A logo with text and a shield

AI-generated content may be incorrect.

CST3990 Undergraduate Individual Project

Stock Price Predictor Using Machine Learning

**Student Name: Rian Qadir**

**Supervisor: Dr Andrew Lewis-Smith**

**Student ID: M00975827**

**A blue rectangular sign with white text

AI-generated content may be incorrect.**

**11th April 2025**

## **Declaration of Authenticity**

I, **Rian Qadir,** ***Student Number: M00975827***, hereby confirm that the work presented in this report and all other related materials is wholly my own work. A list of references is given, and the text includes citations and acknowledgments for the information taken from the literature. This dissertation has never been submitted in part for credit towards any degree or certificate at this or any other university.

A close-up of a signature

AI-generated content may be incorrect.

**Signature: Rian Qadir Date: April 11, 2025**

## **Acknowledgement**

For allowing me to work on this project, I would like to thank and appreciate my supervisor, ***Dr Andrew Lewis-Smith*.** His helpful feedback, advice, and assistance helped me in being able to complete this project successfully. This project allowed me to learn Machine Learning Algorithms deeply and then apply them to my Stock Predictor Project. I would also like to thank my family and friends for their unconditional support throughout this project.

## **Abstract**

Stock price prediction is a complex task in finance due to market volatility brought on by multiple factors. The project develops a web-based stock price predictor based on Long Short-Term Memory (LSTM) networks; a deep learning method well suited for time-series projection. From the historical data retrieved from Yahoo Finance, the system processes and cleanses the stock prices, trains LSTMs, and predicts through an interactive Streamlit interface. The model is found to be of good quality in capturing price patterns in volatile and stable instruments, as validated through quantitative measures. An easy-to-use web-based application provides real-time display of past data as well as predictions, linking academic study to real-world investments. Improvements in the future indicate the addition of sentiment analysis and outside market data to enhance accuracy. The project shows the potential of AI-based applications in financial forecasting with a focus on scalability and user-friendliness for widespread use.

**Table of Contents**

[**Declaration of Authenticity** 1](#_Toc194768975)

[**Acknowledgement** 2](#_Toc194768976)

[**Abstract** 3](#_Toc194768977)

[**1.** **Introduction** 5](#_Toc194768978)

[**1.1** **Introduction** 5](#_Toc194768979)

[**1.2** **Problem Statement** 6](#_Toc194768980)

[**1.3** **Aims and Objectives** 7](#_Toc194768981)

[**2.** **Background and Literature Review** 8](#_Toc194768982)

[**2.1** **Introduction to the Background** 8](#_Toc194768983)

[**2.2 Literature Review** 9](#_Toc194768984)

[**2.3 Conclusion** 25](#_Toc194768985)

[**4.** **Methodology** 27](#_Toc194768986)

[**4.1** **Overview** 27](#_Toc194768987)

[**4.2** **Approach** 27](#_Toc194768988)

[**5.** **Future Directions and Improvement** 44](#_Toc194768989)

[**6.** **Conclusion** 48](#_Toc194768990)

[**References** 49](#_Toc194768991)

## **Introduction**

### **Introduction**

The correct forecasting of stock prices is the core of investment decision-making and finance analysis. Due to the inherent volatility of stock markets, the forecasting of future prices is a difficult yet vital task that arouses the attention of investors as well as researchers (Kumbure et al., 2022). Classical stock market forecasting methods, i.e., the fundamental and the technical analysis methods, offer well-informed investment advice based on past prices and financial measurements. Classical methods, however, often fall short because of the complex, non-linear nature of stock price motion, which results from a tremendous number of unpredictable factors like economic circumstances, geopolitical circumstances, and investors' moods (Kumbure et al., 2022).

The recent advancements of deep learning methods, particularly of recurrent neural networks (RNN), revolutionized the finance forecasting paradigm. Of these methods, the application of special kinds of RNN, i.e., the Long Short-Term Memory (LSTM), has been promising for the modeling of the time-series data dependencies. The LSTM model is the most suitable for the stock prices forecasting because it is particularly effective for learning from sequential data with the ability not to be vulnerable to vanishing gradients, the weakness of the general RNN.

This is a report on the development of a web-based stock price prediction application using the application of LSTM networks. The application will take in historical stock market data, train the prediction models, and produce future stock prices in an easy-to-use interactive interface. This report aims to give a thorough explanation of the development and deployment of this prediction application, from the data acquisition process, model design, the training process, the performance measure, and the deployment procedures.

The report begins with the introduction of the problem of stock price prediction, then proceeds with the discussion on the aims and objectives. The introduction is then succeeded by the literature review, whereby the discussion is on existing stock market prediction models, their advantages, and their disadvantages. The Methodology chapter explains data collection, data preprocessing, training of the model, and the implementation approach. It also discusses the results and problems faced and is focused on the performance of the model on predicted stock prices against real prices with the help of quantitative measurements. The report then concludes with the potential future plans, and potential avenues of improvement for future work.

### **Problem Statement**

Stock prediction is volatile because the stock market is subject to the impact of numerous economic, political, and psychological factors. Different strategies of prediction by investors and traders are used to anticipate the market direction, but the conventional method of forecasting, namely, fundamentals and technical analysis, is incapable of understanding the intricate, non-linear interactions of finance data. Volatility of this nature poses a great challenge for individual investors and institutional traders because it makes the maximization of returns and the reduction of risks challenging (Kumbure et al., 2022).

One of the most difficult things about stock price prediction is handling the noisy and volatile nature of the stock market data. Models such as Autoregressive Integrated Moving Average (ARIMA) and linear regression aren't capable of modeling long-term relationships and complex patterns of the time-series data involved with finance. Models such as these assume some distribution of data beforehand, which may not be reflective of real market dynamics. Unlike this, deep learning models, particularly LSTMs, can be observed to capture complex patterns from sequential data with minimal feature engineering.

Despite contributions from the field of artificial intelligence, current predictive methods are still constrained by such factors as incomplete feature sets, lack of real-time responsiveness, and poor processing of external market data (Pham Hoang Vuong et al., 2024). In addition, the great majority of machine learning-based stock forecasting models are conceived for research goals but with no implementation as a practical, easy-to-use piece of software. The gap herein between research outcomes and actual practice calls for the creation of an effective, easy-to-use, and strong stock price predictor program to offer investors sound guidance (Kumbure et al., 2022).

This project works around these limitations by creating a web application that utilizes an LSTM-based machine learning algorithm for the prediction of stock prices based on historical data. The project endeavors to provide an easy-to-use interface with the user being able to view historical trends, examine predictive outcomes, and make investment decisions based on data. By incorporating interaction-supported visualization, model performance metrics, and real-time data handling, the project strives to contribute to the reliability and usability of stock price prediction systems.

### **Aims and Objectives**

#### **1.3.1 Aims**

The primary objective of this project is the designing, development, and deployment of a web-based stock price prediction system with the application of deep learning methods for better prediction of the stock prices for the future. By the deployment of the implementation of the LSTM neural networks, the system will learn the long-range patterns of the stock prices and make predictions with increased stability. The application will be designed to display a user-friendly interface for the users, enable the users to interact with the prediction models and derive insights from stock market patterns.

This project also aims at bridging the gap from deep learning research theory to practicing forecasting finance with the development of a full-fledged, operational, and scalable prediction system. With the inclusion of real-time fetching of stock data, real-time visualization, and performance monitors, the system aims to be a valuable resource for investors, finance analysts, and researchers examining the application of AI for finance.

#### **1.3.2 Objectives**

**1.** **Collection and preprocessing of stock market data:** Yahoo Finance will be the main source of data, and historical stock prices along with the corresponding financial indicators will be used. Data preprocessing tasks will involve the treatment of missing values and normalization of numerical data, and the dataset will be formatted for effective training and testing.

**2. Model training and optimization with the use of LSTM:** The model structure will be a combination of several layers of the LSTM, dense layers, with appropriate hyperparameters such as learning rate, batch size, and number of epochs for the improvement of predictive ability. The model will be optimized for error reduction factors such as Mean Squared Error (MSE) and Root Mean Squared Error (RMSE).

**3. To evaluate the predictive model performance:** The model will be validated based on the accuracy of the predictions with the use of the RMSE, Mean Absolute Error (MAE), and other evaluation metrics for the assurance of prediction reliability. The baseline models (ARIMA, linear regression) will be compared for the assurance of improvements.

**4. Building an easy-to-use web application with Streamlit:** The web interface will offer users the feature of entering stock tickers, showing the historic stock prices, analyzing the model predictions, and data downloads of the predictions. The application will be built with simplicity and the least technological expertise required.

**5. For creating dynamic visualization tools:** The software will have interactive visualization tools with the help of libraries like Plotly and Matplotlib for appropriately displaying the stock market trends, predictive results, and the confidence interval.

**6.** **To test potential improvements on the predictive model**: More improvements will be tested, including the utilization of other market traits (trading volume, volatility indexes, and sentiment indicators), hyperparameter tuning, and comparing with other deep learning architectures such as Gated Recurrent Units (GRUs) and Transformers.

**7. Future-proof scalability and real-time response:** The system must be designed for expansion, e.g., real-time processing of stock quotes, API-based finance news sentiment analysis addition, and incorporation of retraining the model on newly updated data on a regular schedule.

## **Background and Literature Review**

### **Introduction to the Background**

Stock price forecasting has been among the traditional finance market issues, fueled by an interdependent interaction of a high-dimensional number of factors. While financial markets are not as deterministic as markets are, with changing dynamics and often experiencing unexpected phenomena such as speculation, investors' mood, and exogenous economic shocks, their intrinsic volatility makes forecasting of stock prices a highly significant but very challenging mission, requiring state-of-the-art mathematical methods and highly computation-intensive models. Stock trends result from a combination of different macroeconomic indicators, company financial health statistics, geopolitical conflicts, and government policies. Added on top of these are economic conditions globally and investors' sentiments, bringing in more complexity when making predictions. With such complexities, financial analysts, institutional investors, and hedge funds always look for better methods to make more precise predictions to facilitate informed investment decisions, risk management strategies, and algorithmic trading optimization. The heightened utilization of machine learning, artificial intelligence, and computational simulation has transformed traditional financial analysis into sophisticated predictive methods of volatility management in stock markets. It provides a backdrop to achieve the role of future price forecasting and methodology of financial forecasting being carried out.

### **2.2 Literature Review**

Predicting stock prices has been one of the most complex and extensively studied financial markets' challenges throughout history. This is due to the multivariate relationships between an abundance of factors affecting stock price volatility. Such factors are macroeconomic indicators, the financial health of companies, market sentiments, geopolitical risks, government policies, and current global economic conditions. Unlike purely deterministic systems, financial markets are characterized by dynamic properties and are subject to irrational behavior emanating from speculation, market sentiments, and extrinsic shocks.

