Portfolio Optimization Using a Hybrid Machine Learning Stock Selection Model

by

Joshua S. Masuda

B.S. in Computer Science, Economics, and Data Science and in Business Analytics,
Massachusetts Institute of Technology, 2024

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING IN COMPUTER SCIENCE, ECONOMICS, AND DATA SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2024

Authored by:	Joshua S. Masuda Department of Electrical Engineering and Computer Science August 15, 2024
Certified by:	Nathaniel D. Hendren Professor of Economics, Thesis Supervisor
Accepted by:	Katrina LaCurts Chair, Master of Engineering Thesis Committee

Portfolio Optimization Using a Hybrid Machine Learning Stock Selection Model

by

Joshua S. Masuda

Submitted to the Department of Electrical Engineering and Computer Science
on August 15, 2024 in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING IN COMPUTER SCIENCE, ECONOMICS, AND DATA SCIENCE

ABSTRACT

Portfolio Optimization can be challenging due to the uncertainty about the value of a future asset. With recent developments in machine learning, there are significant prediction tools available that can be applied to portfolio selection. Financial markets are known to be dynamic and complex, but algorithms are designed to capture patterns in the data. In this paper, seven machine learning techniques are used for stock price prediction: Linear Regression, Support Vector Machine, Random Forest, Recurrent Neural Network, Long Short-Term Memory, Bidirectional Long Short-Term Memory, and LightGBM. Additionally, two hybrid machine learning methods are used for prediction: CNN-LSTM and BiLSTM-BO-LightGBM. After training the models, the algorithm creates an optimal portfolio of assets over a simulated year of trading. The symmetric mean absolute percentage error of the algorithms on unseen data evaluates the prediction power. The generated alpha and Sharpe ratio evaluate the quality of the constructed optimal portfolios under Mean-Variance Optimization. Using data on the 50 largest United States companies from January 2, 2019 to December 29, 2023, the results demonstrate that the hybrid models perform better than the individual models, and the CNN-LSTM model outperforms benchmark market indices.

Thesis supervisor: Nathaniel D. Hendren
Title: Professor of Economics

Acknowledgments

First, I would like to thank my academic advisor and thesis supervisor, Professor Nathaniel Hendren. Professor Hendren's support and guidance through my time as both an undergraduate and a Master's student at MIT has been incredibly valuable. He has never hesitated to answer any questions and always provided incredibly helpful feedback on my ideas. I would not be able to complete my degrees without him.

I also want to thank two professors who inspired me to pursue a thesis in Finance and Machine Learning through their classes. Professor Bruce Tidor in Design and Analysis of Algorithms showed me the power of applying efficient algorithms to real world problems. Professor Swati Gupta in The Analytics Edge demonstrated the most fascinating case studies of algorithmic models applied to a business context.

I have met some amazing students at MIT. My classmates are the smartest and most ambitious people I have ever met, and being surrounded by such amazing people was truly an extraordinary experience. To my tennis teammates and coaches, thank you for allowing me to play the sport I love everyday. To my friends, thank you for always being there for me. Thank you to everyone in the MIT community who made this the best four years of my life.

Most importantly, I would like to thank my family. My grandparents have always supported my learning and been my biggest advocates. My parents and my two sisters have always encouraged me to pursue my passions and stayed by my side every step of the way. I would not be where I am today without their love and support, and I am eternally grateful for all that they do.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Financial markets have a significant impact on many areas of society, such as business, technology, and employment. The world's stock market holds a vast amount of wealth; it is estimated that the global stock market has a total equity of 109 trillion USD [1]. Investors have been collecting knowledge and data about companies to increase their investment returns since the creation of financial markets. However, due to the complex and volatile nature of stock prices, it is difficult to predict the direction and magnitude of the movement. In 1970, Gene Fama defined a market to be "informally efficient" if prices are constantly updated with all available information about future prices. The main conclusion of Fama's efficient markets hypothesis is that stock movements are unpredictable [2]. However, in recent times, researchers are using computational power to find predictive patterns in financial data. Establishing a system that improves predictive performance can lead to out-sized profits.

The study of stock price prediction models is an essential research topic for investors aiming to use financial markets to gain a profit. Predicting prices is a difficult problem, due to the non-linear behavior and high number of variables. Statistical and computational methods for prediction have been applied to financial markets with machine learning techniques being the most popular approach due to its ability to discover new patterns in historical data. These

techniques aim to identify the underlying function that is forming the data and realize the relationships that exist between the data and the stock price.

Stock selection is only the first part of generating profits. With the selected stocks, a portfolio of assets must be generated. Portfolio optimization is the distribution of wealth among multiple assets where returns and risks are important factors. The goal of an optimized portfolio is maximization of returns and minimization of risks. Higher expected returns usually bring increased risk, so a model can be used to optimize the portfolio. With this process, investors can have a systematic approach to asset allocation, which is essential for achieving high performance in complex financial markets. The primary benefit of portfolio optimization is risk management by identifying and mitigating risks associated with certain securities [3]. This often means that a portfolio will have diversification, which is guided by the principles of the Modern Portfolio Theory, described later in this section. Portfolio optimization helps investors spread their risk, which leads to greater stability and predictability of investment outcomes. Due to the dynamic nature of the financial landscape, portfolio optimization handles market fluctuations with a balanced approach to risk and returns.

1.1 Problem Description

1.1.1 Best Approach: Machine Learning

As mentioned, accurately predicting stock prices is extremely challenging, and investors grapple with the complexities of market behavior. There are many factors that can have an effect on the prices, including economic indicators, investor sentiment, and company-specific news. The search for reliable forecasting tools has led to the emergence of machine learning as a promising solution. Traditional econometric models heavily rely on assumptions and equations, whereas machine learning can analyze vast amounts of data and uncover intricate patterns and relationships that are not apparent to human analysts [4]. These intricate patterns are especially prevalent in financial markets, which often have nonlinear and chaotic

behavior.

The prediction of stock prices involves leveraging historical market data, such as price movements, trading volumes, and other relevant metrics to forecast future price trends. Machine learning algorithms excel in learning from past data and adapting the predictions as new information becomes available. The adaptability of a financial model is critical because market conditions are constantly changing, providing both risks and opportunities for investors. Financial institutions have access to an incredible amount of data, and machine learning can efficiently use the data to make decisions. Machine learning algorithms can analyze thousands of sources simultaneously, which is not possible for a group of human traders to achieve. The slim advantage that traders have over the market average can translate into significant profits with a high volume of trading operations [5].

1.1.2 Drawbacks and Mitigations

Bias

Machine Learning has many benefits in finance, but there are also some drawbacks that need to be considered. The first major drawback is potential bias that can arise from algorithms. Because machine learning decisions are based on historical data, if the historical data contains bias, the predictions will also be biased. Biased or incomplete data as inputs can lead to a biased model; there are multiple types of biases in machine learning models applied to financial markets [6]:

- Optimization Bias: adjusting or introduction of additional training parameters to inflate the performance on the backtest data set. Therefore, once the strategy is implemented, the performance can be different than what was measured in the backtest. A method to prevent optimization bias is to perform a sensitivity analysis. By varying the parameters incrementally, a surface of performance can be plotted. Any parameter that does not lead to a smooth parameter surface is not reflecting a meaningful

relationship in the data and can be disregarded in the final strategy.

- Survivorship Bias: using disparate sample size, which leads to the model being specific to a group of stocks. One example of this is only using data for securities that have complete data for the duration of the time series and discarding the remaining data. However, eliminating the data for a stock that stopped trading or left the market would lead to major biases within the model and flawed final results. Ensuring that the dataset is cleaned correctly and using more recent data can help mitigate survivorship bias.

- Human Bias: systematic error caused by human tendency to simplify information processing. One example of human bias in algorithmic trading is that there are a significant number of trades completed within 30 seconds of a "round" time, like 9:45 am or 10:00 am. Humans have a tendency to default to round numbers when in an uncertain environment. This can be minimized by using pair programming, where two programmers take turns on the same piece of code, on trading algorithms.

- Look-ahead bias: using information that would not have been available during the period that is being analyzed. A common example is parameter calculation. If the entire data set is used to calculate coefficients of the model and then retroactively applied to a trading strategy, the future data is incorporated. An investor should only use information available at the time of the trade to avoid look-ahead bias in their prediction model.

Automation

Machine learning creates results with minimal human input, which can lead to some drawbacks. Because accurate algorithms are often complex, the reasoning for the prediction cannot be easily interpreted by a human being. This means that if there is an error or biases in the algorithms, it can be difficult to detect. With better interpretability of a machine

learning model, there is easier understanding due to the transparency of the model's decision making process. Oftentimes, machine learning models must balance the trade-off between the accuracy and the interpretability of the model, depending on the context of the problem.

Machine learning can make quick, automated financial decisions, but this means that many decisions are made without human oversight. It is possible that unusual situations occur outside of the model's input which will impact the prediction accuracy. Since this situation is not accounted for when training the model, this can lead to errors. Algorithms can lack the contextual understanding that humans naturally use when making decisions. Additionally, over 90% of machine learning models degrade over time [7], meaning that humans need to always be checking the quality of their models and updating the decision making criteria.

1.1.3 Significance

The selection of securities is the sole determinant of profit for a financial firm. Due to the size of financial markets, the number of financial firms, and retail traders, there is a high volume of trades happening during market hours. In fact, the New York Stock Exchange trades 18.9 billion USD per day on average, while the NASDAQ trades 320 billion USD per day on average [8, 9]. Due to this volume, making quick financial decisions are imperative to creating profitable trades. In a study that used data on the historical constituents of the S&P 500 from 1999 to 2021, it was shown that a machine learning based stock selection model had a substantial and significant risk-adjusted out-performance compared to an equally weighted benchmark [10]. The application of machine learning to stock selection and portfolio optimization can lead to greater returns and better risk management.

1.2 Important Background Information

1.2.1 Modern Portfolio Theory

Modern Portfolio Theory is a method for selecting a portfolio of assets that lead to expected overall returns within an acceptable level of risk. A risky investment that has the same expected return as a less risky investment is typically seen as less desirable, since people are inherently risk-averse. Risk aversion implies that people should invest in multiple assets to ensure that the performance of one asset does not significantly impact the overall return of the portfolio[11]. A portfolio's risk is usually calculated as a function of the variances of each asset and the correlation between each pair of assets. The covariance between assets gives a more accurate metric for the portfolio risk. There are two fundamental assumptions when utilizing Modern Portfolio Theory:

1. You cannot view assets in your portfolio in isolation. Instead, you must view how the assets in the portfolio relate with each other in terms of expected returns and risk level.

2. It's difficult to forecast future investment returns. Investors can make approximations using historical data and mathematical models, but these are far from being completely accurate.

Because the Modern Portfolio Theory looks for reduced volatility, this method can select negatively correlated assets. While negatively correlated assets will likely have opposite directions of expected returns, the trade-off can be worthwhile for hedging risk [11]. The efficient frontier is a graph that represents the most risk-optimized portfolio allocation for expected returns, as seen in Figure 1.1.

