TmLvt_3UOpcA2.

[32] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. “On the difficulty of training recurrent neural networks”. In: *International conference on machine learning*. Pmlr. 2013, pp. 1310–1318.

[33] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *arXiv preprint arXiv:1912.01703* (2019).

[34] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[35] Thai Pham and Steven Lee. “Anomaly Detection in Bitcoin Network Using Unsupervised Learning Methods”. In: *arXiv preprint arXiv:1611.03941* (2017). URL: https://doi.org/10.48550/arXiv.1611.03941.

[36] Lutz Prechelt. “Early stopping-but when?” In: *Neural Networks: Tricks of the trade.* Springer, 2002, pp. 55–69.

[37] S&P Dow Jones Indices. *S&P 500.* URL: https://www.spglobal.com/spdji/en/indices/equity/sp-500/.

[38] S&P Global Market Intelligence. *Compustat Database*. URL: https://www.spglobal.com/marketintelligence/en/solutions/compustat.

[39] Laura Sáez-Ortuño, Rubén Huertas-Garcia, Santiago Forgas-Coll, et al. "How can entrepreneurs improve digital market segmentation? A comparative analysis of supervised and unsupervised learning algorithms". In: *International Entrepreneurship and Management Journal* 19 (2023), pp. 1893–1920. DOI: 10.1007/s11365-023-00882-1. URL: https://doi.org/10.1007/s11365-023-00882-1.

[40] Nuhu A Sansa. "The Impact of the COVID-19 on the Financial Markets: Evidence from China and USA". In: *Electronic Research Journal of Social Sciences and Humanities* 2 (2020).

[41] G William Schwert. "Stock volatility during the recent financial crisis". In: *European Financial Management* 17.5 (2011), pp. 789–805.

[42] Derek Snow. "Financial event prediction using machine learning". In: *Available at SSRN 3481555* (2019).

[43] Thibaut Théate and Damien Ernst. "An application of deep reinforcement learning to algorithmic trading". In: *Expert Systems with Applications* 173 (2021), p. 114632. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2021.114632. URL: https://www.sciencedirect.com/science/article/pii/S0957417421000737.

[44] Ashish Vaswani et al. "Attention is All You Need". In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017.

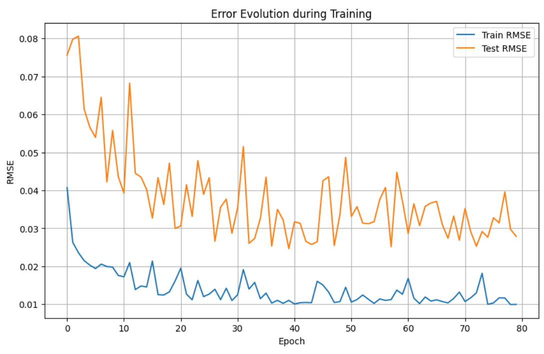
[45] Hao Wang and Baosen Zhang. "Energy Storage Arbitrage in Real-Time Markets via Reinforcement Learning". In: *2018 IEEE Power & Energy Society General Meeting (PESGM)*. 2018, pp. 1–5. DOI: 10.1109/PESGM.2018.8586321.

[46] Wharton Research Data Services. *CRSP/Compustat Merged Database*. Accessed through WRDS. URL: https://www.crsp.org/research/crsp-compustat-merged-database/.

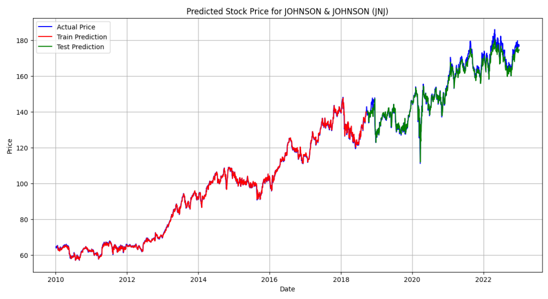
[47] Wharton Research Data Services. *WRDS: Wharton Research Data Services*. URL: https://wrds-www.wharton.upenn.edu/.

