

OTDK-dolgozat

2025

**Budapesti Műszaki és Gazdaságtudományi Egyetem
Gazdaság- és Társadalomtudományi Kar**



LSTM alapú tőzsdei árfolyambecslés technikai indikátorok segítségével

**LSTM based stock market price prediction with the help of technical
indicators**

Veress Dániel

Konzulens: Békési Gergő Bendegúz

Kézirat lezárásának dátuma: 2024. 05. 24.

Table of Contents

1. ABSTRACT.....	5
2. ÖSSZEFOGLALÓ.....	6
3. INTRODUCTION.....	7
3.1. DATA-DRIVEN TECHNOLOGIES IN THE ECONOMIC SECTOR	7
3.1.1. <i>Deep Learning in the economic sector</i>	7
3.2. PRICE PREDICTION	8
4. NEURAL NETWORKS WITH MEMORY	9
4.1. RECURRENT NEURAL NETWORK.....	9
4.1.1. <i>Vanishing and exploding gradient problem</i>	10
4.2. LONG SHORT-TERM MEMORY	11
4.3. GATED RECURRENT UNIT	12
4.4. BIDIRECTIONAL LSTM	13
5. HYPERPARAMETER OPTIMIZATION IN MACHINE LEARNING.....	15
5.1. GRID SEARCH	15
5.2. RANDOM SEARCH	16
5.3. BAYESIAN SEARCH	17
5.3.1. <i>Tree-structured Parzen Estimator</i>	18
6. LSTM NETWORK TRAINING.....	20
6.1. STOCK MARKET DATA.....	20
6.1.1. <i>Technical indicators</i>	21
6.1.2. <i>Data splitting</i>	23
6.1.3. <i>Data normalization</i>	24
6.1.4. <i>Data sequences</i>	25
6.2. MODEL ARCHITECTURE AND TRAINING.....	25
6.2.1. <i>Early stopping</i>	26
6.3. TRADING STRATEGIES	27
6.3.1. <i>Buy and hold</i>	27
6.3.2. <i>Other baselines</i>	28
6.3.3. <i>Model strategies</i>	29
7. RESULTS AND EVALUATION.....	30
7.1. PRICE PREDICTION RESULTS AND EVALUATION	30
7.1.1. <i>Metrics</i>	30
7.1.2. <i>Results</i>	31
7.2. TRADING RESULTS AND EVALUATION	34
7.2.1. <i>Metrics</i>	34

7.2.2.	<i>Results</i>	35
7.3.	OVERALL EVALUATION.....	38
7.3.1.	<i>Possible improvements</i>	38
8.	SUMMARY	40
9.	REFERENCES	41

1. ABSTRACT

For a long time, the best tools available to predict the stock market were statistical models and different equations. The unpredictability of the market makes it so investors are always looking for a better solution. In this work, I will introduce and implement an LSTM based deep neural network model, which will use the general price data and different technical indicators to make predictions. My goal is to build a solution that is capable of learning from the data and predicting the movements of the prices.

With the rise of neural networks, new solutions also emerged. One of these new solutions was the Long Short-Term Memory (LSTM) models. The LSTM was capable of storing important bits of information from past data, which meant you could use a sequence of the previous days as the input to predict a price on a given day. Using sequential data as input, these new models were able to improve prediction and efficiency. In my work, I am predicting the price from a 30 day long sequence.

In the center of my work stands the LSTM model, and its different variants, like the BLSTM and GRU types. The model's architecture is supported by hyperparameter optimization using the Tree-structured Parzen Estimator algorithm, meaning we get an optimal model faster, without so many trial-and-errors. The training itself uses companies from the S&P 500, and then extends that data with different technical indicators, such as the RSI indicator. This additional information improves the accuracy and reliability of the prediction.

Using the predicted price change, I also implemented a simple trading strategy to see the viability of the prediction. In my work, I compared my method and its result with other methods' results. I have discovered methods that can achieve a Sharpe ratio greater than 1. I tested my system on daily stock data, confirming its applicability for further scientific and industrial projects. Moreover, the combination of the Tree-structured Parzen Estimator and the LSTM algorithm on stock data, which contributed to the achieved results, represents a novel approach.

2. ÖSSZEFOGLALÓ

A befektetők többsége sokáig statisztikai modelleket, és különböző egyenleteket használt a tőzsdén történő mozgások szimulálásához és prediktálásához. A részvényt piac kiszámíthatatlansága miatt viszont mindig szükség volt jobb és megbízhatóbb megoldásokra. A TDK munkámban ismertetek és implementálok egy neurális háló alapú, LSTM modellt használó megoldást. Munkám célja egy modell felépítése és betanítása, hogy az képes legyen előrejelezni a tőzsdei részvények árának mozgását.

A neurális hálók megjelenésével új módszerek születtek a tőzsdei árak predikciójához, ezek közül az egyik megoldás a szekvenciális adatokat felhasználó modellek voltak. Az egyik ilyen megoldás az LSTM modell, amely képes régebbi adatokat is felhasználni. A megoldásomban a modell az előző 30 napból prediktálja az ár mozgását.

A munkám középpontjában az LSTM modell, és variánsainak felhasználása áll. A modell architektúrájának kialakítását hiperparaméter optimalizációval, Fa-struktúrájú Parzen Becslővel támogatom, így gyorsabban el lehet jutni egy optimális modellhez. A tanítás különböző S&P 500 indexből vett cégek részvényein történt, és különböző technikai indikátorokat adtam az adatokhoz, hogy azok segítsék a predikció pontosságát.

A megjósolt változást felhasználva, egy egyszerű kereskedő algoritmust is implementáltam, ami befektet a modell által visszaadott eredmény alapján. A megjósolt eredményeket, és az elért befektetési metrikákat elemzem és összehasonlítom más már létező eredményekkel. Számos megalkotott módszerem 1 feletti Sharpe ráta elérésére képes. Valós részvényt piaci adatokon teszteltem rendszerem és értem el eredményeimet, így munkám a későbbiekben teljes mértékben alkalmazható további ipari és tudományos kutatásokhoz. Továbbá újszerű megközelítésnek számít a Fa-struktúrájú Parzen Becslő és az LSTM architektúra együttes alkalmazása, amelyekkel eredményeim elértem.

3. INTRODUCTION

The role of technology in the realm of commerce has always been significant, resulting in the steady evolution of efficient and robust methods and algorithms across various economic sectors. In recent decades, these sectors have undergone significant transformations in data utilization and technological advancements, resulting in profound impacts on risk assessment, investment strategies, and portfolio management.

3.1. Data-driven technologies in the economic sector

For an extended period, data has served as the backbone of strategies within various facets of the economic sector. The development of countless tools and metrics, such as Conditional Value at Risk (CvaR) [1] and Generalized Autoregressive Conditional Heteroskedasticity (GARCH) models [2], has harnessed probability, volatility, returns, and an array of statistical data to make informed decisions in the domains of portfolio management and risk assessment.

Additionally, the forecasting and modeling of time series data have witnessed considerable advancements, contributing to higher investment yields. Notable tools in this domain encompass the AutoRegressive Integrated Moving Average (ARIMA) model [3], suitable for short- and mid-term forecasts, and the Exponential Smoothing model [4].

3.1.1. Deep Learning in the economic sector

The advent of Artificial Intelligence, specifically Deep Learning, has given rise to a new set of methodologies that can further elevate the results of investment and financial management. In the context of risk management, Deep Learning serves as a powerful tool to detect anomalies [5] in the stock market and assess risks [6], enabling more informed investment decisions. Deep Learning Reinforcement Learning also plays a pivotal role in automating and enhancing algorithmic trading strategies [7][8], while Generative Adversarial Networks (GANs) are employed to generate synthetic data for training purposes [9][10].

An additional approach involves leveraging Recurrent Neural Network (RNN)-based Deep Learning models, specifically developed to process time series data. Multiple variations of RNNs have emerged to improve the stability and reliability of predictions. In this research, these RNN variants are utilized for predicting daily closing prices.

3.2. Price Prediction

This research employs various RNN types to predict daily closing prices. The selected models include Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), and Bidirectional Long Short-Term Memory (Bi-LSTM) model types. These models are already in practical use for predicting stock market trends [11] and changes in currency prices [12]. The architecture of these models is determined through a hyperparameter optimization algorithm, a widely adopted technique for identifying the optimal model parameters and configuration [13].

The work starts with an introductory section that lays the foundation for the subsequent discussions. It begins by providing an overview of RNN-based models and their applications in the financial domain. This is followed by an exploration of the diverse hyperparameter optimization algorithms that play a pivotal role in shaping the effectiveness of these models.

The core of this research is then unveiled, presenting a detailed exposition of the methodology employed. This includes a comprehensive description of the data sources used, the data preprocessing techniques, and an in-depth discussion of the various trading strategies that were utilized.

The results of this endeavor are subsequently revealed and meticulously evaluated, bringing the research to a robust conclusion. Price prediction, a fundamental aspect of this study, is rigorously assessed through a range of metrics, providing insights into the performance of the implemented models. Moreover, the trading strategies are subjected to a thorough examination, delving into their effectiveness in real-world scenarios.

4. NEURAL NETWORKS WITH MEMORY

Many problems in the world are inherently sequential, where the outcome of the current state is influenced by a series of preceding states. Tasks like interpreting voice, constructing sentences, or predicting stock market movements all operate in this sequential manner. Consider, for instance, sentence construction – to select the appropriate word in the middle of a sentence, one must take into account all preceding words to maintain cohesion and context.

This category of problems can be effectively addressed by introducing memory into the system. By retaining the outcomes of previous states, we gain the ability to predict the next state accurately. Memory has also found its way into Neural Networks, revolutionizing their capacity to solve various complex challenges, including text translation and speech recognition [14]. As a result of these advancements, models like Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), and many others have been developed.