Finance researchers plot an efficient frontier by examining expected returns and the range of asset allocations. In Figure 1.1, the X-axis represents risk/volatility, and the Y-axis represents expected return. It has been seen in research that based on all possible hypothetical portfolios given a range of asset allocations, the resulting shape for the efficient

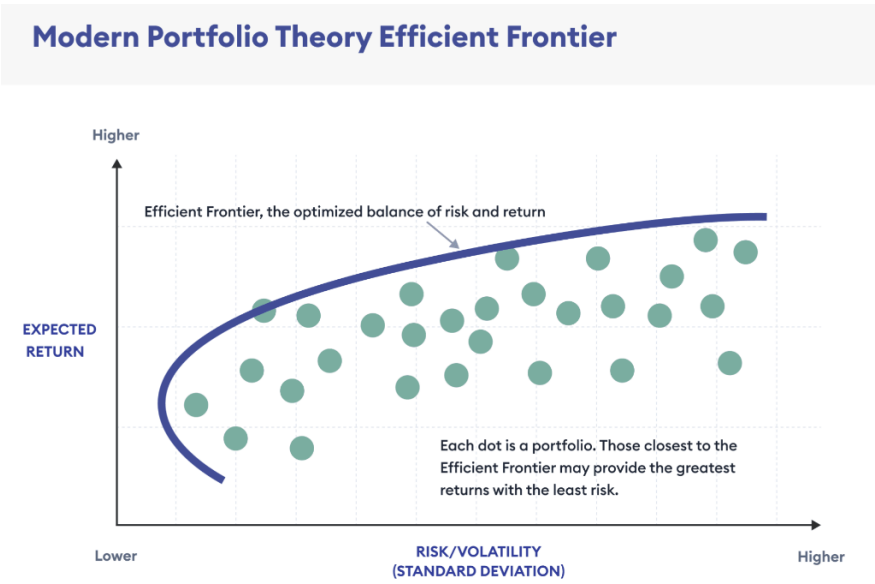

Figure 1.1
Modern portfolio theory efficient frontier graph. Source: Baldridge, 2023 [12]

frontier should be a parabola. Any portfolio that is lying on or above the line represents an optimal portfolio, which means that the portfolio offers the highest return for a specific level of risk. The portfolios under the efficient frontier are not optimized.

1.2.2 Predicting Alpha

To measure the performance of a stock or a fund, it is important to separate the return that is due to overall market volatility and the “excess" return that is a result of your investment strategy [13]. The basic equation for the performance of a stock is expressed as

$$y = \alpha + \beta x + \mu \tag{1.1}$$

where

- y is the performance of the stock
- α is the excess return of the stock
- β is volatility relative to the benchmark (often S&P 500)

- x is the performance of the benchmark
- μ is the residual; unexplained random factor in performance

For a portfolio, getting a positive α is the goal for the investor. The alpha value is the strategy's ability to beat the market, or the edge that the strategy has over the market. Alpha is the primary measure used to evaluate investing strategies because it measures the active return on the investment compared to a market index or benchmark that is considered to represent the entire market movement. In other words, alpha is the additional value that a strategy provides to an investor. This means that an alpha of 0 indicates that the portfolio is tracking perfectly with the benchmark index, and a negative alpha indicates that the portfolio is under-performing the benchmark when adjusting for risk. The α value is represented as a single number that is the percentage of returns compared to the benchmark. For example, $\alpha = 2.4$ means that the stock did 2.4% better than the index, and $\alpha = -1.6$ means that the stock did 1.6% worse than the index.

Beta is an indication of the volatility of a stock in comparison with the market as a whole. A benchmark index, such as the S&P 500, is used as the proxy measurement for the market. The volatility of the stock price will give the investor more information about the risk associated with an asset, and knowing the volatility will allow the investor to determine if an expected return is worth the risk. The baseline number for beta is one, and a value of one means that the price of the security moves exactly as the benchmark market moves [14]. A beta value of greater than one means that the asset is more volatile than the market, and a beta value of less than one means that the asset is less volatile than the market. For example, if $\beta = 1.32$, then the asset is 32% more volatile than the market, and if $\beta = 0.77$, then the asset is 23% less volatile than the market.

1.2.3 Risk Management: Mean-Variance Optimization

Markowitz proposed the mean-variance (MV) model, which as the name suggests, makes use of the expected return and variance of a portfolio. Using historical asset prices, the MV

model calculates the volatility of a generated portfolio. This model assumes that the investor would like to maximize the expected return for a given level of risk [15]. A formulation for MV model can be

$$\begin{aligned} \text{Maximize} \quad & \sum_{i=1}^{N} w_i E_i = \gamma \\ \text{subject to} \quad & \sum_{i=0}^{N} \sum_{j=0}^{N} w_i w_j C_{ij} = \sigma^2 \\ & \sum_{i=0}^{N} w_i = 1 \\ & w_i \geq 0 \ \forall i \end{aligned} \tag{1.2}$$

where N is the total number of assets, w_i is the weight of each asset i in the portfolio to be optimized, σ^2 is the variance of the portfolio (generally referred to as portfolio risk), C_{ij} is the covariance between asset i and asset j, γ is the expected return, and E_i is the average return on asset i.

Using the machine learning techniques to get accurate prediction models, each stock will have an expected return and a variance. However, since there is a maximum level of risk (variance) that an investor would be willing to take on, the next step would be to maximize expected returns given a risk tolerance parameter.

1.3 Research Objectives

This paper has the following high-level research objectives:

1. Preprocess the data in preparation for input to machine learning models
2. Train and test individual and hybrid stock price prediction models
3. Implement the models to construct optimal portfolios
4. Evaluate the portfolios and determine the best approach

Chapter 2

Existing Literature

The application of machine learning to financial markets has been a popular research topic for the past couple of decades. The first part of my thesis is to use machine learning algorithms for stock price predictions. Researchers have applied different algorithms to a variety of datasets to get accurate predictions. The second part of my thesis is to use the trained machine learning models to get an optimized portfolio using Mean-Variance Optimization. Finding the best way to optimize a portfolio has been an interest of researchers since Markowitz's Mean-Variance Optimization in 1952. In this section, I will review existing literature about both stock prediction and portfolio optimization.

2.1 Stock Price Prediction Approaches

2.1.1 Traditional Machine Learning

Many researchers have approached the prediction of stocks by using an ordered sequence of stock prices at equal intervals, called time-series data. Time-series data is a sequence of data points indexed in time order. There is a clear connection with stock prices and time-series data, since it can be inferred that the price of a stock yesterday will be a factor in predicting movement of the stock today, and the price of the stock today will be a factor in predicting the

movement of the stock tomorrow. Idress et. al uses an Auto Regressive Integrated Moving Average (ARIMA) model for prediction of the Indian stock market relative to the Sensex and Nifty [16]. The ARIMA model showed a deviation of roughly 5% mean percentage error. The researchers conclude that the ARIMA model can be very constructive in various real world situations and environments.

Using supervised machine learning, Pahwa and Argarwal use the simplest machine learning algorithm, Linear Regression, to predict stock prices [17]. The dataset is spanning 14 years of GOOGL stock, and they have outlined the process to forecast the closing value of the asset. Misra et. al also uses Linear Regression during their research on categorization of algorithms for predictive power, and they conclude that the Linear Regression model increases accuracy rate when applying Principal Component Analysis on the data during feature selection [18]. While Support Vector Machines lead to higher accuracy on non-linear classification data, Linear Regression is better for linear data because of a higher confidence value. For a binary classification model, Random Forests have high accuracy, and multi-layer perceptron has the lowest error in prediction. It is concluded that choosing the algorithm for prediction largely depends on the type of volume of the analyzed data. Ravikumar and Saraf examine two different methods for prediction: Regression and Classification [19]. In regression, the model will predict the closing price of a company's stock, and in classification, the model will predict if the previous closing price will be greater or less than the closing price of the next trading day. Using S&P 500 data from Yahoo Finance, it is found that the Logistic Regression Model gave the highest mean accuracy.

Several machine learning algorithms, including Support Vector Machine, Random Forest, K-Nearest Neighbor, Naive Bayes, and Softmax, are used to predict market behavior [20]. The data was collected for five stocks over a period of five to ten years from different data sources. After a comparative study of these approaches, the accuracy of each model is compared. It is found that Random Forests is the best performing for large datasets, and Naive Bayesian Classifier is the best for small datasets. Additionally, the reduction in the

number of technical indicators reduces the accuracy of the algorithms. Other researchers focus on forecasting prices of the stocks using certain types of features [21]. Ingle and Deshmukh use the Hidden Markov Model with TF-IDF features extracted to predict the next day's stock market value for a group of companies. The Hidden Markov Model is used to generate the probability of a sequence and the probabilities of the stock switching values. With this method, the actual closing price and predicted closing prices for a particular company show an error rate between 0.2 to 3.9% for weekly collected stock data. It is also found that an increase in data collection for stock news leads to a model with higher accuracy.

In the past, historical financial data was mainly used to forecast security pricing. However, the price is not completely dependent on historical data due to various factors and the dynamic nature of the market. In one paper, the researchers look beyond historical data by analyzing people's sentiments and other news events in the stock market [22]. Different methods are used to predict stock prices with this data, but a high rate of accuracy is not achieved. In the analysis, it is found that fusion algorithms lead to more accurate results than using individual algorithms.

Studies using traditional machine learning techniques have led to promising predictive power. However, some researchers are interested in using deep learning and neural networks to predict market movement, which will be discussed in the following subsection.

2.1.2 Deep Learning and Neural Networks

A Hierarchical Attention Network for stock prediction by using generality of graph neural networks and graph conception is proposed for prediction of stock prices [23]. With this hierarchy, the available data can be clustered on distinctive members of the family and added to the model. In other words, a Hierarchical Attention Network selectively aggregates information on different relation types to learn useful node representations. The paper concludes that this model, which automatically selects which information to use, outperforms existing models.

Li et. al study the effect that the indices have in stock prices [24]. This paper uses Long Short-Term Memory(LSTM), which is a type of recurrent neural network that learns order dependence in sequence prediction problems. LSTM aims to help the model better understand the dynamics of the S&P 500 Index, which a traditional linear regression cannot do. However, because the error of the predictive value becomes large at a certain data size, it is concluded that LSTM cannot be used to develop a profitable trading strategy. Another paper that utilizes LSTM proposes a system that would recommend stocks for buyers to purchase [25]. Their approach is using a combination of the prediction from historical data and real-time data. To get the predicted value, the first layer of the RNN model has the latest trading data and technical indicators as inputs. The second layer is composed of LSTM, a compact layer, and the output layer with the predicted value. This predicted value is combined with the summarized data collected from news analytics, and the final result reports the percent change. It is concluded that this approach will avoid risks that are caused by unforeseen news events, making the model more robust.

One paper observes that only a few input features from a large dataset has a significant effect on the prediction model, so Song and Lee focus on these input features and determine the features most important for prediction [26]. Their approach includes three different Artificial Neural Networks which differ in the input features; multiple, binary, and technical features are used to find the most accurate neural network. The Artificial Neural Network using binary features is the most accurate model, due to the "lightweight" features being most suitable for price prediction. One downside of using binary features, however, is that the conversion of features into binary excludes some important information needed for prediction.

Another study looks at specific prediction techniques for stock closing price prediction on the Stock Exchange of Thailand [27]. Werewithayaset and Tritilanunt use Multi-Layer Perceptron, Sequential Minimal Optimization, and Partial Least Square Classifier to predict the closing price of Thailand stocks. Their idea is that prediction of stock prices can be done using nearby real prices because movement of stock prices is observed to move in a cycle.

Using 12 months of data, they conclude that Partial Least Square is the best algorithm to predict the stock closing price, followed by Sequential Minimal Optimization.

2.2 Portfolio Optimization

Portfolio optimization is an approach to selecting an optimal asset distribution from a set of possible portfolios according to an objective. One limitation of portfolio optimization is that the results can differ in theory and practice due to estimation issues when applied to real data. Ban et. al adapt two machine learning methods, regularization and cross validation, for portfolio optimization in an effort to combat this limitation [28]. The first method they use is performance-based regularization, which constrains the sample variances of estimated portfolio risk and return and drives lower estimation error in performance. Using the performance-based regularization for the mean-variance and the mean-conditional value-at-risk problems lead to better benchmarks in two of the three data sets in the study. The sensitivity of the model is also tested, and the model proves to be robust with nontrivial robust constraints for the portfolio risk. Therefore, it can be concluded that performance-based regularization is a promising model for handling uncertainty in prediction.

