UNIVERSITÄT
KOBLENZ · LANDAU

Fachbereich 4: Informatik

Design a ML system that finds optimal entries and exits with an emphasis on closing times

Projektpraktikumsarbeit
im Studiengang Informatik
mit Anwendungsfach Wirtschaftsinformatik

submitted by

Anchal Gera, Leonard Katz, Muralikrishna Naripeddi, Leo Strauch

Gutachter: Prof. Dr. Thomas Burkhardt, Abteilung Finanzierung, Finanzdienstleistungen und eFinance, Fachbereich 4: Institut für Management, Universität Koblenz-Landau
Zweitgutachter: Dipl. Inf. Heiko Neuhaus, Abteilung Finanzierung, Finanzdienstleistungen und eFinance, Fachbereich 4: Institut für Management, Universität Koblenz-Landau

Koblenz, im Oktober 2023

Abstract - Leo Strauch & Leonard Katz

This paper delves into the utilization of Machine Learning, particularly Long Short-Term Memory (LSTM) networks, for forecasting the stock prices for individual stocks. The study is situated within the broader context of the increasing integration of Artificial Intelligence (AI) in the financial sector, along with the numerous challenges accompanying this advancement. The research methodology employed for this study adheres to the Design Science Research (DSR) approach, which prioritizes the development of practical solutions to address specific issues.

The paper centers on the creation of a tool that leverages an LSTM model to estimate the recommended holding duration for a given stock at a specific price point. The tool aims to benefit smaller investors who may lack the resources or time to continuously monitor their investments. It provides a daily rating indicating the likelihood of a specific stock reaching a predefined value, thereby assisting users in making well-informed decisions.

Various research methodologies were assessed and compared, leading to the selection of LSTM networks due to their superior ability to model temporal patterns and capture long-term dependencies. The LSTM model was implemented using the TensorFlow framework, with the architecture comprising three LSTM layers and dropout layers to mitigate overfitting. The output layer of the model comprises a single neuron responsible for predicting a stock's closing price.

The model's training process involved the collection of historical stock data, data preprocessing to extract relevant features, and model training using historical data spanning from 2019 to 2023. The model's performance was assessed using Mean Squared Error (MSE) and Root Mean Squared Error (RMSE). The paper also elaborates on the data preprocessing and feature engineering steps required to prepare the data for the LSTM model.

The study culminates with an overview of the model's implementation, highlighting the various Python libraries utilized and offering a step-by-step walkthrough of the model's training, testing, and visualization procedures. It showcases the model's efficacy in predicting stock prices, exemplifying a practical application of LSTM networks in the financial sector.

This paper contributes to the expanding body of research concerning AI's role in finance and presents a practical tool for individual investors to enhance their investment decisions. It also underscores the potential of LSTM networks in modeling intricate temporal patterns, rendering it a valuable resource for researchers and practitioners in the AI and finance domains.

Kurzfassung

Diese Arbeit befasst sich mit dem Einsatz von maschinellem Lernen, insbesondere von Long Short-Term Memory (LSTM)-Netzwerken, zur Vorhersage von Aktienkursen für einzelne Aktien. Die Studie ist im breiteren Kontext der zunehmenden Integration von Künstlicher Intelligenz (KI) im Finanzsektor angesiedelt, zusammen mit den zahlreichen Herausforderungen, die mit dieser Entwicklung einhergehen. Die für diese Studie angewandte Forschungsmethodik folgt dem Design Science Research (DSR)-Ansatz, der die Entwicklung praktischer Lösungen für spezifische Probleme in den Vordergrund stellt.

Es wurden verschiedene Forschungsmethoden bewertet und verglichen, wobei die Wahl auf LSTM-Netze fiel, da diese besser in der Lage sind, zeitliche Muster zu modellieren und langfristige Abhängigkeiten zu erfassen. Das LSTM-Modell wurde mit dem TensorFlow-Framework implementiert, wobei die Architektur drei LSTM-Schichten und Dropout-Schichten umfasst, um eine Überanpassung zu vermeiden. Die Ausgabeschicht des Modells besteht aus einem einzigen Neuron, das für die Vorhersage des Schlusskurses einer Aktie verantwortlich ist.

Der Trainingsprozess des Modells umfasste die Sammlung historischer Bestandsdaten, die Vorverarbeitung der Daten zur Extraktion relevanter Merkmale und das Training des Modells anhand historischer Daten von 2019 bis 2023. Die Leistung des Modells wurde anhand des mittleren quadratischen Fehlers (MSE) und des mittleren quadratischen Wurzelfehlers (RMSE) bewertet. In der Studie werden auch die Schritte der Datenvorverarbeitung und des Feature-Engineerings erläutert, die zur Vorbereitung der Daten für das LSTM-Modell erforderlich sind.

Die Studie endet mit einem Überblick über die Implementierung des Modells, wobei die verschiedenen verwendeten Python-Bibliotheken hervorgehoben werden und ein schrittweiser Durchgang durch die Trainings-, Test- und Visualisierungsverfahren des Modells angeboten wird. Es zeigt die Wirksamkeit des Modells bei der Vorhersage von Aktienkursen und veranschaulicht damit eine praktische Anwendung von LSTM-Netzwerken im Finanzsektor.

Diese Arbeit leistet einen Beitrag zur wachsenden Zahl von Forschungsarbeiten über die Rolle der KI im Finanzbereich und stellt ein praktisches Werkzeug für Privatanleger zur Verbesserung ihrer Anlageentscheidungen vor. Sie unterstreicht auch das Potenzial von LSTM-Netzwerken bei der Modellierung komplexer zeitlicher Muster und ist damit eine wertvolle Ressource für Forscher und Praktiker im Bereich KI und Finanzen.

Keywords: Machine Learning, LSTM, Stock Prediction, TensorFlow, FinTech

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Mit der Einstellung dieser Arbeit in die Bibliothek bin ich einverstanden. ja ☐ [✓ nein ☒]

Der Veröffentlichung dieser Arbeit im Internet stimme ich zu. ja ☐ [✓ nein ☒]

Koblenz, den 1st March 2025

Contents

List of abbreviations

MSE	Mean Squared Error
RMSE	Root Mean Squared Error
LSTM	Long short-term memory
BE-LSTM	Backward Elimination Long Short-Term Memory
ML	Machine learning
cuDNN	NVIDIA CUDA Deep Neural Network
AI	Artificial Intelligence
DSR	Design Science Research
RL	Reinforcement Learning
AMZN	Amazon, representing a stock symbol
GPU	Graphics Processing Unit
API	Application Programming Interface
GUI	Graphical User Interface
NN	Neural network
RNN	Recurrent Neural Network
DRL	Deep Reinforcement Learning
DQN	Deep Q-Networks
tanh	Hyperbolic tangent

Chapter 1

Introduction - Leonard Katz

“Machine Learning is on the rise” [Fin22]. The integration of artificial intelligence in the financial sector has begun a new time of data analysis and refining investment strategies. No investor can escape this trend. However, the power of AI coexists with its limitations. Despite the pressure of these new algorithms, it is not possible to predict the entire complexities of financial markets. If it could, the market would change in such a way that it could not be predicted again [MSE+23]. Therefore, AI can only be a tool that helps us make better, more informed decisions.

There are various approaches to creating tools for the financial market. The tools that are widely considered to be useful all have one thing in common: They provide the best-suited solution for one particular task. [YHM+23]. This paper will focus on the concept of holding time for a stock. Utilizing artificial intelligence, a tool is developed to provide an estimate of how long it is advisable to hold a given stock at a specific price.

The determination of holding time is particularly significant for smaller investors. Large investors possess the time and resources to continuously monitor their investments at any time. Consequently, they can promptly respond to market fluctuations and maximize their potential profits. Conversely, smaller investors, who sporadically inspect their portfolios, do not always have the luxury of such swift reactions. Introducing a tool that provides users with a daily score indicating the likelihood of reaching a specific stock value becomes invaluable. This tool informs users whether they need to monitor their portfolios more closely or if minimal movement in their investments is expected for the day.

The methodology for creating this tool involves gathering historical stock data, preprocessing the data to extract pertinent features, and training the AI model using advanced algorithms like neural networks.

1.1 Research question of the work

The focal point of this paper centers on the creation of a machine learning system designed to estimate the holding time of a specific stock. Users will receive a score indicating the probability of a particular stock reaching a predefined value.

1.2 Structure of work

The initial section of this paper will provide a comprehensive background on the fundamental principles of machine learning in fintech and offer insights into the finance market relevant to this study. Subsequently, a chapter on the methodology will follow. This chapter will not only outline the research methods to be employed but also incorporate a literature review addressing the research question posed in this paper.

Following the discussion of these foundational aspects, an in-depth exploration of the development and design of the algorithms' inner workings will be presented. This exploration will culminate in a chapter dedicated to the evaluation of the developed algorithms, followed by a glimpse into future prospects for this research area.

Chapter 2

Background - Muralikrishna Naripeddi

2.1 Challenges in Model Training

To develop a proficient and intelligent machine learning model, it is essential to understand the hidden risks and issues associated with missing values. The development of a machine learning model encompasses several stages, including model selection, data collection, data filtering, data partitioning, data validation, and the utilization of performance metrics. To effectively train a model, it is imperative to comprehend and validate each of these steps. There exist various challenges and difficulties in the domain of model training in machine learning, some of which are detailed below. [MLMLC22]

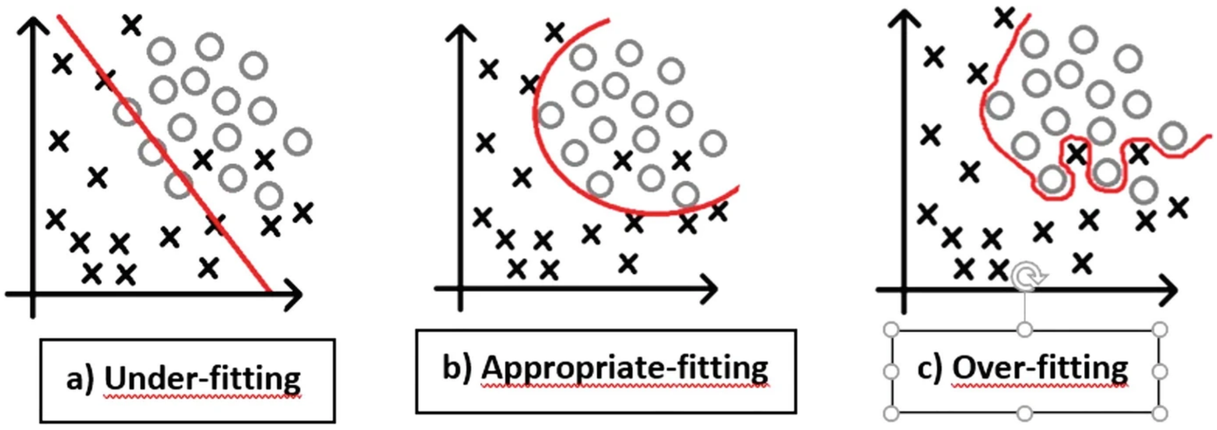

Figure 2.1: Illustration of three models with different fitting[MLMLC22]

2.1.1 Overfitting

Overfitting is a situation in which a machine learning model produces accurate results when applied to a training dataset but fails to generalize well to new datasets, leading to incorrect results. Overfitting occurs when the model meticulously captures every data point in the training dataset. The primary contributors to overfitting are noisy data and an excessive amount of irrelevant information within the dataset. [MLMLC22]

2.1.2 Underfitting

Underfitting arises when a machine learning model is trained on a limited set of data or features, resulting in inefficient performance and inaccurate results. The key reason for underfitting lies in having low variability and high bias during model training. The underfitting problem occurs when the machine learning model fails to encompass all the valuable datasets and features. [MLMLC22]

2.2 Financial Data

Financial data plays an important role in modern investment and trading, serving as the source for making informed decisions, risk management and strategy development. This section delves into the significance and the diverse types of financial data types used in modern market dynamics.