Predictions of stock prices are of more than purely theoretical interest and hold important consequences for investment decisions, risk analysis, portfolio control, and trading algorithms. Hedge funds, institutional investors, and financial professionals are forever looking for robust predictive models that may give them an edge when it comes to forecasting markets. Despite the advancements of orthodox finance and computational methodologies, no one method has proven to repeatedly beat the markets under all situations.

The Efficient Market Hypothesis (EMH) suggests that stock prices are embedded with all available information. Hence, it becomes impossible to achieve returns that are better than those offered by its alternatives (Fama, 1970). Skeptics argue that the efficiency of markets cannot be consistent as it might be affected due to psychological biases, speculation bubbles, and external or political shocks (Shiller, 2003). The recent phenomenon of "meme stocks" reflects how herding behavior among investors can create price volatility inconsistent with standard valuation paradigms (Hayes, 2024).

#### **Traditional Methods of Stock Price Prediction**

##### **Fundamental Analysis**

Fundamental analysis ranks among the longest-standing yet basic methods of stock pricing forecasts. It holds the assumption that the inherent worth of a company, driven by the fundamentals, competitive position, industry direction, and general macroeconomic context, drives the pricing of the stock over the long run (Graham & Dodd, 1984). Investors utilize the tool as a way of calculating a stock's overestimate or underestimate by comparing the current marketplace price with the calculated underlying worth.

Important financial ratios utilized in fundamental analysis include:

* **Earnings per Share:** A simple measure of profitability at the company level, computed by dividing net income by outstanding shares.
* **Price-to-Earnings (P/E) Ratio:** A measure of market expectations for a company’s growth. A high P/E may signal optimism, whereas a low P/E may reflect undervaluation or risk.
* **Return on Assets (ROA) and Return on Equity (ROE):** Financial ratios used in determining the efficacy with which a business organization employs the assets and the shareholders' equity in making profits (Penman, 2012).

Apart from firm-specific issues, macroeconomic events such as GDP growth, inflation, interest policy, and employment opportunities have a profound impact on investors' expectations as well as share price volatility. For example, higher interest rates have a downward effect on share prices since they make capital more expensive and decrease profits for companies.

For all its merits, basic analysis suffers from a few serious defects. While well-predominant with long-term investing, it nearly collapses when trying to forecast short-term direction in the marketplace. It's largely because it fails to take into consideration speculative action, crowd mentality, and sentiment in the marketplace, all too capable of nudging prices away from underlying worth over the short run (Shiller, 2000). It can disregard instant marketplace shock by way of shocking news, political crises, or natural disasters capable of upsetting the marketplace before the money fundamentals take hold.

##### **Technical Analysis**

As compared to this, technical analysis solely examines the direction of price and volume movements to predict future direction in the market. It presumes that prices in the market move in recognizable trends and that previous movements in prices can be utilized to make future movements based on the patterned behavior of investors and market psychology (Murphy, 1999).

Typical tools and indicators used are:

* Moving Averages (SMA/EMA): Utilized in deciding the direction of the trend and smoothing the prices for a specified time frame.
* Bollinger Bands: Developed to measure a stock's volatility and overbought/oversold levels based on standard deviation.
* Relative Strength Index (RSI): Momentum indicator that gauges the magnitude and speed of the direction change in the price.
* Candlestick Patterns: graphical representations of the price behavior that exhibit trend reversals or continuation patterns.

Though very popular among short-term investors and day traders for timeliness and instant responsiveness, the technical analysis has not been free from criticisms. Past data assumes recurring patterns in the past, which collapses in fast-moving markets driven by exogenous shock, announcements by companies, and big events. Fama (1970), among the critics' arguments holds that whenever the markets are considered to be efficient, the public data, including the one built into the technical indicators, would already be reflected in the stock prices, and it would be a requirement for it to be impossible to beat the markets consistently with the use of the charts. Also, many analysts note a self-fulfilling prophecy phenomenon: if enough traders apply the same patterns or indicators, they can inadvertently create the very same price behavior they are expecting, not because they reflect underlying value, but because they reflect the group behavior itself (Lo, Mamaysky, & Wang, 2000).

Murphy (1999) discusses yet another issue. Technical indicators, he argues, may indicate trends in price, but they are frequently inaccurate due to market irregularities. Such irregularities may compromise the forecasting ability of prices, and it may be difficult to rely purely on technical analysis to make good decisions.

##### **Limitations of Conventional Strategies**

Technical analysis based on past market data such as price and volume has been the cornerstone for investors. It is complemented by fundamental analysis, which considers a company’s inherent worth by using financial documents, industry analysis, and macroeconomic data. While traditional methods have been based historically, they have considerable shortcomings.

* Response Inflexibility for Unprecedented Events: Classical models are rule-based and deterministic and are not very responsive to unusual or unprecedented events such as geopolitical crises, pandemics, or stock-market meltdowns. All such “black swan” events, according to the definition by Taleb (2007), fall outside the realm of normal probability distributions and are difficult for classical models to predict as they are unusual and have a drastic impact.
* Shortcomings in Capturing Complex Interdependencies: Stock markets are driven by a web of interconnected, nonlinear variables—from global chain relationships in the supply chain, past investor psychology all the way down to government policies. Traditional indicators like RSI or moving averages are much too simplistic a description for the intricate, dynamic interdependencies. Although the markets in theory can efficiently absorb information, as explained by Fama (1970), actual data exhibit anomalies explainable by no conventional tool.
* Too Much Reliance on the Past: Most conventional models assume that past behavior will be a good predictor of future events. The assumption fails in dynamic settings with the impacts of emerging technology, changes in policy, and instantaneous global interdependence. Lo and MacKinlay (1988) discredited the random walk hypothesis by arguing that past returns are not strictly uncorrelated rather proved that the dynamics of the changing markets make it increasingly irrelevant with a given strategy. Due to these constraints, ML and DL methods have increasingly been used by researchers and practitioners in finance. In offering much-improved improvement through the discovery of intricate, nonlinear links in big and multifaceted data, data-driven methods have been favored more and more. For instance, Fischer and Krauss (2018) demonstrated how LSTM networks exceed the performance of standard models in the S&P 500 index forecast. Similarly, Dixon et al. (2020) showcased the strength of ML models such as random forests and the utilization of XGBoost in using varied data streams such as financial numbers, news sentiment, and macroeconomic data.

Additionally, DL models such as CNNs and transformers are capable of processing and learning data which are unstructured in nature (e.g., news headlines, social media) and combining it with structured numerical data and as a result, are capable of learning both statistical as well as semantic signals (Zhang et al., 2020). DL models are also adaptable, i.e., they can be retrained using additional data almost in real time and thereby become much more responsive to changing market conditions.

#### **Machine Learning for Stock Price Prediction**

##### **Review of Machine Learning Applications in Finance**

Financial forecasting was brought to an advanced height using machine learning as it enabled models to learn from historical stock price behavior and extend trends to the future without any direct programming of rules. When compared to traditional analysis methodologies, ML models can easily deal with high volumes of structure and unstructured data, identify complex patterns, and make predictions from data with accuracy. It is this ability to capture non-linear relationships that makes ML models very helpful in various finance applications, including:

* Algorithmic trading: Employs machine learning algorithms to create trading algorithms that execute trades automatically on predictive price trends, lowering the likelihood of human error and speed of execution.
* Risk Analysis: Banks and financial institutions apply ML algorithms to forecast credit risk due to the behavior of the borrowers, the state of the economy, and past financial data to lend more effectively.
* Portfolio Management: Robo-advisors use artificial intelligence to learn from past trends and current trends to offer customized investment recommendations and maximize portfolio returns.
* Fraud Detection: Machine learning algorithms detect fraudulent spending activities through the detection of unusual spending, facilitating banks and other financial organizations to minimize fraud and cybercrime exposure.

Financial experts utilize ML approaches to peer beyond standard forecasting models, as they are constantly inadequate to gain subtle market trends and volatility.

##### **Regression-Based Stock Market Prediction Models**

Regression models are the first learning machines that were implemented to forecast finances. They create associations between independent variables (e.g., trading volume, economic factors) and dependent variables (stock prices). Some of the most utilized regression models implemented for forecasting the stock market are Linear Regression (LR) and Support Vector Regression (SVR):

**Linear Regression (LR):**

LR establishes a direct correlation between independent and stock prices. It makes use of historical stock prices and uses them to project stock prices from a simple formula. It is an interpretable model, cost-effective computationally, and best suited to situations when price trends are linear. Financial markets are, though, highly non-linear and interdependent, and these cannot be replicated by LR. Therefore, it is not suitable for long-term forecasting of stock prices under volatile situations. Zhang et al. (2016) empirical evidence reveals that LR can be used to detect short-run price trends but cannot adjust to non-linear trends of financial series.

**Support Vector Regression (SVR):**

SVR is one step higher than LR as it maps input features to the high dimensional space by applying kernel functions, and through this, it takes non-linear stock price movement. It tries to minimize error by linking the function to some degree of tolerable deviation, hence making it immune to market fluctuation. Although it is strong, SVR is computationally intensive and does not perform as effectively on very high-dimensionality and large input factor counts. It also requires persistent hyperparameter tuning, i.e., kernel type and regularizes, to avoid overfitting or underfitting of the data.

While regression-based models are suitable when one has to predict plain financial values, they are feature engineering-intensive and cannot capture the intricate, time-varying dynamics of the up-and-down movement of stock prices in the market. Therefore, more sophisticated models, from tree-based models to deep learning models, came to the forefront.

##### **Tree-Based Stock Market Prediction Methods**

Tree-based models are good at predicting finances as they can manage high volumes of data, non-linear associations, and high predictive strength. Decision trees are formed by tree-based models by applying the methodology of recursively partitioning the data according to high-value financial factors. Among the most popular tree-based models are:

**1. Decision Trees (DT):** DTs divide the data according to decision rules, and thus they are easy to interpret and simple. They can effectively deal with categorical and numerical data and hence are suitable for different financial applications. DTs, however, are vulnerable to overfitting as they are capable of learning historical information really effectively but fail to generalize to novel unseen data. They also underperform during high-volatility markets, when prices change suddenly and tend to get the model's predictions confused.

**2. Random Forest (RF):** RF is an ensemble learning method that makes several decision trees and uses their predictions as their average to build accuracy and prevent overfitting. RF is stable and can handle large sets of financial data having complex interactions than one decision tree. RF works effectively as regards stock market trend forecasting due to it taking an adequate number of input variables into account to render forecasting unbiased. However, RF cannot be read because predictions are generated by an ensemble of decision paths, and not one decision path.

**3. Gradient Boosting Machine (GBM):** GBMs like XGBoost and LightGBM are extensions of RF and incorporate one tree at another sequentially and every subsequent tree making alterations to its direct precursor.These models are particularly useful when predicting stock price movement as they are capable of identifying subtle trends from financial data.XGBoost is vastly deployed in algorithmic trading due to its high performance, high efficiency, and abilities to deal with missing values.Though GBMs are highly accurate, they are hyperparameter-intensive and computationally expensive and hence cannot be easily deployed on live trading.