One important aspect of financial theory is the role of correlations of returns on various assets. In particular, the structure and dynamics of correlations are a main interest, since Markowitz's mean-variance theory suffers from the "curse of dimensions," meaning that it is challenging to analyze and organize data in high-dimensional spaces. To tackle the high dimensionality of portfolio optimization, there have been many techniques aimed to reduce the effect dimension of large portfolios. One study analyzes the different dimensionality reduction techniques by using a simulation based approach [29]. After running the simulations, it is found that developing models with finer elements of the structure of financial correlations leads to increased relevance of the results. It is also found that the simulation based analysis can be extended to several other estimation procedures and more complex optimization.

A unique approach toward portfolio optimization is called Particle Swarm Optimization. Particle Swarm Optimization is a powerful, meta-heuristic optimization algorithm that is inspired by swarm behavior observed in nature, such as bees or bats. It is a simulation of a simplified social system, where the swarm is looking for the global minima of a fitness function. Cura applies Particle Swarm Optimization to the portfolio optimization problem and compares it with genetic algorithms, simulated annealing, and tabu search approaches [30]. Using data from Hong Kong, Germany, UK, USA, and Japan, the study finds that none of the four heuristics clearly outperformed the other in all kinds of investment policies. However, for instances where low risk of investment is desired, the Particle Swarm Optimization model leads to better solutions than the other methods.

Oftentimes, a portfolio outperforms the expected returns. The investor would like to know if the added value in performance is worth the extra risks taken. To ensure that portfolio managers do not have an incentive to increase risk, there is often a limit on the volatility of the portfolio deviation from a benchmark, which is called tracking-error volatility. However, this leads to an agency problem due to the manager being incentivized to optimize excess-return without considering overall portfolio risk. Jorion aims to investigate if this agency problem can be corrected with additional restrictions on the portfolio by eliminating the tracking-error volatility constraint [31]. In the analysis, it is found that if tracking-error volatility constraint must be kept, imposing an additional constraint on total volatility will lead to a better balance of returns and risk.

2.3 Addition of My Thesis

In this section, I have described previous research on stock price prediction techniques and portfolio optimization analysis. My thesis adds to this existing literature by merging these two topics into one paper. The first part of my analysis will be using different machine learning algorithms to find the most accurate stock prediction model, and the second part

of my analysis will be finding an optimal portfolio using the trained models from the first part of my analysis. Analysis of both stock price prediction and portfolio optimization on the same set of data will lead to a more complete picture of how to construct a portfolio that maximizes returns without undue risk.

Chapter 3

Data

For any machine learning model, having high quality data as an input is essential. This ensures that the model can learn the correct patterns and relationships for prediction, generalize better to unseen data, and train more efficiently. Data preparation is critical to machine learning because it directly drives the model's performance and accuracy. This section will describe the data preparation process of creating a dataset that will be used to train the machine learning algorithms:

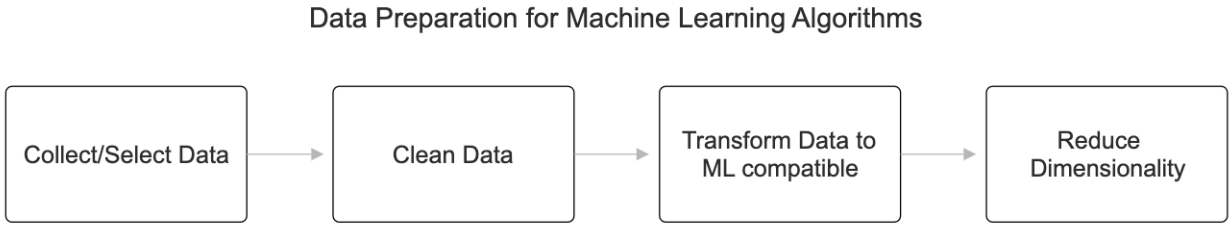

Figure 3.1
Process used to prepare data for training of machine learning algorithms

3.1 Data Collection and Selection

Given the dynamic and complex nature of the financial market, it is easier to make accurate predictions on more stable stocks. The Russell 1000 Index tracks the performance of the 1,000 largest public companies in the United States, which accounts for approximately 92% of the U.S. stock market [32]. In this paper, we start with a dataset of the 1,000 largest public companies in the United States. The data used in this paper is extracted from the Center for Research in Security Prices via Wharton Research and Data Services. The time range of the data set is all dates between January 2nd, 2019, and December 29th, 2023.

In related works, researchers have selected a subset of data to train and test their model. This is due to the understanding that it is not realistic to have thousands of different stocks in a portfolio. For instance, Almahdi and Yang propose an optimized portfolio that includes only five assets [33], and Abrami and Marsoem use a single index model approach that leads to an optimal portfolio of eight assets [34]. Therefore, analyzing the 1,000 stocks that are included in this dataset is not necessary. When a model is deployed in practice, the runtime can be important if there are quick decisions to be made. Using data with 1,000 stocks can slow down the decision making of the model, meaning a smaller data set would be preferred. Wang et. al use deep learning for portfolio optimization, and 21 stocks was deemed to be sufficiently large for asset preselection before portfolio forming [35]. Additionally, Ma et. al use two machine learning models and three deep learning models to predict return on a portfolio. In the paper, they use 49 stocks from China Securities 100 Index to train and test their prediction models [36]. Therefore, for my research, the 50 largest United States companies seem to be an appropriate number of stocks for the data of my analysis. Therefore, from the dataset of the largest 1,000 publicly traded companies in the United States, I will select the largest 50 (for ticker symbols of the largest 50 companies, see Appendix A).

After parsing out only the largest 50 companies from the original dataset, the size of the new dataset is 63,530 rows by 63 columns. Each row represents the closing characteristics

of a stock for each day that the market is open. So for example, one row of data will have the characteristics at 4:30 PM ET on 03-14-2020 for AMZN stock. Some of the potentially important data columns are listed in the table below:

Table 3.1: Potentially Important Features of the Data

CRSP Daily Stock	
Column name	**Description**
ticker	Ticker
permco	CRSP Permanent Company Number
prc	Price (Bid/Ask average)
vol	Volume
askhi	Ask or High Price
bidlo	Bid or Low Price
ret	Returns
vmretd	Value-Weighted Return
ewretd	Equal-Weighted Return
sprtn	Return on S&P Composite Index

A complete list of variables in the dataset is given in Appendix B. To get a better visualization, I have included a graph of one of the companies in the dataset. In Figure 3.2, you can see the time series of Microsoft stock from January 2nd, 2019, to December 29th, 2023, as an example.

3.2 Data Cleaning

The first check was to ensure that there were exactly 50 different stocks in the dataset. After checking the count of unique ticker symbols, there was a count of 51 unique values. The Meta stock changed its ticker symbol from "FB" to "META" in 2022, so I converted all "FB" symbols to "META" to ensure that the data was classified under one security.

Next, there were some obvious redundant stock identification columns. Examples of redundant stock identification columns are CRSP Permanent Company Number, Company Name, and CUSIP identifier. Also, there are some redundant columns for the exchange

Figure 3.2
Microsoft Stock Price from January 2nd, 2019, to December 29th, 2023.

where the stock is listed. Examples of a redundant column for exchange are Exchange Code, Header Exchange Code, and Nasdaq Issue Number, because only including the Exchange code column contains all the information that we need. Finally, there are some columns that only have one distinct value for the entire dataset. Because there is only one distinct value, these variables will not be useful to our model and we can remove these columns. Examples of one-distinct value columns are Trading Status and Security Status. After looking through the columns that are redundant or only have one distinct value, there are a total of 18 columns removed.

It is noticeable that there are many missing values. In order to look at the columns with many values missing, I found the 13 columns that have more than 90% of their values as missing values. I then looked at the meaning of each of these columns to get a better understanding of which of these variables can be disregarded. After looking through the 13 columns, I determined that 3 columns are not adding any information and can be disregarded. One example of a disregarded column is Delisting Code, which is not relevant to the 50 largest U.S. companies from 2019 to 2023.

There were originally 63 columns in the dataset, but through the data cleaning process, I have eliminated 21 columns. This means that we are left with 42 remaining variables. The next step is to transform the variables into a form that the model can extract value from. In the next section, we will do data transformation.

3.3 Data Transformation

There are some components of the data that need to be transformed to be understood by the machine learning algorithm. This is important to model performance because creating new features or modifying existing ones can help models better capture underlying patterns in the data. Additionally, it can help with handling different data types, such as transforming categorical data into a numerical format or extracting useful information about dates.

The main data transformations in this dataset are modifying the variables into numerical format. There are some variables that are in numerical format, but these numbers do not have meaning. One example is Exchange Code, where 1 means the stock is traded on NYSE, and 3 means the stock is traded on NASDAQ. Keeping this numerical format will represent an ordering of the exchanges, which is not desired. Therefore, we need to represent the Exchange Code as separate exchanges without any ordering. There are also categorical variables that should be represented as numerical, such as Share Class, where the letter is the class of the stock. To handle the categorical data in the dataset, I used one hot encoding for nominal data and a dummy variable for binary variables.

By transforming the categorical variables into numerical variables, the dataset can now be used as an input for machine learning algorithms (the summary statistics of the numerical variables are seen in Appendix C). However, efficiency can be important for machine learning models that need to make quick decisions. To create a more efficient model, we must reduce the number of variables as inputs. We can do this by Principal Component Analysis, which is described in the next subsection.

3.4 Dimensionality Reduction

Principal Component Analysis (PCA) is commonly used for data preprocessing in preparation for machine learning algorithms. The goal of PCA is to preserve the most relevant features of large datasets while reducing dimensionality. A greater number of features negatively affects the time-complexity of a model, meaning that running a model on data after PCA will reduce model time-complexity. PCA also minimizes two other common issues with larger datasets, multicollinearity and overfitting. In this subsection, we will reduce the dimensionality using the PCA process [37].

The first step is to standardize the data so that the features are along the same scale. In order to do this, we will use a standardization formula[1] given by the following:

$$z_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j} \tag{3.1}$$

where z_{ij} is the standardization of feature j at row i, x_{ij} is the value of feature j at row i, μ_j is the mean of feature j, and σ_j is the standard deviation of feature j. The result is a mean of 0 and a standard deviation of 1 for the standardized variable.

The next step is to find how the features vary with each other. For two feature vectors x_j and x_k, the covariance between them, σ_{jk}, is calculated using the following equation:

$$\sigma_{jk} = \frac{1}{n-1} \sum_{i=1}^{n} \left(x_j^i - \mu_j\right) \left(x_k^i - \mu_k\right) \tag{3.2}$$

Using the calculated covariance, we can create a covariance matrix. This symmetric matrix contains the covariance values between each pair of features, and it has dimension d

[1] only makes sense for numerical values, so this was not applied to created encodings

by d. This means that the covariance matrix has the form:

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \sigma_{13} & \sigma_{14} \\ \sigma_{21} & \sigma_2^2 & \sigma_{23} & \sigma_{24} \\ \sigma_{31} & \sigma_{32} & \sigma_3^2 & \sigma_{34} \\ \sigma_{41} & \sigma_{42} & \sigma_{43} & \sigma_4^2 \end{bmatrix} \tag{3.3}$$

Using the standardized data on the numerical values and the equations above, I have created a covariance matrix of the features (Figure 3.3).

