# A NOVEL PREDICTION BASED PORTFOLIO OPTIMIZATION MODEL USING DEEP LEARNING

*Report submitted to the SASTRA Deemed to be University*
*as the requirement for the course*

**INT424: Algorithmic Trading**

*Submitted by*

**NARAYANAN G**
**(Reg.no 124150030)**

**May 2023**

**SCHOOL OF ARTS, SCIENCES, HUMANITIES & EDUCATION**

**THANJAVUR, TAMIL NADU, INDIA – 613 401**

SASTRA

ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION

DEEMED TO BE UNIVERSITY

(U/S 3 of the UGC Act,1956)

THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

THANJAVUR | KUMBAKONAM | CHENNAI

**SCHOOL OF ARTS, SCIENCES, HUMANITIES & EDUCATION**
**THANJAVUR, TAMIL NADU, INDIA – 613 401**

# Bonafide Certificate

This is to certify that the report titled "A Novel Prediction Based Portfolio Optimization Model Using Deep Learning**"** submitted as a requirement for the course, **INT424 ALGORITHMIC TRADING** for M.Sc. Data Science programme, is a bonafide record of the work done by **Mr. NARAYANAN G (124150030)** during the academic year 2022-23, in the School of Arts Science and Humanities & Education, under my supervision.

| | |
|---|---|
| **Signature of Project Supervisor** | **:** |
| **Name with Affiliation** | **: Dr.Ashok Palaniappan P.hd** |
| **Date** | **: 29.05.2023** |

Project *Viva voc*e held on 29.05.2023

**Examiner 1** **Examiner 2**

SASTRA

ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION

DEEMED TO BE UNIVERSITY

(U/S 3 of the UGC Act,1956)

THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

THANJAVUR | KUMBAKONAM | CHENNAI

**SCHOOL OF ARTS, SCIENCES, HUMANITIES & EDUCATION**
**THANJAVUR, TAMIL NADU, INDIA – 613 401**

## Bonafide Certificate

This is to certify that the thesis titled "A Novel Prediction Based Portfolio Optimization Model Using Deep Learning**"** submitted in partial fulfillment of the requirements for the award of the degree of M.Sc. Data Science to the SASTRA Deemed to be University, is a bonafide record of the work done by **Mr. NARAYANAN G** (Reg. No. 124150030) during the final semester of the academic year 2022-23, in the **School of Arts Science and Humanities & Education**, under my supervision. This has not formed the basis for the award of any degree, diploma, associateship, fellowship or other similar title to any candidate of any University.

**Signature of Project Supervisor :**
**Name with Affiliation : Dr.Ashok Palaniappan P.hd**
**Date : 29.05.2023**

Project *Viva voc*e held on 29.05.2023

**Examiner 1** **Examiner 2**

SASTRA

ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION

DEEMED TO BE UNIVERSITY

(U/S 3 of the UGC Act, 1956)

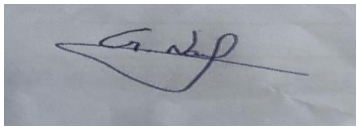
THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

THANJAVUR | KUMBAKONAM | CHENNAI

**SCHOOL OF ARTS, SCIENCES, HUMANITIES & EDUCATION**
**THANJAVUR, TAMIL NADU, INDIA – 613 401**

## Declaration

I/We declare that the thesis titled "A Novel Prediction Based Portfolio Optimization Model Using Deep Learning" submitted by me is an original work done by me/us under the guidance of **Dr. Ashok Palaniappan P.hd, Associate Professor, School of Chemical and Biotechnology, SASTRA Deemed to be University** during the second semester of the academic year 2022 - 2023, in the **School of Arts Science, Humanities & Education**. The work is original and wherever I/We have used materials from other sources, I/We have given due credit and cited them in the text of the thesis. This thesis has not formed the basis for the award of any degree, diploma, associate-ship, fellowship or other similar title to any candidate of any University.

**Signature of the candidate(s) :** 

**Name of the candidate(s) : NARAYANAN G**

**Date : 29.05.2023**

# Table of Contents

**INDEX**

# ABSTRACT

Portfolio optimization is an important part of portfolio management. It realizes the trade-off between maximizing expected return and minimizing risk. A better portfolio optimization model helps investors achieve higher expected returns under the same risk level.

This paper proposes a novel prediction-based portfolio optimization model. This model uses an autoencoder (AE) for feature extraction and long short term memory (LSTM) network to predict stock return, then predicted and historical returns are utilized to build a portfolio optimization model by advancing worst-case omega model.