**Relative Performance of Tree-Based Structures for Finance Prediction**

Research by Patel et al. (2015) demonstrated RF to perform superiority over classical regression models to predict the stock market by: Having many properties of data and decreasing bias. Prevention of Overfitting Using Ensemble Learning This makes more stable predictions than when classical regression methods are utilized. Whereas tree-based models give tremendous improvements when it comes to forecasting stock prices, they are weak at detecting sequential information and hence are not as good at capturing long-run trends and time-series relations. To overcome these deficiencies, deep learning models, i.e., Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM) networks, are increasingly being deployed.4. Deep Learning for Stock Price Prediction

#### **Deep Learning and Stock Market Prediction**

##### **An Overview of Stock Prediction Using Deep Learning**

Deep learning significantly improved stock price forecasting by utilizing multi-layer neural network models to capture complex, *non-linear relationships* and learn *temporal dependencies* amongstock price variances. In comparison to the standard machine learning method that entails lengthy *feature engineering*, deep learning methodologies can *extract important features automatically* from raw financial inputs and hence are extremely capable when it comes to time-series forecasting (Kumbure et al., 2022).

Financial markets are complex and are affected by numerous factors, including *historical price trends, macroeconomic indicators, trading volume, sentiment analysis, and world events*. Deep learning models can learn from these factors dynamically without having any predefined rules and, hence, become immune to *disequilibria of market conditions.*

##### **Stock Market Prediction Using Artificial Neural Networks (ANNs)**

Artificial Neural Networks (ANNs) are among the most established deep learning approaches to forecasting money. ANNs are composed of *multiple layers of interconnected neurons*, and they all take and process inputs and then transmit them to the next (Kumbure et al., 2022).

**How ANNs Function in Stock Prediction:**

* ANNs analyze historically observed stock prices and technical indicators (e.g., moving average, RSI, MACD) and try to identify trends and project future price actions.
* Every neuron in the network uses an *activation function*, assisting in representing intricate, non-linear relationships in financial data.
* Maximum learning is obtained by applying *backprop and gradient descent*, through which the network can adjust its weight upon experiencing errors made during predictions.

**Benefits of ANNs:**

* Very flexible: Able to capture subtle features and non-linear price movement.
* Scalability: Effective handling of large financial data sets.
* Pattern recognition: Capable of identifying hidden associations between share prices.

**Limitations of Stock Prediction Using ANNs:**

* Data requirements: ANNs need *huge amounts of training data* to generalize well.
* Overfitting: They memorize historical stock prices rather than learning useful trends and hence end up having *bad generalization on novel input*.
* No memory: Basic ANNs *don't have long-term dependencies*, thus they aren't very good at time-series predictions.
* Computational cost: Such deep networks require *considerable computational resources* to train and are susceptible to hyperparameter tuning.

Kim and Han's (2000) research reports that while ANNs perform well when working on big financial databases, they are not very successful when it comes to long-term stock market forecasting.

##### **Recurrent Neural Networks (RNN) and LSTM Networks Stock Prediction**

Breaking the limits of traditional ANNs, Recurrent Neural Networks (RNNs) were created to integrate the mechanism of memory to allow models to keep track of information from one step to another. RNNs perform best at time-series forecasting, where earlier stock prices influence subsequent movement (Kumbure et al., 2022).

**Recurrent Neural Networks (RNN)**

RNNs use back propagation, whereby the output from one step is fed back again as input to the network. They are valid for dealing with sequential data, hence valid to use when making stock market predictions. There are vanishing gradients during training, and due to this, RNN cannot learn long-term dependencies.

**Long Short-Term Memory (LSTM) Networks**

To solve the vanishing gradient problem, Long Short-Term Memory (LSTM) networks were devised. LSTMs have memory cells that hold up-to-useful information on long sequences. They employ gates (input, forget, and output gates) to control the movement of information and get rid of redundant information. They are very good at forecasting long-term trends within the stock market.

A diagram of a tank

AI-generated content may be incorrect.

**Major Strengths of LSTMs in Finance Prediction:**

* Memory Retention: LSTMs are capable of learning long-term dependencies between stock price trends.
* Reduction of overfitting: With the use of the gate mechanism, excessive noise from financial information is not stored.
* Time-series aptitude: LSTMs are more adept than standard ML models at learning sequences over time.

Fischer and Krauss (2018) proved through research that LSTM networks are superior to those typical regression-based models and RNNs when it comes to stock market forecasting by being able to learn long-term dependencies.

##### **Stock Market Prediction Using Combined Deep Learning Models**

Although LSTMs improved predictive ability further, hybrid deep learning models were developed to further enhance predictive ability. These models combine multiple deep learning models, benefiting from each architecture's strength:

**1. LSTM-CNN Models:**

* **Convolutional Neural Networks (CNNs)** are typically deployed when image recognition is involved, but when predicting stock markets, they are responsible for identifying spatial trends on stock price graphs.
* **CNN-LSTM** hybrid models use CNNs to extract salient features from stock charts before inputting them into an LSTM network to learn sequential relationships.
* **Example:** Hochreiter and Schmidhuber (1997) demonstrated that more complex market trends can be learned using CNN-LSTM models compared to LSTMs.

**2. Attention-Based Models:**

* Attention mechanisms bolster LSTMs by allowing the model to focus on the most relevant time steps and disregard irrelevant information.
* Through **weights** being given dynamically to different earlier observations, these models are made more predictive.
* Wang et al. (2020) demonstrated one of those hybrid LSTM-Attention models that outperformed standard LSTMs when forecasting stock prices.

**3. Transformer-Based Models:**

* **Transformer models** (e.g., GPT and BERT models) are currently trending as they are able to process sequences simultaneously as opposed to sequentially processed LSTMs.
* They are particularly adept at **financial sentiment analysis**, whereby they scan news, social media, and earnings calls and then utilize these to project stock price trends.
* The Future of Finance and Deep Learning Deep learning revolutionized stock market forecasting by allowing models to automatically extract features, uncover intricate non-linear relationships, and capture long-term dependencies between financial data.
* **Artificial Neural Networks (ANN)** laid the foundation but were weak in long-term memory.
* **RNNs and LSTMs** improved stock forecasting by adding memory mechanisms, but they are still limited from being able to capture all the market signals that are of relevance (Kumbure et al., 2022).

#### **Stock Market Prediction Using Sentiment Analysis**

One of the major determinants of stock price movement is investor sentiment as it reflects the combined tone and expectation among investors. Predictive models have increasingly incorporated natural language processing (NLP) as a major practice to glean sentiment-based trends. News reports, social media posts, companies' earnings call transcripts, and even entries from the financial blogosphere are all subject to analysis to derive valid conclusions about market sentiment and, consequently, stock price movement (Kumbure et al., 2022).

Recent studies show the impact of sentiment, particularly from media outlets such as Twitter, upon stock price movement. Bollen et al. (2011) research established that Twitter sentiment correlates directly to stock price movement and, as an extension, suggests social media sentiment as another predictive variable when integrated into stock forecasting models. This has spurred the creation of various sentiment analyses as ways to enhance forecasting ability.

##### **Stock Prediction Using Sentiment Analysis Methods**

There are several ways of measuring market sentiment, and all of these have their strengths and applications:

**1. Lexicographic Methods**

Lexicon-based sentiment analysis relies on dictionaries or sentiment lexicons to assign sentiment scores to terms. They are classified as positive, negative, or neutral as per the dictionary and they are aggregated to determine the sentiment of the document, post, or news article. This is simple to use and provides an easy means of sentiment analysis. However, this may be poor when identifying context, especially when recognizing sarcasm or irony.

**2. Pre-Trained Language Models**

Pre-trained models such as BERT (Bidirectional Encoder Representations from Transformers) and FinBERT (specialist financial document model) are being leveraged to extract sentiment from more advanced text inputs. They are pre-trained on large corpora first and then fine-tuned to learn the finer distinction between financial terminologies so that they are able to understand more context-sensitive sentiment indicators from news articles, social media, and other financial reports.

For example, FinBERT is designed to operate on texts relating to finance and hence is more capable of detecting positive, adverse, or neutral sentiment when processing financial news or earnings calls.

1. **LSTM-Based Sentiment Models**

Researchers such as Kumar & Ravi (2016) have proposed implementing Long Short-Term Memory (LSTM) networks to perform improved predictions through sentiment analysis. LSTM is able to learn the time dependencies from text-based data and thus suits the task of capturing the dynamic aspect of sentiment, particularly where sentiment is time-sensitive, e.g., financial markets. These models can be exploited to enhance stock price forecasting by adding sequential sentiment input, which is especially relevant to forecasting stock movement from time-series sentiment. Such sentiment analysis methodologies can be integrated into stock market forecasting models to make predictions more accurate and investment decisions more informed based on market fundamentals and investors' sentiments.

#### **Web-based Stock Forecast Software**

Applications of stock price forecasts using machine learning models are being taken to the live web as web and cloud computing technology is continually developing. Such web applications enable users to interact directly with predictive models and can incorporate features like visualizations, predictive recommendations, and personal portfolio recommendations.

These web applications are designed using web frameworks including Streamlit, Flask, and Django. They are lightweight web applications and can give stock predictions in real time. They can integrate various sources of information, display dynamic visualization, and give investment advice depending on stock market predictions. They offer simple and direct interaction between retail investors and experts as well as advanced predictive models.

Key Features of Web-Based Stock Forecasting Systems:

* Real-Time Forecasting: The ability to create stock price predictions in real-time from current data.
* Interactive Visualizations: Dynamic graphs and charts presenting stock price trends, sentiment analysis, and other key indicators.
* Portfolio Recommendations: Through predictive models, the investors can be offered personal investment recommendations or portfolio optimizations.
* Easy-to-use Dashboards: Interactive and intuitive dashboards enable users to make informed decisions without having to know much about money or machine learning.

Xu et al. (2021) emphasize the relevance of including visualizations and easy-to-use dashboards within stock forecasting web applications as measures to facilitate usability and make complex models understandable to large populations. The applications are crucial to summarize the data in an interpretable and actionable format so users can easily interpret the results of the model and make reasonable judgments.

Although there has been a tremendous progress achieved to date with the application of machine learning (ML) and deep learning (DL) algorithms in predicting stock prices, there are still a number of issues hindering their reliability and accuracy when applied to real-world applications. A few of the major issues include market volatility, data availability and quality, and model overfitting, and multi-modal and heterogeneous data fusion. They need to be addressed to render predictive models more practical and feasible to be applied to real-market scenarios.

#### **Challenges and Future Directions**

**Market Volatility and Uncertainty**

One of the greatest obstacles to stock price predictions is the volatility of the market. The share markets are extremely sensitive to news regarding the economic climate, political change, slowdown abroad, and investors' psychology and, due to this, may make sudden and unexpected changes to the stock price.