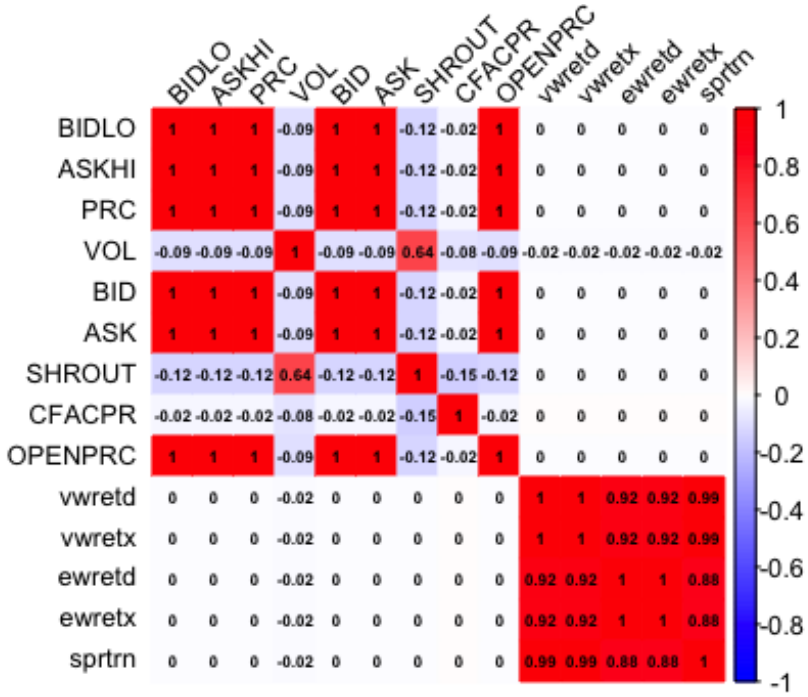

Figure 3.3
Covariance Matrix of Numeric Features of the Dataset

After finding the covariance matrix of the features, the next step is to find the eigenvectors and the eigenvalues. The eigenvectors represent the directions of maximum variance of the covariance matrix, called the principal components. The eigenvalues are the corresponding magnitude of the eigenvectors. Therefore, the eigenvector that has the largest eigenvalue is the direction that has the most variance. After computing, the eigenvectors are sorted by

the magnitude of the eigenvalues. To determine how many principal components to select for our model, we plot the cumulative sum of the eigenvalues (Figure 3.4).

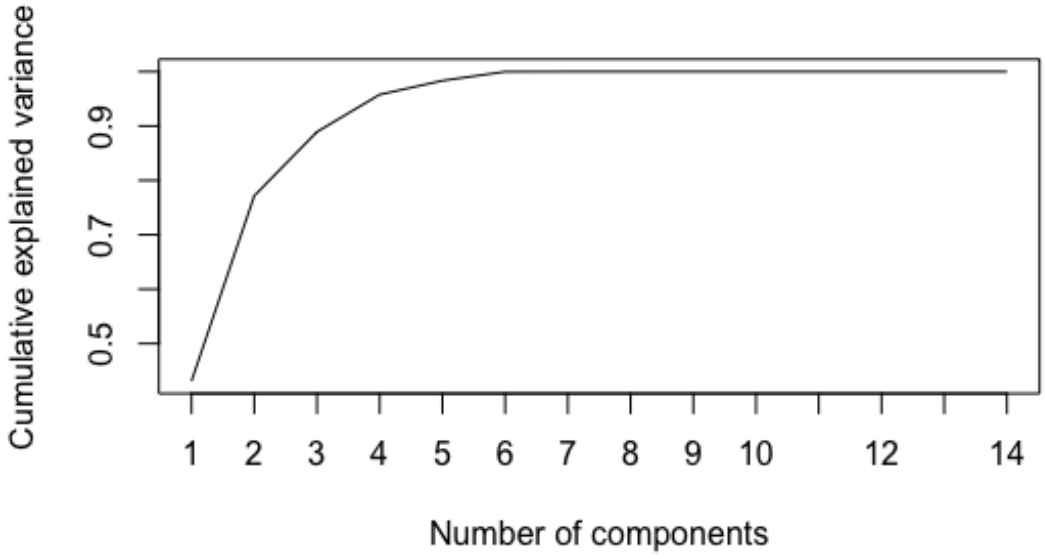

Figure 3.4
Cumulative Explained Variance by Number of Principal Components

From the plot, it is evident that about 99% of the variance is captured in the six largest principal components. Therefore, it is reasonable that we can use the six largest principal components as inputs to our prediction model. These six components will make up the "projection matrix" of our standardized data. With data preprocessing, we have reduced our dimensionality to six features, which is critical to having models that can run efficiently.

In this paper, this dataset will be used when the pre-PCA dataset is too large to run efficiently. We can do this because the six columns will cover most of the variation in the data.

3.5 Splitting the Data

Since we have a time-series dataset, the train-test split of the data is a special case. The goal of a testing set is to validate the model predictability that was trained on the training set. However, for time-series data, the test data must be later in time than the training data. This is the most realistic situation because the model in practice will not have access to any future data, but only the data that is available at that point in time.

For this data in this paper, the data will be split based on the dates. The training data will be the data from January 2nd, 2019, to December 30th, 2022, and the testing data will be data from January 3rd, 2023, to December 29th, 2023. In other words, all data from 2019 to 2022 will be the training data, and data from 2023 will be the testing data. This is a train-test split of 80.13% training and 19.87% testing (an approximate 80-20 split).

Chapter 4

Methodology

In this section, I will describe the methodology for the stock prediction models, hybrid models, and portfolio optimization, where each one of these topics will have a subsection. For each of the algorithms I am using, there will be a description of the algorithm, as well as the training procedure. These methods will then be applied to the data and analyzed in the next section of this paper.

4.1 Stock Prediction Algorithms

Linear Regression

Linear Regression is important for machine learning analysis because it evaluates and establishes a linear relationship between multiple variables. Although it is a simple supervised machine learning model that cannot capture complex intricacies within the data, it provides an easily understandable baseline model. The equation of the model has clear coefficients that show the impact that each independent variable has on the dependent variable, which can help in understanding the dynamics of the data.

For our linear regression, we have multiple independent variables and a dependent variable

of stock price. This means that the linear regression will take the following form:

$$y = \beta_0 + \beta_1 X + \beta_2 X + \ldots\ldots\ldots \beta_n X \tag{4.1}$$

where:

- Y is the dependent variable (price)
- $\mathrm{X}_1, \mathrm{X}_2, \ldots, \mathrm{X}_\mathrm{p}$ are the independent variables
- β_0 is the intercept
- $\beta_1, \beta_2, \ldots, \beta_n$ are the coefficients

I will apply linear regression as a baseline machine learning model, meaning that any model that has a worse performance than linear regression will be disregarded.

Support Vector Machine

Support Vector Machine (SVM) is a supervised machine learning algorithm that finds the optimal hyperplane in a N-dimensional space that can separate the data points into different classes. It is best suited for classification problems, but it can also be applied to regression. The hyperplane attempts to maximize the distance between the closest points of the different classes. This means that the extreme points on either side of the hyperplane are critical to determining the hyperplane, which is also known as the Support Vector Classifier (SVC). An example of a hyperplane and the maximum margin are shown in Figure 4.1. Note that a hyperplane is not always linear.

Because stock price is a continuous target variable, an extension of SVM called Support Vector Regression (SVR) will be used. SVR differs from conventional regression because the hyperplane is selected by maximizing the distance between the extreme points on either side of the hyperplane, whereas conventional regression minimizes some error function. Therefore, SVR aims to minimize the upper bound of generalization error rather than training error, which is called Structural Risk Minimization. SVR is expected to perform better than

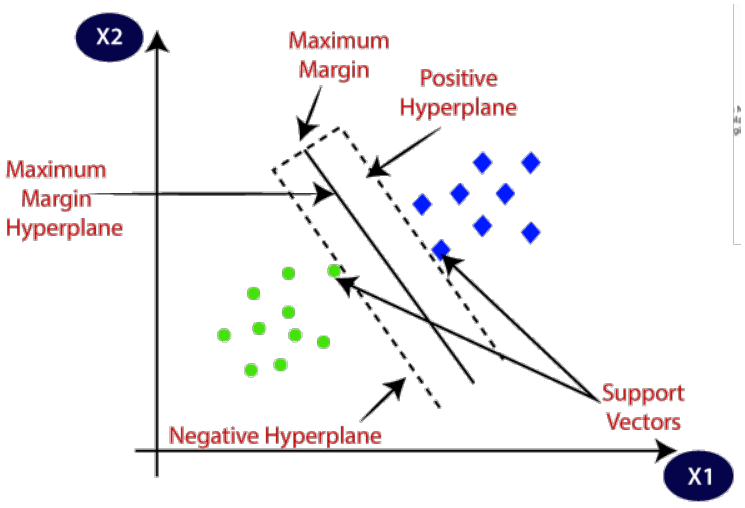

Figure 4.1
Example of Hyperplane of Support Vector Machine in 2-D [38]

conventional techniques because it is usually resistant to both over-fitting and outliers. Due to this robustness, SVR is often used for time-series data, similar to the stock price prediction problem in this paper.

Random Forest

Random forest combines the output of multiple decision trees and gets a single result. It is widely used for both classification and regression predictive modeling problems since it is a further refinement of classification and regression tree (CART) models. In a CART model, the algorithm selects the optimal decision at each split by recursive partitioning, or analyzing the value of different thresholds for each predictor and implementing the most valuable threshold until there is not a significant improvement to predictions [39]. This uses a single-tree, which usually cannot handle complex data with many features. However, in a random forest, many different decision trees are averaged to produce more accurate predictions.

In this paper, we will use Random Forest as a time-series forecasting. To create a Random Forest model, there are a number of decision trees that are created from a different bootstrap

sample of the training dataset. A bootstrap sample means that a sample with replacement is taken from the training data. Using different bootstrap samples is effective because each decision tree is fit on a slightly different dataset, which means slightly different results. In addition to bootstrap sampling, Random Forests also select a subset of input features that each split point in the trees are determined from. With each split point coming from a random subset of features, this forces each decision tree to be slightly different. Predictions from all the trees are aggregated, and this leads to a more accurate result than using any of the single trees.

Recurrent Neural Network

Recurrent Neural Network (RNN) is a variation of a neural network such that the output from the previous step is fed as an input to the current step. This works better than a classic neural network for sequential data because in classic neural networks, the inputs and outputs are independent of each other. The difference in architecture between a RNN and a traditional feed forward network can be seen in Figure 4.2. For time-series data, it would follow intuition that the output of the previous step will likely be correlated with the prediction of the current step. The most distinctive feature of a Recurrent Neural Network is the hidden state, where the model has memory about the sequence. RNN uses the same parameter for each input since it performs the same task on all inputs and hidden layers to get the output, which leads to a reduction in the complexity of the parameters.

A Recurrent Unit is a fundamental processing unit in a RNN. Each Recurrent Unit has the ability to store information about a hidden state, which allows the model to capture time dependencies of previous inputs. For each time step, a RNN has an activation function unit with a hidden state. Also at each time step, the hidden state is updated to reflect the change in knowledge of the network. The hidden state and the parameters of the network are updated using backpropagation.

While RNN offers several advantages such as handling sequential data and capturing long-

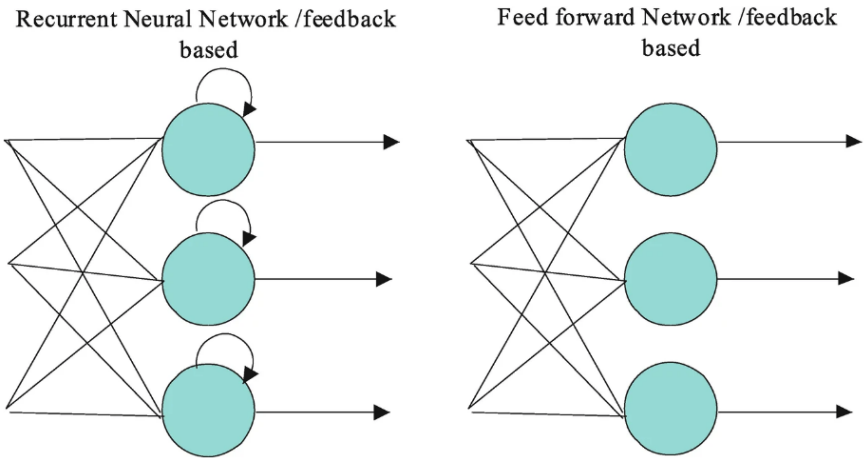

Figure 4.2
Difference in the Architecture of Recurrent Neural Network and Feed Forward Network [40]

term dependencies, the vanishing gradient problem is a challenge that affects the training of RNN. It is possible that the gradients, which update the direction and magnitude of the network weights during training, will become very small as they backpropagate. One solution to combat the vanishing gradient problem is Long Short-Term Memory, which will be described in the next section.