4.1. Recurrent neural network

The simple Recurrent Neural Network (RNN) stands as one of the most fundamental implementations of a model designed to incorporate a form of "memory" [15]. This network comprises an input layer, a hidden state, and an output layer. The fundamental architecture is then sequentially unrolled for each element in the sequence, with the hidden states connected to both the previous and subsequent hidden states. In the context of stock market prediction, the input could consist of a sequence of daily closing prices for a particular stock, and the final element in the output sequence would represent the closing price for the following day. The information learned is subsequently passed on to the next hidden state, thereby ensuring that the prediction of future prices is influenced by all preceding prices. The standard and unrolled structure are depicted in Figure 1.

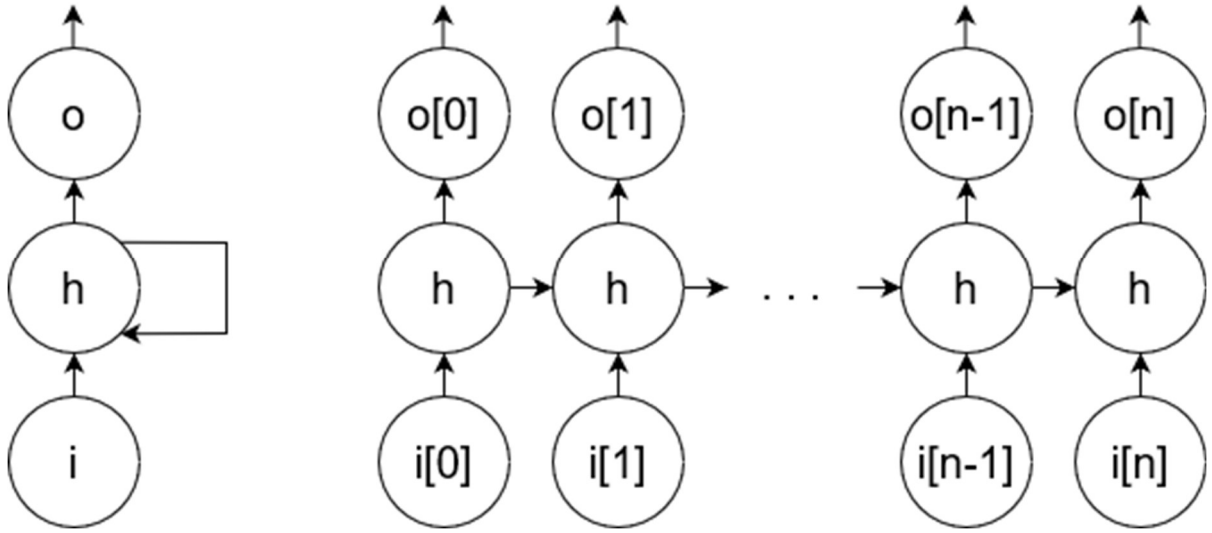


Figure 1. - An RNN and its unrolled representation, where i is the input, h is the hidden state and o is the output. The sequence consists of n items.

An RNN cell's operation can be divided into two steps. The first step calculates the new hidden state, while the second step calculates the output. These steps are described by the following two equations:

$$H_t = g(W * I_t + U * H_{t-1} + B) \quad (1)$$

$$O_t = g(V * H_t + C) \quad (2)$$

In these equations, H_t represents the hidden state at time t , while H_{t-1} represents the hidden state of the previous time step. I_t is the input, and O_t is the output at time t . The matrices W , U , and V , as well as the bias terms B and C , are involved in the calculations, with 'g' representing the activation function.

4.1.1. Vanishing and exploding gradient problem

As the length of the input sequence increases, two major challenges emerge when using the simple RNN: the exploding gradient and the vanishing gradient problems [15][16].

In the case of the exploding gradient problem, long-term components within the network become increasingly larger, eventually approaching infinity from a practical point of view. Conversely, the vanishing gradient problem is characterized by long-term components diminishing to near zero, making it difficult for the model to learn effectively.

These issues manifest as the input sequence grows, rendering the simple RNN less efficient. Consequently, various solutions have been developed to enhance the reliability of RNNs [17]. Today, the most widely employed variant to overcome these problems is the Long Short-Term Memory (LSTM) model.

4.2. Long Short-Term Memory

The Long Short-Term Memory (LSTM) model was developed to address several key challenges. It enables the handling of significantly larger sequences and effectively mitigates the vanishing and exploding gradient problems. This is achieved through the use of specialized units that enforce a consistent error flow.

The core structure of an LSTM unit closely resembles that of a simple RNN cell, encompassing an input, an output, and a hidden state that is passed on to the next cell. However, what sets LSTM apart is the introduction of an additional element, the cell state, which plays a pivotal role in governing the flow of information. The presence of this cell state empowers the model to capture longer dependencies in the input sequence.

The unique architecture of LSTM cells allows control over the information flow within a single cell. This control is accomplished through the utilization of three essential elements: the input gate, the output gate, and the forget gate.

The forget gate allows the cell to determine how much information should be discarded from the cell state, effectively eliminating useless data. In contrast, the input gate dictates whether new information should be integrated into the cell state. Lastly, the output gate regulates what information is added to the subsequent hidden state. The equations for these gates are as follows:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t]) \quad (3)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t]) \quad (4)$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t]) \quad (5)$$

In these equations, f_t , i_t and o_t are the outputs of the forget, input, and output gates, respectively. σ represents the sigmoid activation function, and W_f , W_i and W_o are the weight matrices for the gates. The structure of an LSTM cell is presented in Figure 2.

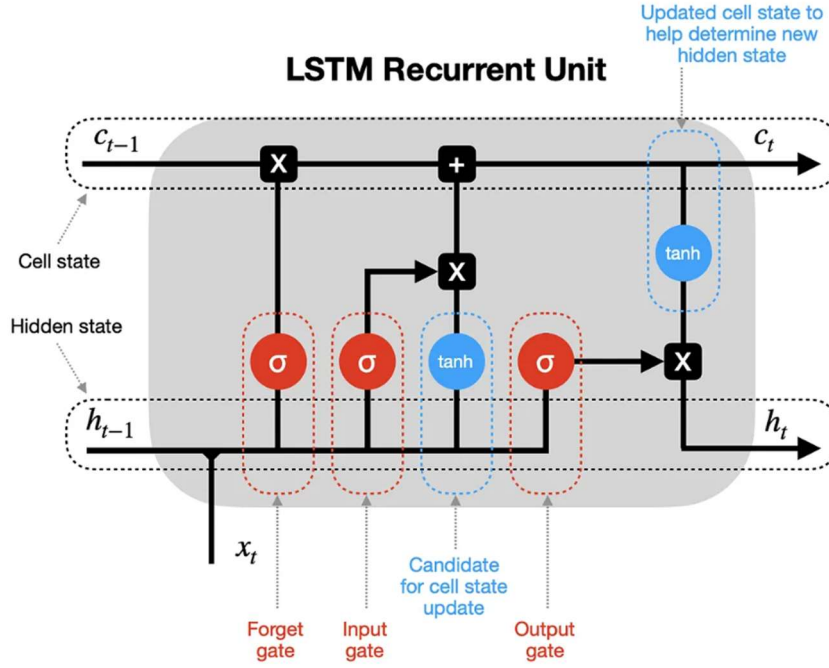


Figure 2. – Structure of an LSTM cell

Source: Dobilas, S., (2022). *LSTM Recurrent Neural Networks — How to Teach a Network to Remember the Past. Towards Data Science*

4.3. Gated Recurrent Unit

Much like the LSTM model, the Gated Recurrent Unit (GRU) was developed with the primary goal of mitigating the vanishing gradient problem. Notably, both LSTM and GRU have demonstrated the capacity to yield comparably high results in various applications [18], making them common choices for addressing Time Series problems.

The GRU model incorporates two key gates: the update gate and the reset gate. The update gate determines how much of the past hidden state should be added to the current hidden state, facilitating the assimilation of relevant past information. Conversely, the reset gate governs the extent to which past information should be forgotten from the previous hidden state, allowing the model to discard irrelevant data. A fundamental distinction between these two gates is that the output of the reset gate sets the candidate state, while the update gate decides how much of the past hidden state should be added to this candidate state. Figure 3 visually illustrates the structure of a GRU cell.

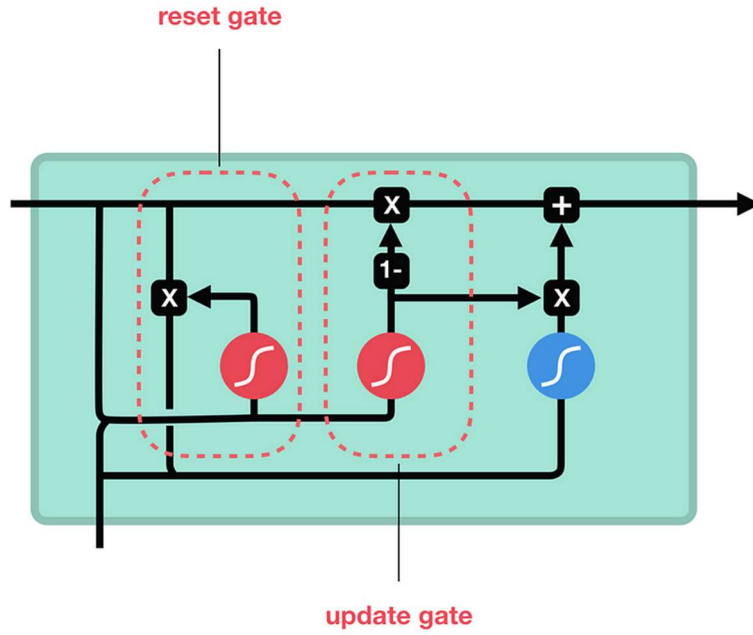


Figure 3. – Structure of a GRU cell

Source: Phi, M., (2018). *Illustrated Guide to LSTM's and GRU's: A step by step explanation. Towards Data Science*

The equations for the reset and update gates are as follows:

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t]) \quad (6)$$

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t]) \quad (7)$$

Here, r_t and z_t represent the outputs of the reset and update gates, respectively. σ denotes the sigmoid activation function, and W_r and W_z are the corresponding weight matrices.