***Impact of Sudden Events***

* Political upheavals, monetary policy shocks, wars, and trade wars are events that trigger extreme stock price dynamics. Such events are stochastic and happen unexpectedly, and hence, predictive models are even more challenging.
* Natural calamities (e.g., pandemics, hurricanes, earthquakes) make global supply chains, production, and investors' sentiment freeze and this makes stock price forecasting more complex.
* One good example is the COVID-19 2020 pandemic, when the market saw sharp falls and vigorous swings, corresponding to how external influences create excessive volatility.

***High-Frequency Market Adjustments***

* It differs from regular data as stock price activity is influenced by current information from macroeconomic indicators, financial reports, and sentiment.
* Even the fastest high-speed trading software operating at millisecond speeds cannot always respond immediately to unexpected global events, and they cannot be counted on to discern sudden market activity.

**Potential Research Directions**

* Event-oriented predictive models that incorporate macro announcements and extrinsic events can make predictions more accurate as they can capture these sudden market moves.
* Reinforcement learning-based adaptive models can help algorithms learn and adjust automatically to changing market conditions, potentially decreasing the errors of predictions when markets are volatile.
* Hybrid approaches that integrate ML and econometric approaches have the ability to more effectively capture market recuperation dynamics and absorption of shocks.

**Data Quality and Availability Challenges**

ML and DL models' performance are subject to the extent of accuracy, size, and quality of training and testing provided. The quality and availability of financial information are often one of the main stumbling blocks and may impact the performance of the forecasting models adversely (Pham Hoang Vuong et al., 2024).

**Partial incomplete and gaps within data**

* For developing countries, stock market data can be less standardized and less reliable, and therefore, it will give biased estimates.
* Regulatory changes create inconsistency among the data, especially when different nations present their financial information differently.

**Noisy and Unreliable Data:**

* Financial information is noisy, i.e., subject to manipulative market forces, trading irregularities, or inaccuracies due to reports.
* Hedge fund manipulations or pump and dump schemes may create artificial stock price volatility.
* False accounting statements and manipulative social media posts can upset market stability and induce price volatility at the current moment.
* Faulty automated trading may provoke flash crashes or temporary price deviations from trends.

**Research Directions**

The development of outlier detection algorithms would help us remove noisy data and make it possible to build stable predictive models. Mechanisms of trust to screen disinformation from non-verifiable sources like social media can help increase the quality of financial forecasting information.

**Overfitting and Generalization Problem**

One of the major pitfalls of stock price forecasting deep models is overfitting. Overfitting occurs when the model becomes too sophisticated and accommodates the training set too much, failing to generalize to novel, unseen data.

**Why does overfitting occur in stock market models?**

* Advanced deep learning models such as LSTMs, CNNs, and Transformers contain millions of parameters, and these may lead the model to memorize the training set patterns, hence weakening the model when it must generalize to new unseen data.
* The stock prices are subject to random movement and temporary volatility and are often erroneously identified as long-term trends by deep models, and the models are overfit.
* Such a lack of diversity in training sets may drive the models to overfitting under certain economic conditions and hence perform poorly under recession or crises (Pham Hoang Vuong et al., 2024).

**Integration from Multiple Sources**

Today's financial market generates vast quantities of information, both structured and unstructured. The issue is how to utilize these various sources of information to be useful when making stock price predictions.

**Structured and Unstructured Data:**

* Structured data includes fiscal information such as share price, market trends, accounting statements, and macro data.
* Unstructured data, however, includes more varied sources, ranging from news articles and social media opinions to earnings called transcripts and satellite images to read economic trends.

**Difficulties in Data Integration**

* It is challenging to sync up live stock price volatility, earnings announcements, and sentiment on social media.
* Handling mass quantities of text content (e.g., news articles, social media posts, and research reports by analysts) requires advanced NLP skills to be able to extract usable information.
* Data integration: Data received from various sources is generally integrated by eliminating duplicates and inconsistencies.

**Computational and Ethical Challenges**

Deep learning and stock price forecasting are as computationally as they are ethically daunting.

**Computational Complexity**

* High-frequency trading algorithms must support massive streams of data on the fly, creating enormous computational needs.
* Deep learning models (e.g., LSTMs and Transformers) are computationally expensive to fine-tune and train and are hardware and time intensive.

**Ethical and Regulatory Challenges**

* Bias within algorithms can result in unethical trading activities, such as market manipulation or discriminatory trading.
* Risks of insider trading: The ability to make more accurate predictions of stock movement in the future may pose insider trading concerns, particularly when ML models can make stock predictions more consistently correct than human traders.

Breaking the constraints of market volatility, quality of data, overfitting, and processing diverse sources of information is central to optimizing stock price forecasting models' efficiency and dependability. Research on new methodologies, i.e., event-based models, reinforcement learning, and ensemble models, and advancements of computing and handling unstructured information and computational capabilities, will continue to be of essence as stock price forecasting moves along. Ethic and regulatory considerations as well will continue to be at the center to facilitate responsible application of these models and prevent them from generating market imbalances.

### **2.3 Conclusion**

Stock price prediction is an essential and complex task within the field of finance that determines investments, risk analysis, and trading decisions. The classical methodology fails to capture the complex and nonlinearity of the stock market, and hence the use of deep learning and machine learning methodologies is inevitable. A trend increasingly recognized as important in stock market forecasting research is sentiment analysis, where natural language processing (NLP) methodology is utilized when tracking investors' opinions, social media, and economic news. The methodology provides useful predictive indicators besides the numerical values conventionally associated with share prices, providing clearer intuition of market direction.

Despite this dramatic improvement in computational finance, there remain issues. Surprise turns of economics, market deviations, random investors' moods, and black swans create tremendous unpredictabilities, and long-term accuracy eludes. In addition, deep learning models become susceptible to overfitting and require extensive preprocessing and regularization methods to be implemented to make them generalizable. With the progressive development of the financial markets, research over the next few years is bound to center on making models more interpretable, adding alternative sources of information, and applying reinforcement learning approaches to help models learn to change more quickly to evolving situations in real time. With the development of web-based forecasting applications, these advanced predictive models are made accessible to retail and institutional investors as well. No forecasting model is able to eliminate market unpredictabilities, and improvements made to sentiment analysis, deep learning, and machine learning continue to push frontiers of forecasting projections in finance, steadily enhancing stock price forecasting models to become ever more accurate and responsive (Ding, X., Zhang, Y., Liu, T., & Duan, J., 2015).

## **Methodology**

### **Overview**

The section presents a step-by-step approach to forecasting financial time series using deep learning. It begins with data gathering from Yahoo Finance with a focus on high-frequency stock data like closing prices for Google and Bitcoin assets. It then describes a stringent preprocessing pipeline—comprising data cleaning, forward-fill imputation, outlier smoothing using the IQR approach, and normalization via MinMax scaling—before reshaping the data into sequences using a sliding window approach for LSTM compatibility. It then discusses the use of LSTM networks over baseline approaches, describing in detail their architecture, dropout layers, and hyperparameter tuning for effectively capturing long-term temporal dependencies. The section then discusses the training process, evaluation via metrics like MSE, RMSE, and MAE, and adopting a recursive forecasting approach. Finally, it describes the integration of such models into an interactive Streamlit web application that allows exploration of historical stock trends and prediction of future prices in real time.

A diagram of a software development process

AI-generated content may be incorrect.

### **Approach**

A diagram of a data processing process

AI-generated content may be incorrect.

#### **Data Collection**

A screen shot of a computer program

AI-generated content may be incorrect.

It depends on the quality and pertinence of the data it learns for the success of any strong machine learning model. Data acquisition for the project centered on the utilization of Yahoo Finance's API (yfinance), a well-developed and reliable source of past stock prices. I used Yahoo Finance because of the wide coverage they have across global markets, ease and convenience in access and retrieval, and the potential for the retrieval of high-frequency data including Open, High, Low, Close (OHLC) prices and volume. I focused on these parameters because they have application universally across asset classes so that the model would be capable of making cross-asset class generalizations between stocks, cryptocurrencies, commodities, and indices. Bitcoin (BTC-USD) and Google (GOOG), for instance, were employed as base data in a way that included high-volatility cryptocurrencies and stable equities, respectively, so that the model would be able to handle a wide variety of behavior.

Data preprocessing involved a series of steps with the goal of consistency and reliability. Missing values were first imputed, a typical characteristic of stock data based on the closing of the stock market, the celebration of a holiday, or other technical breakdowns. Gaps were filled using a forward-fill procedure, and missing values were filled using the last observation. This ensures temporal consistency without the addition of artificial noise. Outliers were identified using the Interquartile Range (IQR) procedure. Outliers were tagged as data points greater than the mean by 1.5 times the IQR and were smoothed using a rolling median over a 7-day window. This procedure in place of mean-based smoothing for the purpose of diminishing the impact of outliers.

A computer screen shot of a program

AI-generated content may be incorrect.

Normalization was then performed, with MinMaxScaler being employed in rescaling raw prices into a [0, 1] scale. It's a critical step for the use with neural networks because it prevents features with greater magnitudes (i.e., Bitcoin's 50,000prices)from dominating those with smaller ranges (ifGoogle’s$100–$200 range). The scaler was only imposed on the training data in order not affect data leakages so the model’s performance metrics would be unbiased. Temporal integrity was also checked by the count of trading days per year. It had the Bitcoin dataset of 2015–2025 with 3,654 entries and the dataset of Google in the year 2005–2025 with a countof5,833 entries.

Preprocessing difficulties involved managing non-standard trading hours for the cryptocurrency markets as well as volume adjustment for reporting discrepancies done at the exchange level. I addressed these by converting timestamps into UTC and aggregating exchange-by-day volume for the cryptocurrencies. Final preprocessed data were as Pandas Data Frames indexed by date and stored as Parquet files for rapid retrieval as well as taking into consideration distributed computing architectures.

#### **Data Preprocessing**

A screen shot of a computer program

AI-generated content may be incorrect.

Preprocessing data forms the foundation for the development of any deep learning model, particularly for time-series forecasting. In the current project, as I venture into predicting future prices of the GOOG stock and the BTC-USD with the aid of the LSTM networks, appropriate preprocessing becomes the utmost consideration in getting the raw data into a shape that will allow the model to learn and generalize well. This section provides a step-by-step illustration of the preprocessing pipeline employed in the current research. It provides the various stages involved—from preprocessing and normalization to windowing and reshaping for LSTM input—and elaborates on reasons why each of the above stages had to be performed and the challenges encountered when performing them.

The preprocessing pipeline began with the first step of understanding the very nature of the data I was dealing with. Financial time-series data involves a special nature with strong temporal dependences, inherent noise, and non-stationarity. Stock and cryptocurrency markets both have trends, cycles, and spike volatilities with varying frequency and magnitude. Stock prices at places like Google reflect the patterns in conventional equity markets as they experience normal trading days and weekends and holidays when they remain shut. Bitcoin prices, being operational on a 24x7 schedule, form different kinds of dynamics such as heightened volatility and round-the-clock price movements. Preprocessing thus had to be such that it could accommodate two such disparate time-series data and yet have the same methodological orientation.