Long Short-Term Memory

Long Short-Term Memory (LSTM) is a type of recurrent neural network that is designed to handle the issue of vanishing and exploding gradients. To capture long-term dependencies in sequential data, LSTM introduces a memory cell, which is a container that can hold information for an extended period. Each memory cell has three components: an input gate, a forget gate, and an output gate. These three gates determine what information is added, removed, and outputted from the memory cell [41]. The input gate decides on what information to store, the output gate determines the next hidden state based on the updated cell state, and the forget gate determines the information that will be discarded. A visualization of the memory cell can be seen in Figure 4.3. This memory cell is how the model regulates the flow of information and allows storage of long-term dependencies in

time-series data.

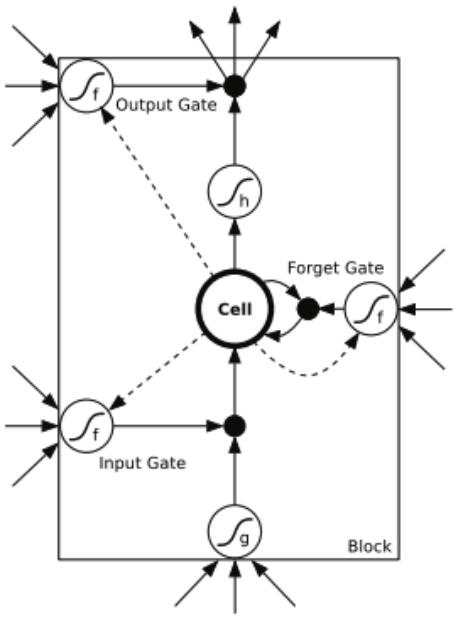

Figure 4.3
Memory Cell of a LSTM model: The input and output gates multiply the input and the output of the cell, and the forget gate multiplies the cell's previous state. The gate activation function 'f' is usually the logistic sigmoid, meaning that activations are between 0 (gate closed) and 1 (gate open). Functions 'g' and 'h' are typically tanh or logistic sigmoid functions. [41]

LSTM is similar to a RNN, but it uses memory cells to better capture long-term patterns in sequential data and handles the problem of vanishing gradients. Therefore, it is expected that LSTM performs better than RNN for prediction.

BiLSTM

Bidirectional Long Short-Term Memory (BiLSTM) is a model that consists of two LSTMs, with one taking the input in the forward direction and the other taking input in the backward direction. By having two LSTMs, the model can increase the amount of available information and better understand the data in both directions. The architecture of BiLSTM can be thought of as two separate LSTM networks going in opposite directions. Both of these LSTMs will produce an output, and the final output is the combination of these probabilities.

Figure 4.4 shows a visual representation of the architecture. Because the BiLSTM has more

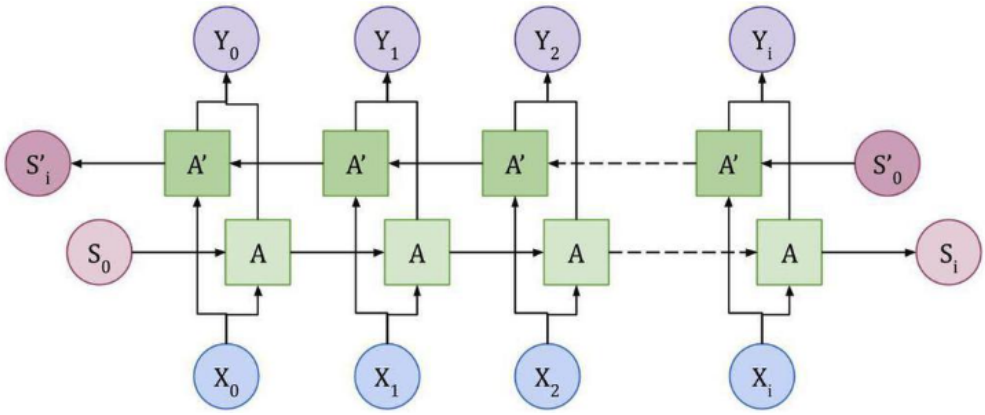

Figure 4.4
Bidirectional LSTM Layer Architecture: X_i is the input token, Y_i is the output token, A are the nodes of one LSTM, A' are the nodes of the other LSTM, and Y_i is the combination of the two LSTMs [42]

available information and combines the output of two LSTMs, it is expected to perform better than the single LSTM.

LightGBM

LightGBM is a gradient boosted framework developed by Microsoft based on decision trees designed to improve efficiency and reduce memory usage. This model has some similarities to Gradient Boosted Decision Trees and XGBoost, but it was created to be applicable to more solutions. To reduce memory usage, it retains instances with large gradients during training, called Gradient-based One-Side Sampling. For efficient tree construction, this model employs a histogram-based algorithm. Arguably the most critical feature of the model, LightGBM has leaf-wise tree growth rather than depth-wise tree growth, which means that the leaf node enlarges by finding the biggest split gain out of all the tree's leaves. This allows the model to produce a more accurate model while having fewer nodes [43].

4.2 Hybrid Models

In the previous subsection, there are descriptions of many machine learning algorithms. Applying them individually has a lot of value, but it is possible that combining two (or more) of the techniques above can result in more predictive power. A hybrid approach attempts to leverage the strengths of multiple machine learning techniques to create more robust and accurate models. Hybrid machine learning models have the potential to create greater efficiency, versatility, and accuracy.

CNN-LSTM

The first hybrid model explored in this paper is CNN-LSTM. The idea to have CNN used for feature engineering and LSTM used for stock prediction was introduced by Lu et. al [44]. CNN stands for Convolutional Neural Network, and this is composed of convolutional layers, pooling layers, and fully connected layers. The architecture of CNN is inspired by visual processing that occurs in human brains, and they are used to detect hierarchical patterns and spatial recognitions within images. The weight sharing and local perception of CNN can greatly reduce the number of parameters in the data, which will improve the model's efficiency. The convolutional layer contains multiple convolution kernels where the features of the data are extracted. In order to reduce the number of feature dimensions, there is a pooling layer that is added after the convolution layer:

$$o_t = tanh(x_t * k_t + b_t) \tag{4.2}$$

where o_t is the output value after convolution, $tanh$ is the activation function, x_t is the input vector, k_t is the convolution kernel weight, and b_t is the bias of the convolution kernel [44].

As described above, the output data of the CNN layer are calculated through the LSTM layer, and the output value is obtained. By using CNN and LSTM, we can capture the

strength of CNN for feature selection and LSTM for predictive power to get an improved model. The complete process for the CNN-LSTM layer explained by Lu et. al is outlined in Appendix D.

BiLSTM-BO-LightGBM

The second hybrid model that I am evaluating in this paper is the BiLSTM-BO-LightGBM hybrid model. The idea of using BiLSTM to forecast each selected attribute to build a new "text" testing set for the LightGBM model was introduced by Tian et. al [45]. For this paper, I will employ a similar method, using Bayesian optimization to optimize the parameters for the LightGBM model. Bayesian optimization (BO) is used for parameter tuning because it uses a probability model of the objective function to select the parameters that will approximate the true objective function. Using BO will allow our LightGBM model to have the optimal parameters.

The first step for the BiLSTM-BO-LightGBM algorithm is to use the six attributes found in the PCA from the data section of this paper. Using BiLSTM, each of the six attributes will be trained using a stock price forecasting model. This means that the BiLSTM forecasts each attribute, and the predicted results are recombined to make a new "text" testing set. LightGBM will be used to build a stock price forecasting model on the original data, and Bayesian Optimization algorithm is used to optimize the parameters. The new BO-LightGBM stock price forecasting model is used on the "text" dataset from the BiLSTM model. The difference between the real value and the predicted value will be the performance of the model.

By using BiLSTM, BO, and LightGBM, we hope to get a more accurate prediction model. We use the strength of BiLSTM to obtain a new "text" testing set, BO to optimize parameters for LightGBM, and LightGBM for the strong predictive power of gradient boosted decision trees.

4.3 Portfolio Optimization

For the portfolio optimization section of my paper, I will use Mean-Variance Optimization. The Mean-Variance model is described in the Introduction section of my paper, and I will more formally describe how it will be utilized to evaluate the portfolios in this paper. The goal of the Mean-Variance Optimization is to maximize returns given an allowed variance of a portfolio. The application of Mean-Variance Optimization is as follows [46].

Let's say there is a portfolio constructed from N different assets. The weights, or proportion, of the different assets in a portfolio can be written as $w = (w_1, w_2, \dots w_N)$, with the constraint given $\sum_{i=1}^{N} w_i = 1$ since there is only one portfolio. Let us call I the N-dimensional vector $I = (1, 1, \dots, 1)$. Therefore, the portfolio constraint can be written as $w^T I = 1$. The random returns on stocks are $r_1, \dots, r_N$, and let the vector of expected return by $\mu = (\mu_1, \mu_2, \dots, \mu_N)$ with $\mu_i = E(r_i)$ for $i = 1, 2, \dots, N$. The covariances between returns are represented by $\sigma_{ij} = \mathrm{Cov}(r_i, r_j)$. These covariances (and variances along the diagonal) are the entries of the $N \times N$ covariance matrix Φ.

$$\Phi = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1N} \\ \sigma_{21} & \sigma_{22} & \cdots & \sigma_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{\mathrm{N1}} & \sigma_{\mathrm{N2}} & \cdots & \sigma_{\mathrm{NN}} \end{bmatrix} \tag{4.3}$$

The expected return $\mu_P = E(R_P)$ and variance $\sigma_P^2 = \mathrm{Var}(R_P)$ of a portfolio with weights w are seen in these two equations

$$\mu_P = \sum_{i=1}^{N} w_i \mu_i = w^T \mu, \tag{4.4}$$

$$\sigma_P^2 = \mathrm{Var}(R_P) = \sum_{i,j=1}^{N} w_i w_j \sigma_{ij} = w^T \Phi w. \tag{4.5}$$

The return maximizing mean-variance portfolio allocation problem is formulated as follows, with k being the allowable variance parameter:

$$\begin{aligned} \text{Maximize} \quad & w^T \mu = r_{\text{target}} \, , \\ \text{subject to} \quad & w^T \Phi w \leq k \\ & w^T I = 1. \end{aligned} \tag{4.6}$$

Using the Mean-Variance model, I will look for the algorithms that lead to the highest returns for a given level of variance tolerance. The algorithms with higher returns at the same level of variance are better portfolio constructions.

In particular, the main goal of portfolio optimization is to gain excess returns, also known as alpha. Alpha is calculated using Equation 1.1: $y = \alpha + \beta x + \mu$, where y is the returns and β is the volatility of the stock relative to the benchmark. In our data, we are not given beta values for the individual securities. In order to calculate the beta values, I will use the formula for β:

$$\beta = \frac{\text{Cov}(R_e, R_m)}{\text{Var}(R_m)} \tag{4.7}$$

where R_e is the expected return on an individual stock and R_m is the return of the benchmark (in this case S&P 500). For each of 50 stocks in the data set, the beta value is calculated over the duration of the training set data, January 2nd, 2019, to December 30th, 2022. The model will use these beta values as inputs when selecting the stocks with the greatest alpha return, as defined in Equation 1.1. Thus, in the portfolio optimization, stocks will be selected by the maximum of α_i for the 50 stocks:

$$\alpha_i = R_{ei} - \beta_i * R_m \tag{4.8}$$

where

- α_i is expected alpha for stock i

- R_{ei} is expected return of stock i
- β_i is volatility of stock i relative to the benchmark (in this case S&P 500)
- R_m is the performance of the benchmark (in this case S&P 500)

Because the portfolio optimization is selecting stocks based on expected alpha, a risk-adjusted return is being maximized, which is usually preferred for investors.