2.2.1 Types of Financial Data

Financial data encompasses various types of data that solve various kinds of machine learning practices in financial analysis. These data types provide the base for training and testing various machine learning practices.[SZK17]

Price Data

Price data is time series data including open, high, low, and close prices of financial instruments over a specific time period. These data are indispensable for modeling price movements and predicting future price changes using regression models and time series analysis.[SZa]

Volume Data

Volume data is data showing the number of shares or contracts traded during a specific time interval. Volume data plays a critical role in assessing liquidity and market activity. In machine learning, they are employed for developing trading volume prediction models and detecting unusual trading patterns.

Fundamental Data

Fundamental data is data encompassing financial statements, earnings, revenue, debt, and other fundamental metrics of companies. Machine learning models utilize fundamental data for quantitative analysis, including financial statement prediction and credit risk assessment.[SZb]

Economic Data

Economic data is data such as GDP growth rates, inflation, interest rates, and employment figures. With the help of economic data machine learning models analyze the impact of economic indicators on asset prices and develop economic forecasting models.[APF]

News and Sentiment Data

News and Sentiment data is data that is extracted from news articles, social media, and financial news sources reflecting market sentiment. Machine learning models analyze sentiment data to assess market sentiment and predict price movements based on public sentiment.[Alp]

2.2.2 Financial Data Sources

Financial data comes from various sources. This kind of data source is used by investors, traders and analysts Some of the sources include[Alp]

Stock Exchanges

Stock exchanges constitute the primary source of real-time and historical trading data, providing official market prices and trading volumes.

Financial News Outlets

Prominent news outlets like Bloomberg, Reuters, and CNBC disseminate financial news, analysis, and market data to the public. With their in-house financial analysts, they analyze the market data to cater to the needs of the viewer.

Data Providers

Specialized data providers furnish comprehensive datasets encompassing a wide spectrum of financial information. Frequently, these providers offer data through APIs, facilitating seamless integration into trading and analytical platforms.

APIs

Application Programming Interfaces (APIs) have gained increasing popularity for accessing financial data. APIs such as the Yahoo Finance API (yfinance) empower developers and analysts to programmatically retrieve data, streamlining the process of building trading strategies and conducting financial analysis.

2.2.3 yFinance API

[Ran23] Yahoo Finance API often referred to as yFinance is an API to get financial data. yFinance has many compelling reasons to use it as a financial data source.

1. Simplicity and Ease of use: yFinance is simple and easy to use. With only a few lines of Python code users can draw a wealth of financial data. This ease of use makes it easy for yFinance against experienced programmers and those who are new to coding.

2. Comprehensive Data Coverage: yFinance offers a wide range of financial data, including historical and real-time price data, volume data and more.

3. Integration with Python: As a Python library, yFinance seamlessly integrates into Python environments. This is very advantageous for analysts as they can make use of native Python libraries.

4. Open source and Free: yFinance is open-source, which means that it is free to use. This accessibility makes it a cost-effective choice for both trading enthusiasts and veteran analysts alike.

5. Community Support: yFinance library benefits from an active and always engaged user community. This support system is very valuable when tackling an error.

Installation and Setup

1. Installing yFinance can be accomplished using the standard Python package management tool 'pip'. The library can be installed by using the following command: 'pip install yFinance'.

2. Once installed yFinance can be imported to the Python script with the following line of code.

```
import yFinance as yf
```

3. With yFinance imported to the Python script, users can access the financial data by creating a ticker object. This can be done by using the following lines of code.

```
# To retrieve historical data for Apple Inc. (AAPL)
apple = yf.Ticker("AAPL")
historical_data = apple.history(period="5y")
```

Retrieving data is as simple as specifying just the ticker symbol and the date range. With the above two lines of code, five years' worth of historical price data from Apple can be retrieved.

2.3 Machine Learning Techniques

In this section, we will look into a variety of machine learning algorithms and techniques that are employed in the predictive analysis done on the stock market. Although it is not feasible to predict market movements consistently, the use of predictive stock price systems is seen to be increasing in stock trading mechanisms as they can be useful for the basis of risk management tools[HSK18]. The machine learning techniques and a study using the technique follows

2.3.1 Regression models

Regression models are used for predicting continuous values, such as the future price of an asset. Some common regression models used in trading are Linear Regression, Polynomial Regression, and Support Vector Regression.

Stock price prediction using support vector regression on daily and up-to-the-minute prices

[HSK18] This study done by Henrique et al. (2018) is centered on the prediction of stock prices utilizing Support Vector Regression (SVR) with daily and high-frequency price data. It assesses SVR models with diverse kernel functions and periodic update strategies on various Brazilian, American, and Chinese stocks. The investigation revealed that SVR when employed with a linear kernel and periodical updates, outperformed random walk-based models in predicting stock prices and this was particular for blue-chip and small-cap stocks across the three countries. The study highlights the importance of model updates and a connection between SVR price prediction and volatility. The limitations of this study encompass issues related to data quality and computational complexities, along with recommendations for future research to contemplate additional variables and update frequencies.

2.3.2 Time Series Analysis

Time series forecasting methods are specifically designed for sequential data like historical stock prices. Techniques like Auto Regressive Integrated Moving Average (ARIMA), Generalized Auto Regressive Conditional Heteroskedasticity (GARCH), and Exponential Smoothing are commonly applied to predict future prices.

Study of Effectiveness of Time Series Modeling (Arima) in Forecasting Stock Prices

[MSG14] This study done by Mondal et al. (2014) is focused on the effectiveness of the ARIMA model in predicting the stock prices for 56 Indian stocks from various sectors. The ARIMA models were selected based on the Akaike Information Criterion. The study examined the impact of varying the span of previous period data on the prediction accuracy. The results showed that ARIMA provided relatively high accuracy. The best results came in the Fast Moving Consumer Goods sector, while the banking and automobile sectors had lower accuracy. The study also indicated that ARIMA is a useful model for stock price prediction, but some sectors may require alternative approaches for getting higher accuracy. The standard deviation analysis demonstrated variations in accuracy among different sectors. The paired t-tests did not reveal significant differences in accuracy across various training datasets, suggesting that training data length did not significantly affect the accuracy of predictions.

2.3.3 Machine Learning Classifiers

Classification models can be used to predict binary outcomes, such as whether to buy or sell an asset. Random Forest, Decision Trees, and Support Vector Machines can be employed for classification tasks.

Forecasting Stock Index Movement: A Comparison of Support Vector Machines and Random Forest

[KT06] In this study the authors Kumar et al. (2006) conducted a comparative analysis of Support Vector Machines (SVM) and Random Forest in predicting the daily movements of the S&P CNX NIFTY Index, alongside traditional discriminant and logit models, as well as neural networks and other techniques. The results demonstrated that SVM performed better than Random Forest, neural networks, and traditional models, mainly because it followed the principle of structural risk minimization, which prioritizes reducing generalization error over training error alone.

The authors concluded that while SVM exhibited a slight advantage over Random Forest, both models show promise for further assessment across diverse financial time series and stock market indices. These machine learning methods have the potential to aid traders, investors, and financial experts in rendering more informed investment decisions, presenting various trading strategies that can lead to financial gains. Subsequent research could explore the amalgamation of SVM with other classification models to balance their strengths and weaknesses and consider additional macroeconomic variables for stock market index forecasting.

2.3.4 Neural Networks

Deep learning techniques, like the Long Short-Term Memory (LSTM) and Convolutional Neural Networks (CNN), can capture complex patterns in time-series data. LSTMs are particularly effective for sequential data like stock prices.

Short-term Stock Market Price Trend Prediction Using a Comprehensive Deep Learning System

[SS20] This study, conducted by Shen et al. (2020), comprises three major components: the extraction and pre-processing of a Chinese stock market dataset, feature engineering, and stock price trend prediction using Long Short-Term Memory (LSTM) neural networks. The authors collected, cleaned, and organized two years of Chinese stock market data, incorporating a unique computational element termed feature extension.This feature engineering method enhances the efficacy and efficiency of model construction by employing principal component analysis (PCA) and recursive feature elimination (RFE). By integrating these feature engineering techniques with the LSTM prediction model, the system outperformed leading models in related studies in terms of prediction accuracy. In an effort to bridge the knowledge gap between investors and researchers, the authors conducted a comprehensive evaluation and identified topics for further research. The study stands out for its substantial feature engineering and the customization of a deep learning prediction system, resulting in a noticeable enhancement in model performance.

While the research yielded positive results, the authors note that there is ample room for further investigation. Given the sensitivity of the RFE algorithm to term lengths beyond two days, weeks, and biweekly intervals, a thorough exploration of the influence of technical indices on irregular term lengths is warranted. Furthermore, the incorporation of sentiment analysis techniques and diverse information sources such as tweets and news into feature engineering and deep learning models holds promise for the development of a more comprehensive prediction system. This study underscores the potential for further advancements at the intersection of technical and financial research domains.

Stock Market Prediction using CNN and LSTM

[HE21] In this study conducted by Hamoudi et al. (2021), the primary focus was on developing prediction models for high-frequency automated algorithmic trading using 1-Dimensional Convolutional Neural Networks (1D CNNs) and Long Short-Term Memory (LSTM) networks. In contrast to traditional regression models, they approached the problem through binary classification, with the goal of classifying trading opportunities as either positive or negative. The positive class encompasses trades resulting in returns in the top ten percentile, underscoring high-return opportunities. Additionally, the study addresses missing data by augmenting the feature matrix with a logical array, ensuring the preservation of pertinent data. They evaluate the performance of various CNN and LSTM models using diverse machine learning metrics and investment risk and return measures. The results indicate that the LSTM model with a deeper architecture attains the best performance in terms of total return, average return per trade, and risk, despite the lower recall value. This novel approach in the study highlights the effectiveness of 1D CNNs in identifying promising trading opportunities in high-frequency stock trading, offering valuable insights for algorithmic trading strategies.

2.3.5 Reinforcement Learning

Reinforcement learning algorithms, such as Q-learning or Deep Q-Networks (DQN), can be used to determine optimal trading actions. They learn to make decisions by maximizing cumulative rewards over time.

An integrated framework of genetic network programming and multi-layer perception neural network for prediction of daily stock return: An application in Tehran stock exchange market

[RPE19] This study done by Ramezanian et al., introduces an integrated framework that merges Genetic Network Programming (GNP) and Multi-Layer Perceptron (MLP) neural networks to predict daily stock returns, applied within the context of the Tehran Stock Exchange market. This integration of GNP, a machine learning algorithm grounded in genetic programming, with the MLP neural network aims to heighten prediction accuracy for stock returns. GNP is used for feature selection, facilitating the identification of the most pertinent input features for stock return prediction, thereby improving the model's efficiency and interpretability. The MLP neural network, a potent tool for capturing intricate patterns in financial data, is subsequently utilized to model the relationships between the chosen features and stock returns. Additionally, the study incorporates reinforcement learning techniques to further optimize the model's performance. This integrated framework combines the strengths of GNP, MLP, and reinforcement learning to present a comprehensive approach to stock return prediction. The results underscore the potential of this integrated model in effectively forecasting daily stock returns in the Tehran Stock Exchange market, providing valuable insights for investors and market participants.

2.3.6 Ensemble Methods

Ensemble methods are the combination of multiple models into an ensemble for getting better predictions. Techniques like Bagging (e.g., Random Forest) and Boosting (e.g., AdaBoost) are commonly used.

Ensemble system based on genetic algorithm for stock market forecasting

[GPB15] This study by Gonzalez et al. (2015) aims to predict the weekly directional movement of the Bovespa Index. To accomplish this, they employ a combination of genetic algorithms (GA), technical analysis tools, and machine learning techniques. The authors devise an ensemble system that enhances the classification accuracy of Support Vector Machine (SVM) Ensembles by optimizing parameters and selecting features using GA. Technical indicators are employed to extract relevant aspects from historical data encompassing stock and forex markets.