To show the effect of AE, the LSTM network without any feature extraction methods is used as a benchmark in stock prediction. Also, an equally weighted portfolio is considered as a comparison to reveal the advantage of the worst-case omega model.

Empirical results show that the proposed model significantly outperforms the equally weighted portfolio, and a high risk–return preference is more suitable to this model. In addition, even after deducting transaction fees, this model still achieves a satisfying return and performs better than the state-of-art prediction-based portfolio optimization models.

# Chapter 1

# INTRODUCTION

The introduction provides an overview of portfolio management and the importance of prediction-based portfolio optimization models in improving the performance of investment portfolios. It mentions the use of machine learning models such as support vector regression (SVR), random forest (RF), and artificial neural networks (ANNs) in stock prediction and their potential integration with classical portfolio optimization models.

We are discuss three research directions that have been explored to enhance classical portfolio optimization models using ML models:

1. Using ML models for stock preselection before applying classical portfolio optimization models.
2. Using ML models to predict future stock returns and incorporating the predicted information into objective functions of classical portfolio optimization models.
3. Utilizing ML models for stock return prediction and using predictive errors instead of historical returns in building portfolio optimization models.

To further improve the out-of-sample performance of prediction-based portfolio optimization models, the paper proposes a new model called AE+LSTM+OMEGA.

This model combines an autoencoder (AE) for feature extraction, a long short-term memory (LSTM) network for stock return prediction, and the worst-case omega model for portfolio optimization.

It applies AE for feature extraction, uses LSTM with extracted features to predict stock returns, selects stocks with higher predicted returns, and incorporates predictive errors in the worst-case omega model.

The model generates multiple objective functions based on predicted and historical returns and converts them into a single objective using the linearity weighted method.

The contributions of the paper are highlighted, including the application of LSTM in stock prediction with AE-extracted features, the use of predictive errors in the worst-case omega model, the addition of four objective functions based on predicted and historical returns, and the outperformance of the proposed model compared to existing prediction-based portfolio optimization models.

In summary, the paper aims to develop an improved prediction-based portfolio optimization model that integrates AE, LSTM, and the worst-case omega model. The model enhances stock selection, incorporates predictive errors, generates multiple objective functions, and demonstrates superior performance compared to existing models.

## 1.1 Problem Statement

The main problem is to find optimal portfolio by selecting stocks in stock market analysis is the difficulty in predicting how the market will behave.so we compare the portfolio with method proposed portfolio and we conclude which one is best among those portfolio . Traditional methods of portfolio is to find individual return and individual risk of particular stock and we have to find portfolio risk and portfolio return for overall stocks.

**Objectives***:* The primary objectives of this project are Optimal Portfolio Optimization .

## 1.2 Portfolio Optimization

The project aims to optimize portfolios by determining the optimal weights for a given set of stocks. This is achieved through the implementation of the Sequential Least Squares Programming (SLSQP) optimization algorithm. The objective function used is the negative Sharpe ratio, which measures the risk-adjusted return of the portfolio. By maximizing the Sharpe ratio, the project seeks to identify the optimal allocation of weights that balances the expected return and risk of the portfolio.

## 1.3 AE+LSTM+OMEGA

the concepts of Autoencoder (AE), Long Short-Term Memory (LSTM), and OMEGA briefly, along with their formulas:

### 1.3.1 Autoencoder (AE):

An autoencoder is a type of artificial neural network used for unsupervised learning and dimensionality reduction.

It consists of an encoder and a decoder. The encoder compresses the input data into a lower-dimensional representation, while the decoder reconstructs the original input from the compressed representation.

The goal of an autoencoder is to minimize the reconstruction error between the input and the output.

**Formula:**

Encoder: encoded_data = Encoder(input_data)

Decoder: reconstructed_data = Decoder(encoded_data)

### 1.3.2 Long Short-Term Memory (LSTM):

LSTM is a type of recurrent neural network (RNN) architecture designed to overcome the vanishing gradient problem and effectively model sequences.

It has memory cells and gates that allow it to capture long-term dependencies and process sequential data efficiently.

LSTM cells have input gates, forget gates, and output gates, which control the flow of information in and out of the cell.

The LSTM model takes a sequence of inputs and predicts the next element in the sequence or performs other sequence-based tasks.