Chapter 5

Analysis of Results

5.1 Stock Prediction Accuracy Evaluation

The performance of the machine learning algorithms on unseen data must be evaluated after they are implemented. As described in the data section, the data was split into a training data set of January 2nd, 2019, to December 30th, 2022, and a testing data set of January 3rd, 2023, to December 29th, 2023. I used the training data to train the models and then implemented the forecast predictions on the testing data. To evaluate the performance of the models, I used three criteria: mean absolute error (MAE), mean square error (MSE), and symmetric mean absolute percentage error (SMAPE). These three criteria are the benchmark evaluators for many prediction models, and they are often used in research:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{p}_i - p_i)^2 \tag{5.1}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{p}_i - p_i| \tag{5.2}$$

$$SMAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\hat{p}_i - p_i|}{(|p_i| + |\hat{p}_i|)} \tag{5.3}$$

where n is the number of forecasted predictions, $\hat{p}_i$ is the predicted price for stock i, and p_i is the true price for stock i.

The models are implemented as described in the earlier section. In Figure 5.1, there is an output of the predicted values for Walmart and the Long Short-Term Memory algorithm. The algorithm predicted the values of the Walmart stock for all trading dates in 2023 based on the trained LSTM model.

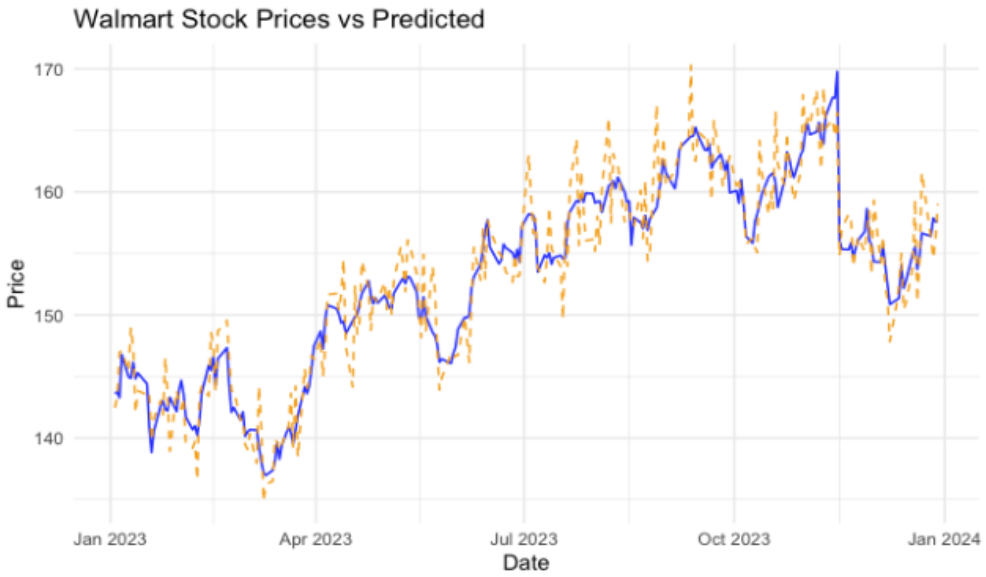

Figure 5.1
Walmart Stock Price vs. Predicted Value using LSTM algorithm: The actual price is the blue solid line and the LSTM predicted price is the orange dotted line

After training all of the models, the test data was predicted using the trained model. The evaluation performance results over all 50 stocks are shown in Table 5.1. It is worth noting that SMAPE is represented in the table as a percentage.

The first observation from looking at the table is that the MSE numbers are high in magnitude. Due to the different prices of each stock, stocks that have higher prices are skewing the MSE. Let's say that a prediction is 2% off the actual value for two companies: company A and company B. If the stock price of company A is \$800, then the MSE of the prediction is $(0.02 * 800)^2 = 256$. However, if the stock price of company B is \$20, then the

Table 5.1: Performance Results of Algorithms

Model	**MSE**	**MAE**	**SMAPE**
Linear Regression	265.398	27.341	8.364
Support Vector Machine	178.093	14.173	5.185
Random Forest	129.104	12.398	4.472
Recurrent Neural Network	92.827	9.836	3.631
Long Short-Term Memory	51.993	7.839	2.703
BiLSTM	53.761	7.624	2.655
LightGBM	74.928	9.386	3.390
CNN-LSTM	40.745	6.538	2.349
BiLSTM-BO-LightGBM	34.181	6.323	2.284

MSE of the prediction is $(0.02 * 20)^2 = 0.16$. Even though these predictions were off by the same percentage, the mean square errors have different values. For investing, the percentage is often seen as more important because a return on investment is measured in percentage. Therefore, the MSE and MAE values are skewed due to higher stock prices holding more weight on the averages. SMAPE should be the criteria that we judge the quality of the model.

Looking at the symmetric mean absolute percentage error values for the different models, BiLSTM had the lowest value for the individual machine learning algorithms with a SMAPE of 2.655. The LSTM algorithm also had a relatively low SMAPE of 2.703, which is just barely greater than the BiLSTM. The highest SMAPE is the Linear Regression model at 8.364, and this was expected for a time-series as complex as financial data. The linear regression model does not have the architecture to deal with financial data intricacies. Both of the hybrid models had lower SMAPE values than all of the individual models, which signal that hybrid models are better at stock price predictions. For forecasting future stock prices, the results show that BiLSTM-BO-LightGBM has the most predictive power.

Creating a model to predict stock prices is only the first part of the solution. The next part is to use Mean-Variance Optimization to create an optimal investment portfolio.

5.2 Portfolio Optimization Performance

The stock prediction model predicts what the price of the stock will be in the next day. To make this information useful, an investor needs to create a portfolio that looks to maximize returns while minimizing risks. I created a 2023 simulation that selects stocks each day based on the Mean-Variance Optimization formulation. To evaluate the performance of different optimal portfolios, I will use two different criteria: excess returns (α) using the Mean-Variance Optimization framework and the Sharpe Ratio.

5.2.1 Mean-Variance Optimization Framework

To evaluate using the Mean-Variance Optimization framework, I created a simulation of the trading days of 2023, which is the unseen data. This was done by running a loop for each of the trading days and updating available data with the previous day's information because the investor would have access to the prior day's data. Each day, the model will choose the top N stocks that will be purchased for the day without exceeding a set value for variance, and the position will be cleared at the end of the trading day at the closing price. To determine which N stocks are chosen, the model will select the stocks with the highest expected alpha, as defined in Equation 1.1. This means that both expected returns and beta values will be factors in the expected alpha value.

Before running the simulation, I set the upper bound of the variance each day at 0.01 and chose $N = 5$. Since we are looking at the largest 50 U.S. companies, I do not expect to have too much single day variation in any of the assets. Therefore, I believe that a modest variance of 0.01 is an appropriate number for the model to make sound financial decisions. I chose $N = 5$ because in prior work, portfolios are often optimized at fewer than 10 stocks. For instance, Almahdi and Yang propose an optimized portfolio that includes only five assets [33], and Abrami and Marsoem use a single index model approach that leads to an optimal portfolio with just eight assets [34]. Therefore, each trading day, the model will select 5 stocks with weight vector w that does not exceed a variance of 0.01.

I ran the simulation using each of the models for the trading days in 2023. I then found the α value of the portfolio in relation to the 2023 S&P 500, which means α represents the active return on the investment compared to the overall market. The S&P had a strong performance in 2023 with a return of 24.14%. The final alpha values for the different models are in Figure 5.2. The model with the highest alpha value was the CNN-LSTM hybrid model, which had the second lowest SMAPE in the stock prediction. With an alpha value of 6.18%, this means that the overall return on the portfolio for the year was 30.32%. The two hybrid

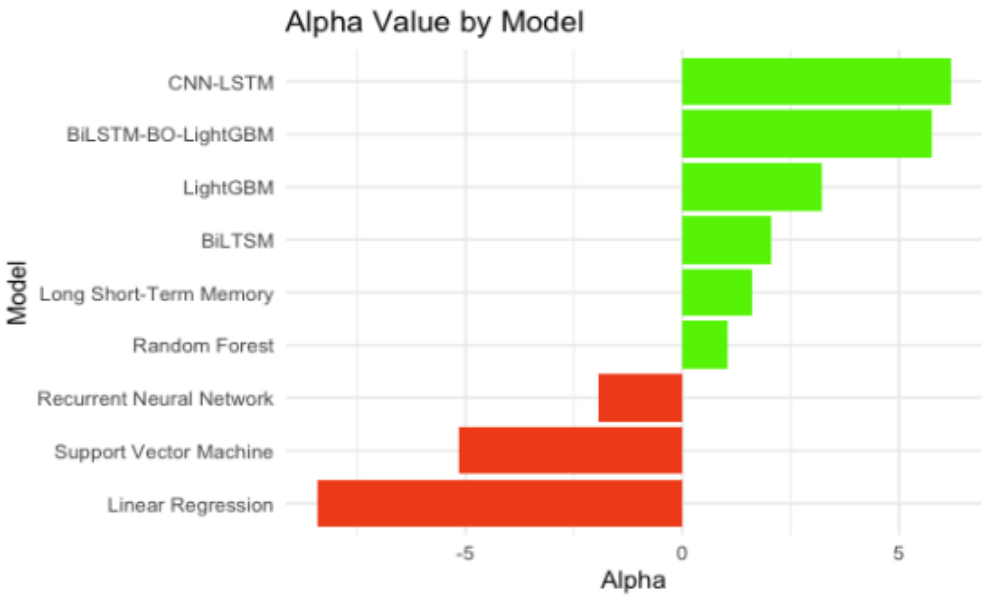

Figure 5.2
Alpha (α) Value by Model for 2023 Simulation: CNN-LSTM model has the highest alpha value at 6.18% above the S&P 500

models seem to outperform the individual algorithms with significantly higher alpha values.

Note of Definition of Alpha

In this paper, I have defined alpha as the excess return on the beta-adjusted S&P 500 index. Therefore, to analyze the performance of the 2023 testing data, I reported "alpha" as the value of the returns above the S&P 500 index over the sample window of the simulation. However, it is possible that there are risks outside the sample window that are being priced into the stocks [47], which would mean that the model observes realizations of these risks that are outside of the time-frame of the simulation. The potential risk premium that is priced into the stocks are included in this definition of alpha, since it is difficult to separate the risk premiums from the pure alpha value. Hence, the alpha value reported in the analysis is the excess return on the beta-adjusted S&P 500 index with potential risk premium included.

5.2.2 Sharpe Ratio

The Sharpe Ratio is a number that compares the return of an investment with its risk. The numerator is the difference in expected return in the investment and a benchmark, often a risk-free benchmark. The denominator is the standard deviation of returns over that period of time. Since the paper is using United States data, the risk-free benchmark will be the U.S. Treasury bonds 10-year yield of 3.8%. The Sharpe Ratio equation is as follows:

$$SR = \frac{R_p - R_f}{\sigma_p} \tag{5.4}$$

where R_p is the expected return of the portfolio, R_f is the risk-free return, and σ_p is the standard deviation of the portfolio's excess return.

The Sharpe Ratios were calculated using the 2023 daily simulation results. The resulting Sharpe Ratios are seen in Figure 5.3. The CNN-LSTM also has the highest Sharpe Ratio of 2.41, which approximately means that the weight of excess returns outweighs the weight of volatility by 2.41 times. Another observation is that using a predictive model, like Linear Regression in this case, can be worse than investing in a market index like the S&P 500. The two hybrid models again outperformed the individual algorithms in Sharpe Ratio.