GRU possesses certain advantages over LSTM, primarily rooted in its simplicity. Whereas LSTM employs three gates that can interact in complex ways, GRU relies on just two gates, simplifying its structure. This streamlined design can lead to faster and more efficient learning capabilities [19]. However in some cases the more complex architecture of LSTM can be a better solution for more complex problems, such as speech recognition [20].

4.4. Bidirectional LSTM

The Bidirectional Long Short-Term Memory (Bi-LSTM) differs from a simple LSTM by introducing bidirectionality, which enables information to flow in both forward and backward directions in time. This unique behavior is realized through the utilization of two LSTM layers, where the second layer operates in a reverse manner, allowing information to flow in the opposite direction.

The bidirectional information flow allows the model to harness both past and future information to make predictions, thereby often achieving superior results compared to the straightforward LSTM model [21]. This utilization of both preceding and succeeding data

within a sequence, without actual foresight, enables the model to capture context more effectively, often proving crucial for nuanced tasks. The structure of the Bi-LSTM model is depicted in Figure 4.

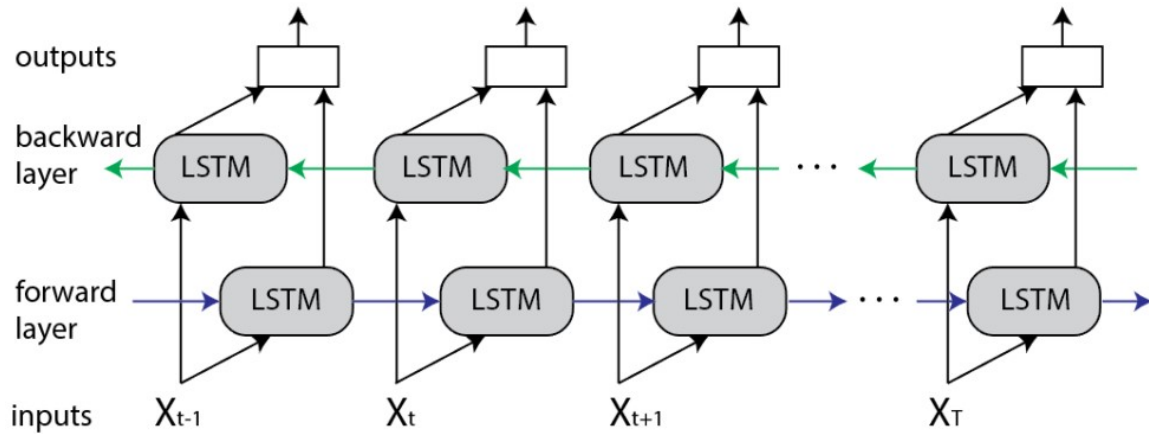


Figure 4. – Structure of a Bi-LSTM layer

Source: Zvornicanin, E., (2023). *Differences Between Bidirectional and Unidirectional LSTM*. Baedung

5. HYPERPARAMETER OPTIMIZATION IN MACHINE LEARNING

Before embarking on the training of our model, the crucial first step involves the creation of its structure and configuration of its parameters. These hyperparameters wield significant influence over the behavior and performance of our model. For instance, when designing an LSTM model, we can determine how many layers the model should comprise, whether these layers should be bidirectional, the number of hidden nodes within each layer, and various other parameters.

However, manually specifying and experimenting with all possible combinations can be a highly labor-intensive and exhaustive process. Consequently, researchers often turn to the aid of automated hyperparameter optimization algorithms. These algorithms systematically select hyperparameter combinations and evaluate the model based on defined metrics. We can specify the number of experiments to be conducted, and in return, these algorithms give us the optimal hyperparameter combination. Such optimization techniques range from the simpler approaches like grid search or random search [22] to more sophisticated and efficient Bayesian search methods [23].

5.1. Grid Search

Grid search stands as the most straightforward automated solution for hyperparameter optimization. For each hyperparameter that requires tuning, we define an array of discrete values. Subsequently, the algorithm systematically tries all possible combinations of these values and evaluates the corresponding results.

This method, while effective in its comprehensive exploration of the parameter space, comes with certain limitations. It can be notably slow and time-consuming because it adheres to a brute-force approach that doesn't consider the change in outcome of past trials. In essence, grid search allocates the same amount of computational effort to both insignificant and crucial parameters [22]. Grid search, specifically with two hyperparameters, is illustrated in Figure 5.

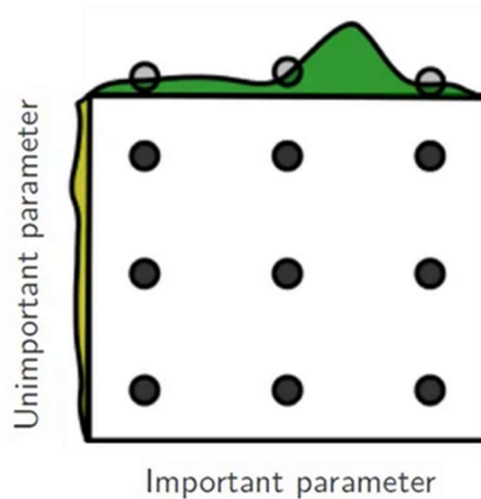


Figure 5. – Grid Search optimizing two hyperparameters, where one parameter disproportionately affects the outcome.

Source: Senapati, D., (2018). Grid Search vs Random Search. Medium

5.2. Random Search

Random Search, similar to grid search, is a straightforward hyperparameter optimization algorithm. However, it diverges from grid search in its approach. With random search, we define a range or distribution for each hyperparameter, and the algorithm randomly selects a value within the interval. We can also specify the number of experiments, enabling the exploration of numerous distinct random combinations.

One distinct advantage of random search over grid search lies in its efficiency. It avoids wasting time on unimportant parameters, ultimately leading to faster and often better results [22]. This distinction is visually demonstrated in Figure 6, where Grid Search uses the same discrete values for a parameter in multiple combinations, resulting in less coverage for the important parameter. In contrast, random search, due to its random selection approach, is less likely to repeat the same value for a parameter in different combinations.

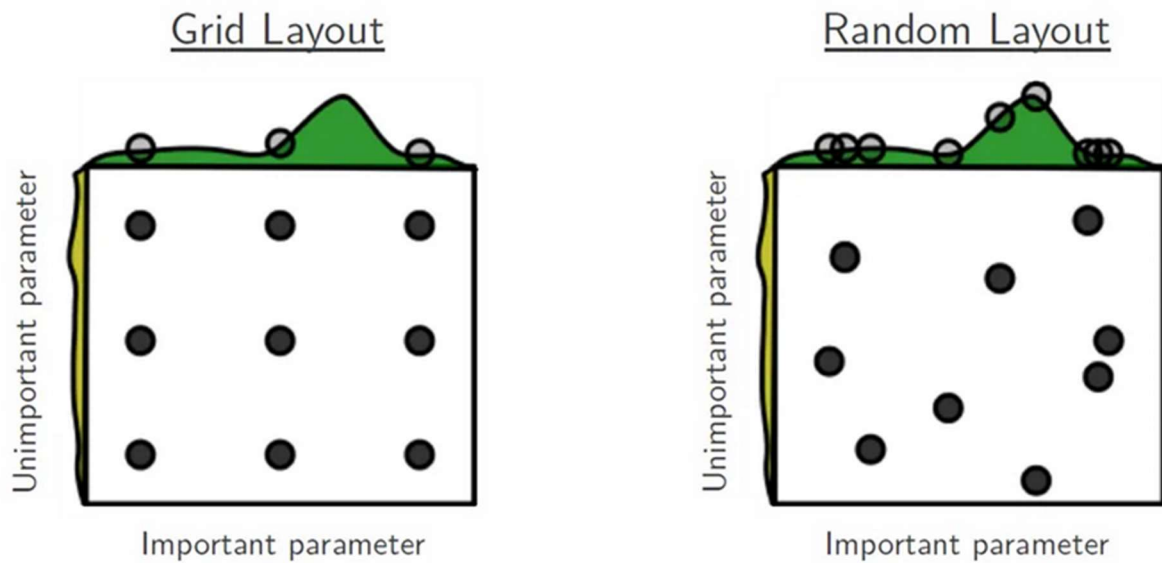


Figure 6. – Comparison of Grid Search and Random Search

Source: Senapati, D., (2018). *Grid Search vs Random Search*. Medium

5.3. Bayesian Search

Bayesian Search is a sophisticated hyperparameter optimization approach that leverages past optimization results, resulting in a faster and more efficient overall optimization process. At its core lies the objective function, often referred to as a "black-box" function, which takes hyperparameter inputs and returns the metric used for optimization. Since the inner workings of this function are unknown for the optimization algorithm, the algorithm's primary mission is to estimate it through a series of iterative trials.