Formula:

LSTM cell: h_t = LSTM(x_t, h_{t-1}, c_{t-1})
x_t: Input at time step t
h_t: Hidden state/output at time step t
c_t: Cell state at time step t
LSTM(x_t, h_{t-1}, c_{t-1}) represents the LSTM cell operation.

### 1.3.3 OMEGA:

OMEGA is a combination of an autoencoder and an LSTM model. The autoencoder compresses the input data into a lower-dimensional representation, which is then fed into the LSTM model for sequence prediction.

OMEGA learns to predict the future sequence based on the encoded representation of the input data.

Formula:

Encoded data: encoded_data = Autoencoder(input_data)
LSTM prediction: predicted_data = LSTM(encoded_data)

In this project , the Autoencoder (AE) is used to compress the input stock price data into a lower-dimensional representation. The LSTM model takes this encoded representation and predicts the future stock prices. The OMEGA model combines the AE and LSTM by using the encoded data as input to the LSTM for sequence prediction.

# Chapter 2

# METHODOLOGY

To achieve the objective function, we used three methods

1. Autoencoder
2. Long Shot Term Memory
3. omega

## 2.1 Method for This Project:

The given code appears to be a Python script that calculates the risk and return for multiple portfolios consisting of different stocks. Let's go through the code step by step:

The script begins with importing necessary libraries, such as `statistics`, `yfinance`, `numpy`, `matplotlib.pyplot`, `random`, `pandas`, `yahoo_fin`, `sklearn.preprocessing`, and `tensorflow.keras`. These libraries are commonly used for data analysis, financial calculations, and machine learning.

It defines a class called `Portfolio` that represents a portfolio of stocks. The class has methods to calculate the risk and return for individual stocks (`risk_return_individual()`), portfolio return (`portfolio_return()`), and portfolio risk (`portfolio_risk()`).

The `risk_return_individual()` method takes in the ticker symbols of the stocks in the portfolio, downloads the historical price data using `yfinance`, calculates the daily returns for each stock, and then calculates the expected returns and risk (standard deviation of returns) for each stock. It stores the individual returns and risks in `indi_ret` and `indi_risk` lists, respectively.

The `portfolio_return()` method calculates the expected return for the portfolio by multiplying the weights of each stock with their respective expected returns and summing them up. The portfolio return is stored in the `port_ret` list.

The `portfolio_risk()` method calculates the portfolio risk using the formula: sqrt(w1^2

* σ1^2 + w2^2 * σ2^2 + ... + 2 * w1 * w2 * Cov1,2 + ...), where wi is the weight of the ith stock, σi is the risk (standard deviation) of the ith stock, and Cov1,2 is the covariance between stocks 1 and 2. It uses a covariance matrix to calculate the covariances between stocks. The portfolio risk is stored in the `portfo_risk` list.

The code then prompts the user to enter the number of portfolios to be evaluated (`get_por`). For each portfolio, it asks for the number of companies in the portfolio (`n_companies`), the ticker symbols of the companies, and the weights assigned to each company. It creates a `Portfolio` object for each portfolio and calculates the portfolio return and risk.

Finally, the code creates separate pandas DataFrames (`p`, `p1`, `p2`, `p3`) to store the individual risks, individual returns, portfolio returns, and portfolio risks, respectively. These DataFrames can be used for further analysis or visualization.

It's worth noting that the code provided may require additional error handling and modifications to ensure it runs smoothly and produces accurate results.

The additional code provided appears to be performing stock price prediction using an LSTM-based autoencoder model. Let's go through the code step by step:

1.The code starts by initializing an empty list `n_com` and an empty list `tc`.

2. It prompts the user to enter the number of ticker symbols (`c`) and then asks for the ticker symbols of the companies one by one. The ticker symbols are stored in the `tc` list.

3. Next, the code sets the start and end dates for downloading the historical stock price data.

4. For each ticker symbol in the `tc` list, the code uses the `yfinance` library to download the historical price data between the specified start and end dates. The downloaded data is then saved to a CSV file named 'stock.csv'.

5. After downloading the data for all ticker symbols, the code proceeds to load the stock data from the 'stock.csv' file using pandas.

6. The code then imports necessary libraries, such as `numpy`, `pandas`, `sklearn.preprocessing`, `tensorflow.keras.models`, `tensorflow.keras.layers`, and `datetime`.

7. Preprocessing of the data begins by scaling the 'Close' column values using the MinMaxScaler from `sklearn.preprocessing`.

8. The data is split into training and testing sets, where 80% of the data is used for training.

9. The training data is reshaped to have a shape of (number of samples, 1) to fit the input shape requirements of the LSTM model.