In their experimental assessments, the suggested approach is compared to well-established ensemble learning strategies such as Bagging, Boosting, and Random Forest. The results consistently demonstrate that the suggested strategy outperforms these alternative approaches in forecasting the weekly price movement of the Bovespa Index. Consequently, the study concludes that their approach presents a viable alternative for stock market forecasting.

The authors also identify several potential directions for future research. First, they propose exploring new technical indicators or alternative types of financial data to generate more informative features. Second, fine-tuning the parameters of each SVM through a grid search based on the final GA solution could potentially enhance model performance. Third, an analysis of financial series with more extensive historical data may enable the proposed method to uncover additional patterns from past data. Lastly, they recommend developing an algorithm that leverages the parallelization capabilities of Genetic Algorithms and Ensemble Systems to expedite the training process.

Chapter 3

Methodical part - Leonard Katz

This chapter consists of two substantial sections. The first section is dedicated to a comprehensive exploration of research methods, wherein a range of methodologies is deliberated. Following this exploration, the most appropriate methodology is carefully selected for further investigation. The subsequent section delves deeply into the literature, employing a systematic approach to present a thorough analysis.

3.1 Research method

In this section, an evaluation of various research methodologies was conducted to determine the most suitable approach. Design Science Research (DSR) was selected for this paper due to its primary focus on developing practical solutions, whether in the form of software or hardware, to address specific problems. DSR's emphasis on actionable outcomes aligns perfectly with the objectives of this study, making it the ideal choice for conducting the research.

The paper will evaluate three different approaches to determine the most appropriate one for the study.

3.1.1 Peffer

The Design Science Research (DSR) approach outlined in Peffer et al. [PRTV12] is visualized in Figure 3.1. Various research entry points are depicted in circles within the graphic,

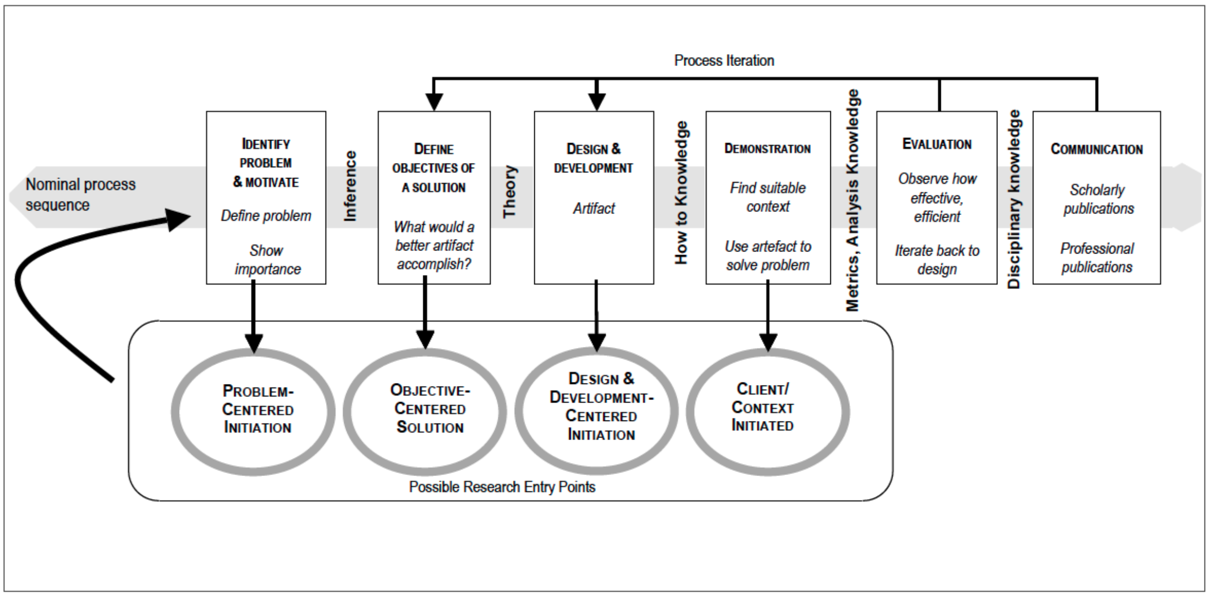
Process Iteration
Nominal process sequence
IDENTIFY PROBLEM & MOTIVATE
Define problem
Show importance
Inference
DEFINE OBJECTIVES OF A SOLUTION
What would a better artifact accomplish?
Theory
DESIGN & DEVELOPMENT
Artifact
How to Knowledge
DEMONSTRATION
Find suitable context
Use artefact to solve problem
Metrics, Analysis Knowledge
EVALUATION
Observe how effective, efficient
Iterate back to design
Disciplinary knowledge
COMMUNICATION
Scholarly publications
Professional publications
PROBLEM-CENTERED INITIATION
OBJECTIVE-CENTERED SOLUTION
DESIGN & DEVELOPMENT-CENTERED INITIATION
CLIENT/CONTEXT INITIATED
Possible Research Entry Points

Figure 3.1: DSR from Peffer et al. [PRTV12]

offering multiple options. Depending on the research intention, one or several entry points are chosen, initiating the process sequence with the identification of the problem and the motive (Phase 1). Here, the key problem is precisely defined. Subsequently, the objective of a solution is articulated (Phase 2), and the design and development of an artifact commence (Phase 3). Following this, a demonstration of the artifact is constructed (Phase 4). It is possible to adjust the entry point to align with the workflow of phases 1 through 4, a modification is necessary if the key problem is not specified adequately. Phase 5 involves evaluating the artifact, while in Phase 6, the artifact is presented and published to an audience. In both final stages, it is feasible to return to Phases 2 and 3. This approach allows for the redefinition of objectives if the artifact does not align with the application, or the artifact can be refined to better meet its demand.

3.1.2 Hevner

Hevner et al. [Hev07] divides the DSR process into three cycles:

1. Relevance cycle

2. Rigor cycle

3. Design cycle

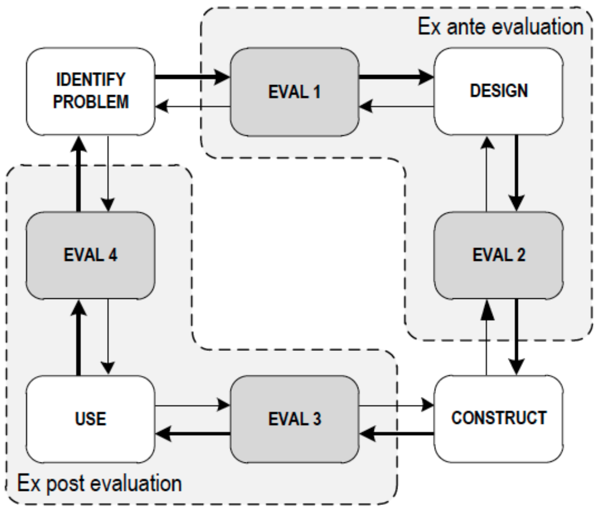

Figure 3.2: DSR from Sonnenberg [SVB12]

In the relevance cycle, the environment is evaluated to define the so-called "business needs". In this cycle, the goal is to find the actors, structures, problems and opportunities involved on the environmental side. This is similar to the first two phases of Peffers.

The rigor cycle creates a knowledge base in which knowledge, theory and other artifacts are collected. This is necessary to abstract the acquired knowledge and to draw principles and theoretical conclusions. The rigor cycle can be passed through simulations as the relevant cycle.

The final cycle is the design cycle, in which the artifact is built and developed. After the development, the artifact is evaluated with the help of the, in the relevance cycle defined, "business needs". This cycle is reiterated until all requirements have been satisfied. It is also possible to run through the relevance and rigor cycle again at any given time as needed.

3.1.3 Sonnenberg

In [SVB12], the Design Science Research approach of Hevner is further elaborated. Here, the method is divided into two distinct parts: "Build" and "Evaluate". These parts correspond to the six phases outlined in Hevner's approach. The "Build" part encompasses the first three phases, while the "Evaluate" part encompasses phases 4, 5, and 6. What sets Sonnenberg's approach apart is the iterative nature of transitioning between these two parts, as illustrated in Figure 3.2. This iterative process allows for continuous refinement and improvement. Notably, this DSR methodology is particularly valuable for developing software tailored to precisely defined tasks.

3.1.4 Conclusion

For this , I've chosen Sonnenberg's Design Science Research methodology because it's highly effective. It gives a clear structure that's perfect for solo work like mine. Sonnenberg's approach emphasizes thorough evaluation, which fits well with the detailed focus of this project. By following this methodology, I aim to deliver robust insights and innovative solutions.

3.2 Literature review

The following section will begin with an introductory overview outlining the methodology employed for the literature review. Subsequently, the individual papers under consideration will be systematically presented. Following this, a comprehensive classification of the aforementioned papers will follow, culminating in a nuanced and substantiated conclusion.

3.2.1 Method

To provide a comprehensive overview of AI in exit point calculation, a systematic approach has been adopted for the literature review. The following steps have been meticulously followed:

1. Brief overview of the topic
2. Creation of a query of keywords
3. Selection of a database of scientific research
4. Refining the Keywords and setting other parameters

5. Examination of the found papers

For obtaining a concise understanding of the topic, the search engine Google was utilized. This approach demonstrated the increasing popularity of the topic not only in academic circles but also in non-scientific media. A list of keywords was generated from online articles and subsequently inputted into Google Scholar.

"Google Scholar provides a simple way to broadly search for scholarly literature" [Abo23]. Its advanced article scanning capabilities allow Google Scholar to offer a more extensive collection of papers compared to similar databases. In certain respects, Google Scholar functions as a meta-database, guiding users to other databases. However, a drawback lies in its inclusivity, it encompasses a wide array of sources. Therefore, it becomes imperative to assess the credibility of individual articles.

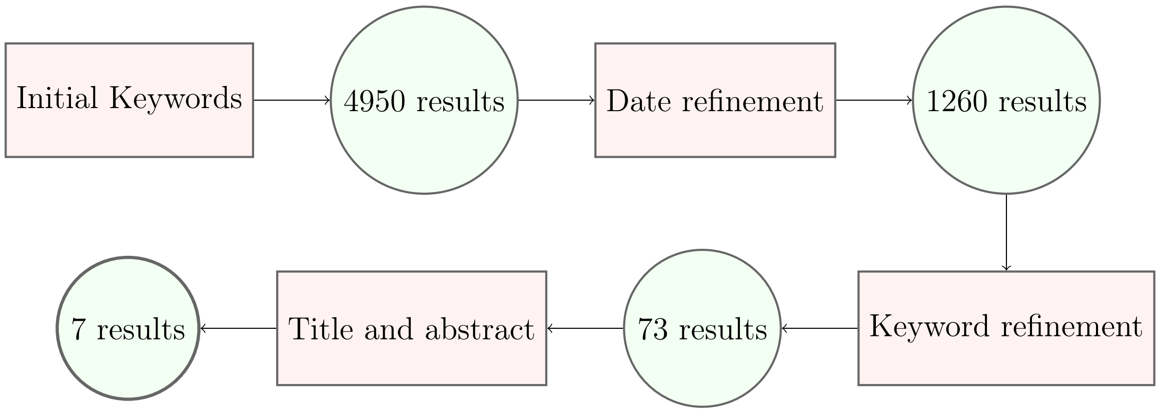

Figure 3.3: Keyword/Litterateur pipeline

The initial set of keywords comprised the following terms: "ML systems, financial market, opening, closing, exit points, AI, FinTech". This initial query yielded 4950 results, which were reduced to 1260 when limited to the year 2023. Following multiple refinements, only 73 results remained as of October 3rd, 2023. The final query of keywords includes: "financial machine learning, opening, closing, exit points, ML, AI, FinTech, trading, enclosure length, stock market, trading strategies, portfolio optimization, Algorithmic Trading, Time Series Analysis". The last step involves sifting through the remaining results and extracting those relevant to the study's focus and integrity. This is all illustrated in fig. 3.3.