Appendix

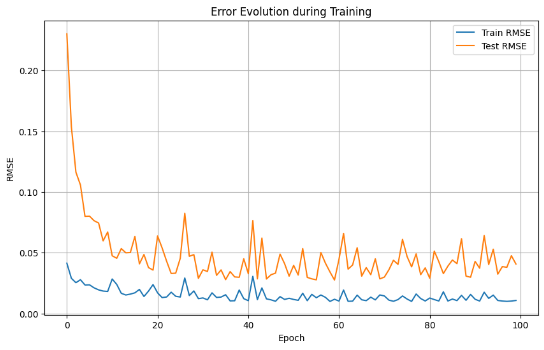
Additional Model Visualizations



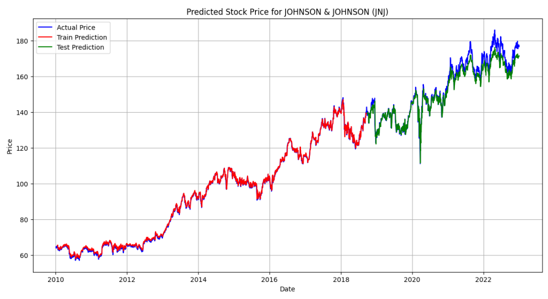
(a) Error evolution for 2 LSTM layers.



(b) Predictions for 2 LSTM layers.

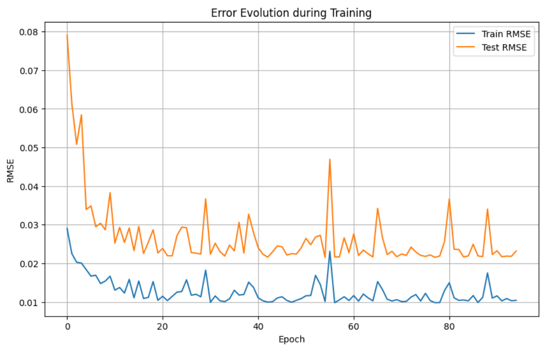


(c) Error evolution for 3 LSTM layers.

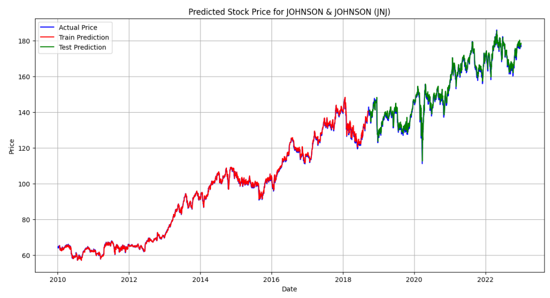


(d) Predictions for 3 LSTM layers.

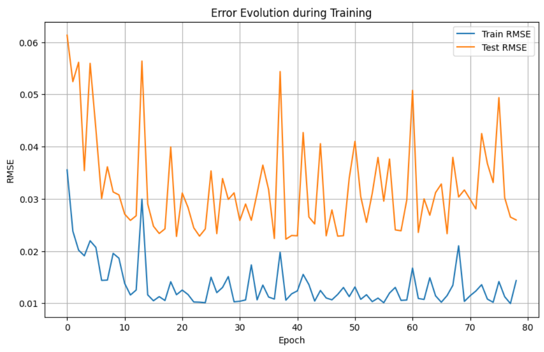
Figure 7.1: Error evolution and predictions for models with different depths.



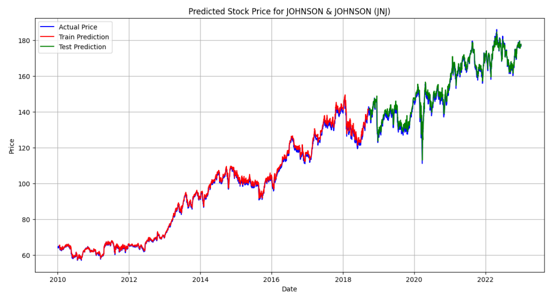
(a) Error evolution for 100 hidden units.



(b) Predictions for 100 hidden units.



(c) Error evolution for 150 hidden units.



(d) Predictions for 150 hidden units.

Figure 7.2: Error evolution and predictions for models with different hidden layer sizes.