10. The autoencoder model is defined using `tensorflow.keras.models.Sequential`. It consists of a single dense layer for encoding and another dense layer for decoding.

11. The Omega model is defined by creating an input layer (`autoencoder_input`) and connecting it to the encoded data obtained from the autoencoder model.

12. The LSTM model is defined using `tensorflow.keras.models.Sequential` and consists of an LSTM layer followed by a dense layer for prediction.

13. The model is compiled with the Adam optimizer and mean squared error (MSE) loss.

14. The model is trained by fitting the training data to itself for a specified number of epochs.

15. The input data for the LSTM model is created by repeating the encoded data for each timestep using `RepeatVector`. The predicted data is then obtained by passing the LSTM input through the model.

16. The Omega model is created by defining the input and output layers and compiling it with the Adam optimizer and mean squared error loss. It is then trained using the training data, excluding the timesteps, and the target data, starting from the timesteps.

17. Predictions are generated on the test data by first encoding the test data using the autoencoder and then passing it through the LSTM model. The predicted data is reshaped to (-1, 1).

18. The predicted data is scaled back to the original range using the `inverse_transform` method of the MinMaxScaler.

19. The difference between the actual test data and the predicted test data is calculated, flattened, and stored in the variable `D`.

20. The model is evaluated by calculating the mean squared error (MSE) between the actual and predicted test data.

21. The MSE value is printed as "Mean Squared Error".

Overall, the code performs stock price prediction using an LSTM-based autoencoder and LSTM model. The autoencoder is used to learn a compressed representation of the input data, and the LSTM model is used for sequence prediction based on the encoded data. The model is evaluated using mean squared error to assess its prediction performance.

# Chapter 3

# IMPLEMENTATION

## 3.1 About the data

Based on the code provided earlier, the data used for analysis is historical stock price data. The code downloads the daily closing prices for a set of ticker symbols from the Yahoo Finance API using the ‘yfinance’ library in python. We are getting the ticker from user to check with portfolio is best .The some of the ticker symbols used in the code are: IBM , GOOG (Google),. These ticker symbols represent a diversified set of companies from different sectors.

# Chapter 4

# RESULTS & DISCUSSION

**4.1 portfolio for IBM & GOOG:** The normal portfolio of IBM and GOOG are produce some individual risk, individual return, portfolio risk and portfolio return. Based on weights for each stock in portfolio .

```
enter the no of portfolio count: 1
Enter the number of companies in your portfolio : 2
enter a company name : ibm
enter a company name : goog
Enter your weights of the  companies ibm : 0.8
Enter your weights of the  companies goog : 0.2
[**********************100%**********************]  1 of 1 completed
[**********************100%**********************]  1 of 1 completed

Risk of the individual companies :  [1.591392178861302, 1.9439851175631495]
Portfolio expected return is :  0.045346600645564686
Portfolio risk is : 1.4603275810043541
                    individual risk
0  [[1.591392178861302, 1.9439851175631495]]
                     individual return
0  [[0.031269421373339906, 0.1016553177344638]]
          portfolio ret
0  [0.045346600645564686]
         portfolio risk
0  [1.4603275810043541]
```

These weights indicate the proportion of each stock's contribution to the overall portfolio.