These data were initially retrieved using the Python library yfinance via Yahoo Finance. All these data had a few features such as Open, High, Low, Close, Adj Close, and Volume. For the purpose of the project, I have taken into consideration the ‘Close’ price alone because it is generally the most stable and last daily level, often used by analysts and traders. With the features thus constrained, the first thing done was the removal of missing or null values and preparation of the data for analysis. Missing values for time-series stock data could be for a variety of reasons all the way up to public holidays, data outage, or data web-scraping discrepancies. Missing public holidays and weekends when the markets are shut are the standard with stock data. In this case, I have not interpolated these missing values since this would add artificial trends and lower the dataset's realism. With the data for the cryptocurrencies, which ideally should be continuous with no missing days, I found random missing values, possibly because of API errors or transient data lags. All these were forward-fill imputed, which retains the last valid observation and is appropriate if prices are unlikely to be very volatile between consecutive points.

A screenshot of a computer

AI-generated content may be incorrect.

Following data cleaning, I proceeded with reformatting the datetime column and setting the same as the index. Conversion of the 'Date' column into a datetime object and setting the same as the index played a significant role in enabling time-series operations. It helped us slice, plot, and arrange well in the case of windowing. I made use of line plots after data cleaning for visual data inspection to identify outliers, trends, or sharp changes. I used the plots to validate the assumption of data continuity, and they gave the first impressions about the outliers that might be impacting the learning.

A screen shot of a computer

AI-generated content may be incorrect.

Next came the feature scaling or normalization. Neural networks, particularly the gradient descent-based ones, are sensitive to the scale of the input. In case the features have a very large scale, the network might not be able to converge or would take a very long time and might not be very effective. To circumvent this, I normalized the closing prices using Min-Max normalization into a scale between 0 and 1. It maintains the distribution as well as the relation among values but gets it into a uniform scale, which is ideal for models based on deep learning. I used the MinMaxScaler class in the scikit-learn library for implementing this transformation. I needed to only fit the scaler into the training set and use the same on the test set to not leak the data—a very important practice in machine learning.

One of the primary data preprocessing steps for LSTM was the formation of the sequence using sliding windows. As opposed to other conventional machine learning algorithms based on a point, LSTM models are learned over a series of observations. I thus had to transform the 1D time-series into a supervised learning framework in which the input would be a sequence of past observations and the output would be the observation that would follow. I achieved this by specifying a fixed window length, say 60 in the experiments, so the model would use the last 60 days' closing prices for the prediction of the next day's price. I determined the length of the window empirically as well as based on the observation that a two-month long window captures enough behavior in the marketplace without overfitting the model.

Windowing was done using a custom function that cycled over the dataset, adding rolling series into input (X) and the next value into the label (y). These series were added into numpy arrays for performance. Once the sequence data had been created, this had to be reshaped into the input shape expected by the LSTM layer in Keras, i.e., a three-dimensional input shape: [samples, time steps, features]. I had a univariate feature (normalized closing price), so each series would have needed to be reshaped accordingly into this structure. How critical this reshape is cannot be over-emphasized, as LSTM layers are very sensitive with regards to input shape and will err, or be learning sub-optimally, if this shape isn't honored.

Following generation and sequence shaping, the data were split into a test dataset and a training dataset. A split of 80/20, using which the model would be trained using the first 80% data and tested using the last 20%, was employed. Division by the time sequence was done on purpose in line with the practice followed in time-series forecasting whereby random scrambling of the data would be precluded in service of being able to preserve the timeline. Model training using past data and testing using future data more closely simulates real-world forecasting environments, whereby a more authentic test of the generalizability of the model would be achieved.

For the sake of repeatability and maintaining the workflow with the organization, all the preprocessing operations were refactored into reusable functions. This modularity enabled us to use the same pipeline with little duplicated code on different datasets. The preprocessing module contained a series of functions including load\_and\_clean\_data(), scale\_and\_window\_data(), reshape\_for\_lstm(), and split\_train\_test(). This organization proved itself gold worth when I scaled the model for usage with both GOOG and BTC-USD datasets because it enabled us to maintain consistency and prevent errors.

Although the pipeline itself was well-organized, preprocessing caused a degree of problems. Most significant were the methods for addressing the volatility in the prices of Bitcoin. In contrast with the typical stock prices, which fluctuate incrementally, Bitcoin's undergone dramatic fluctuations over short durations. While enlightening, this made the volatility problematic with respect to normalization since the scaled data were largely skewed by massive spikes, rendering even the smallest movements imperceptible. In response, I experimented with a variety of different methods for scaling but ultimately stuck with the use of Min-Max scaling owing to the way it interacted with the optimized use of windowing.

#### **Model Selection**

Model selection is one of the most critical aspects of any machine learning or deep learning project. The model acts as the core engine that learns from the data, identifies patterns, and generates predictions. In this project, which involves forecasting the future prices of both Google stock (GOOG) and Bitcoin (BTC-USD), the model had to be capable of handling complex, non-linear, and highly temporal data. Time-series forecasting brings its own set of challenges, such as autocorrelation, seasonality, noise, and irregular trends. Therefore, choosing the appropriate model required a thorough understanding of various available options, their strengths, and their limitations in the context of financial data.

Initially, I considered a broad range of models that are typically used for time-series forecasting. These included traditional statistical models like ARIMA (Auto-Regressive Integrated Moving Average), exponential smoothing techniques such as Holt-Winters, and modern machine learning models like Decision Trees, Random Forests, and gradient-boosted machines. Each of these models offers unique benefits. For instance, ARIMA is well-suited for univariate time-series forecasting with linear trends and seasonality, and it provides explainable components such as autoregression and moving averages. Similarly, Random Forests are excellent for capturing complex relationships between variables, and they handle non-linearities well without requiring extensive parameter tuning. However, these models have limitations when applied to sequential data, especially when the relationship between past and future values is complex, long-term dependencies are significant, or the dataset is non-stationary—as is often the case with financial data.

After careful consideration, I chose to use Long Short-Term Memory (LSTM) networks for this project. LSTM is a type of Recurrent Neural Network (RNN), specifically designed to overcome the shortcomings of traditional RNNs, which struggle with vanishing and exploding gradients during training. LSTMs are exceptionally well-suited for time series because they can retain and leverage long-term dependencies. In the context of financial forecasting, this is crucial. For example, an unusual market movement from two months ago may still influence investor behavior and pricing today, and LSTM’s ability to “remember” relevant past information makes it a powerful tool in such scenarios.

The internal architecture of LSTM is what sets it apart. Unlike traditional RNNs that have a single repeating neural network structure, LSTM introduces a more complex memory cell mechanism consisting of gates—specifically the input gate, forget gate, and output gate. These gates regulate the flow of information through the network, allowing it to selectively remember or forget specific data points. The input gate determines how much new information should be added to the cell state, the forget gate controls which parts of the existing memory should be discarded, and the output gate decides what the next hidden state should be. This gating mechanism provides LSTMs with a dynamic memory, enabling them to learn from both short- and long-term trends, which is essential in modeling price behavior that does not follow simple linear patterns.

In designing our LSTM model, I aimed to strike a balance between model complexity and generalization ability. Overly complex models with too many parameters are prone to overfitting, especially when the training dataset is not very large. On the other hand, shallow models may lack the capacity to capture the intricate relationships within the data. After conducting multiple experiments and referencing previous literature on financial time-series forecasting, I opted for a moderately deep LSTM network architecture. The core architecture consisted of an input LSTM layer with 50 units, followed by a Dropout layer, a second LSTM layer with 50 units, another Dropout layer, and finally a Dense output layer with one neuron to predict the next closing price. This multi-layered architecture allowed the model to learn hierarchical representations of the data—where the first LSTM layer could capture local temporal dependencies, and the second layer could learn higher-level abstractions.

The inclusion of Dropout layers after each LSTM layer was a strategic decision aimed at preventing overfitting. Dropout works by randomly setting a fraction of the input units to zero at each update during training time, which forces the network to not rely on specific neurons and promotes generalization. In our model, I used a dropout rate of 0.2 (or 20%), which is commonly accepted as a good starting point in LSTM networks. This helped the model generalize better to unseen data, a critical requirement for making reliable future forecasts.

The output layer of our model was a Dense layer with a single neuron and a linear activation function. The choice of a linear activation in the final layer was deliberate and appropriate for a regression task such as price forecasting. Unlike classification problems, where the output needs to be bound within a range (e.g., probabilities between 0 and 1), regression tasks benefit from a linear activation that allows the model to output a continuous range of values without any constraint. This configuration ensured that the model could learn to predict a price value directly without transformation.

Selecting the right loss function and optimizer was another key step in model configuration. Since our problem is a regression task, I used Mean Squared Error (MSE) as the loss function. MSE is a widely used metric in regression problems and measures the average squared difference between the actual and predicted values. It penalizes large errors more heavily than small ones, which is suitable in financial forecasting, where large prediction deviations can have severe implications. For optimization, I used the Adam optimizer, a state-of-the-art method that combines the advantages of both RMSprop and Stochastic Gradient Descent with momentum. Adam dynamically adjusts the learning rate based on the first and second moments of the gradients, which speeds up convergence and handles noisy data well.

I also explored hyperparameter tuning to improve model performance. This involved experimenting with different configurations for the number of LSTM units (ranging from 20 to 100), dropout rates (0.1 to 0.5), batch sizes (32, 64, 128), and learning rates (0.001, 0.0005, 0.0001). The goal was to find the optimal setup that yielded the lowest validation loss without overfitting. This was achieved through a manual grid search and performance monitoring using validation datasets. In future iterations, more sophisticated tuning methods like Randomized Search or Bayesian Optimization could be employed to further streamline the process.

To assess the suitability of our LSTM model compared to alternatives, I implemented a baseline model using a naive persistence method, which simply assumes that the next day’s closing price will be the same as today’s. I also briefly tested a Dense Neural Network (DNN) without any recurrence, but the performance was significantly inferior, especially on volatile data like Bitcoin. These comparisons validated the hypothesis that LSTM’s sequential learning capability provides a significant edge in forecasting applications where time dependencies are crucial.

Moreover, I considered the possibility of using bidirectional LSTM layers, which process sequences both forward and backward. However, since I am forecasting future data and not classifying sequences where past and future context are equally important, bidirectional LSTM was not suitable in this specific scenario. Similarly, more advanced architectures such as attention-based models or transformers were deemed out of scope for this project, though they represent promising directions for future work as they have shown impressive results in sequence modeling across various domains.

In terms of computational efficiency, training the LSTM model required access to GPU acceleration to significantly reduce training time. While LSTM models can be trained on CPU, the time required to train multiple epochs over large datasets becomes impractical. Our implementation was carried out using TensorFlow and Keras, which seamlessly utilize GPU resources when available. I also leveraged callbacks such as EarlyStopping to monitor the validation loss and halt training once it stopped improving, thereby saving computation time and preventing overfitting.