For the two criteria, alpha value and Sharpe Ratio, the CNN-LSTM model had the most promising results. However, the other hybrid model, BiLSTM-BO-LightGBM, had promising results as well, only being slightly behind the CNN-LSTM model. In both criteria, the hybrid models seem to yield better results for the investor.

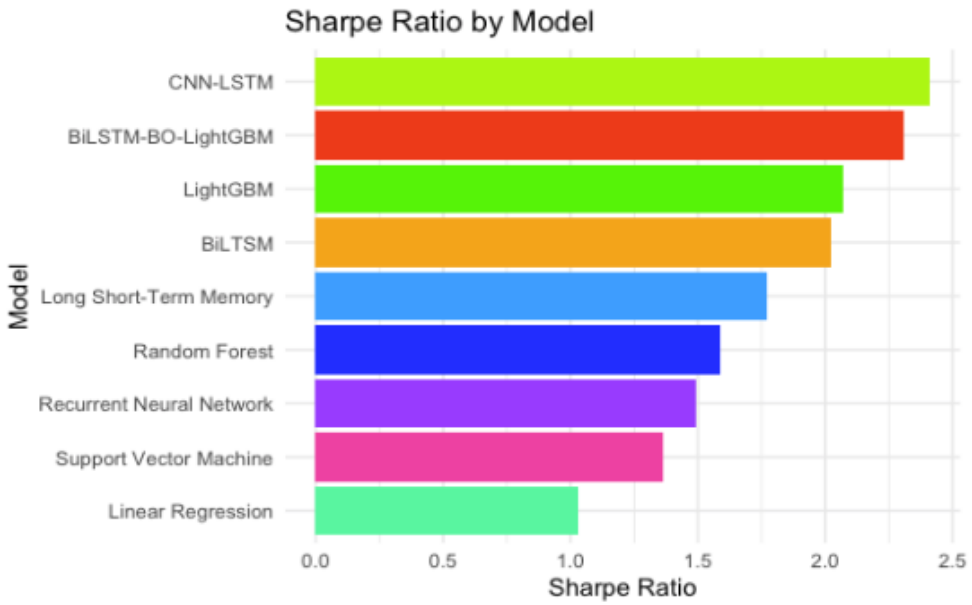

Figure 5.3
Sharpe Ratio by Model for 2023 Simulation: CNN-LSTM model has the highest Sharpe Ratio value at 2.41

Chapter 6

Conclusion

6.1 Discussion and Key Findings

This thesis aims to contribute to the existing literature by advancing portfolio optimization techniques through machine learning stock selection methods. There are two parts to this problem: find an accurate stock price prediction model and select an optimal portfolio that balances return and risk.

I sourced detailed data on the 50 largest U.S. companies and preprocessed it so that it can be used as an input for the machine learning algorithms. For each of the securities, I also calculated the β value over the duration of the training data, which is used for the calculation of expected alpha in the portfolio optimization of this paper. In addition to individual algorithms, I implemented two hybrid machine learning models. I trained nine algorithms: Linear Regression, Support Vector Machine, Random Forest, RNN, LSTM, BiLSTM, LightGBM, CNN-LSTM, and BiLSTM-BO-LightGBM. With each of these models, I forecasted a year of daily data to evaluate the prediction accuracy. To evaluate how these prediction models would create a portfolio, I created a simulation of all the trading days in 2023. Each day, the model chose five stocks with a variance limit constraint using the

Mean-Variance Optimization framework. The excess returns, α[1], of optimal portfolios were analyzed. Additionally, the Sharpe Ratio of each portfolio was calculated to measure the risk-adjusted performance.

After the analysis of my model, the BiLSTM-BO-LightGBM has the smallest symmetric mean absolute percentage error (SMAPE) out of the implemented models with a SMAPE of 2.655. However, when crafting an optimal portfolio, the CNN-LSTM model measured an alpha value of 6.18% above the S&P 500 and a Sharpe Ratio of 2.41, which were both the best results. It is evident that the hybrid models performed better than the individual models in both stock prediction and portfolio optimization.

6.2 Theoretical Implications and Future Work

This paper enhances the theoretical research on portfolio optimization using a stock selection model, especially for United States financial markets. I have implemented nine prediction models on the most prominent stocks in the United States, and these models have the ability to seek appreciating assets for an investor's portfolio. The performance evaluation shows that there are some machine learning algorithms that are better suited for the complex nature of financial time-series data, like CNN-LSTM and BiLSTM-BO-LightGBM. Hybrid models seem to have better predictive and portfolio construction power than individual models.

One limitation of this thesis is that only blue-chips stocks are evaluated. These assets are seen to be safer investments because these companies are usually established and have more steady pricing with less volatility relative to the market. It would be interesting to implement similar algorithms to riskier assets, such as options, derivatives, commodities, or cryptocurrency. With more volatile movement in the market, there is a greater opportunity to generate return, but that comes with higher risk. It is also worth noting that the testing data was during a bull market with an annual S&P 500 return of 24.14%. Even though

[1] as defined in this paper, which can potentially include risk premiums

the S&P 500 return was taken into consideration during analysis in this paper, it would be interesting to determine if the performance is replicated when implementing and analyzing the algorithms during a market environment with lower returns. Another component of algorithmic trading that can be explored is the time complexity of the methods. With faster computing, being ahead of your competitors is incredibly important, and having less expensive computation can make the difference.

In this paper, I have analyzed nine different algorithms for stock selection and portfolio optimization on the 50 largest U.S. companies. Both the accuracy of the stock price forecasting prediction (SMAPE) and the construction of portfolios (α and Sharpe Ratio) were evaluated. The most promising algorithm is the hybrid CNN-LSTM model, followed closely by the other hybrid BiLSTM-BO-LightGBM method.

Appendix A

Selected U.S. Companies for Data

The ticker symbols of the largest 50 United States companies, which were used for data:

Table A.1: The 50 Corporations Used for Data

First column	Second column
AAPL	Apple Inc.
MSFT	Microsoft Corporation
NVDA	NVIDIA Corporation
GOOG	Alphabet Inc.
AMZN	Amazon.com, Inc.
META	Meta Platforms, Inc.
BRK-B	Berkshire Hathaway Inc.
LLY	Eli Lilly and Company
TSLA	Tesla, Inc.
AVGO	Broadcom Inc.
JPM	JPMorgan Chase & Co.
WMT	Walmart Inc.
UNH	UnitedHealth Group Incorporated
XOM	Exxon Mobil Corporation
V	Visa Inc.
MA	Mastercard Incorporated
PG	Procter & Gamble Co.
JNJ	Johnson & Johnson
ORCL	Oracle Corporation

Continued on next page

Table A.1 – continued from previous page

First column	Second column
COST	Costco Wholesale Corporation
HD	The Home Depot, Inc.
ABBV	AbbVie Inc.
BAC	Bank of America Corporation
KO	The Coca-Cola Company
MRK	Merck & Co., Inc.
CVX	Chevron Corporation
NFLX	Netflix, Inc.
CRM	Salesforce, Inc.
ADBE	Adobe Inc.
PEP	PepsiCo, Inc.
TMO	Thermo Fisher Scientific Inc.
TMUS	T-Mobile US, Inc.
AMD	Advanced Micro Devices, Inc.
DHR	Danaher Corporation
MCD	McDonald's Corporation
WFC	Wells Fargo & Company
ABT	Abbott Laboratories
CSCO	Cisco Systems, Inc.
GE	General Electric Company
QCOM	Qualcomm Incorporated
PM	Philip Morris International Inc.
AMGN	Amgen Inc.
INTU	Intuit Inc.
AXP	American Express Company
TXN	Texas Instruments Incorporated
IBM	International Business Machines Corporation
PFE	Pfizer Inc.
VZ	Verizon Communications Inc.
DIS	The Walt Disney Company
NOW	ServiceNow, Inc.

Appendix B

Variable Description

Table B.1: Variable Description

Variable Name	**Type**	**Description**
cusip	Char	Cusip (cusip)
ncusip	Char	Ncusip (ncusip)
comnam	Char	Company Name (comnam)
ticker	Char	Ticker (ticker)
permco	Float	CRSP Permanent Company Number (permco)
shrcd	Float	Share Code (shrcd)
shrcls	Char	Share Class (shrcls)
issuno	Float	Nasdaq Issue Number (issuno)
exchcd	Float	Exchange Code (exchcd)
hexcd	Float	Header Exchange Code (hexcd)
siccd	Float	SIC Code (siccd)
hsiccd	Float	Header SIC Code (hsiccd)
hsicmg	Float	Header SIC Major Group (hsicmg)
hsicig	Float	Header SIC Industry Group (hsicig)
nameendt	Date	Names Ending Date (nameendt)
tsymbol	Char	Trading Symbol (tsymbol)
naics	Char	North American Industry Class System (naics)
primexch	Char	Primary Exchange (primexch)
trdstat	Char	Trading Status (trdstat)
secstat	Char	Security Status (secstat)
prc	Float	Price or Bid/Ask Average (prc)
vol	Float	Volume (vol)
openprc	Float	Price Alternate (openprc)
askhi	Float	Ask or High Price (askhi)
bidlo	Float	Bid or Low Price (bidlo)

Variable Name	Type	Description
bid	Float	Bid (bid)
ask	Float	Ask (ask)
numtrd	Float	Number of Trades (numtrd)
ret	Float	Returns (ret)
retx	Float	Returns without Dividends (retx)
shrout	Float	Number of Shares Outstanding (shrout)
shrflg	Float	Share Flag (shrflg)
shrenddt	Date	Shares Observation End Date (shrenddt)
dlstcd	Float	Delisting Code (dlstcd)
nwperm	Float	New CRSP Permno (nwperm)
nextdt	Date	Date of Next Available Information (nextdt)
dlamt	Float	Amount After Delisting (dlamt)
dlprc	Float	Delisting Price (dlprc)
dlpdt	Date	Date of Delisting Payment (dlpdt)
dlret	Float	Delisting Return (dlret)
dlretx	Float	Delisting Return without Dividends (dlretx)
distcd	Float	Distribution Code (distcd)
divamt	Float	Dividend Cash Amount (divamt)
facpr	Float	Factor to Adjust Price (facpr)
facshr	Float	Factor to Adjust Shares (facshr)
cfacpr	Float	Cumulative Factor to Adjust Price (cfacpr)
cfacshr	Float	Cumulative Factor to Adjust Shares (cfacshr)
dclrdt	Date	Declaration Date (dclrdt)
rcrddt	Date	Record Date (rcrddt)
paydt	Date	Payment Date (paydt)
acperm	Float	Acquiring PERMNO (acperm)
accomp	Float	Acquiring PERMCO (accomp)
trtscd	Float	Traits Code (trtscd)
nmsind	Float	National Market Indicator (nmsind)
mmcnt	Float	Market Maker Count (mmcnt)
nsdinx	Float	NASD Index (nsdinx)
vwretd	Float	Value-Weighted Return (includes distributions) (vwretd)
vwretx	Float	Value-Weighted Return (excluding dividends) (vwretx)
ewretd	Float	Equal-Weighted Return (includes distributions) (ewretd)
ewretx	Float	Equal-Weighted Return (excluding dividends) (ewretx)
sprtrn	Float	Return on S&P Composite Index (sprtrn)