**4.2 Portfolio Expected Return and Risk by using AE+LSTM+OMEGA:** The expected return of the portfolio is estimated to be approximately 2.42%. This value represents the average potential gain of the portfolio based on the chosen allocation. The portfolio risk is measured by the standard deviation, and it is estimated to be approximately 15.35%. This metric provides an indication of the potential volatility or variability in the portfolio's value.

```
Enter the number of ticker: 2
enter a company name : IBM
enter a company name : goog
['IBM', 'goog']
[**********************100%**********************]  1 of 1 completed
```

```
[**********100%**********]  1 of 1 completed
Epoch 1/100
19/19 [==============================] - 1s 3ms/step - loss: 0.3338
Epoch 2/100
19/19 [==============================] - 0s 3ms/step - loss: 0.2704
Epoch 3/100
19/19 [==============================] - 0s 2ms/step - loss: 0.2110
Epoch 4/100
19/19 [==============================] - 0s 2ms/step - loss: 0.1538
Epoch 5/100
19/19 [==============================] - 0s 2ms/step - loss: 0.1044
Epoch 6/100
19/19 [==============================] - 0s 2ms/step - loss: 0.0649
Epoch 7/100
19/19 [==============================] - 0s 2ms/step - loss: 0.0416
Epoch 8/100
19/19 [==============================] - 0s 2ms/step - loss: 0.0298
Epoch 9/100
19/19 [==============================] - 0s 2ms/step - loss: 0.0254
Epoch 10/100
19/19 [==============================] - 0s 2ms/step - loss: 0.0235
Epoch 11/100
19/19 [==============================] - 0s 2ms/step - loss: 0.0219
Epoch 12/100
19/19 [==============================] - 0s 2ms/step - loss: 0.0203
Epoch 13/100
19/19 [==============================] - 0s 2ms/step - loss: 0.0186
Epoch 14/100
19/19 [==============================] - 0s 2ms/step - loss: 0.0170
Epoch 15/100
19/19 [==============================] - 0s 2ms/step - loss: 0.0154
Epoch 16/100
19/19 [==============================] - 0s 2ms/step - loss: 0.0138
Epoch 17/100
19/19 [==============================] - 0s 2ms/step - loss: 0.0122
Epoch 18/100
19/19 [==============================] - 0s 2ms/step - loss: 0.0106
Epoch 19/100
19/19 [==============================] - 0s 2ms/step - loss: 0.0092
Epoch 20/100
19/19 [==============================] - 0s 2ms/step - loss: 0.0078
Epoch 21/100
19/19 [==============================] - 0s 3ms/step - loss: 0.0065
Epoch 22/100
```

```
19/19 [==============================] - 0s 2ms/step - loss: 0.0054
Epoch 23/100
19/19 [==============================] - 0s 2ms/step - loss: 0.0044
Epoch 24/100
19/19 [==============================] - 0s 2ms/step - loss: 0.0035
Epoch 25/100
19/19 [==============================] - 0s 3ms/step - loss: 0.0028
Epoch 26/100
19/19 [==============================] - 0s 2ms/step - loss: 0.0022
Epoch 27/100
19/19 [==============================] - 0s 2ms/step - loss: 0.0017
Epoch 28/100
19/19 [==============================] - 0s 2ms/step - loss: 0.0013
Epoch 29/100
19/19 [==============================] - 0s 2ms/step - loss: 0.0011
Epoch 30/100
19/19 [==============================] - 0s 2ms/step - loss: 8.5823e-04
Epoch 31/100
19/19 [==============================] - 0s 3ms/step - loss: 7.1021e-04
Epoch 32/100
19/19 [==============================] - 0s 2ms/step - loss: 5.9823e-04
Epoch 33/100
19/19 [==============================] - 0s 2ms/step - loss: 5.2053e-04
Epoch 34/100
19/19 [==============================] - 0s 2ms/step - loss: 4.6469e-04
Epoch 35/100
19/19 [==============================] - 0s 3ms/step - loss: 4.2695e-04
Epoch 36/100
19/19 [==============================] - 0s 2ms/step - loss: 3.9468e-04
Epoch 37/100
19/19 [==============================] - 0s 2ms/step - loss: 3.7192e-04
Epoch 38/100
19/19 [==============================] - 0s 3ms/step - loss: 3.5531e-04
Epoch 39/100
19/19 [==============================] - 0s 2ms/step - loss: 3.3870e-04
Epoch 40/100
19/19 [==============================] - 0s 2ms/step - loss: 3.2685e-04
Epoch 41/100
19/19 [==============================] - 0s 2ms/step - loss: 3.1626e-04
Epoch 42/100
19/19 [==============================] - 0s 3ms/step - loss: 3.0733e-04
Epoch 43/100
19/19 [==============================] - 0s 2ms/step - loss: 2.9807e-04
Epoch 44/100
```

```
19/19 [==============================] - 0s 2ms/step - loss: 2.9091e-04