Lastly, the model architecture was designed to be flexible and adaptable across different datasets. The same LSTM structure was applied to both the Google stock and Bitcoin datasets with only minor adjustments in input shape, ensuring a consistent approach and allowing for a fair comparison between the two financial instruments. This generalizability is a strength of the chosen model design and supports future extension of the project to other stocks or cryptocurrencies.

In conclusion, the decision to use LSTM for time-series forecasting in this project was driven by its superior ability to handle temporal dependencies, its proven performance in financial modeling tasks, and its scalability across different types of data. The model architecture was thoughtfully designed to balance complexity and generalization, incorporating dropout layers for regularization, linear activation for regression, and a robust optimizer for efficient training. By comparing its performance with simpler models, I validated its appropriateness and set a strong foundation for the training and evaluation phases that follow. LSTM remains one of the most powerful tools for time-series forecasting, and its successful application in this project underscores its value in the domain of financial analytics.

#### **Model Training and Evaluation**

With the model architecture completed, model training and testing followed as the second essential aspect in our deep learning venture. It's a point where the theoretical promise of the model comes alive—by the model's fitting into historical data, calibration of inner parameters through backpropagation, and analysis of how well the model performs against past unseen data. Given the nature of the project being a prediction over time-series data—with increasingly more accurately daily closing prices of GOOG and BTC-USD, the LSTM model's training consisted of very detailed data set creation and cleaning, judicious selection of time windows, much hyperparameter tuning, and stringent test protocols in an effort to attain prediction accuracy as well as reliability.

The first and foremost component in the learning process was preparing the data for use by the LSTM model. While normal machine learning models operate with each data row in a silo, LSTM models have some awareness of the time sequence of input data. It was therefore essential to get the data into sequence form. I framed our time-series data as a supervised learning problem using a sliding window approach. That involved choosing some number of past time intervals (also referred to as “look-back” windows) with which to forecast the next time step. In playing with changing the widths of the windows, I opted for a look-back period of 60 days. That is, the model had the closing prices for the last 60 days input for each forecast for a stock or a cryptocurrency price and had to make an educated guess as to what the 61st day would be. Not only did this provide the LSTM with a nice amount of history with which to work, but it fit well into notions generally accepted in the convention for forecasting time-series data in the finance arena.

After partitioning the data into sequences, the dataset was split into a test dataset and a train dataset. In the majority of machine learning problems, data can be normally shuffled first before being split into a test dataset and a train dataset. In time series data, however, coherence by time must be preserved. So, I split the data chronologically, I split the sequence data into the first 80% for train data and the last 20% for test data. It enabled the model to be trained on historical data and tested on future data, which simulates a real-world forecasting situation with the future never being known.

Before feeding the data into the LSTM model, normalization through MinMax scaling had already been performed. This had been done as a hedge against destabilizing the learning as well as against the varied ranges of prices disrupting the learning by the model. LSTM models are particularly vulnerable to the scale of the input features; unnormalized large numerical data can potentially make the gradients too big or too small, resulting in learning at a very slow pace or even failure to train. MinMaxScaler had scaled the closing prices into a scale between 0 and 1, maintaining the relative gap among the values but squashing the magnitude. Reverse transformation had been done after the prediction so that values could be scaled back into the original scale for ease of interpretation as well as visual observation.

I specified a finite number of epochs and batch sizes in the model's training procedure. I began training the model with more than 100 epochs and a batch size of 64. I had determined these as a result of a few experiments so as to attain the balance between the model's stability and the speed of convergence. A batch context can be explained as a full iteration over the entire dataset for training, whereas a batch size describes the quantity of samples being fed into the model before updating the weights. High batch sizes make the model's training faster but with a less general model, whereas the models with lower batch sizes are stable but at the expense of the model's speed being trained. In this context, a batch-size of 64 provided the optimal balance. Additionally, in order to avoid overfitting and early stopping in the case of no improvement, I used the EarlyStopping function of Keras. It tracked the validation loss and stopped the training if the model didn't improve over a series of consecutive epochs (patience parameter = 10) so that I save computational power as well as avoid over-training.

As the model learned over a few ephochs, it continuously updated its internal weights and biases such that the loss function kept decreasing, which in our case equated to Mean Squared Error (MSE). Our regression problem had a natural fit with MSE as it penalizes larger errors more severely, and so the model felt the pressure to minimize large errors in the predicted prices. While learning in the model, I were monitoring the training loss as well as the validating loss so as to analyze the model’s generalization ability. A declining training loss with a rising validating loss generally signifies overfitting, but the two declining together indicates effective learning. In the experiments conducted by us, I found the LSTM model had consistent convergence with both the losses decreasing gradually and settling in the range of the epoch 80–90, particularly when the Dropout layers were used.

With training finished, the model was then evaluated using the test set, which had been completely hidden the whole time during training. One objective of this exercise was to measure how well the model was able to generalize—how well it would be able to make predictions about future prices given previous trends. I generated predictions by passing the test sequences through the trained model and compared the resulting values against actual closing prices. I employed a range of measurements to quantify model performance. In addition to the utilization of MSE, I used Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE). RMSE places more weight on larger errors, while MAE treats all instances of actual deviation equally. These measurements gave us a well-rounded insight into the accuracy of the prediction, allowing us to compare how closely the model's prediction would follow actual marketplace activity. On the dataset for the stock prices of Google, the LSTM model had an RMSE of around 21.65, so the stock price varied on average by some $21.65 away from the actual. Keeping in mind that the stock price of Google fluctuates in hundreds and thousands of dollars, the error margin makes sense for short-term prediction. On the dataset for the Bitcoin asset, notoriously volatile and unpredictable, the model's performance was slightly lower, with the RMSE at around 780. Bitcoin's behavior on a daily basis could be outrageous, and the asset's price reacts to a greater variety of external events than a normal stock. Yet, the LSTM model still managed to pick up dominant trends and reversals, a sign of the model's potential even for volatile assets.

Another critical component of the performance evaluation procedure involved a plotting of the predictions. By comparing predicted vs actual prices in a time-series plot, I could ascertain the degree by which model output matched actual data. From the plots, it appeared the LSTM model performed well at tracking the general direction of the trend in prices, including the highs and lows, but would consistently trail by a little when experiencing sharp spikes and drops. This accounts for LSTM's capacity for tracking steady trends but failing at catching the sudden changes in the marketplace, which will likely be the kind that will call for additional exogenous features (i.e., news sentiment, volume traded) so as to effectively model.

I supplemented the analysis throughout with a look at the model's residuals that is, predicted minus actual. I used a residual plot to check whether the model had systematic errors. Predominantly scatter-plotted residual with no pattern would be ideal, indicating the model learned as much as it could with the data. In our case, residual plots revealed around uniformly scatter, with the model being resistant to biased errors or omitting significant structural information from the data. Slightly higher residual when the security's underlying increased significantly reflected that LSTM performed very well but still holds room for improvement for outliers.

Another aspect considered in the analysis was the rolling forecast strategy. Rather than forecasting one day ahead for each test case, I imitated a realistic forecasting scenario in which the model predicted prices recursively. That is, the forecasted value at each iteration was fed back into the input series to predict the next. Recursion mirrors how future price estimations are used by financial analysts with models. While the strategy makes forecasting over long time frames possible, it introduces the effect of compounding errors. A little prediction error at one time step has the potential to carry over and propagate at the next time steps. Our model predicted well for short-term recursive forecasts (5 days ahead), after which the performance began to degrade progressively. This mirrors a general tradeoff in time series forecasting—horizon vs. accuracy. For a detailed vision of how well our model performs, I have also plotted a comparison with a basic Moving Average (MA) model, which averages the last 'n' days' prices and then projects the average as the future day's price. Our MA model performed much worse than the LSTM model when the markets were fluctuating fast. While MA makes a good comparison, it lacks the capability to pick up patterns, cyclic trends, and context—which reflects the LSTM's strength in catching intricate time-series patterns. In addition to model-based estimation, I also emphasized interpretability and trust in estimates. While generally black-boxed, in finance, it is important to know what the model learned as decision-makers have to make decisions based on model estimates. While LSTM does not have inherent feature importance, SHAP (SHapley Additive exPlanations) and attention mechanisms can be added in future work so that it can be more explainable. In the short run, as a policy, I attempted to build trust based on robust validation and detailed analysis on errors.

#### **Prediction and Visualization**

The end goal with the development and training of any machine learning model, and specifically for this research, predicting the Google and Bitcoin prices with a time-series, is the testing of the model's use in real-world scenarios as well as interpreting the results that it generates. I present the performance of the LSTM model built in the context of the prices of Google and Bitcoin, explain the results by context with the markets' dynamics, and compare the performance among the different experiments. I compare not only the model's strengths and limitations but put the research into the context of the use of deep learning in predicting the stock.

To begin, after I had trained the LSTM model and run it on previously untested test data, I graphed predicted vs. actual prices for all assets. I immediately had visual evidence of the model's ability to capture underlying patterns with these graphs. On the Google stock prices, the LSTM predicted lines closely tracked the actual closing prices throughout the test run, particularly in intervals of modest price fluctuation. Smooth predicted lines were evidence of LSTM's competence at tracking continuous change and cyclical fluctuation. Though the model would fall behind sudden pivoting points—a natural shortcoming with the lack of exogenous variables—it could still track the direction of price fluctuation, the biggest consideration in financial forecasting for investors and algorithmic trading programs alike.

Bitcoin (BTC-USD) dataset presented an additional challenge with its inherent volatility as well as exposure to external market forces in the form of regulatory news, opinionated tweets, and big macroeconomic moves. Despite all this, the LSTM model still tracked the overall direction of price fluctuations, accurately forecasting periods of upside or down trends. However, with Bitcoin's tendency to make frequent and drastic swings, the model would at times under-predict the amplitude of sudden shots followed by reversals. For instance, in regions where BTC recorded rapid-fire gains of more than $1,000 a day, the LSTM forecasts were more subdued. That the model would be so is understandable, as it had been trained on historical prices but had not been shown any other context or fundamental indicators in the market.

Quantitatively, in performance terms, I tested the model against three significant measures of error: Mean Squared Error, Root Mean Squared Error, and Mean Absolute Error. With these measures, I were able to gauge how far, on average, the estimates were away from actual prices. In the case of Google's stock, the RMSE averaged around 21.65 and the MAE around 15.4. Given the average stock price for the test period for Google hovered around $2,800, the model's average deviation of $21 represented a lesser than a 1% error—a performance optimum for financial data. A low MAE score again verified the model's reliability in making quite precise estimations with no outliers.

Bitcoin performance, as less precise, still impressed. The BTC predictions' MAE had been in the order of 1000, with MAE close to 620. With the Bitcoin price in the test interval varying between $80,000 and $96,000. Although more constricted than the performance with Google, such an accuracy degree isn't bad with the degree of volatility associated with cryptocurrencies. In addition, the LSTM correctly predicted the direction each time, and being correct direction-wise more frequently than incorrect can often be more useful than exact accuracy with the majority of trading tactics, primarily momentum or directionally based trading.