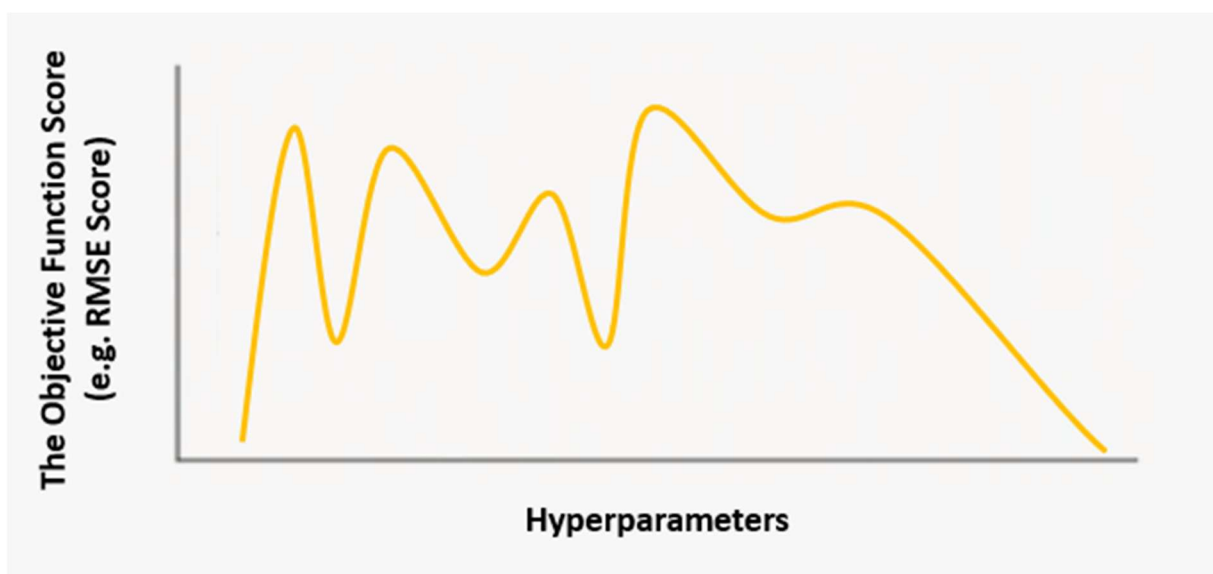


Figure 7. – Example objective function

Source: Wang, W., (2020). *Bayesian Optimization Concept Explained in Layman Terms*. Towards Data Science

The algorithm starts with a random search to construct a probabilistic model of the objective function. This model serves as a representation of the potential forms the objective function might take. To determine the next hyperparameter combination for a trial, an acquisition function is made, instructing the algorithm on which combination to utilize. The acquisition function strives to find a balance between exploration and exploitation, ultimately seeking the best estimation of the objective function. Common types of acquisition functions include Probability of Improvement (PI), Expected Improvement (EI), and Upper Confidence Bound (UCB).

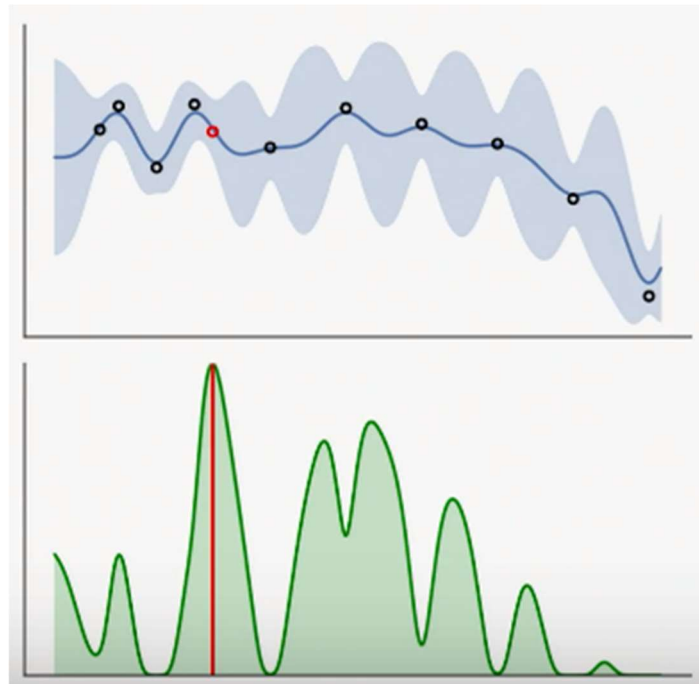


Figure 8. – Example of a probabilistic model and an acquisition function. The red circle indicates the hyperparameter combination chosen for the next trial, as it minimizes the acquisition function.

Source: Wang, W., (2020). *Bayesian Optimization Concept Explained in Layman Terms. Towards Data Science*

Following each new trial, the probabilistic model is updated to account for the newly acquired information. This iterative approach leads to progressive enhancements in the estimation of the objective function.

5.3.1. Tree-structured Parzen Estimator

A variant of Bayesian Search, known as the Tree-structured Parzen Estimator (TPE), is designed to estimate an objective function through exploration and exploitation. Similar to other Bayesian optimization algorithms, it employs a probabilistic model for the objective function. However, TPE introduces an additional probabilistic model, one that gauges the likelihood of

a particular combination being good or not. This dual-model approach enhances the exploration and exploitation aspects, with combinations sampled from the categorical distribution [23].

In my research, I employ the TPE optimization method with the assistance of the Optuna optimization library. This enables the automatic optimization of the model's final structure, increasing the likelihood of achieving an optimal solution.

6. LSTM NETWORK TRAINING

In this chapter, I will go through my solution and its various components. I will start by explaining the data utilized for model training and evaluation. Notably, I will delve into the new features incorporated into the data to enhance the robustness of our predictions.

Prior to training, it's crucial to navigate through the preprocessing steps that are essential to prepare the data effectively. This chapter will provide a step-by-step breakdown of the preprocessing procedures applied.

With the data being ready, our focus turns to the model architecture and its training process. A thorough analysis of the architecture and the training methods employed will be presented, giving a deeper understanding of the inner workings of our LSTM-based solution.

To conclude, various trading strategies will be shown that are employed to assess the practicality and viability of our model's predictions. This chapter offers a comprehensive view of the entire process, from data preparation to model evaluation and real-world application.

6.1. Stock market data

The world of LSTM-based stock market prediction begins with the raw material: the daily stock market data from Google, one of the most influential companies in the tech industry. This dataset offers many basic daily information essential for our model's training and evaluation.

The data encompasses a range of critical features:

- **Date:** This column represents the date on which the data is recorded.
- **Open Price (Opening):** The opening price is the price of the stock at the beginning of a trading session. It's the price at which the first trade occurs when the stock market opens for the day.
- **High Price (Daily High):** The daily high price is the highest price that the stock reaches during a trading session. It represents the peak price at which the stock traded on that day.
- **Low Price (Daily Low):** The daily low price is the lowest price that the stock reaches during a trading session. It represents the lowest price at which the stock traded on that day.
- **Close Price (Closing):** The closing price is the final price of the stock at the end of a trading session. It's the price at which the last trade occurs before the stock market closes for the day.

- **Adjusted Close Price (Adj. Close):** The adjusted closing price takes into account corporate actions such as dividends, stock splits, and new stock offerings.
- **Volume:** The volume column represents the total number of shares traded for that stock on a given day. It reflects the liquidity and activity of the stock.

Our dataset covers a significant period, spanning from August 19, 2004, to December 12, 2022. This extensive historical data forms the foundation for our stock market predictions. Figure 9 offers a representation of Google's daily closing prices throughout this substantial timeframe.

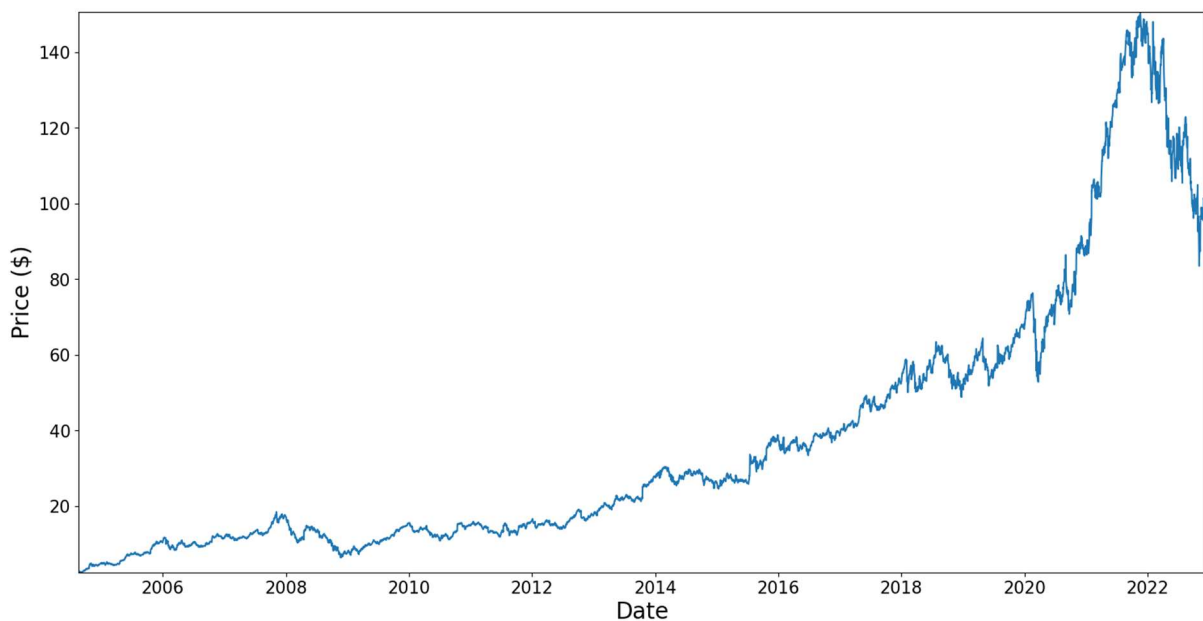


Figure 9. – A diagram of the daily closing price for Google

6.1.1. Technical indicators

In addition to the foundational stock market data, I augmented our dataset with essential technical indicators to provide an expanded information set for our model. These technical indicators, often used by traders and investors, furnish our model with valuable insights into market trends and potential price movements.