3.2.2 The AI Revolution

The report [MSE+23] from the Alan Turing Institute offers a comprehensive and systematic analysis of the application of AI in the financial sector. This in-depth study meticulously examines the current utilization of AI technologies in finance, delves into their potential applications, and carefully explores the associated risks and challenges posed by these transformative technologies. In conclusion, it is imperative to provide recommendations for effectively managing this new technology across three key sectors: academia, industry, and regulation.

After an introduction to the basics of AI, the report proceeds to go over the benefits of AI in finances. The benefits can be summarized into three categories:

1. Improving existing processes
2. Reducing potential flaws
3. Automatic trading

Next, the report discusses "threats & potential pitfalls"[MSE+23, p. 19]. AI can not only exacerbate existing systemic risks ask the question of responsibility. Whether the responsibility of job displacement and de-skilling of employees or challenges with privacy and transparency in model training, the report appeals to the responsible. Finally, the report presents possible regulations. "However, not all outcomes of the use of AI in finance should be regulated" [MSE+23, p. 26].

3.2.3 Comprehensive Review on Financial Explainable AI

Financial Explainable AI (FinXAI) focuses on white-box AI systems. These systems adhere to a transparent line of reasoning, ensuring that users have a clear understanding of the system's operations and the reasons behind its decisions. Consequently, users are always aware of what the system is doing and why it is making specific choices.

The literature review conducted by Zhang et al. analyzed over 50 studies on FinXAI, checking if they met the specified criteria [YHM+23, p. 111:7-111:8]:

- *Trustworthiness*: Explainable AI focuses on enhancing user trust by ensuring transparency in AI models' decisions. It establishes trust by providing a clear understanding of how the model operates, fostering confidence in users affected by its decisions.

- *Fairness*: Refers to providing AI solutions and explanations uniformly to every user and stakeholder, eliminating potential biases in the process.

- *Informativeness*: Clear problem statements and explanations play a vital role in enhancing both social and performance aspects. They assist in identifying relevant features, facilitate debugging, and prevent overfitting by emphasizing crucial parts of the data.

- *Accessibility*: Increasing accessibility of algorithms to non-experts promotes broader acceptance of AI. Complex algorithms often discourage adoption due to the required training and concerns about unintended consequences. Simplifying explanations alleviates user concerns, thereby encouraging organizations to embrace AI technologies.

- *Privacy Awareness*: Accountable personnel must restrict third-party access to user data, ensuring integrity and privacy, which are crucial in the sensitive financial sector.

- *Confidence*: AI models must provide not only outcomes but also confidence levels, allowing experts to identify uncertainties in results and captured data regions. Stable predictions demonstrate confidence, and explanations are deemed trustworthy when they remain consistent across diverse data inputs.

- *Transferability*: Understanding the behavior of models facilitates adaptation, reducing the workload for experts and enhancing transferability – an essential aspect for future AI models.

As observed in Table 3.1, most studies do not meet the criteria of trust, fairness, privacy, confidence, and transferability. Accessibility is the only criterion that is commonly fulfilled in the studies examined.

Trustworthiness	9
Fairness	2
Informativeness	22
Accessibility	44
Privacy Awareness	0
Confidence	2
Transferability	2
Total papers reviewed	54

Table 3.1: Results of the analysis of Zheng et al. [YHM+23]

3.2.4 AI Techniques for Decision-Making in Market Environments

This thesis [Thé23] encompasses applied research, fundamental research, and sustainable research. Applied research concentrates on AI-driven solutions for decision-making in energy and stock markets using Deep Reinforcement Learning (DRL). Fundamental research enhances DRL techniques for stochastic markets, with a focus on distributional RL for learning complete probability distributions. Additionally, the thesis addresses a sustainable dimension by formalizing a decision-making problem in energy markets to improve the synchronization of power consumption and renewable energy production.

3.2.5 Artificial intelligence in EU securities markets

This article [ESM23], by the European Security and Markets Authority (ESMA), offers an overview of AI applications in EU securities markets, emphasizing the growing adoption in asset management for investment, risk management, and compliance purposes.

The integration of AI in securities markets exhibits variations across sectors and entities, with Natural Language Processing (NLP) gaining widespread usage for text processing. In asset management, AI finds applications in investment, risk management, and compliance; however, few funds openly disclose their utilization of AI. In trading, AI enhances execution and post-trade processes, optimizing the impacts of large orders and predicting settlement failures. Although AI supports specific activities, its adoption has not resulted in rapid overhauls of business processes due to technological constraints, client preferences, and regulatory uncertainty. Market participants view potential regulatory frameworks as crucial for instilling trust in AI adoption. Risks associated with AI implementation include complexity, lack of transparency, and the potential concentration of AI systems among major players. Operational concerns and compatibility issues with legacy technology hinder some firms from implementing AI and Machine Learning (ML) algorithms. Additionally, existing governance structures often lack specific compliance personnel overseeing AI development.

3.2.6 On Forecasting Cryptocurrency Prices

This paper [MRCV23] addresses the challenge of accurately predicting cryptocurrency prices, a task complicated by their uncertainty and volatility. Existing research in this area is limited, often concentrating on well-known cryptocurrencies and lacking consistency and generality. The study introduces a comprehensive framework that compares statistical, machine learning (ML), and deep learning (DL) approaches for five popular cryptocurrencies. The research explores the novel use of the Temporal Fusion Transformer (TFT), along with hybrid models and ensembles. Results indicate that DL methods, particularly Long Short-Term Memory (LSTM), consistently outperform other approaches across all cryptocurrencies.

LSTM, ARIMA and SVR are already discussed in Chapter 2. The other methods examined by Murray et al. [MRCV23] are the following:

- *k-Nearest Neighbor (kNN)*: This is a straightforward and intuitive machine learning algorithm commonly employed for both classification and regression tasks. "The kNN model has been used in financial forecasting, electric market price prediction, and in the prediction of Bitcoin." [MRCV23, p. 198].

- *Random Forest (RF) Regressor*: Operates as an ensemble of decision trees, each constructed on a random subset of the training set. Predictions made by Random Forest are derived by averaging individual tree predictions, demonstrating its strong generalization capability and minimal sensitivity to hyperparameters.

- *Gated Recurrent Unit (GRU)*: Similar to LSTM, GRU is also a type of recurrent neural network (RNN). GRU simplifies the LSTM structure by merging the memory cell and hidden state, thereby reducing the number of gating mechanisms. Therefore, achieving a better runtime.

- *LSTM-GRU (HYBRID)*: This method was proposed by Patel et al. [PTGK20] to avail of the advantages of both LSTM and GRU.
- *Temporal Convolution Network (TCN)*: In contrast to traditional RNNs and LSTMs, TCN employs 1D dilated convolutions to capture long-range dependencies in sequential data more efficiently.
- *Temporal Fusion Transformer (TFT)*: This is an advanced deep learning model specifically designed for time series forecasting tasks. It combines the strengths of the transformer architecture, which excels at capturing complex patterns in sequential data, with techniques for handling temporal patterns.

Murray et al. proposed training various NN on five cryptocurrencies:

1. XRP
2. Bitcoin (BTC)
3. Litecoin (LTC)
4. Ethereum (ETH)
5. Monero (XMR)

As seen in tab.3.2 “LSTM consistently outperforms [...] the other models” [MRCV23, p. 204], but the other methods do not lag far behind.

3.2.7 Forecasting of NIFTY 50 Index Price by Using Backward Elimination with an LSTM Model

This paper [JAEC$^+$23] explores the integration of emerging technology and artificial intelligence (AI) for predicting the NIFTY 50 index price, proposing the BE-LSTM model as an effective alternative to traditional methods such as fundamental and technical analysis.

Modell	RMSE
LSTM	0.023
GRU	0.0231
Hybrid	0.0231
KNN	0.0232
TCN	0.0232
Arima	0.0232
TFT	0.0232
RF	0.0232
SVR	0.0233

Table 3.2: Simplified results from [MRCV23]

The study demonstrates the superiority of the Backward Elimination Long Short-Term Memory (BE-LSTM) over LSTM in accurately predicting stock prices. The financial industry is increasingly adopting technology, and this research provides valuable insights into portfolio management, equity analysis, and market trends. By leveraging AI, this study assists policymakers and investors in understanding market volatility and making informed investment decisions. It underscores the importance of investor education, risk management, and the use of AI-driven predictive models to enhance market participation and ensure a stable economy.

3.2.8 Automated market maker inventory management with deep reinforcement learning

This paper [VFG23]presents a robust approach for developing a market-making agent based on Reinforcement Learning (RL). The agent dynamically adjusts its portfolio based on a desired liquidity ratio by employing a reward function. This reward function enables the agent to modulate portfolio risk dynamically while seeking profitability. Unlike static policies driven by conventional reward functions, the approach in this paper allows the market-making agent to adapt to market conditions, considering factors like cash and portfolio holdings. Experimental results, comparing the approach with random baseline agents and alternative reward functions, demonstrate its efficiency and robustness in modulating operations and managing portfolio risk, shedding light on the underlying strategies employed by the agent.

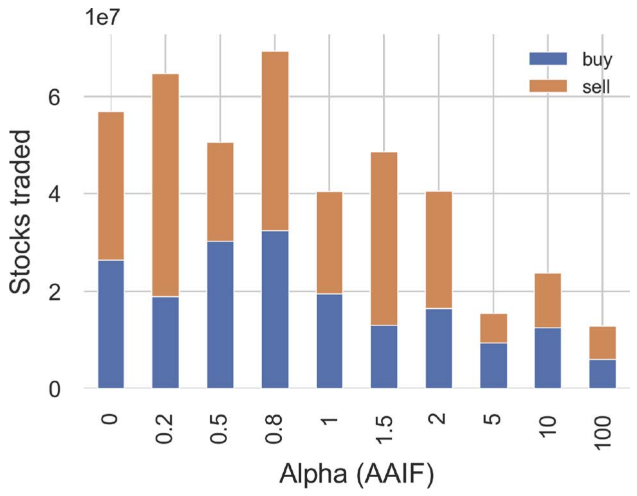

Figure 3.4: Results from [VFG23]

The approach presented in this paper still needs improvement in risk management. The “Alpha Inventory Impact Factor” (AIIF) is a factor that attempts to incorporate risk. The higher the factor becomes, the more the AI is penalized for holding a stock. As the factor increases, the number of stocks traded decreases (see Figure 3.4).

3.2.9 Conclusion of the literature review

In summary, this literature review offers a comprehensive overview of the intersection between artificial intelligence and the financial sector. By exploring various innovative models, such as BE-LSTM for stock price prediction and RL-based market-making agents, it becomes evident that the industry is embracing data-driven solutions at an unprecedented pace.

Furthermore, this review sheds light on the diverse applications of AI in finance, ranging from predicting cryptocurrency prices using advanced algorithms to enhancing market liquidity through intelligent agents. The comparison of these methodologies against traditional techniques underscores the transformative power of artificial intelligence in revolutionizing financial decision-making processes.

However, amid the rapid advancements and transformative potential, it’s imperative to acknowledge the challenges and potential pitfalls accompanying the integration of AI in finance. One of the significant concerns is the ethical implications of AI-driven decision-making, especially in high-stakes financial scenarios. Algorithmic biases, if not carefully addressed, can perpetuate existing inequalities, leading to unfair market advantages and biased outcomes. Moreover, the complexity of these AI systems poses a challenge in terms of regulation and oversight. Striking a balance between fostering innovation and ensuring regulatory compliance remains a daunting task.