Appendix C

Summary Statistics of Data

Variable	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
BIDLO	5.48	93.87	149.53	254.36	272.64	3696.79
ASKHI	5.66	96.22	152.68	260.60	279.50	3773.08
PRC	5.49	95.01	151.07	257.57	276.24	3731.41
VOL	305728	3191884	6360114	14654760	15745683	401048654
BID	5.49	95.03	151.07	257.53	276.20	3731.41
ASK	5.50	95.04	151.09	257.61	276.26	3733.39
SHROUT	172602	749748	1389241	2336796	2831953	17102536
CFACPR	0.1609	1.0000	1.0000	1.7005	1.0000	20.0000
OPENPRC	5.61	95.02	151.07	257.51	276.42	3744.00
vwretd	-0.1182	-0.0052	0.0009	0.0006	0.0071	0.0916
vwretx	-0.1182	-0.0053	0.0009	0.0005	0.0071	0.0915
ewretd	-0.1076	-0.0059	0.0006	0.0005	0.0073	0.0822
ewretx	-0.1078	-0.0059	0.0005	0.0004	0.0072	0.0820
sprtrn	-0.1198	-0.0051	0.0009	0.0006	0.0072	0.0938

Table C.1: Summary Statistics

Appendix D

CNN-LSTM detailed process

The main steps for the CNN-LSTM training and prediction are as follows [44]:

1. Input standardized training data using the z-score standardization method
2. Initialize the weights and biases of each layer in the CNN-LSTM
3. Calculate the CNN layer: the input data is passed through the CNN and the feature extraction is obtained
4. Calculate the LSTM layer: the output data of the CNN is passed through the LSTM layer to get an output value
5. Output layer calculation: the output value calculated by the output layer is input into the full connection layer to get an output value
6. Calculate the error of the the output layer
7. Continue backpropagation until the the set parameters are satisfied
8. Save the model and input the standardized testing data
9. Forecast on the testing data and output the result to get a prediction using CNN-LSTM

References

[1] D. Neufeld. *The $109 Trillion Global Stock Market in One Chart*. Visual Capatalist, Sept. 2023. URL: https://www.visualcapitalist.com/the-109-trillion-global-stock-market-in-one-chart/.

[2] J. H. Cochrane. *Eugene F. Fama, Efficient Markets, and the Nobel Prize*. Chicago Booth, May 2014. URL: https://www.chicagobooth.edu/review/eugene-fama-efficient-markets-and-the-nobel-prize.

[3] E. Reed. *A Guide to Portfolio Optimization Strategies*. Smart Asset, Mar. 2023. URL: https://smartasset.com/investing/guide-portfolio-optimization-strategies.

[4] T. Phuoc, P. T. K. Anh, P. H. Tam, and C. V. Nguyen. *Applying machine learning algorithms to predict the stock price trend in the stock market - The case of Vietnam*. Nature, Mar. 2024. URL: https://www.nature.com/articles/s41599-024-02807-x.

[5] K. Didur. *Machine learning in finance: Why, what how*. Medium, July 2018. URL: https://towardsdatascience.com/machine-learning-in-finance-why-what-how-d524a2357b56.

[6] F. Patrawala. *How does bias in machine learning affect algorithmic trading?* Medium, Jan. 2020. URL: https://medium.com/@fatshusami1/how-does-bias-in-machine-learning-affect-algorithmic-trading-cef4d830a127.

[7] P. Carpenter. *AI may revolutionize security, but not without human intuition*. Security Magazine, Feb. 2024. URL: https://www.securitymagazine.com/articles/100378-ai-may-revolutionize-security-but-not-without-human-intuition.

[8] *Trading & Data*. NYSE, May 2024. URL: https://www.nyse.com/trading-data.

[9] *Daily Market Summary*. Nasdaq, May 2024. URL: https://www.nasdaqtrader.com/Trader.aspx?id=DailyMarketSummary.

[10] D. Wolff and F. Echterling. *Stock picking with machine learning*. Wiley, Sept. 2023. URL: https://onlinelibrary.wiley.com/doi/full/10.1002/for.3021.

[11] G. Smith. *Modern Portfolio Theory: What Is It?* US News, July 2023. URL: https://money.usnews.com/investing/articles/modern-portfolio-theory-what-is-it.

[12] R. Baldridge. *Understanding Modern Portfolio Theory*. Forbes, June 2024. URL: https://www.forbes.com/advisor/investing/modern-portfolio-theory/.

[13] L. Rodini. *What Is Alpha in Finance? Definition, Formula Examples*. The Street, Feb. 2023. URL: https://www.thestreet.com/dictionary/alpha.

[14] W. Kenton. *What Beta Means for Investors*. Investopedia, June 2024. URL: https://www.investopedia.com/terms/b/beta.asp.

[15] L. F. Torres. *Portfolio Optimization: The Markowitz Mean-Variance Model*. Medium, Apr. 2023. URL: https://medium.com/latinxinai/portfolio-optimization-the-markowitz-mean-variance-model-c07a80056b8a.

[16] S. Indrees, M. A. Alam, and P. Agarwal. *A Prediction Approach for Stock Market Volatility Based on Time Series Data*. IEEE, Feb. 2019. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8626097.

[17] K. Pahwa and N. Agarwal. *Stock Market Analysis using Supervised Machine Learning*. Amity University, Feb. 2019. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8862225.

[18] M. Misra, A. Yadav, and H. Kaur. *Stock Market Prediction using Machine Learning Algorithms: A Classification Study*. Thapar Institute of Engineering and Technology, July 2018. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9009178.

[19] S. Ravikumar and P. Saraf. *Prediction of Stock Prices using Machine Learning (Regression,Classification) Algorithms*. Vishwakarma Institute of Technology, June 2020. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9154061.

[20] I. Kumar, K. Dogra, C. Utreja, and P. Yadav. *A comparative study of supervised machine learning algorithms for stock market trend prediction*. NIT Kurukshetra, July 2018. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8473214.

[21] V. Ingle and S. Deshmukh. *Hidden Markov Model Implementation for Prediction of Stock Prices with TF-IDF features*. Dr.B.A.Mm University, Aug. 2016. URL: https://dl.acm.org/doi/pdf/10.1145/2979779.2979788.

[22] S. Singh, T. Madan, J. Kumar, and A. Singh. *Stock Market Forecasting using Machine Learning: Today and Tomorrow*. National Institute of Technology, June 2019. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8993160.

[23] R. Kim, C. H. So, and M. Jeong. *HATS: A Hierarchical Graph Attention Network for Stock Movement Prediction*. Korea University, Jan. 2019. URL: https://arxiv.org/pdf/1908.07999.

[24] L. Xingzhou, R. Hong, and Z. Yujun. *Predictive Modeling of Stock Indexes Using Machine Learning and Information Theory*. In Proceedings of the 2019 10th International Conference on E-business, Management and Economics, July 2019. URL: https://www.researchgate.net/publication/336201707_Predictive_Modeling_of_Stock_Indexes_Using_Machine_Learning_and_Information_Theory.

[25] S. Sarode, H. Tolani, and P. Kak. *Stock Price Prediction Using Machine Learning Techniques*. 2019 International Conference on Intelligent Sustainable Systems (ICISS), Feb. 2019. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8907958.

[26] Y. Song and J. Lee. *Design of stock price prediction model with various configuration of input features*. In Proceedings of the International Coference on Artificial Intelligence, Information Processing and Cloud Computing, Dec. 2019. URL: https://dl.acm.org/doi/10.1145/3371425.3371432.

[27] P. Werawithayaset and S. Trilanunt. *Stock Closing Price Prediction Using Machine Learning*. In 2019 17th International Conference on ICT and Knowledge Engineering, Nov. 2019. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8966836.

[28] G.-Y. Ban, N. E. Karoui, and A. Lim. *Machine Learning and Portfolio Optimization*. London Business School, Nov. 2016. URL: https://pubsonline.informs.org/doi/pdf/10.1287.

[29] S. Pafka and I. Kondor. *Estimated correlation matrices and portfolio optimization*. Collegium Budapest - Institute for Advanced Study, Jan. 2004. URL: https://www.sciencedirect.com/science/article/abs/pii/S0378437104007447?via%3Dihub.

[30] T. Cura. *Particle swarm optimization approach to portfolio optimization*. Istanbul University, Apr. 2008. URL: https://www.sciencedirect.com/science/article/abs/pii/S1468121808001259.

[31] P. Jorion. *Portfolio Optimization with Tracking-Error Constraints*. University of California at Irvine, Jan. 2003. URL: https://www.tandfonline.com/doi/abs/10.2469.

[32] E. Birken. *The Russell 1000 Index*. Forbes, June 2022. URL: https://www.forbes.com/advisor/investing/russell-1000-index/.

[33] S. Almanhdi and S. Yang. *An adaptive portfolio trading system: A risk-return portfolio optimization using recurrent reinforcement learning with expected maximum drawdown*. Stevens Institute of Technology, Nov. 2017. URL: https://www.sciencedirect.com/science/article/pii/S0957417417304402.

[34] R. Abrami and B. Marsoem. *Optimal Portfolio Formation with Single Index Model Approach on Lq-45 Stocks on Indonesia Stock Exchange*. University of Mercu Buana, Mar. 2021. URL: https://ijisrt.com/optimal-portfolio-formation-with-single-index-model-approach-on-lq45-stocks-on-indonesia-stock-exchange.

[35] W. Wuyu, L. Weizi, Z. Ning, and L. Kecheng. *Portfolio formation with preselection using deep learning from long-term financial data*. Central University of Finance and Economics, Apr. 2020. URL: https://www.sciencedirect.com/science/article/abs/pii/S0957417419307596.

[36] Y. Ma, R. Han, and W. Wang. *Portfolio optimization with return prediction using deep learning and machine learning*. Southeast University, Mar. 2021. URL: https://www.sciencedirect.com/science/article/abs/pii/S0957417420307521.

[37] C. O'Sullivan. *A Step-By-Step Introduction to PCA*. Toward Data Science, Apr. 2020. URL: https://towardsdatascience.com/a-step-by-step-introduction-to-pca-c0d78e26a0dd.

[38] A. Saini. *Guide on Support Vector Machine (SVM) Algorithm*. Analytics Vidhya, May 2024. URL: https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/.

[39] S. Rigatti. *Random Forest*. Journal of Insurance Medicine, Jan. 2017. URL: https://doi.org/10.17849/insm-47-01-31-39.1.

[40] G. Kanagachindambaresan, A. Ruwali, D. Banerjee, and K. Prakash. *Recurrent Neural Network*. Springer Innovations, Jan. 2021. URL: https://link.springer.com/chapter/10.1007/978-3-030-57077-4_7.

[41] A. Graves. *Long Short-Term Memory*. Studies in Computation Intelligence, Jan. 2012. URL: https://link.springer.com/chapter/10.1007/978-3-642-24797-2_4.

[42] A. Taparia. *Bidirectional LSTM in NLP*. GeeksforGeeks, June 2023.

[43] G. Ke, Q. Meng, T. Finely, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. *LightGBM: A Highly Efficient Gradient Boosting Decision Tree*. Microsoft, Oct. 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

[44] W. Lu, J. Li, Y. Li, A. Sun, and J. Wang. *A CNN-LSTM-Based Model to Forecast Stock Prices*. Jiangsu Second Normal University, Nov. 2020. URL: https://onlinelibrary.wiley.com/doi/epdf/10.1155/2020/6622927.

[45] L. Tian, L. Feng, L. Yang, and Y. Guo. *Stock price prediction based on LSTM and LightGBM hybrid model*. The Journal of Supercomputing, Feb. 2022. URL: https://doi.org/10.1007/s11227-022-04326-5.

[46] V.-D. Ta, C.-M. Liu, and D. Tadesse. *Portfolio Optimization-Based Stock Prediction Using Long-Short Term Memory Network in Quantitative Trading*. Taipei University of Technology, Jan. 2020. URL: https://www.mdpi.com/2076-3417/10/2/437.

[47] A. Damodaran. *Equity Risk Premiums: Determinants, Estimation and Implications - The 2020 Edition*. NYU Stern, Mar. 2020. URL: http://dx.doi.org/10.2139/ssrn.3550293.