Let's explore the key technical indicators integrated into our dataset:

- **MACD (Moving Average Convergence Divergence):** MACD is a trend-following momentum indicator that calculates the difference between two Exponential Moving Averages - EMAs (the "fast" and "slow" EMAs). It helps identify potential buy and sell signals based on crossovers of the MACD line and the signal line [24].

- SMA (Simple Moving Average): The simple moving average is a trend-following indicator that calculates the average price of a stock or asset over the previous n trading days, where n 's most usual value is 50 or 200. It's used to identify trends and potential support or resistance levels [25]. The following equation shows its calculation, where n is the calculated interval's size and P_i is the i -th price in this interval.

$$SMA = \frac{1}{n} \sum_{i=1}^n P_i \quad (7)$$

- RSI (Relative Strength Index): The RSI is a momentum oscillator that measures the speed and change of price movements. It provides values between 0 and 100, with readings above 70 considered overbought and readings below 30 considered oversold. RSI helps traders identify potential reversal points [26]. The following equations show the 14 days long RSI calculation:

$$RSI = 100 - \frac{100}{1 + RS} \quad (8)$$

where $RS = \frac{\text{Average Gain}}{\text{Average Loss}}$ in a 14 days interval.

- ROC (Rate of Change): ROC calculates the percentage change in price over a specific period. Usually a 14-day ROC is used, that would measure the percentage change in price over the last 14 days. It can help identify the speed and magnitude of price movements [27].

$$ROC = 100 \cdot \frac{P_i - P_{i-n}}{P_{i-n}} \quad (9)$$

where n is 14 in my case, and P_i is the current price.

For the technical indicators, I have opted for common settings widely used by investors, recognizing that these settings can influence stock prices as well. Specifically, the MACD parameters include a faster EMA of 12 days and a slower EMA of 26 days, with a 9-day signal line. Two SMAs are applied, one computed over 50 days and the other over 200 days. Both the RSI and ROC are calculated over 14-day periods.

In Table 1, the table representation of an organized and expanded dataset that incorporates these technical indicators can be seen. It is important to note that after adding these features, some missing values were also introduced to the dataset. For instance, the calculation of the 200-day SMA leaves the first 199 rows empty. After incorporating these technical indicators, the missing values and their rows were dropped from the dataset. This data cleansing procedure

resulted in a shift in our dataset's time range, which now spans from June 3, 2005, to November 18, 2022.

Table 1. – A sample of the data used after cleaning it from missing values

	Date	Low	Open	Volume	High	Close	Adjusted Close	ticker	MACD_12_26_9	MACDh_12_26_9	...	RSI_14	ROC_14
Date													
2005-06-03	2005-06-03	6.909352	7.142976	754108027	7.205492	6.980336	6.980336	GOOG	0.449593	0.075923	...	79.570214	22.256154
2005-06-06	2005-06-06	7.019439	7.033387	904413304	7.316327	7.246339	7.246339	GOOG	0.467185	0.074812	...	82.771400	25.920792
2005-06-07	2005-06-07	7.230399	7.399764	976566743	7.461781	7.300635	7.300635	GOOG	0.479975	0.070082	...	83.345063	25.732409
2005-06-08	2005-06-08	6.924047	7.293911	1031889331	7.302379	6.962901	6.962901	GOOG	0.457585	0.038153	...	68.145384	16.892449
2005-06-09	2005-06-09	6.987808	7.091420	660109011	7.185567	7.131021	7.131021	GOOG	0.448239	0.023046	...	70.982302	19.704824

6.1.2. Data splitting

After the data was cleaned, the dataset is partitioned into three distinct sets, each serving a specific purpose. The initial segment, constituting 70% of our original dataset, is designated as the training set. This segment is fundamental for the model's training, allowing it to learn and adapt to the underlying patterns within the data.

The second segment, known as the validation set, plays a critical role in assessing the optimization, discussed in chapter 5, and training processes. This segment comprises 15% of the original dataset and is indispensable for testing the model's validity and reliability.

Finally, the third segment, equally comprising 15% of the original dataset, is the test set. This segment is reserved for the final evaluation of the model's performance, providing an unbiased examination of its predictive capabilities.

Maintaining a clear separation between these three segments is of great importance. It safeguards against the potential issue of evaluation on already seen data, which could skew the accuracy and lead to overly optimistic but false results [28][29].

Additionally, it is important to preserve the sequentiality of our data throughout this process. Therefore, shuffling of the data is strictly prohibited to ensure that the temporal order of the data remains intact. The distribution of our data across these segments is shown in Figure 10, where the training set is represented by the orange line, the validation set by the yellow, and the test set by the green.

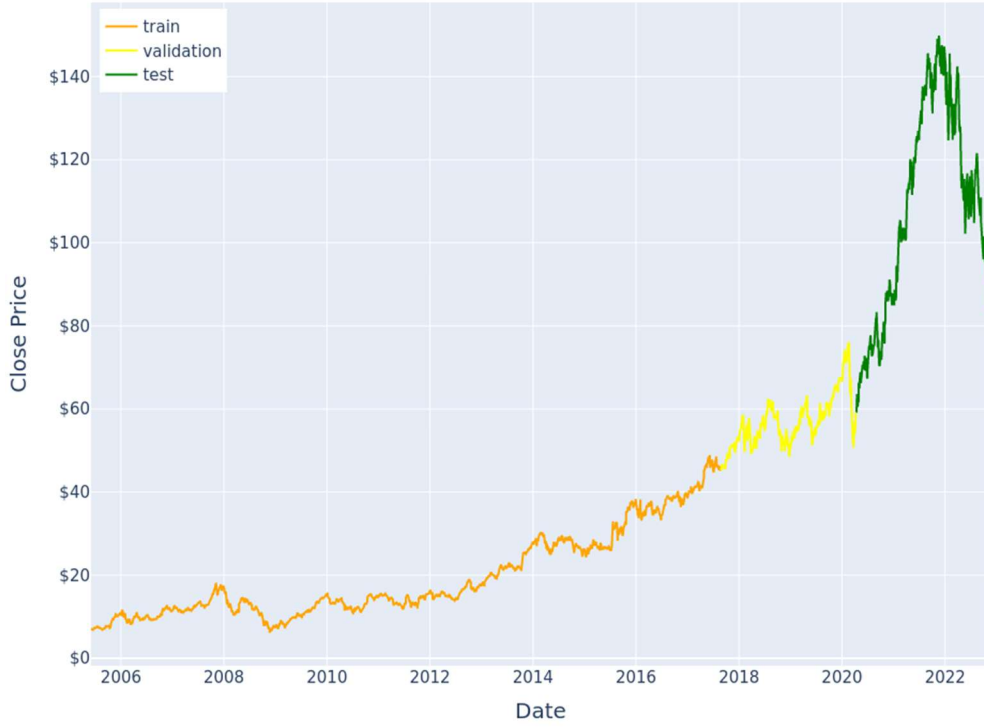


Figure 10. – The 70%-15%-15% split of our dataset

6.1.3. Data normalization

Following the partitioning of our dataset into three distinct sets, the next critical step is data normalization or standardization. This process is indispensable due to the disparate ranges our features can encompass. For instance, while the stock price may surge above \$1000, the Relative Strength Index (RSI) remains confined within the 0 to 100 range. This significant disparity in feature ranges can lead to training inefficiencies, prolonging the training process [30-32]. However, normalization and standardization serve as the solution to this issue, ensuring that our diverse features share similar or identical ranges.

For this study, I have used a straightforward standardization approach utilizing the scikit-learn library. This method entails removing the mean (μ) and scaling to the unit variance (σ) for each data point (x), this way producing a standardized value (z) according to the formula (1):

$$z = \frac{(x - \mu)}{\sigma} \quad (10)$$

6.1.4. Data sequences

The final processing step before starting the model training involves the construction of sequences from our data. This step is imperative due to the operational mechanism of the LSTM network, where sequences serve as inputs, and the model unrolls for each element within the sequence [15].

In this context, I have chosen to implement sequences with a length of 30 days. These sequences facilitate the training process, breaking down the data into manageable chunks for the model to process. The target value we aim to predict corresponds to the price on the 31st day minus the price on the 30th day. In essence, our model is tasked with forecasting the difference between the future and current price.

6.2. Model Architecture and Training

To explore and optimize the performance of our model, multiple runs were conducted, each employing different types of neural networks. Four primary network types were considered: LSTM, GRU, Bi-LSTM, and Hybrid. The Hybrid architecture uniquely allows the optimization process to determine the network type for each individual layer. Consequently, the final model could comprise a blend of LSTM, GRU, and Bi-LSTM layers.

For each network type, the architecture was determined using the Tree-structured Parzen Estimator (TPE) optimization method, with the help of the Optuna library. The following crucial parameters were optimized through this process:

- **Loss function:** The loss function is a crucial component of the model optimization process. It defines the measure of error or discrepancy between the predicted values and the actual target values during training. Options for the loss function included Mean Squared Error (MSE), Mean Absolute Error (MAE), and Huber loss [33].
- **Optimizer:** The optimizer determines the algorithm used to update the model's weights during training. It plays a significant role in how quickly and effectively the model converges to the optimal solution. The optimization process considered options such as Adam, Stochastic Gradient Descent (SGD), and Adagrad.
- **Layer count:** The layer count refers to the number of layers in the neural network architecture. In the context of an LSTM-based model, this parameter specifies the depth of the recurrent neural network. The choice of layer count impacts the model's capacity to capture complex temporal dependencies in the data. The optimization process was allowed to add 1 to 6 layers to the model.