As the financial industry continues its rapid technological evolution, this literature review underscores the importance of these advanced methodologies. These innovations not only enhance profitability but also provide valuable insights for investors, policymakers, and industry professionals. The findings emphasize the critical role of artificial intelligence and reinforcement learning in reshaping financial strategies and risk management paradigms. Looking ahead, the integration of these technologies is poised to redefine the future landscape of finance, ushering in an era of adaptive, data-informed decision-making processes.

Chapter 4

Design and Development - Leo Strauch

4.1 Data Collection and Preprocessing

We utilized the **yfinance** library [Ran23] to retrieve historical data. Our data collection began on January 1, 2019, and concluded on January 1, 2023. To prevent the LSTM from being trapped in a local optima, we implemented data scaling, transforming the financial data into a range between 0 and 1. This scaling procedure is vital for improving the model's performance.

Date	Open	High	Low	Close	Adj Close	Volume
2020-01-02	93.750	94.905	93.207	94.901	94.901	80580000
2020-01-03	93.225	94.310	93.225	93.748	93.749	75288000
2020-01-06	93.000	95.185	93.000	95.144	95.144	81236000
2020-01-07	95.225	95.695	94.602	95.343	95.343	80898000
2020-01-08	94.902	95.550	94.322	94.599	94.599	70160000
2020-01-09	95.494	95.891	94.790	95.052	95.052	63346000
2020-01-10	95.269	95.347	94.000	94.158	94.158	57074000
2020-01-13	94.565	94.900	94.040	94.565	94.565	55616000
2020-01-14	94.294	94.355	92.927	93.472	93.472	68818000
2020-01-15	93.613	93.943	92.755	93.101	93.101	57932000

Table 4.1: Received Data from the yfinance API

4.1.1 Feature Engineering

Feature engineering is a crucial step in the data preparation process for machine learning models. In this section, we provide a detailed explanation of the specific steps taken to transform raw financial data into informative features that serve as inputs for our predictive models. This meticulous feature engineering process is instrumental in enhancing the accuracy and effectiveness of our predictive algorithms. [DE21]

Lookback period

To provide our predictive models with valuable historical context, we selected a lookback period of p=60 trading days. This timeframe allows the model to consider previous market trends and conditions, which are critical elements in forecasting potential future stock prices. The choice of a 60-day lookback period was a well-considered decision, striking a delicate balance between capturing essential historical data and maintaining computational efficiency. This careful consideration ensures that our models remain both effective and computationally manageable.

Model Selection

The process of selecting the most suitable machine learning model is a pivotal and critical decision in the context of our stock price prediction system within financial markets. In this section, we offer a comprehensive explanation of our model selection procedure. Here, we delve into the underlying rationale that informs our choices, elucidating why these models are particularly well-aligned with the distinct and specific requirements of our predictive task. This systematic and considerate model selection process stands as the foundational pillar, ensuring both the accuracy and overall effectiveness of our stock price prediction system. [DE21]

1. **MLP Models:** Multi-layer perceptron (MLP) models provide the versatility and robust performance required to handle non-linear data patterns. This flexibility is particularly valuable when addressing the intricate and dynamic nature of financial

markets. The second paper showcased their potential by attaining superior results compared to traditional factor models. [KK19]

2. **1D CNNs:** While 1D CNN models are frequently employed in image classification, their effectiveness in analyzing sequential data is well-documented. Their capability to discern local data patterns aligns with our goal of predicting market trends. [KK19]

3. **LSTM networks:** LSTM networks have been chosen as the foundational element of our model architecture due to their ability to deliver a 4% to 7% improvement in accuracy compared to other models. Furthermore, they are particularly well-suited for modeling temporal patterns, as exemplified in [ZYM20]. Additionally, they can autonomously identify the most pertinent patterns, thereby mitigating the risk of overfitting.

   LSTM networks have a more sophisticated architecture that incorporates memory cells and gating mechanisms. This design enables them to capture long-term dependencies more effectively. [KGC+23]

4.2 Model Architecture

The architectural framework of our machine learning model plays a vital role in the program's ability to predict stock prices in financial markets. In this section, we provide a comprehensive explanation of this architectural design, which encompasses insights into the layer configuration, unit quantities, and the specific design decisions that underlie our models.

4.2.1 TensorFlow

TensorFlow is an open-source machine learning framework developed by the Google Brain team. It is designed to facilitate the creation and deployment of machine learning models, including deep learning models, for a wide range of applications. TensorFlow provides a flexible and scalable platform for building and training machine learning models using both CPUs and GPUs. [AAB+15]

We will be using **Tensorflow** for our model creation as it suits the scope of our problem. When deploying an LSTM with TensorFlow the Framework decides what layer implementations will chosen, the available runtime hardware, and constraints [AAB+15]:

1. cuDNN-based
2. pure-TensorFlow

Wenn certain configuration requirements of the LSTM are met. Namely the default [AAB+15]:

1. activation == tanh
   (a) Activation function to use.
   (b) Default: hyperbolic tangent (tanh).
2. recurrent_activation == sigmoid
   (a) Activation function to use for the recurrent step.
   (b) Default: sigmoid
3. recurrent_dropout == 0
   (a) Float between 0 and 1. Fraction of the units to drop for the linear transformation of the recurrent state. Default: 0.
4. unroll == False
   (a) Boolean (default False). If True, the network will be unrolled otherwise, a symbolic loop will be used. Unrolling can speed up an RNN, although it tends to be more memory-intensive. Unrolling is only suitable for short sequences.
5. use_bias is True
   (a) Boolean (default True), whether the layer uses a bias vector.

6. Inputs, if using masking, are strictly right-padded.
    (a) A 3D tensor with shape [batch, timesteps, feature].
7. Eager execution is enabled in the outermost context.
    (a) Eager execution is enabled by default.

Since our Code meets these criteria, depending on the available runtime hardware and constraints TensorFlow decides what layers are used. [AAB+15] We are using a total of three LSTM layers, where each of these Layers processes the input sequences with 50 neurons each. Between each of the layers is one dropout layer which randomly sets input bits to 0 with a specified frequency of r = 0.2 at each training step. [AAB+15] After the Layers follows one Dense Layer, with a single neuron. Said layer is responsible for predicting the closing price of one neuron.

In the following, there is a more detailed description of the layers which are present in our deployed model.

1. **Input Layer:** The cornerstone of our LSTM model is the input layer, which assumes a crucial role in processing input sequences. These sequences represent historical data and serve as the fundamental building blocks for predicting future market trends. This initial input layer is pivotal to the model's ability to analyze and interpret historical information, facilitating accurate trend forecasting. [AAB+15]
2. **Multiple Hidden Layers:** We include multiple LSTM hidden layers, with each layer housing a customizable number of units. This multi-layer architecture is intended to capture intricate temporal dependencies inherent in financial time series data. The precise number of hidden layers and the units per layer are determined through experimentation and optimization to guarantee that the model effectively learns from the data. [AAB+15]
3. **Dropout Layers:** To mitigate overfitting and bolster generalization, we introduce dropout layers after each LSTM hidden layer. These dropout layers randomly deactivate a portion of the neurons during training, effectively reducing the likelihood

of overfitting and enhancing the model's capacity to generalize to unseen data. [AAB+15]

4. **Output Layer:** The final layer of our LSTM model is the output layer, tasked with generating predictions. In our particular case, the output layer consists of a single neuron, a decision made because we are forecasting a continuous variable, specifically the closing price. The primary aim of this model is to predict the closing price of a financial instrument with precision and accuracy. [AAB+15]

4.2.2 Training and Evaluation

Training and evaluation are crucial phases in the development of our machine learning models. In this section, we provide a comprehensive overview of both the training process and the rigorous evaluation criteria employed to assess their performance.

Training Process: Our models undergo a systematic and methodical training process to extract insights from historical financial data and refine their predictive capabilities. This iterative training procedure is essential for fine-tuning the models to achieve optimal performance.

1. **Normalize Data:** We utilize the **MinMaxScaler** from **sklearn.preprocessing** to standardize feature values. This scaling guarantees that all features contribute uniformly to the model by preventing any single feature from overpowering the learning process. [PVG+11]

2. **Model Training:** We define the number of epochs and the batch size for optimization, and we perform error backpropagation using the Adam optimizer from TensorFlow/Keras. [AAB+15]

3. **Dropout Layers:** In our LSTM models, we include Dropout layers from Keras to bolster model robustness. The dropout rate is fine-tuned during training to strike a balance between reducing overfitting while preserving the model's capacity to learn and generalize effectively, as discussed in [AAB+15]. This strategic use of dropout layers significantly enhances the model's overall performance and reliability [FK17].

4. **Mean Squared Error (MSE):** The computes the average of the squared differences between predicted values and actual values. A lower MSE indicates better predictive accuracy. Mathematically, it is defined as:

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

5. **Root Mean Squared Error (RMSE):** The RMSE involves taking the square root of the MSE to render the error metric more interpretable. It is a valuable metric for understanding the scale of prediction errors. It is calculated as:

$$RMSE = \sqrt{MSE}$$

Therefore, we can assess how closely the model predicts stock prices in comparison to the real data.

Here, we present the prediction of the stock price for AMZN, using training data from 2021-01-01 to 2023-01-01, with a

1. Mean Squared Error (MSE) of 61.867081419568585
2. Root Mean Squared Error (RMSE) of 7.865563007157758

4.3 Implementation

In this section, we will delve into the details of how the implementation works and elucidate the reasons behind the necessity of certain aspects.

4.3.1 Libraries

The code heavily relies on various Python libraries that empower the project to leverage the capabilities of Machine Learning. In addition, a few smaller libraries are in use, although they won't be further elaborated upon in this discussion.

1. **Pandas** [tea20]

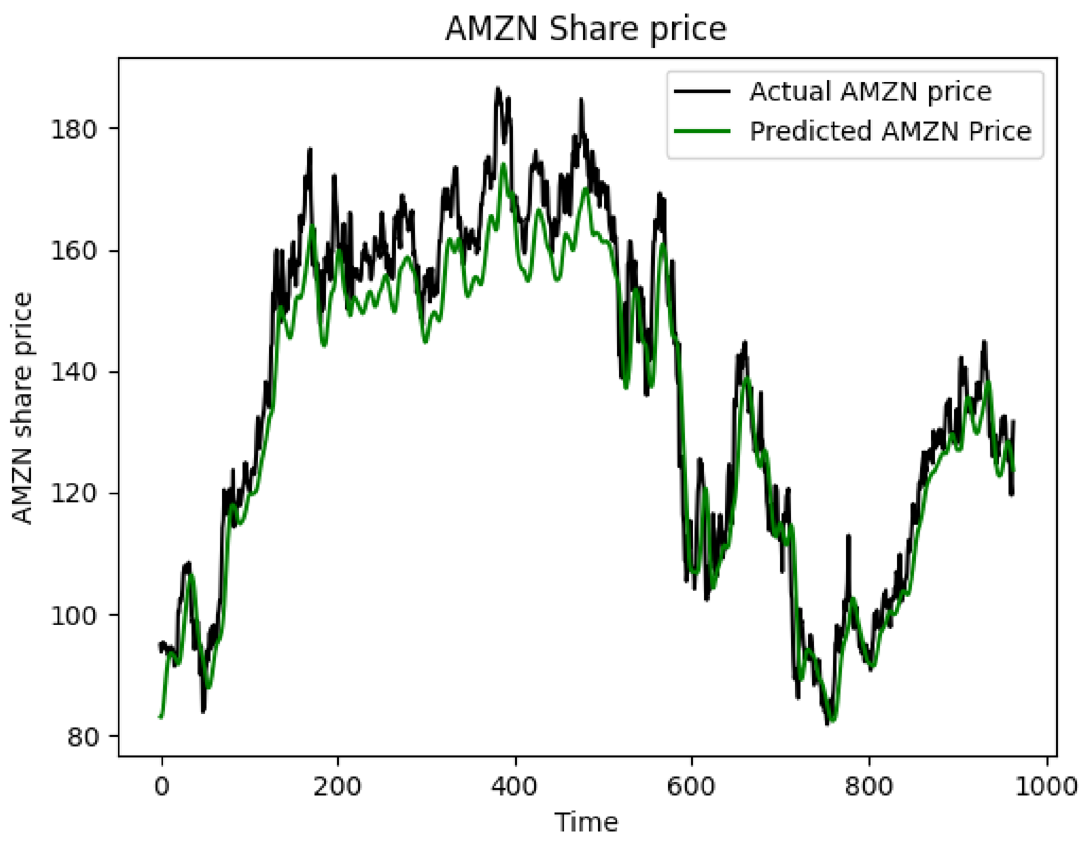

Figure 4.1: Predicted and real price of the AMZN price.