For the purpose of comparison, I have also used a baseline model with a basic Moving Average (MA) algorithm. While the method produced a smooth trend line estimation,

It placed significant importance on the advantage of utilizing methods such as the LSTM with the capacity to learn sequential dependencies rather than using stationary averaging methods as they depend on the data being stationary.

The second essential part of our review of the results was residual analysis—the variations between predicted and actual values. In a good model, ideally the residuals would be spread equally around zero with no pattern. In examining for Google stock, residual plots were well-distributed with no significant autocorrelation, indicating the model, if anything, lacked little in the data. In Bitcoin, the residuals were more spread out and had moderate clustering when the asset prices were volatile, again reflecting the challenges inherent in modeling such a volatile stock with a univariate model. This analysis with the residuals proved helpful in validating the model as well as the placement of the model's weaknesses.

I also tested the model's performance with a recursive forecast setup—where it makes a forecast for the next day, feeds that forecast back into the input sequence, and then makes a forecast with it for the next day, and so forth. This mimics a real-world operational scenario where future data are not yet known and the forecasts have to be computed sequentially. Not surprisingly, the model's accuracy suffered as the forecast horizon increased. While the 1-day ahead forecast was incredibly precise, the 5-day ahead forecasts began diverging more appreciably, mostly for Bitcoin. This worsening is an old issue with time-series modeling known as the accumulation of errors—in which minor errors compound over a series of stages. Still, the model captured a realistic pattern in the majority of the instances, which hints at it being used for short-term forecasting or trend analysis.

One interesting aspect which can be discussed is the model's conservativeness in smoothing over wild fluctuations. LSTMs possess the built-in capacity for discovering and representing long-term patterns, so they get cautious when they perceive outliers or wild behavior. If Bitcoin experienced a wild price surge resulting from a significant announcement, say, then the model forecast a more modest increase. While this may initially be seen as a failure, it instead reflects a desirable property in some instances: robustness. A model too sensitive, reacting to every fluctuation, will likely generate jittery and unpredictable output. So, this conservativeness can be seen as a balance between sensitivity and generalization.

Aside from numerical and visual evaluations, business use is considerable. For stock/option traders, a modest improvement in prediction accuracy can translate into massive returns, especially if compounded over thousands or hundreds of trades. Institutionally invested groups might find the LSTM model useful as a decision support tool to validate output by other model alternatives, assist with risk management, or provide asset-allocation suggestions. On the other hand, individual traders might have the direction accuracy of the model included in trading bots or platforms providing buy/sell suggestions based on momentum.

This does identify a set of limitations. First, our LSTM model had been formally trained only on historical prices (univariate), and the relevant features such as trading volume, social sentiment, macroeconomic data, or even indicators such as RSI and MACD had not been included. It would undoubtedly boost the predictive power if these variables were input into a multivariate LSTM. Second, whereas the model generalised well enough over our test data, actual market performance would need close watch on a rolling basis, with the attendant dynamism thereto. Third, LSTM still lacks explainability—while stable, the inner mechanisms of the model remain much a mystery for the non-technologist.

Such constraints could be overcome by future advances, including the use of exogenous data (news headlines, economic news), the use of attentions with a focus on important time steps in the series, or even the use of LSTM with other models, including ARIMA/Exponential Smoothing. Experimentation with other architectures, including GRU (Gated Recurrent Units) or Transformer models for the series, could provide greater possibilities for enhanced performance and faster learning.

In summary, the results of the present work unambiguously validate the use of LSTM networks in prediction of financial time series. Notwithstanding the inherent challenges in handling noisy, non-linear, and volatile data streams, the model proved itself to possess a superb ability in learning from past patterns and making rational anticipations. In comparing two hugely disparate assets—Google stock and Bitcoin—I highlighted the universality and impartiality of the LSTM technique per asset class. Our findings provide a firm foundation for further inquiry and development, and provide real-world insight into the evolving nature of AI-based prediction.

#### **Web App:**

A screenshot of a computer

AI-generated content may be incorrect.

To make the machine learning model interactive, easy to use, and pre-trained, the interactive web application using Streamlit, an open-source Python data-driven application development library, was created by me. A full-fledged front-end application wherein students can analyze past stock prices, visualize the technical indicators, and get past and future stock prices predicted in real time is the web application.

A screenshot of a computer

AI-generated content may be incorrect.

The program is divided into two main function tabs: Stock Analysis and Future Prediction. In the Stock Analysis tab, the user will be able to input any stock ticker symbol (for example, GOOG, TSLA, AAPL), select a target date range, and fetch past stock data using the Yahoo Finance API. Then, the system will graph the latest data points, the 100-day Moving Averages, and the 250-day Moving Averages.

A graph of a price

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

The user will be presented with an interactive line chart using Plotly that superimposes the historical closing stock price of the stock and the moving averages so that trends will be easily analyzed.

A graph on a screen

AI-generated content may be incorrect.

One of the functionalities with the highest importance in this tab includes the integration of the pre-trained model into the real-world. This model, pre-trained with preprocessed stock closing prices using an LSTM architecture, is loaded and employed for predicting past prices. Scaling down the closing prices data, the web application renders sliding windows consisting of 100-day series for input into the model before making the forecast prediction inverse-transformed into original prices. By plotting the predicted values against actual closing prices, the users are shown a graphical display of model performance against untrained data.

A screen shot of a black screen

AI-generated content may be incorrect.

In the Future Prediction tab, future stock prices for a specified number of days (e.g., 10, 15, 30 days) can be predicted. The app takes the last 100 days' closing prices as the input and recursively feeds the input data one day at a time into the LSTM model to forecast future prices. The predicted future price obtained is added as the input sequence in the next forecast and simulates a rolling forecast. Future prices forecast by the app are listed in a table with dynamically calculated future dates. Future prices are plotted by the app in a readable graph, providing a good vision for the predicted future trend in the market.

Together, the web application bridges the gap between intelligent machine learning algorithms and natural decision aids. Coupled with the integration of modern UI modules, real-time forecast, interactive visualizations, and dynamic input facilities, the application serves as a useful, informative tool for traders, investors, and researchers who want to analyze and forecast stock market trends.

## **Future Directions and Improvement**

While this research has demonstrated the potential capacity of LSTM networks to forecast financial time-series data based on past price trends, it is also a precursor to far more widespread and sophisticated future work. The implications observed in this project, particularly in forecasting the price direction of assets as different in nature as Google stock and Bitcoin, are a source of confidence in the deep learning model employed. But financial markets in their very nature are complex and are determined by not only price movements in the past but by a set of externalities such as market psychology, macroeconomics, world events, geo-political activities, social networking and global news-driven speculation movements. Thus, the next point in the organic development of this work is to incorporate these externalities into the model and therefore create a fuller and stronger predicting tool.

One of the most important additions being considered for the next phase of this project is the integration of social media sentiment analysis into the forecasting process. In today's internet-dependent world, sources like Twitter, Reddit, have emerged as powerful platforms where views, rumors, news, and information about stocks and cryptocurrencies propagate and get absorbed by the market participants at a phenomenal speed (Bollen, Mao and Zeng, 2011). Especially valuable for risky assets like Bitcoin is the authority of one influential person—CEO, regulator, or public figure—to move short-term prices by a single tweet. One can quantify emotive sentiment (positive, neutral, or negative) on an individual asset based on real-time data from these channels as well as sentiment analysis employing natural language processing models like BERT or RoBERTa.

For example, a sudden rise in Bitcoin-supportive tweets that include hashtags such as "*#bullish*," "*#to the moon*," or "*#buy the dip*" can be an indicator of a buy sentiment on the market that can be an immediate lead-in to an increase in price (Greyling and Rossouw, 2025). These sentiment scores can then be transformed into numerical features and fed into a multivariate LSTM model along with historical prices. This integration of quantitative and qualitative data is done in the hopes of making the model more predictive, especially for assets like cryptocurrencies that are highly sensitive to public sentiment and collective behavior.

Another area of interest is integrating financial and political news analysis into the model. Financial markets react in real-time to economic signals, policy news, central bank initiatives, and international events. For instance, inflation rate news headlines, interest rate releases, company profits announcements, or regulatory crackdown headlines can significantly influence investor mood and bring about gigantic price movements. Similarly, political events such as election times, war threats, trade talks, or sanctions can indirectly influence market activity (ARQ Wealth Advisors, 2020). By headline and article web scraping from reputable financial news outlets (e.g., Bloomberg, Reuters, CNBC), and processing such content with advanced NLP models, the model would learn to identify patterns between specific keywords or stories and their subsequent market impacts.

Such news sentiment extraction would be quantifiable on a time-series level e.g., in units of daily average sentiment polarity—and aligned with historical price data in order to give additional context to the model. Furthermore, by tracking geopolitical news across the globe, it is possible to add features that provide early warning or risk indications for asset classes like tech stocks or cryptocurrencies that have a tendency to behave as risk-on assets in times of uncertainty (US Bank, 2024). This would then turn the model from a basic price-based predictor to a multi-dimensional market-aware forecasting engine.

In addition, later iterations of the model may include visual and statistical integration of technical analysis graphs—such as moving averages, Bollinger Bands, RSI (Relative Strength Index), MACD (Moving Average Convergence Divergence), and Fibonacci retracements. These charts are analyst and trader favorites, and their prediction trends could be encoded as additional features. With the capacity of deep learning to handle high-dimensional data, these technical indicators can be utilized as overlay signals to reinforce or adjust the model's output based on historically sound patterns.

With these enhancements in place for sentiment analysis, news impact analysis, and technical chart patterns, the next step would be to convert this advanced model into a full-scale application. The mission is to develop a scalable, smart, and intuitive platform where users can track, visualize, and even act upon asset price predictions powered by a robust AI engine. The app would allow users to select assets of interest (e.g., Bitcoin, Ethereum, Google, Amazon), view real-time sentiment scores, review past and predicted price charts, and get forecast-based notifications for potential market movement. The users would be able to personalize their dashboards with favorite coins or stocks, track portfolio projections, and even get daily or weekly market projections.

From a development standpoint, the app would require good backend architecture—most likely through the use of cloud infrastructure for ingestion of real-time data (through APIs for news, tweets, and prices) and database storage, along with hosting models. On the front-end side, frameworks like React or Vue.js could render an interactive user interface, while RESTful APIs would connect the front end to the AI engine. The predictive model itself could be containerized using Docker and deployed through scalable platforms like AWS, Google Cloud, or Azure, ensuring high availability and performance.

The objective is not just to make a self-contained utility but to compete with existing market leaders such as Coinbase, CoinMarketCap, and Binance—not necessarily in the exchange functionality arena, but rather in the forecasting and analytics arena. While these websites do offer historical data, basic charts, and news feeds, they fail to deliver actionable forecasts or AI-driven insights related to user behavior and sentiment (Chimelu, 2021). By integrating a model blending deep learning with real-time sentiment tracking and global market insights, the app would be able to position itself as a "smart financial forecasting engine" that would empower novice as well as seasoned investors.