- Hidden unit count for each layer: This parameter determines the number of hidden units (neurons) in each LSTM layer of the model. The hidden units are responsible for learning and representing patterns in the sequential data. The count of hidden units per layer influences the model's capacity to capture and model intricate patterns within the sequences. To manage training time effectively, discrete values were used, with the optimization process choosing numbers between 8 and 128 in 8-unit increments.
- For the Hybrid network type, an additional parameter was used to optimize the type of each layer in the model.

It's important to note that various metrics were used during the optimization process, including Mean Absolute Error (MAE), Mean Squared Error (MSE), and Directional Accuracy (DA). Detailed descriptions of these metrics will be provided in the next chapter, where the results will be evaluated.

In addition to the parameters optimized, several other key parameters influenced the training process:

- Epoch number: The epoch number represents the count of times the entire dataset is processed during the training phase. Each epoch consists of one forward pass and one backward pass of all training examples. The choice of the epoch number affects how many times the model learns from the data. Too few epochs may result in underfitting, while too many epochs may lead to overfitting [28]. I used 50 epochs for each run which is a common approach and also was my best choice based on experiments.
- Batch size: The batch size determines the number of training examples used in each forward and backward pass during a single epoch. It impacts the training process's speed and memory requirements. Smaller batch sizes provide a more frequent update to the model's weights, while larger batch sizes are computationally efficient but may result in slower convergence and less noise in gradient estimation [34]. In my case, I set the batch size to 32 which is a common methodology as well.

These parameters, combined with the previously optimized ones, play a significant role in shaping the training process of the model, ensuring effective learning and convergence to the optimal solution.

6.2.1. Early stopping

If the model is allowed to learn indefinitely or for a high number of epochs, we can expect the model to overfit the training data, resulting in poor generalization on unseen data, such as

the test data [28]. While one could manually search for an optimal epoch number to avoid overfitting, I chose a more automated solution: early stopping.

Early stopping halts the training if signs of overfitting become apparent during the training process. These signs can include a decline in accuracy on the validation set. To implement early stopping, I use the validation loss and set a patience of 3. This means that if the validation loss doesn't improve over the next 3 epochs, the training process is stopped. This automated self-control ensures that we can expect an optimally trained final model from our process.

6.3. Trading strategies

Once the model is trained, multiple trading strategies are tested and analyzed. These strategies begin with an initial investment of \$10 000, and at every buy signal, they purchase as many stocks as possible, while at the sell signal, they sell all of them. In this section of the chapter, I will discuss the various trading strategies I experimented with. I will start by mentioning some baseline strategies that do not rely on the model's predictions for trading, such as the Buy and Hold strategy or the 200-day SMA baseline strategy.

6.3.1. Buy and hold

The buy and hold strategy is inherently simple yet highly effective, as praised by influential investors like Warren Buffett [35]. To implement this strategy, an investor merely acquires the stock once at the beginning and retains it until the end. In the case of Google, Figure 11 illustrates this trading strategy in action, with the green dashed line denoting the start and end of the position.



Figure 11. – The Buy and Hold strategy used on Google stock

6.3.2. Other baselines

The first baseline strategy employs the 200 days Simple Moving Average (SMA) technical indicator as a trading signal. If the 200 days SMA value surpasses the current price, it signifies a buy signal. Conversely, if it's less than or equal to the current price, it indicates a sell signal. This strategy has demonstrated the ability to avoid significant drawdowns while achieving favorable returns.



Figure 12. – The 200 days SMA strategy on Google stock

The second baseline strategy utilizes the Relative Strength Index (RSI) indicator as its primary signal. A buy signal is generated when the current RSI value exceeds 70, while a sell signal is indicated when its value falls below 30.



Figure 13. – The RSI strategy on Google stock

The third and final baseline strategy combines the previous two strategies to enhance reliability and robustness. It triggers a buy action only when both the RSI and the 200 days SMA indicators signal a buy. Conversely, a sell action is executed when both indicators agree on a sell signal.



Figure 14. – The combined 200 days SMA and RSI strategy on Google stock

6.3.3. Model strategies

The final set of strategies employs the model's predictions as signals for buying or selling the stock. The first such strategy, referred to as the 'Trust Strategy', relies on the predicted change to determine the future price. A buy signal is generated when the predicted price falls within an acceptable 10% error margin of the current price and is higher than the current price. Conversely, a sell signal is generated when the predicted price is lower than the current price.

The second strategy, named as the 'Direction Strategy', analyzes the direction of change predicted by the model. If the predicted change is positive, it triggers a buy action; otherwise, it initiates a sell action. Unlike the Trust Strategy, it does not consider whether the predicted change is within an acceptable range.

The other two strategies are variations of the previous ones but incorporate the 200 days SMA technical indicator to create more robust and less volatile trading strategies. Both strategies use the SMA200 signal and their original buy signal to buy, while only the SMA200 Strategy's signal is considered for selling. These strategies are referred to as 'Trust & SMA200 Strategy' and 'Direction & SMA200 Strategy', respectively.

The results and workings of these strategies will be discussed in the next chapter during the final evaluation of our model and training process.

7. RESULTS AND EVALUATION

In this chapter, I will present the results of the model training and the performance of the trading strategies. I will also describe and discuss the metrics used to evaluate the results.

7.1. Price prediction results and evaluation

7.1.1. Metrics

The following metrics were used during the evaluation:

- **MAE (Mean Absolute Error):** MAE measures the average absolute differences between predicted values and actual values. It provides insight into the model's absolute prediction accuracy, with lower values indicating better performance.

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \quad (11)$$

where y_i represents the actual values, and $\hat{y}_i$ represents the predicted values, and n is the number of samples.

- **MSE (Mean Squared Error):** MSE calculates the average of the squared differences between predicted values and actual values. It is widely used and emphasizes larger errors due to the squaring operation, making it sensitive to outliers [36].

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (12)$$

where y_i represents the actual values, and $\hat{y}_i$ represents the predicted values, and n is the number of samples.

- **ME (Max Error):** Max Error is a metric used to quantify the maximum difference between predicted values and actual values in a dataset. It is a measure of the largest individual error within the prediction set. The calculation involves finding the absolute difference between each predicted and actual value and identifying the maximum among these differences.

$$\text{ME} = \max_{i=1}^n (|y_i - \hat{y}_i|) \quad (13)$$

where y_i represents the actual values, and $\hat{y}_i$ represents the predicted values, and n is the number of samples.

- **R2 (R-squared):** R2 is a statistical measure of the proportion of the variance in the dependent variable that is explained by the independent variables. It provides an

understanding of how well the model's predictions fit the actual data, with values closer to 1 indicating a better fit.

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2} \quad (14)$$

where y_i represents the actual values, and $\hat{y}_i$ represents the predicted values, $\bar{y}$ represents the mean of the actual values, and n is the number of samples.

- **MAPE (Mean Absolute Percentage Error):** MAPE calculates the average percentage difference between predicted and actual values. It's a useful metric when you want to understand the relative size of errors, making it easy to interpret and compare across different datasets.

$$\text{MAPE} = \frac{1}{n} \sum_{i=1}^n \left| \frac{y_i - \hat{y}_i}{y_i} \right| \quad (15)$$

where y_i represents the actual values, and $\hat{y}_i$ represents the predicted values, and n is the number of samples.

- **DA (Directional Accuracy):** DA measures the model's ability to predict the correct direction (positive or negative) of the price change. It is particularly relevant in stock market predictions, where predicting the correct trend is crucial.

$$\text{DA} = \frac{\text{Number of correctly predicted price change directions}}{n} \quad (16)$$

where n is the number of samples.

7.1.2. Results

The evaluation was done on the test set of the data. As the optimization was performed for three different metrics: MAE, MSE, and DA, the following three tables showcase the results for each of them:

Table 2. – Results achieved by optimizing for the MAE metric

<i>Model type</i>	MAE	MSE	ME	R2	MAPE	DA
<i>LSTM</i>	1.561	4.679	10.15	-2.105e-4	0.999	0.545
<i>GRU</i>	1.555	4.662	10.11	0.003	1.042	0.561
<i>Bi-LSTM</i>	1.602	4.81	10.16	-0.028	1.21	0.537
<i>Hybrid</i>	1.56	4.68	10.14	-4.331e-4	1.016	0.545

Table 3. – Results achieved by optimizing for the MSE metric

<i>Model type</i>	MAE	MSE	ME	R2	MAPE	DA
<i>LSTM</i>	1.561	4.679	10.15	-1.083e-4	1.005	0.545
<i>GRU</i>	1.557	4.675	10.15	7.364e-4	1.042	0.548
<i>Bi-LSTM</i>	1.852	15.09	48.39	-2.225	1.486	0.542
<i>Hybrid</i>	1.562	4.71	10.19	-0.007	1.134	0.545

Table 4. – Results achieved by optimizing for the DA metric

<i>Model type</i>	MAE	MSE	ME	R2	MAPE	DA
<i>LSTM</i>	1.627	4.993	10.44	-0.067	2.147	0.559
<i>GRU</i>	1.553	4.667	10.33	0.002	1.171	0.563
<i>Bi-LSTM</i>	2138	59778353.6	75515.5	-12778202.9	2641.2	0.552
<i>Hybrid</i>	1.562	4.679	10.15	-2.685e-4	1.002	0.532

The best value for each metric is highlighted. It is evident that the GRU model type achieved the overall best results in all three optimizations, while the Bi-LSTM model type had the worst results among the four types. Notably, when the optimization was done based on the DA metric, the metrics for the Bi-LSTM model were the worst. This was expected as the DA metric does not consider outliers in its calculation, as illustrated in Figure 15. A normal prediction result can be seen in Figure 16 for comparison, and Figure 17 shows the comparison of the real future price and the predicted future price.