(a) Pandas is a powerful library for data manipulation and analysis in Python.

(b) It provides data structures like DataFrames and Series, which are well-suited for handling and analyzing structured data.

(c) Pandas allows you to read, write, filter, and manipulate data easily, making it a popular choice for data preprocessing and cleaning in data science and analysis tasks.

2. **Tkinter** [Lun99]

   (a) Tkinter is Python's standard library for creating graphical user interfaces (GUIs).

   (b) Tkinter is a simple tool to develop GUI applications quickly, which is why it was chosen.

3. **NumPy** [HMW+20]
   (a) NumPy is a fundamental library for numerical computing in Python.
   (b) It provides support for arrays and matrices, along with a wide range of mathematical functions to perform operations on these data structures which makes it essential for most mathematical computing.
4. **sklearn** [PVG+11]
   (a) Scikit-Learn is a machine learning library that provides tools for data analysis and modeling, including classification, regression, clustering, and more.
   (b) **mean_squared_error** is a function in Scikit-Learn that calculates the mean squared error, a common metric for evaluating the performance of regression models. It measures the average squared difference between the actual and predicted values.

   (a) **yfinance** [Ran23]
      i. yfinance is a Python library that provides a simple and convenient interface to access financial data from Yahoo Finance.
      ii. yfinance offers a straightforward way to download financial data for analysis, research, and modeling in Python.
      iii. It was selected for this project because it streamlines the process of obtaining financial data in comparison to manual web scraping or the use of intricate APIs.

4.3.2 Date Handling and Stock Data Retrieval

The user is required to enter a stock handle and an entry date, which then is used to acquire the data from said date till 2023-01-01 from the **yfinance**.

4.3.3 Data Preprocessing

In the data preprocessing stage, the code initially employs the scikit-learn library to execute MinMaxScaling [PVG+11] on the Closeprices of the stock data, which standardizes the values within the range of 0 to 1. Subsequently, it defines the time window the machine learning model uses to consider past data when predicting future stock prices. Finally, it prepares the training data by constructing sequences.

4.3.4 Building and Training the LSTM Model

The code generates a Sequential model utilizing the TensorFlow/Keras API, a high-level neural network library. It incorporates three LSTM layers with dropout layers [FK17] in between to mitigate overfitting. The ultimate layer is a Dense layer responsible for forecasting the closing price of the following day. The model undergoes compilation with the Adam optimizer and employs the mean squared error (MSE) loss function. Subsequently, it undergoes training on the training data.

4.3.5 Testing and Visualization

The code retrieves the current stock prices for the test period. It preprocesses the test data in a manner consistent with how the training data is prepared. The model generates predictions for the test data, and these predictions are subsequently converted back to their original scale using the Min-Max scaler [PVG+11]. The actual and predicted stock prices are graphed using matplotlib. The code calculates and displays the Root Mean Squared Error (RMSE) and Mean Squared Error (MSE) between the actual and predicted prices. The results are also displayed within the Tkinter GUI.

Chapter 5

Overview - Anchal Gera

5.1 Overview

This chapter serves as a compilation of previously covered material and provides a comprehensive summary of the previous chapters in abbreviated bullet points. This approach is intended to provide readers with a quick and concise overview of the work done in earlier sections of the paper. In the last section a short explanation of code is also provided.

5.2 Introduction-Summary

The rise of artificial intelligence and machine learning in the financial sector has led to a shift in data analysis and investment strategies. Investors now see AI tools as essential; However, they also have their own limitations. Financial markets, characterized by their immense complexity, elude AI's perfect prediction. Rather, AI serves as a tool to facilitate more informed decisions.

5.2.1 Focus on Holding Time

This paper centers its attention on the notion of the holding time for stocks and harnesses the capabilities of artificial intelligence to ascertain the optimal duration for retaining a specific stock at a given price. This aspect assumes paramount importance, particularly for smaller investors who may lack the means for perpetual investment monitoring.

5.2.2 Methodology

The methodology encompasses the compilation of historical stock data, the extraction of pertinent features through preprocessing, and the subsequent training of an AI model utilizing advanced algorithms, such as neural networks.

5.2.3 Research Question

At the heart of this work lies the research question: Can a machine learning system be devised to estimate the holding duration of an individual stock and furnish users with a probability score for attaining a predefined value?

5.2.4 Paper Structure

The structure of this paper encompasses an introductory exposition of machine learning in the financial technology domain, an extensive treatment of the employed methodology, an exhaustive examination of algorithmic development, an evaluation of the effectiveness of the formulated algorithms, and a glimpse into prospective research directions.

The integration of artificial intelligence in the financial sector has ushered in a new era of data analysis and the refinement of investment strategies. In this paper, we explore the concept of holding time for stocks, utilizing artificial intelligence to provide an estimate of how long it is advisable to hold a given stock at a specific price. This is particularly significant for smaller investors who may not have the resources to continuously monitor their investments. We present a machine learning system that offers users a daily score indicating the likelihood of a stock reaching a predefined value, thus informing them whether they need to monitor their portfolios more closely. The paper outlines the methodology for creating this tool, including data collection, feature extraction, and AI model training. It also delves into the research question and structure of the work, offering insights into the development and design of the algorithms, their evaluation, and future prospects.

5.3 Background Summary

5.3.1 Challenges in Model Training

Developing a proficient and intelligent machine learning model necessitates a comprehensive understanding of the challenges associated with missing values in the model development process. The model development process encompasses various stages, including model selection, data collection, data filtering, data partitioning, data validation, and the utilization of performance metrics. To effectively train a model, it is imperative to comprehend and validate each of these steps. Within the domain of machine learning model training, a multitude of challenges and difficulties exist, some of which are elaborated upon in the subsequent sections.

5.3.2 Overfitting-Underfitting

Overfitting is a scenario in which a machine learning model produces accurate results when applied to a training dataset but falters in generalizing well to new datasets, thereby yielding inaccurate results. This phenomenon materializes when the model meticulously captures every data point within the training dataset. Overfitting predominantly stems from noisy data and an excessive inclusion of irrelevant information within the dataset.

Underfitting materializes when a machine learning model is trained on a limited set of data or features, resulting in suboptimal performance and inaccurate outcomes. The primary reason for underfitting lies in the manifestation of low variability and high bias during model training. Underfitting occurs when the machine learning model fails to encompass all valuable datasets and features.

5.3.3 Financial Data

Financial data assumes a pivotal role in contemporary investment and trading, serving as the linchpin for informed decision-making, risk management, and strategy formulation. This section delves into the significance and the diverse typologies of financial data utilized in modern market dynamics.

5.3.4 Types of Financial Data

Financial data spans an array of data categories that cater to distinct machine learning practices in the realm of financial analysis. These diverse data categories form the bedrock for training and assessing a myriad of machine learning methodologies. They encompass price data, volume data, fundamental data, economic data, and news and sentiment data.

5.3.5 Financial Data Sources

Financial data emanates from diverse founts and constitutes an indispensable resource for investors, traders, and analysts. Among the provenance of financial data are stock exchanges, financial news agencies, data providers, and Application Programming Interfaces (APIs).

5.3.6 Machine Learning Techniques

This section embarks on an exploration of an assortment of machine learning algorithms and techniques employed in the predictive analysis of stock markets. Although the consistent prediction of market movements poses a formidable challenge, the adoption of predictive stock price systems is witnessing an upswing owing to their instrumental role in risk management and informed decision-making.

The techniques that come under scrutiny encompass regression models, time series analysis, machine learning classifiers, neural networks, and reinforcement learning. Each technique is illustrated through pertinent studies that underscore their efficacy and potential in the domain of stock market prediction.

5.4 Research Methodologies Summary:

- A full mapping of research methods is done.
- Design Science Research (DSR) was chosen as the most appropriate research methodology because it focuses on practical solutions to specific problems.
- Three different DSR approaches are evaluated: Peffer, Hevner and Sonnenberg.

5.4.1 Peffer:

- Peffer's DSR approach includes several starting points in the research process.
- The process includes problem identification, objective formulation, artifact design and development, artifact presentation, evaluation, and public presentation.
- Starting points can be adjusted according to research intent, allowing to redefine targets or refine artifacts.

5.4.2 Hevner:

- Hevner's DSR process is divided into three cycles: importance cycle, density cycle and planning cycle.
- The relevance cycle focuses on understanding business needs and environmental factors.
- Rigor cycle collects information and theories, summarizes information obtained.
- The design cycle involves building and evaluating the artifact, ensuring that all requirements are met.

5.4.3 Sonnenberg:

- The DSR approach of Sonnenberg shares the methodology "Buildänd TaxicParts corresponding to stages of Hevner.
- The iterative nature allows for continuous refinement and transition between these parts.
- Valuable for developing software adapted for specific tasks.

5.4.4 Conclusion:

- Sonnenberg's DSR methodology has been chosen for its effectiveness, especially for large projects.

- It provides a well-defined framework and enables collaboration.

5.5 Design and Development - Summary :

- The chapter describes the methodology used in the literature review.
- The systematic approach includes an overview of the topic, selection of keywords, selection of the database and subsequent refinement.
- Google Scholar is used for many types of scientific literature.
- The original set of keywords is expanded and refined to narrow down relevant articles.

5.5.1 The Revolution of Artificial Intelligence:

- Alan Turing Institute report examines artificial intelligence and applications in finance.
- It discusses the benefits, threats and possible regulations of artificial intelligence in finance.

5.5.2 Comprehensive overview of economic explanatory artificial intelligence:

- The review analyzes more than 50 studies of Financial Explainable AI (FinXAI) based on reliability, fairness and accessibility, among others.
- Most studies do not fully meet these criteria.

5.5.3 Artificial intelligence technologies to make decisions in the market environment:

- The thesis combines applied, basic and sustainable research on AI-based decision-making in energy and stock markets.

- Handles the synchronization of electricity consumption and renewable energy production.

5.5.4 Artificial intelligence on the EU securities market:

- The ESMA article provides an overview of AI applications in EU securities markets, with an emphasis on adoption in asset management, trading and risk management.
- It addresses the challenges and risks associated with the implementation of artificial intelligence in the financial sector.

5.5.5 Cryptocurrency price forecast:

- The work focuses on accurately predicting cryptocurrency prices using various machine learning and deep learning methods.
- LSTM consistently outperforms other models in this context.

5.5.6 NIFTY 50 index price forecast:

- This paper presents the BE-LSTM model for NIFTY 50 index price forecasting and highlights its superiority over traditional methods.
- It highlights the importance of AI-based forecasting models to understand market volatility.

5.5.7 Automated Market Maker Stock Management with Deep Reinforced Learning:

- The paper presents a robust approach to developing a market making agent using reinforcement learning to manage portfolios and liquidity.
- The agent adjusts its portfolio dynamically based on the reward function.

5.5.8 Conclusion of the literature review:

- The review highlights the transformative power of artificial intelligence in financial decision-making.
- Challenges, ethical implications and the need for regulation in the financial sector are discussed.
- Artificial intelligence and reinforcement learning are poised to redefine the future of finance through data-driven decision-making.

Collection and Pre-processing of Data:

- Historical data was collected using the yfinance library from January 1, 2019 to January 1, 2023.
- Data was scaled between 0-1 to improve model performance.