Other market apps just display charts and numbers, this tool would give predictive insight telling users what can happen next, why, and based on what indicators. With such capabilities as trend confidence scores, volatility estimates, risk-level classification, and NLP-enabled news digests, users would not only receive predictions but also the reasons why they were made in simple human language. Additionally, with features like explainable AI (XAI), users would be able to understand how various features (e.g., sentiment spike, breaking news) affected a prediction, thereby increasing transparency and trust in the system.

To enable continuous learning and real-world responsiveness, the system could even use online learning, wherein the model is retrained or refined periodically as fresh data arrives. This would enable the model to stay sensitive to shifting market dynamics, emergent trends, and black swans. In addition to feedback cycles from user activities (e.g., how many users followed a signal and how much return on investment they have), the system could become self-improving over time.

Ultimately, a monetization strategy could be to offer tiered levels of access—from free access to general forecasts to premium access with high-frequency forecasts, trading signals, portfolio recommendations, and higher-end charting tools. API access could be offered to institutional clients or fintech developers who wish to integrate intelligent forecasting capabilities into their own platforms.

To sum it up, the future research being conducted as a continuation of this work is not merely about improved prediction rates—it's about an entire paradigm shift in how people understand and interact with financial data. By combining social media sentiment, financial and political news analysis, historical trends, and technical indicators into a single unified deep learning paradigm and serving it up via an application that prioritizes user experience, this project has the aim of raising the bar in the domain of fintech. It is a bold step towards democratizing predictive financial intelligence, where anyone, anywhere, can make educated choices powered by AI. As the application keeps developing, it could eventually become a game-changing platform that takes on the biggest players in crypto and stock analysis—offering not just data, but actionable foresight.

## **Conclusion**

This project was successful in effectively analyzing the application of Long Short-Term Memory (LSTM) neural networks in the forecasting of financial asset prices from the past. By working on two different case studies—Google shares and Bitcoin—it was successful in showing that deep learning methods can be used in forecasting stable and highly volatile financial assets. The model of LSTM was found to capture temporal relationships effectively and thus proved to be an ideal device in time-series forecasting in finance.

Whereas the current approach was heavily based on historical pricing data, the findings uphold the potential of deep learning in prediction and set the foundation upon which to take it even further. Sentiment analysis, worldwide news analysis, and technical analysis to take the predictability of the model even further are future improvements in consideration. This study is essentially a foundation upon which to develop a mature AI-based application delivering valuable market forecasts to clients.

## **References**

1. ARQ Wealth Advisors. (2020). *How Is the Stock Market Impacted By Politics?* [online] Available at: <https://arqwealth.com/how-is-the-stock-market-impacted-by-politics/>. (Accessed: 20th Feb 2025)
2. Association For Computing Machinery Special Interest Group On Management Of Data and Data, I. (2016). *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* New York, Ny Acm. (Accessed: 20th March 2025)
3. Bahdanau, D., Cho, K. and Bengio, Y. (2014). *Neural Machine Translation by Jointly Learning to Align and Translate*. [online] arXiv.org. Available at: <https://arxiv.org/abs/1409.0473>. (Accessed: 29th Feb 2025)
4. Bollen, J., Mao, H. and Zeng, X. (2011). Twitter mood predicts the stock market. *Journal of Computational Science*, [online] 2(1), pp.1–8. Available at: <https://www.sciencedirect.com/science/article/abs/pii/S187775031100007X>. (Accessed: 27th Feb 2025)
5. Breiman, L. (2001). Random Forests. *Machine Learning*, [online] 45(1), pp.5–32. Available at: <https://link.springer.com/article/10.1023/A:1010933404324>. (Accessed: 21st March 2025)
6. Burton Malkiel’s A Random Walk Down Wall Street. (2018). Macat Library. doi: <https://doi.org/10.4324/9781912281169> , (Accessed: 25th Feb 2025).
7. Chaajer, P., Shah, M. and Kshirsagar, A. (2021). The applications of artificial neural networks, support vector machines, and long-short term memory for stock market prediction. *Decision Analytics Journal*, 2, p.100015. doi: <https://doi.org/10.1016/j.dajour.2021.100015> , (Accessed: 2nd March 2025).
8. Chimelu, G.C. (2021). *Binance vs. Coinbase: Which Should You Choose?* [online] Investopedia. Available at: <https://www.investopedia.com/binance-vs-coinbase-5120852> , [Accessed 5 Apr. 2025].
9. Ding, X., Zhang, Y., Liu, T., & Duan, J. (2015) *Deep learning for event-driven stock prediction | proceedings of the 24th International Conference on Artificial Intelligence*. Available at: <https://dl.acm.org/doi/10.5555/2832415.2832572> , (Accessed: 23 February 2025).
10. Fama, E. (1970). Efficient capital markets: A review of theory and empirical work. *The Journal of Finance*, [online] 25(2), pp.383–417. doi: <https://doi.org/10.2307/2325486> . (Accessed: 23rd Feb 2025)
11. Fischer, T.G., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. *Eur. J. Oper. Res., 270*, 654-669., (Accessed: 25 February 2025).
12. Goodfellow, I., Bengio, Y. and Courville, A. (2016). *Deep Learning*. Cambridge, Massachusetts: The MIT Press (Accessed: 20th Feb 2025).
13. Greyling, T. and Rossouw, S. (2025). *Twitter sentiment and stock market movements: The predictive power of social media*. [online] CEPR. Available at: <https://cepr.org/voxeu/columns/twitter-sentiment-and-stock-market-movements-predictive-power-social-media>? (Accessed: 19th Feb 2025).
14. Higher Intellect Documents (2025). *John J. Murphy - Technical Analysis of the Financial Markets (1999) - Higher Intellect Documents*. [online] Higher Intellect Documents. Available at: <https://preterhuman.net/docs/John_J._Murphy_-_Technical_Analysis_of_the_Financial_Markets_(1999)#google_vignette> . [Accessed 4 Apr. 2025].
15. Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8), pp.1735–1780. doi: <https://doi.org/10.1162/neco.1997.9.8.1735> ,(Accessed: 23rd Feb 2025).
16. Karazmodeh, M., Nasiri, S. and Hashemi, S.M. (2013). Stock Price Forecasting using Support Vector Machines and Improved Particle Swarm Optimization. *Journal of Automation and Control Engineering*, 1(2), pp.173–176. doi: <https://doi.org/10.12720/joace.1.2.173-176> ,(Accessed: 20th Feb 2025).
17. Kim, K. and Han, I. (2000). Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index. *Expert Systems with Applications*, 19(2), pp.125–132. doi: <https://doi.org/10.1016/s0957-4174(00)00027-0> , (Accessed: 20th Feb 2025).
18. Kumar, B.S. and Ravi, V. (2016). A survey of the applications of text mining in financial domain. *Knowledge-Based Systems*, 114, pp.128–147. doi: <https://doi.org/10.1016/j.knosys.2016.10.003> , (Accessed: 20th Feb 2025).
19. Kumbure, M.M., Lohrmann, C., Luukka, P. and Porras, J. (2022). Machine learning techniques and data for stock market forecasting: A literature review. *Expert Systems with Applications*, [online] 197, p.116659. doi:https://doi.org/10.1016/j.eswa.2022.116659 (Accessed: 29th Feb 2025).
20. Madhu, B., Rahman, Md.A., Mukherjee, A., Islam, Md.Z., Roy, R. and Ali, L.E. (2021). A Comparative Study of Support Vector Machine and Artificial Neural Network for Option Price Prediction. *Journal of Computer and Communications*, 09(05), pp.78–91. doi: <https://doi.org/10.4236/jcc.2021.95006> , (Accessed: 20th March 2025).
21. Pai, P.-F. and Lin, C.-S. (2005). A hybrid ARIMA and support vector machines model in stock price forecasting. *Omega*, 33(6), pp.497–505, (Accessed: 15th March 2025).
22. Patel, J., Shah, S., Thakkar, P. and Kotecha, K. (2015). Predicting stock and stock price index movement using Trend Deterministic Data Preparation and machine learning techniques. *Expert Systems with Applications*, [online] 42(1), pp.259–268. doi: <https://doi.org/10.1016/j.eswa.2014.07.040> , (Accessed: 2nd April 2025).
23. Pham Hoang Vuong, Lam Hung Phu, Hong, T., Le Nhat Duy, Pham The Bao and Tan Dat Trinh (2024). A bibliometric literature review of stock price forecasting: From statistical model to deep learning approach. *Science progress*, 107(1). doi:https://doi.org/10.1177/00368504241236557 , (Accessed: 25th Feb 2025).
24. Shiller, R.J. (2003). From Efficient Markets Theory to Behavioral Finance. *Journal of Economic Perspectives*, [online] 17(1), pp.83–104. Available at: <https://ideas.repec.org/a/aea/jecper/v17y2003i1p83-104.html> , (Accessed: 25th Feb 2025).
25. US Bank (2024). *How Presidential Elections Affect the Stock Market | U.S. Bank*. [online] www.usbank.com. Available at: <https://www.usbank.com/investing/financial-perspectives/market-news/how-presidential-elections-affect-the-stock-market.html> , (Accessed: 29th Feb 2025).
26. Vargas, M.R., de Lima, B.S.L.P. and Evsukoff, A.G. (2017). Deep learning for stock market prediction from financial news articles. *2017 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA)*. doi: <https://doi.org/10.1109/civemsa.2017.7995302> , (Accessed: 22th March 2025).
27. Wang, Y. (2024). Analyzing the impact of financial news on the stock market using Natural Language Processing techniques. *Applied and Computational Engineering*, 76(1), pp.294–301. doi: <https://doi.org/10.54254/2755-2721/76/20240622> , (Accessed: 29th March 2025).
28. Yang, N., Fernandez‐Perez, A. and Indriawan, I. (2024). The price impact of tweets: A high‐frequency study. *The financial review/The Financial review*. doi: <https://doi.org/10.1111/fire.12406> , (Accessed: 17th March 2025).
29. Yujun, Y., Yimei, Y. and Jianhua, X. (2020). A Hybrid Prediction Method for Stock Price Using LSTM and Ensemble EMD. *Complexity*, 2020, pp.1–16. doi: <https://doi.org/10.1155/2020/6431712> , (Accessed: 20th Feb 2025).
30. Zhang, L., Aggarwal, C. and Qi, G.-J. (2017). Stock Price Prediction via Discovering Multi-Frequency Trading Patterns. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. doi: <https://doi.org/10.1145/3097983.3098117> , (Accessed: 17th March 2025).
31. Zou, J., Zhao, Q., Jiao, Y., Cao, H., Liu, Y., Yan, Q., Abbasnejad, E., Liu, L. and Shi, J.Q. (2023). Stock Market Prediction via Deep Learning Techniques: A Survey. *arXiv:2212.12717 [q-fin]*. [online] Available at: <https://arxiv.org/abs/2212.12717> , (Accessed: 20th March 2025).