Figure 15. – Large outliers skew the evaluation of the Bi-LSTM model

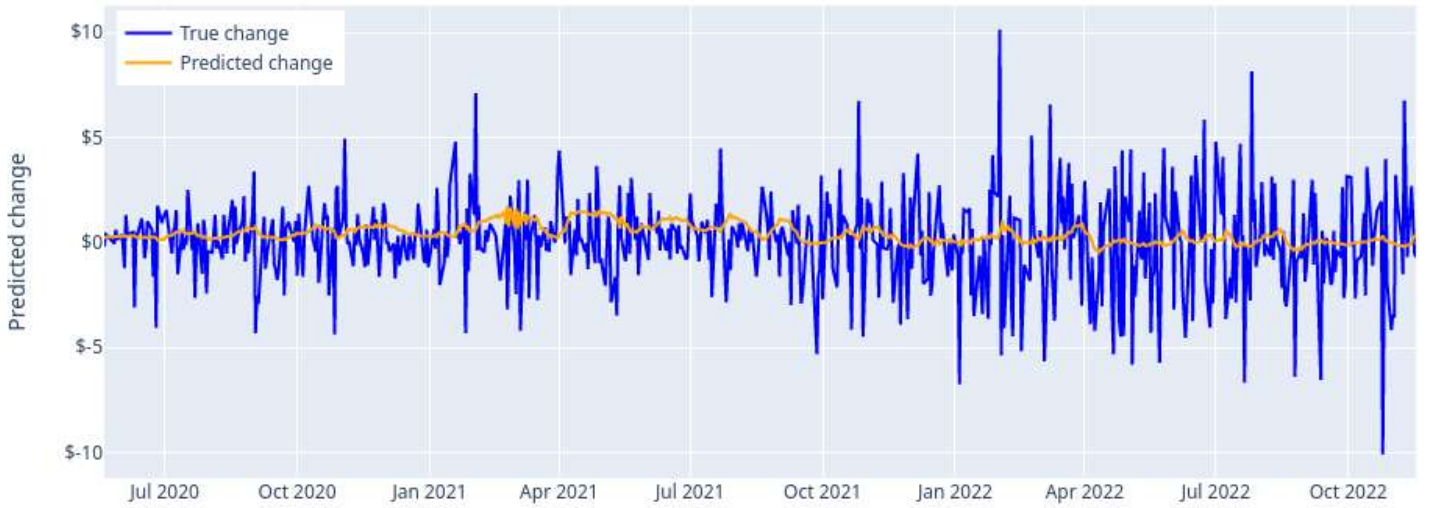


Figure 16. – Prediction results for the LSTM model type when DA is optimized

The results align with existing research, showing that the GRU model type slightly outperforms simple LSTM models [37][38]. It is also interesting to note that the GRU model, consisting only of GRU layers, outperforms the Hybrid model type, where all the types could be used. It appears that the optimization process could not find the optimal architecture for the Hybrid model. This suboptimal solution is also evident in Table 2, where the Bi-LSTM model achieved worse values than expected, suggesting that the optimization process was not able to find the optimal solution it had found before.



Figure 17. – Predicted (yellow line) compared to real future (blue line) price

7.2. Trading results and evaluation

7.2.1. Metrics

To comprehensively evaluate the trading strategies and their effectiveness, two main metrics are employed:

- **Overall Return:** The overall return metric assesses the cumulative profit or loss generated by the trading strategies over the test period. It provides a measure of the strategies' financial performance.
- **Sharpe Ratio (SR):** The Sharpe Ratio is a risk-adjusted measure that evaluates the strategies by considering both their return and the level of risk taken to achieve that return. It helps assess whether the strategies provide an adequate return for the level of risk involved [39].

$$SR = \frac{R_p - R_f}{\sigma_p} \quad (17)$$

where R_p represents the expected portfolio return, R_f is the risk-free rate (typically a government bond yield), and σ_p is the standard deviation of the portfolio's excess return.

These metrics are fundamental in gauging the performance and success of trading strategies, offering valuable insights into their profitability and risk management capabilities.

7.2.2. Results

For each of the optimized metrics, I have provided two tables. One table outlines the overall return of each strategy, and the other presents the Sharpe Ratio (SR), which helps assess the strategies' risk-adjusted performance. It is also essential to emphasize that the Buy and Hold strategy yielded a noteworthy 40.04% profit.

Table 5. – Returns achieved by optimizing for the MAE metric

<i>Model type</i>	SMA200	RSI	SMA200 & RSI	Trust Strategy	Direction Strategy	SMA200 & Trust	SMA200 & Direction
<i>LSTM</i>	76.72%	52.01%	52.01%	7.899%	40.99%	90.34%	90.34%
<i>GRU</i>	76.72%	52.01%	52.01%	18.93%	40.99%	76.72%	76.72%
<i>Bi-LSTM</i>	76.72%	52.01%	52.01%	62.49%	40.99%	90.34%	90.34%
<i>Hybrid</i>	76.72%	52.01%	52.01%	26.5%	40.99%	76.72%	76.72%

Table 6. – SR achieved by optimizing for the MAE metric

<i>Model type</i>	SMA200	RSI	SMA200 & RSI	Trust Strategy	Direction Strategy	SMA200 & Trust	SMA200 & Direction
<i>LSTM</i>	0.912	0.795	0.795	0.501	0.402	1.076	1.076
<i>GRU</i>	0.912	0.795	0.795	0.727	0.402	0.912	0.912
<i>Bi-LSTM</i>	0.912	0.795	0.795	1.158	0.402	1.076	1.076
<i>Hybrid</i>	0.912	0.795	0.795	0.299	0.402	0.912	0.912

Table 7. – Returns achieved by optimizing for the MSE metric

<i>Model type</i>	SMA200	RSI	SMA200 & RSI	Trust Strategy	Direction Strategy	SMA200 & Trust	SMA200 & Direction
<i>LSTM</i>	76.72%	52.01%	52.01%	45.9%	40.99%	76.72%	76.72%
<i>GRU</i>	76.72%	52.01%	52.01%	42.31%	40.99%	76.72%	76.72%
<i>Bi-LSTM</i>	76.72%	52.01%	52.01%	16.02%	23.76%	83.92%	83.92%
<i>Hybrid</i>	76.72%	52.01%	52.01%	19.27%	23.76%	87.55%	87.55%

Table 8. – SR achieved by optimizing for the MSE metric

<i>Model type</i>	SMA200	RSI	SMA200 & RSI	Trust Strategy	Direction Strategy	SMA200 & Trust	SMA200 & Direction
<i>LSTM</i>	0.912	0.795	0.795	0.791	0.402	0.912	0.912
<i>GRU</i>	0.912	0.795	0.795	0.547	0.402	0.912	0.912
<i>Bi-LSTM</i>	0.912	0.795	0.795	0.869	0.255	1.043	1.043
<i>Hybrid</i>	0.912	0.795	0.795	0.863	0.255	1.054	1.054

Table 9. – Returns achieved by optimizing for the DA metric

<i>Model type</i>	SMA200	RSI	SMA200 & RSI	Trust Strategy	Direction Strategy	SMA200 & Trust	SMA200 & Direction
<i>LSTM</i>	76.72%	52.01%	52.01%	1.295%	-100%	74.65%	74.65%
<i>GRU</i>	76.72%	52.01%	52.01%	0.767%	37.22%	90.34%	90.34%
<i>Bi-LSTM</i>	76.72%	52.01%	52.01%	48.35%	40.99%	75.29%	75.29%
<i>Hybrid</i>	76.72%	52.01%	52.01%	2.452%	40.99%	79.39%	79.39%

Table 10. – SR achieved by optimizing for the DA metric

<i>Model type</i>	SMA200	RSI	SMA200 & RSI	Trust Strategy	Direction Strategy	SMA200 & Trust	SMA200 & Direction
<i>LSTM</i>	0.912	0.795	0.795	0.019	0	0.929	0.929
<i>GRU</i>	0.912	0.795	0.795	0.292	0.371	1.076	1.076
<i>Bi-LSTM</i>	0.912	0.795	0.795	1.155	0.402	0.93	0.93
<i>Hybrid</i>	0.912	0.795	0.795	0.338	0.402	0.968	0.968

What is evident from our results is that, despite the GRU model type delivering the best outcomes in price prediction, the strategies with the most promising returns and Sharpe ratios are those associated with LSTM, Bi-LSTM, Hybrid, and GRU models. Notably, the Trust Strategy displayed substantial volatility in its performance. However, when combined with the 200-day Simple Moving Average (SMA) indicator, it exhibited more stable results.

This combination with the technical indicator appeared to significantly benefit the strategies, enabling them to achieve returns that are not only equivalent but sometimes even surpass the baseline strategy. On average, the combined strategies outperformed the baseline strategies, both in terms of returns and Sharpe ratios. We can see one such case in Figure 18, where the combination of the model and the technical indicator achieved the best results.



Figure 18. – Trading results for the GRU model type when DA is optimized

7.3. Overall evaluation

In summary, our evaluation reveals that the GRU model excelled in price prediction, while the most favorable trading results were achieved by integrating model strategies with a stable and static technical indicator. A noteworthy outcome of the hyperparameter optimization is its ability to consistently find optimal architectures, evident in the repeated occurrence of identical metric values, such as an ME of 10.15.

7.3.1. Possible improvements

My work offers several aspects for improvement to enhance efficiency, robustness, and reliability. One part involves diversifying the model types used. While my work exclusively explored RNN-based neural networks, there are alternative approaches to price prediction, such as Autoregressive Integrated Moving Average (ARIMA) [40], Support Vector Machine (SVM) [41], or Reinforcement Learning Models [42]. These additional model types could provide valuable alternatives and may yield improved results in different contexts.