Schedule Features:

- Feature planning is essential when preparing data for machine learning models.
- The historical context of the model was chosen for the observation period of 60 business days.
- Model selection process including selection of MLP models, 1D-CNN networks, and LSTM networks.

Model Architecture:

- Explanation of the model and architectural design.
- Using TensorFlow to build and train models.
- Description of LSTM model layers including input, hidden, output, and output layers.

Training and Evaluation:

- Training process including data normalization and mean squared error (MSE) calculations.
- Using root mean square error (RMSE) in evaluating model forecasts.

Commission:

- Explanation of Python libraries used, including Pandas, Tkinter, NumPy, scikit-learn, and yfinance.
- The process of information processing and inventory information retrieval.

Pre-processing of Data:

- Data preprocessing steps, including MinMaxScaling and sequence construction.

LSTM Model Building and Training:

- Creating a sequence model with LSTM layers.
- Building the model with Adam optimizer and MSE loss function.
- Modeling training from training data.

Testing and Imaging:

- Preprocessing test data consistent with training data.
- The model makes predictions for experimental data.
- Convert forecasts to original scale using Min-Max scale.
- Graph of actual and predicted stock prices.
- Calculation and display of RMSE and MSE between actual and predicted prices.

5.6 Explanation of code using ChatGPT

The script begins by importing various essential libraries such as `numpy`, `matplotlib`, `pandas`, `tkinter`, and more, which are required for data manipulation, machine learning, and visualization.

At the heart of the script is the `run_prediction` function, which serves as the core functionality and is executed when the user interacts with the UI. This function starts by gathering input information, including the company name and entry date from the user.

The data retrieval process involves fetching historical stock price data for the specified company using the `yfinance` library. This data is collected within a defined time frame, starting one year before the user's entry date and ending on January 1, 2023. The collected data is stored in the `data` variable.

Additional test data is retrieved, covering the period from January 1, 2020, up to the current date. This test data is stored in the `test_data` variable, enabling model evaluation.

The feature engineering stage involves scaling the Closeprices to a range between 0 and 1 using the `MinMaxScaler`. The scaled data is stored in the `scaled_data` variable, which is crucial for machine learning.

To prepare the model for training, the script defines a variable, `prediction_days`, set to 60, indicating how many past days' closing prices will be used to predict future prices. Training data is then constructed, with `x_train` and `y_train` lists being populated in a loop. The `x_train` list contains sequences of past closing prices, while the `y_train` list holds the corresponding next day's closing price.

For the machine learning model, a Sequential model is created using the `tensorflow,keras,models` library. The model architecture includes three LSTM (Long Short-Term Memory) layers, essential for capturing temporal dependencies in the data. Dropout layers are also added to prevent overfitting, and the model ends with a Dense layer for predicting future closing prices.

The model is compiled using the ädamöptimizer and the mean_squared_errorloss function. Subsequently, it is trained on the training data with 25 epochs and a batch size of 32 using the `fit` method.

Predictions are made by the model for the test data, and the results are stored in the `predicted_price` variable.

To provide insight into the model's performance, the script employs visualization. It plots both actual and predicted prices using `matplotlib` and calculates the Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) to assess the model's accuracy.

The script also creates a user-friendly interface (UI) using `tkinter`. Users can input the company name and entry date, and upon clicking the "Run Prediction" button, the `run_prediction` function is invoked. The results, including actual and predicted stock prices, MSE, and RMSE, are displayed in the UI using labels and graphs.

In summary, this script offers an end-to-end solution for stock price prediction, encompassing data retrieval, preprocessing, model development, training, and visual representation, all seamlessly presented through a user-friendly interface.

List of Figures

List of Tables

Bibliography

[AAB+15] Abadi, Martín ; Agarwal, Ashish ; Barham, Paul ; Brevdo, Eugene ; Chen, Zhifeng ; Citro, Craig ; Corrado, Greg S. ; Davis, Andy ; Dean, Jeffrey ; Devin, Matthieu ; Ghemawat, Sanjay ; Goodfellow, Ian ; Harp, Andrew ; Irving, Geoffrey ; Isard, Michael ; Jia, Yangqing ; Jozefowicz, Rafal ; Kaiser, Lukasz ; Kudlur, Manjunath ; Levenberg, Josh ; Mané, Dandelion ; Monga, Rajat ; Moore, Sherry ; Murray, Derek ; Olah, Chris ; Schuster, Mike ; Shlens, Jonathon ; Steiner, Benoit ; Sutskever, Ilya ; Talwar, Kunal ; Tucker, Paul ; Vanhoucke, Vincent ; Vasudevan, Vijay ; Viégas, Fernanda ; Vinyals, Oriol ; Warden, Pete ; Wattenberg, Martin ; Wicke, Martin ; Yu, Yuan ; Zheng, Xiaoqiang: *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. https://www.tensorflow.org/. Version: 2015. – Software available from tensorflow.org 44, 45, 46

[Abo23] About Google Scholar: *Stand on the shoulders of giants*. https://scholar.google.com/intl/de/scholar/about.html, 2023. – [Online; accessed 09-September-2023] 31

[Alp] Alpha Vantage: *API Documentation*. https://www.alphavantage.co/documentation/, . – [Online; accessed 31-October-2023] 17

[APF] APFM: *What are the differences between financial and economic analysis?* https://www.floodmanagement.info/what-are-the-differences-between-financial-and-economic-analysis/, . – [Online; accessed 31-October-2023] 17

[DE21] Dami, Sina ; Esterabi, Mohammad: Predicting stock returns of Tehran exchange using LSTM neural network and feature engineering technique. In: *Multimedia Tools and Applications* 80 (2021), 05, S. 1–24. http://dx.doi.org/10.1007/s11042-021-10778-3. – DOI 10.1007/s11042–021–10778–3 42

[ESM23] ESMA, TRV: Artificial intelligence in EU securities markets. (2023) 34

[Fin22] FinBridge Article: *ML im Bankenwesen: Informationen in der Finanzanalyse, dem Portfoliomanagement und dem Risikomanagement.* https://www.finbridge.de/ml-artikel/2022/03/01/ml-im-bankenwesen, 2022. – [Online; accessed 05-September-2023] 13

[FK17] Fischer, Thomas ; Krauss, Christopher: Deep learning with long short-term memory networks for financial market predictions. In: *European Journal of Operational Research* 270 (2017), 12, S. 7 – 12. http://dx.doi.org/10.1016/j.ejor.2017.11.054. – DOI 10.1016/j.ejor.2017.11.054 46, 50

[GPB15] Gonzalez, Rafael T. ; Padilha, Carlos A. ; Barone, Dante Augusto C.: Ensemble system based on genetic algorithm for stock market forecasting. In: *2015 IEEE Congress on Evolutionary Computation (CEC)*, 2015. – ISSN 1941–0026, S. 3102–3108 25

[HE21] Hamoudi, Hamdy ; Elseifi, Mohamed A.: Stock Market Prediction using CNN and LSTM. In: *Computer Science Department, Stanford University* (2021) 23

[Hev07] Hevner, Alan R.: A three cycle view of design science research. In: *Scandinavian journal of information systems* 19 (2007), Nr. 2, S. 4 28

[HMW$^+$20] Harris, Charles R. ; Millman, K. J. ; Walt, Stéfan J. ; Gommers, Ralf ; Virtanen, Pauli ; Cournapeau, David ; Wieser, Eric ; Taylor, Julian ; Berg, Sebastian ; Smith, Nathaniel J. ; Kern, Robert ; Picus, Matti ; Hoyer, Stephan ; Kerkwijk, Marten H. ; Brett, Matthew ; Haldane, Allan ; Río, Jaime F. ; Wiebe, Mark ; Peterson, Pearu ; Gérard-Marchant, Pierre ; Sheppard, Kevin ; Reddy, Tyler ; Weckesser, Warren ; Abbasi, Hameer ; Gohlke, Christoph ; Oliphant, Travis E.: Array programming with NumPy. In: *Nature* 585 (2020), September, Nr. 7825, 357–362. http://dx.doi.org/10.1038/s41586-020-2649-2. – DOI 10.1038/s41586–020–2649–2 49

[HSK18] Henrique, Bruno M. ; Sobreiro, Vinicius A. ; Kimura, Herbert: Stock price prediction using support vector regression on daily and up to the minute prices. In: *The Journal of Finance and Data Science* 4 (2018), Nr. 3, 183-201. http://dx.doi.org/https://doi.org/10.1016/j.jfds.2018.04.003. – DOI https://doi.org/10.1016/j.jfds.2018.04.003. – ISSN 2405–9188 20

[JAEC+23] Jafar, Syed H. ; Akhtar, Shakeb ; El-Chaarani, Hani ; Khan, Parvez A. ; Binsaddig, Ruaa: Forecasting of NIFTY 50 Index Price by Using Backward Elimination with an LSTM Model. In: *Journal of Risk and Financial Management* 16 (2023), Nr. 10, S. 423 36

[KGC+23] Kumar, Arun ; Gaur, Nishant ; Chakravarty, Sumit ; Alsharif, Mohammed H. ; Uthansakul, Peerapong ; Uthansakul, Monthippa: Analysis of spectrum sensing using deep learning algorithms: CNNs and RNNs. In: *Ain Shams Engineering Journal* (2023), 102505. http://dx.doi.org/https://doi.org/10.1016/j.asej.2023.102505. – DOI https://doi.org/10.1016/j.asej.2023.102505. – ISSN 2090–4479 43

[KK19] Kim, Sangyeon ; Kang, Myungjoo: Financial series prediction using Attention LSTM. (2019), 02 43

[KT06] Kumar, Manish ; Thenmozhi, M: Forecasting stock index movement: A comparison of support vector machines and random forest. In: *Indian institute of capital markets 9th capital markets conference paper*, 2006 22

[Lun99] Lundh, Fredrik: An introduction to tkinter. In: *URL: www. pythonware. com/library/tkinter/introduction/index. htm* (1999) 48

[MLMLC22] Montesinos López, Osval A. ; Montesinos López, Abelardo ; Crossa, Jose: Overfitting, model tuning, and evaluation of prediction performance. In: *Multivariate statistical machine learning methods for genomic prediction.* Springer, 2022, S. 109–139 15, 16, 63

[MRCV23] Murray, Kate ; Rossi, Andrea ; Carraro, Diego ; Visentin, Andrea: On Forecasting Cryptocurrency Prices: A Comparison of Machine Learning, Deep Learning, and Ensembles. In: *Forecasting* 5 (2023), Nr. 1, S. 196–209 35, 36, 37, 65

[MSE$^+$23] Maple, Carsten ; Szpruch, Lukasz ; Epiphaniou, Gregory ; Staykova, Kalina ; Singh, Simran ; Penwarden, William ; Wen, Yisi ; Wang, Zijian ; Hariharan, Jagdish ; Avramovic, Pavle: The AI Revolution: Opportunities and Challenges for the Finance Sector. In: *arXiv preprint arXiv:2308.16538* (2023) 13, 32

[MSG14] Mondal, Prapanna ; Shit, Labani ; Goswami, Saptarsi: Study of Effectiveness of Time Series Modeling (Arima) in Forecasting Stock Prices. In: *International Journal of Computer Science, Engineering and Applications* 4 (2014), 04, S. 13–29. `http://dx.doi.org/10.5121/ijcsea.2014.4202`. – DOI 10.5121/ijcsea.2014.4202 21

[PRTV12] Peffers, Ken ; Rothenberger, Marcus ; Tuunanen, Tuure ; Vaezi, Reza: Design science research evaluation. In: *Design Science Research in Information Systems. Advances in Theory and Practice: 7th International Conference, DESRIST 2012, Las Vegas, NV, USA, May 14-15, 2012. Proceedings 7* Springer, 2012, S. 398–410 27, 28, 63

[PTGK20] Patel, Mohil M. ; Tanwar, Sudeep ; Gupta, Rajesh ; Kumar, Neeraj: A deep learning-based cryptocurrency price prediction scheme for financial institutions. In: *Journal of information security and applications* 55 (2020), S. 102583 36

[PVG+11] Pedregosa, F. ; Varoquaux, G. ; Gramfort, A. ; Michel, V. ; Thirion, B. ; Grisel, O. ; Blondel, M. ; Prettenhofer, P. ; Weiss, R. ; Dubourg, V. ; Vanderplas, J. ; Passos, A. ; Cournapeau, D. ; Brucher, M. ; Perrot, M. ; Duchesnay, E.: Scikit-learn: Machine Learning in Python. In: *Journal of Machine Learning Research* 12 (2011), S. 2825–2830 46, 49, 50

[Ran23] Ran Aroussi: *Download market data from Yahoo! Finance API*. https://pypi.org/project/yfinance/, 2023. – [Online; accessed 15-September-2023] 18, 41, 49

[RPE19] Ramezanian, Reza ; Peymanfar, Arsalan ; Ebrahimi, Seyed B.: An integrated framework of genetic network programming and multi-layer perceptron neural network for prediction of daily stock return: An application in Tehran stock exchange market. In: *Applied Soft Computing* 82 (2019), 105551. http://dx.doi.org/https://doi.org/10.1016/j.asoc.2019.105551. – DOI https://doi.org/10.1016/j.asoc.2019.105551. – ISSN 1568–4946 24

[SS20] Shen, Jingyi ; Shafiq, M O.: Short-term stock market price trend prediction using a comprehensive deep learning system. In: *Journal of big Data* 7 (2020), Nr. 1, S. 1–33 22

[SVB12] Sonnenberg, Christian ; Vom Brocke, Jan: Evaluation patterns for design science research artefacts. In: *Practical Aspects of Design Science: European Design Science Symposium, EDSS 2011, Leixlip, Ireland, October 14, 2011, Revised Selected Papers 2* Springer, 2012, S. 71–83 29, 63

[SZa] Shi, Xiang ; Zhang, Peng: Quantitative Data Analysis in Finance. 17

[SZb] Shi, Xiang ; Zhang, Peng: Quantitative Data Analysis in Finance. 17

[SZK17] Shi, Xiang ; Zhang, Peng ; Khan, Samee U.: Quantitative data analysis in finance. In: *Handbook of big data technologies* (2017), S. 719–753 16

[tea20] team, The pandas d.: *pandas-dev/pandas: Pandas*. http://dx.doi.org/10.5281/zenodo.3509134. Version: Februar 2020 47

[Thé23] Théate, Thibaut: Artificial Intelligence Techniques for Decision-Making in Market Environments. (2023) 34

[VFG23] Vicente, Óscar F. ; Fernández, Fernando ; García, Javier: Automated market maker inventory management with deep reinforcement learning. In: *Applied Intelligence* (2023), S. 1–18 37, 38, 63

[YHM+23] Yeo, Wei J. ; Heever, Wihan van d. ; Mao, Rui ; Cambria, Erik ; Satapathy, Ranjan ; Mengaldo, Gianmarco: A comprehensive review on financial explainable AI. In: *arXiv preprint arXiv:2309.11960* (2023) 13, 32, 34, 65

[ZYM20] Zhang, Yong'an ; Yan, Binbin ; Memon, Aasma: A novel Deep Learning Framework: Prediction and Analysis of Financial Time Series using CEEMD and LSTM. In: *Expert Systems with Applications* 159 (2020), 05, S. 113609. http://dx.doi.org/10.1016/j.eswa.2020.113609. – DOI 10.1016/j.eswa.2020.113609 43

Chapter 6

Attachment

6.1 Use of AI in this paper - Leonard Katz & Leo Strauch

For the creation of this paper, ChatGPT 3.5 was utilized. It was primarily employed to enhance writing through prompts like: “Make this text sound more academic” or “Please lengthen this paragraph slightly.” We also utilized the myfix prompt, made available by Heiko Neuhaus University of Koblenz. For the exact prompt, please refer to the next subsection. Before MyFix the export method was utilized to ensure quality standards.

ChatGPT was also used for reading comprehension in the literature review section, albeit primarily for an initial check of the paper. All selected papers were subsequently read conventionally.

Finally, we used ChatGPT to assist in structuring the paper and to ensure that parts could be written without significant delays, thus optimizing the workflow among group members.

In addition to ChatGPT, DeepL and Grammarly were used for translation and spell checking, as none of the participants were native English speakers.

MyFix Define task `‘‘myfix’’` as following steps 1 and 1.1 to 1.17 and 3 on a provided input text. If you understand this task, simply reply with “OK”, nothing else.

1. These steps describe how text is rewritten.

   1.1 Rewrite text to sound much more academic than before.

   1.2 Remove all passive voice, rewrite the text to contain active voice only. This rule is more important than every other rule. If a change leads to the introduction of passive voice, do not make it.

   1.3 Fix grammar and stylistic mistakes.

   1.4 If it does not alter the meaning, prefer present tense.

   1.5 Avoid word repetitions by using synonyms, omission of the repetition or other ways.

   1.6 Never remove latex tags for a given word in the output.

   1.7 Force use of the noun form, remove all instances of the saxon genitive.

   1.8 Be precise, not vague and subjective.

   1.9 If the text contains latex cite tags then include them in the output.

   1.10 Keep the quotation marks.

   1.11 Fix punctuation errors.

   1.12 If you find factual, contextual or other problems, fix them.

   1.13 Write without possessives.

   1.14 Avoid addressing readers or authors with "I", "We", "One" or "You".

   1.15 Never change words in brackets.

   1.16 Pretty print latex code.

2. The result of step 1 is put in a paragraph named `''Corrected Text''` in which you put the fixed text. Place a line break after the title of the paragraph. Try not to remove special characters, such as backslashes, percentage or dollar signs.

3. Create table `''Changes''` with 3 columns, in this order: "New", "Old" and "Reason". The first column ("New") shows the corrected text of step 1, the second ("Old") column the original text and in column "Reason" you briefly argue why the change was made. This table should not contain latex source code.

Expert method Minik a discussion between three experts, that discuss the following text. Each expert can't agree with the other two and they have to discuss the wording of each sentence. When all three have stated their opinion, they determine the best wording and move on to the next sentence.

6.2 Code - Leo Strauch

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import tkinter as tk
import math
from dateutil.parser import parse
from sklearn.metrics import mean_squared_error

import datetime as dt

from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, LSTM

from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import yfinance as yf

def run_prediction():
    company = company_entry.get()
    entry_date_str = entry_date_entry.get()
    entry_date = parse(entry_date_str).date()
    start = dt.datetime(entry_date.year -1, entry_date.month, entry_date
                                        .day)
    end = dt.datetime(2023, 1, 1)

    # define ticker symbol
    data = yf.download(company, start, end)
```

```
# Load Test Data
test_start = dt.datetime(2020, 1, 1)
test_end = dt.datetime.now()
test_data = yf.download(company, test_start, test_end)

data.tail(10)
# Feature Engineering part
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(data['Close'].values.reshape(-1,
                                     1))

# How many days we want to look at the past to predict
prediction_days = 60

# defining two empty lists for preparing the training data
x_train = []
y_train = []

# we are counting from the 60th index to the last index
for x in range(prediction_days, len(scaled_data)):
    x_train.append(scaled_data[x - prediction_days:x, 0])
    y_train.append(scaled_data[x, 0])

x_train, y_train = np.array(x_train), np.array(y_train)
x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1
                                     ))

# Sequential provides training and inference features on this model.
model = Sequential()
# Add an LSTM layer with:
#        - unit: Positive integer, the dimensionality of the output
                                     space.
#        - return_sequences: Boolean. Whether to return the last
                                     output in the output sequence,
                                     or the full sequence.
```

```
model.add(LSTM(units=50, return_sequences=True, input_shape=(x_train
                                    .shape[1], 1)))
# Add a dropout layer: To prevent 'overfitting' by setting 0.2 of
                                    values to 0
model.add(Dropout(0.2))
model.add(LSTM(units=50, return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(units=50))
model.add(Dropout(0.2))
# prediction of the next closing value
model.add(Dense(units=1))

model.compile(optimizer='adam', loss='mean_squared_error')
# fit the model in the training data
model.fit(x_train, y_train, epochs=25, batch_size=32)

actual_prices = test_data['Close'].values
total_dataset = pd.concat((data['Close'], test_data['Close']), axis=
                                    0)

model_input = total_dataset[len(total_dataset) - len(test_data) -
                                    prediction_days:].values
# reshaping the model
model_input = model_input.reshape(-1, 1)
# scaling down the model
model_input = scaler.transform(model_input)

x_test = []
for x in range(prediction_days, len(model_input)):
    x_test.append(model_input[x - prediction_days:x, 0])

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

# Use the model to predict the prices
predicted_price = model.predict(x_test)
predicted_price = scaler.inverse_transform(predicted_price)

# plot the test Predictions into the IDE
plt.plot(actual_prices, color="black", label=f"Actual {company}
                                    price")
plt.plot(predicted_price, color='green', label=f"Predicted {company}
                                     Price")
plt.title(f"{company} Share price")
plt.xlabel('Time')
plt.ylabel(f'{company} share price')
plt.legend()
plt.show()

# Create the real data to compare to
real_data = total_dataset.tail(len(test_data) + 1)
real_data = np.array(real_data).reshape(-1, 1)
real_data = np.reshape(real_data, (real_data.shape[0], real_data.
                                    shape[1], 1))

# Create prediction for the next day
prediction = model.predict(real_data)
prediction = scaler.inverse_transform(prediction)
print(f"Prediction: {prediction}")

# Calculate MSE and RMSE
mse = mean_squared_error(actual_prices, predicted_price)
rmse = math.sqrt(mse)

print(f"Mean Squared Error (MSE): {mse}")
print(f"Root Mean Squared Error (RMSE): {rmse}")

# Plot the test predictions for the ui -  The plt part is for
                                    debugging in pycharm
figure, ax = plt.subplots(figsize=(6, 4))
ax.plot(actual_prices, color="black", label=f"Actual {company} price
                                    ")
```

```
    ax.plot(predicted_price, color='green', label=f"Predicted {company}
                                    Price")
    ax.set_title(f"{company} Share price")
    ax.set_xlabel('Time')
    ax.set_ylabel(f'{company} share price')
    ax.legend()

    # Create a FigureCanvasTkAgg object to display the graph in the UI
    canvas = FigureCanvasTkAgg(figure, master=app)
    canvas_widget = canvas.get_tk_widget()
    canvas_widget.pack()

    highest_value = np.max(prediction)
    lowest_value = np.min(prediction)

    # Display the prediction in the UI
    prediction_label.config(text=f"Our model predicts, that the highest
                                    value tomorrow is: {
                                    highest_value} and the "
                             f"lowest: {lowest_value}")

    # Display the MSE in the UI
    mse_label = tk.Label(app, text=f"Mean Squared Error: {mse}")
    mse_label.pack()
    rmse_label = tk.Label(app, text=f"Mean Squared Error: {rmse}")
    rmse_label.pack()

# Create the main application window
app = tk.Tk()
app.title("Stock Price Prediction")

# Create labels and entry fields for the company name and entry date
company_label = tk.Label(app, text="Company Name")
company_label.pack()
company_entry = tk.Entry(app)
company_entry.pack()
```

```
entry_date_label = tk.Label(app, text="Entry Date (YYYY-MM-DD)")
entry_date_label.pack()

entry_date_entry = tk.Entry(app)
entry_date_entry.pack()

# Create a button to run the prediction
run_button = tk.Button(app, text="Run Prediction", command=
                                    run_prediction)
run_button.pack()

# Label to display the prediction
prediction_label = tk.Label(app, text="")
prediction_label.pack()

app.mainloop()
```