Furthermore, the trading strategy could be extended with the use of Machine Learning models [43]. By introducing such models into the trading framework, the strategies could become more dynamic, adaptable, and responsive. In particular, Reinforcement Learning Models stand out as a promising alternative for replacing static trading strategies [43][44]. These models can be designed to learn and adapt their strategies based on the feedback and profits they generate.

Improvements in the optimization process are also feasible. One approach is to provide Tree-structured Parzen Estimator (TPE) with a broader array of parameters to tune or to devise a more complex architecture. This enhancement would enable the optimizer to explore a wider spectrum of configurations, which could lead to better results and refined model architectures.

Lastly, expanding our dataset to include a variety of new data types could further elevate the predictive capabilities of our model. Integrating additional technical indicators,

incorporating data from quarterly earning reports, and factoring in real-time news sentiment on the company could significantly extend our dataset. In the highly volatile world of the stock market, any additional data is potentially invaluable for our model in accurately forecasting market movements.

8. SUMMARY

This research undertook the task of employing RNN-based Neural Networks to predict Google's daily closing prices, aiming to translate these predictions into viable trading strategies. The study methodically explored the theoretical foundations of the employed model types and the intricacies of hyperparameter optimization. The culmination was an insightful interaction between RNN-based models and hyperparameter optimization, showcasing its potential in the realm of practical trading.

The initial segment of the study provided a detailed introduction to the model types selected for price predictions. It elucidated their distinguishing features and operational mechanisms. In tandem, the research also addressed various hyperparameter optimization techniques, objectively discussing their respective merits and limitations.

Shifting its lens to the applicative side, the research carefully implemented trading strategies, testing them against actual stock market data. Notably, the use of the 200-day Simple Moving Average (SMA) was integrated to augment the stability of the trading approach. This combination of a conventional technical indicator with the predicted prices yielded promising results, as reflected by a commendable Sharpe Ratio of 1.

The research didn't shy away from a thorough evaluation, encompassing diverse trials and configurations to validate its findings. Through this process, the study managed to discern the nuances between different models and strategies, shedding light on configurations that seemed to resonate best with the data at hand.

Concluding, the research marked a step forward in the realm of trading strategies backed by neural network predictions. While the implemented strategies showcased potential, the study also acknowledged the broader scope available within the dataset. It subtly hinted at the prospect of refining these strategies further, emphasizing the continuous journey towards improved trading methodologies.

9. REFERENCES

- [1] Kang, Z., Li, X., Li, Z., & Zhu, S. (2019). Data-driven robust mean-CVaR portfolio selection under distribution ambiguity. *Quantitative Finance*, 19(1), 105-121.
- [2] Bauwens, L., Laurent, S., & Rombouts, J. V. (2006). Multivariate GARCH models: a survey. *Journal of applied econometrics*, 21(1), 79-109.
- [3] Piccolo, D. (1990). A distance measure for classifying ARIMA models. *Journal of time series analysis*, 11(2), 153-164.
- [4] Gardner Jr, E. S. (1985). Exponential smoothing: The state of the art. *Journal of forecasting*, 4(1), 1-28.
- [5] Chalapathy, R., & Chawla, S. (2019). Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407*.
- [6] Heaton, J. B., Polson, N. G., & Witte, J. H. (2017). Deep learning for finance: deep portfolios. *Applied Stochastic Models in Business and Industry*, 33(1), 3-12.
- [7] Jiang, Z., & Liang, J. (2017, September). Cryptocurrency portfolio management with deep reinforcement learning. In *2017 Intelligent systems conference (IntelliSys)* (pp. 905-913). IEEE.
- [8] Fischer, T. G. (2018). *Reinforcement learning in financial markets-a survey* (No. 12/2018). FAU Discussion Papers in Economics.
- [9] Li, J., Wang, X., Lin, Y., Sinha, A., & Wellman, M. (2020, April). Generating realistic stock market order streams. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 34, No. 01, pp. 727-734).
- [10] Assefa, S. A., Dervovic, D., Mahfouz, M., Tillman, R. E., Reddy, P., & Veloso, M. (2020, October). Generating synthetic data in finance: opportunities, challenges and pitfalls. In *Proceedings of the First ACM International Conference on AI in Finance* (pp. 1-8).
- [11] Chen, K., Zhou, Y., & Dai, F. (2015, October). A LSTM-based method for stock returns prediction: A case study of China stock market. In *2015 IEEE international conference on big data (big data)* (pp. 2823-2824). IEEE.
- [12] Islam, M. S., & Hossain, E. (2021). Foreign exchange currency rate prediction using a GRU-LSTM hybrid network. *Soft Computing Letters*, 3, 100009.
- [13] Yang, L., & Shami, A. (2020). On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415, 295-316.

- [14] McCann, B., Bradbury, J., Xiong, C., & Socher, R. (2017). Learned in translation: Contextualized word vectors. *Advances in neural information processing systems*, 30.
- [15] Sherstinsky, A., 2020. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *Physica D: Nonlinear Phenomena*, 404, p.132306.
- [16] Y. Bengio, P. Simard and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," in *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157-166, March 1994, doi: 10.1109/72.279181.
- [17] Pascanu, R., Mikolov, T. and Bengio, Y., 2013, May. On the difficulty of training recurrent neural networks. In *International conference on machine learning* (pp. 1310-1318). Pmlr.
- [18] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," in *Neural Computation*, vol. 9, no. 8, pp. 1735-1780, 15 Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.
- [19] Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.
- [20] Irie, K., Tüske, Z., Alkhouli, T., Schlüter, R., & Ney, H. (2016, September). LSTM, GRU, highway and a bit of attention: An empirical overview for language modeling in speech recognition. In *Interspeech* (pp. 3519-3523).
- [21] Jang, B., Kim, M., Harerimana, G., Kang, S. U., & Kim, J. W. (2020). Bi-LSTM model to increase accuracy in text classification: Combining Word2vec CNN and attention mechanism. *Applied Sciences*, 10(17), 5841.
- [22] Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2).
- [23] Bergstra, J., Bardenet, R., Bengio, Y., & Kégl, B. (2011). Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24.
- [24] Appel, G., & Dobson, E. (2007). *Understanding MACD* (Vol. 34). Traders Press.
- [25] Hyndman, R. J. (2011). Moving Averages.
- [26] Gumparthi, S. (2017). Relative strength index for developing effective trading strategies in constructing optimal portfolio. *International Journal of Applied Engineering Research*, 12(19), 8926-8936.
- [27] Paul, P. V. (2022). Stock Market Prediction Based on Technical-Deviation-ROC Indicators Using Stock and Feeds Data. *Recent Advances in Computer Science and Communications (Formerly: Recent Patents on Computer Science)*, 15(3), 367-375.

- [28] Ying, X. (2019, February). An overview of overfitting and its solutions. In *Journal of physics: Conference series* (Vol. 1168, p. 022022). IOP Publishing.
- [29] Hawkins, D. M. (2004). The problem of overfitting. *Journal of chemical information and computer sciences*, 44(1), 1-12.
- [30] Jo, J. M. (2019). Effectiveness of normalization pre-processing of big data to the machine learning performance. *The Journal of the Korea institute of electronic communication sciences*, 14(3), 547-552.
- [31] Ali, P. J. M., Faraj, R. H., Koya, E., Ali, P. J. M., & Faraj, R. H. (2014). Data normalization and standardization: a technical report. *Mach Learn Tech Rep*, 1(1), 1-6.
- [32] Singh, D., & Singh, B. (2020). Investigating the impact of data normalization on classification performance. *Applied Soft Computing*, 97, 105524.
- [33] Iliev, A. I., & Anand, A. (2023, March). Huber Loss and Neural Networks Application in Property Price Prediction. In *Future of Information and Communication Conference* (pp. 242-256). Cham: Springer Nature Switzerland.
- [34] Kandel, I., & Castelli, M. (2020). The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. *ICT express*, 6(4), 312-315.
- [35] Beers, B. (2020). What Is Buy and Hold? How the Investing Strategy Works. *Investopedia*
- [36] Wallach, D., & Goffinet, B. (1989). Mean squared error of prediction as a criterion for evaluating and comparing system models. *Ecological modelling*, 44(3-4), 299-306.
- [37] Saud, A. S., & Shakya, S. (2020). Analysis of look back period for stock price prediction with RNN variants: A case study on banking sector of NEPSE. *Procedia Computer Science*, 167, 788-798.
- [38] Yamak, P. T., Yujian, L., & Gadosey, P. K. (2019, December). A comparison between arima, lstm, and gru for time series forecasting. In *Proceedings of the 2019 2nd international conference on algorithms, computing and artificial intelligence* (pp. 49-55).
- [39] Sharpe, W. F. (1998). The sharpe ratio. *Streetwise—the Best of the Journal of Portfolio Management*, 3, 169-85.
- [40] Ariyo, A. A., Adewumi, A. O., & Ayo, C. K. (2014, March). Stock price prediction using the ARIMA model. In *2014 UKSim-AMSS 16th international conference on computer modelling and simulation* (pp. 106-112). IEEE.

- [41] Lin, Y., Guo, H., & Hu, J. (2013, August). An SVM-based approach for stock market trend prediction. In *The 2013 international joint conference on neural networks (IJCNN)* (pp. 1-7). IEEE.
- [42] Meng, T. L., & Khushi, M. (2019). Reinforcement learning in financial markets. *Data*, 4(3), 110.
- [43] Wu, X., Chen, H., Wang, J., Troiano, L., Loia, V., & Fujita, H. (2020). Adaptive stock trading strategies with deep reinforcement learning methods. *Information Sciences*, 538, 142-158.
- [44] Yang, H., Liu, X. Y., Zhong, S., & Walid, A. (2020, October). Deep reinforcement learning for automated stock trading: An ensemble strategy. In *Proceedings of the first ACM international conference on AI in finance* (pp. 1-8).