Epoch 45/100
19/19 [==============================] - 0s 2ms/step - loss: 2.8467e-04
Epoch 46/100
19/19 [==============================] - 0s 3ms/step - loss: 2.7854e-04
Epoch 47/100
19/19 [==============================] - 0s 3ms/step - loss: 2.7311e-04
Epoch 48/100
19/19 [==============================] - 0s 3ms/step - loss: 2.6842e-04
Epoch 49/100
19/19 [==============================] - 0s 2ms/step - loss: 2.6350e-04
Epoch 50/100
19/19 [==============================] - 0s 2ms/step - loss: 2.6098e-04
Epoch 51/100
19/19 [==============================] - 0s 2ms/step - loss: 2.5635e-04
Epoch 52/100
19/19 [==============================] - 0s 2ms/step - loss: 2.5286e-04
Epoch 53/100
19/19 [==============================] - 0s 2ms/step - loss: 2.4998e-04
Epoch 54/100
19/19 [==============================] - 0s 2ms/step - loss: 2.4700e-04
Epoch 55/100
19/19 [==============================] - 0s 2ms/step - loss: 2.4540e-04
Epoch 56/100
19/19 [==============================] - 0s 2ms/step - loss: 2.4159e-04
Epoch 57/100
19/19 [==============================] - 0s 2ms/step - loss: 2.3840e-04
Epoch 58/100
19/19 [==============================] - 0s 3ms/step - loss: 2.3649e-04
Epoch 59/100
19/19 [==============================] - 0s 2ms/step - loss: 2.3337e-04
Epoch 60/100
19/19 [==============================] - 0s 2ms/step - loss: 2.3152e-04
Epoch 61/100
19/19 [==============================] - 0s 2ms/step - loss: 2.3003e-04
Epoch 62/100
19/19 [==============================] - 0s 2ms/step - loss: 2.2724e-04
Epoch 63/100
19/19 [==============================] - 0s 3ms/step - loss: 2.2546e-04
Epoch 64/100
19/19 [==============================] - 0s 2ms/step - loss: 2.2269e-04
Epoch 65/100
19/19 [==============================] - 0s 2ms/step - loss: 2.2225e-04
Epoch 66/100
```

```
19/19 [==============================] - 0s 2ms/step - loss: 2.2034e-04
Epoch 67/100
19/19 [==============================] - 0s 2ms/step - loss: 2.1649e-04
Epoch 68/100
19/19 [==============================] - 0s 2ms/step - loss: 2.1583e-04
Epoch 69/100
19/19 [==============================] - 0s 2ms/step - loss: 2.1368e-04
Epoch 70/100
19/19 [==============================] - 0s 3ms/step - loss: 2.1189e-04
Epoch 71/100
19/19 [==============================] - 0s 2ms/step - loss: 2.1043e-04
Epoch 72/100
19/19 [==============================] - 0s 2ms/step - loss: 2.0922e-04
Epoch 73/100
19/19 [==============================] - 0s 2ms/step - loss: 2.0965e-04
Epoch 74/100
19/19 [==============================] - 0s 2ms/step - loss: 2.0946e-04
Epoch 75/100
19/19 [==============================] - 0s 2ms/step - loss: 2.0573e-04
Epoch 76/100
19/19 [==============================] - 0s 3ms/step - loss: 2.0415e-04
Epoch 77/100
19/19 [==============================] - 0s 2ms/step - loss: 2.0168e-04
Epoch 78/100
19/19 [==============================] - 0s 2ms/step - loss: 2.0097e-04
Epoch 79/100
19/19 [==============================] - 0s 2ms/step - loss: 1.9850e-04
Epoch 80/100
19/19 [==============================] - 0s 2ms/step - loss: 1.9805e-04
Epoch 81/100
19/19 [==============================] - 0s 2ms/step - loss: 1.9698e-04
Epoch 82/100
19/19 [==============================] - 0s 2ms/step - loss: 1.9892e-04
Epoch 83/100
19/19 [==============================] - 0s 2ms/step - loss: 1.9436e-04
Epoch 84/100
19/19 [==============================] - 0s 3ms/step - loss: 1.9178e-04
Epoch 85/100
19/19 [==============================] - 0s 3ms/step - loss: 1.9080e-04
Epoch 86/100
19/19 [==============================] - 0s 3ms/step - loss: 1.8990e-04
Epoch 87/100
19/19 [==============================] - 0s 3ms/step - loss: 1.8739e-04
Epoch 88/100
```

```
19/19 [==============================] - 0s 3ms/step - loss: 1.8604e-04
Epoch 89/100
19/19 [==============================] - 0s 2ms/step - loss: 1.8474e-04
Epoch 90/100
19/19 [==============================] - 0s 2ms/step - loss: 1.8237e-04
Epoch 91/100
19/19 [==============================] - 0s 2ms/step - loss: 1.8140e-04
Epoch 92/100
19/19 [==============================] - 0s 2ms/step - loss: 1.8003e-04
Epoch 93/100
19/19 [==============================] - 0s 2ms/step - loss: 1.7918e-04
Epoch 94/100
19/19 [==============================] - 0s 2ms/step - loss: 1.7748e-04
Epoch 95/100
19/19 [==============================] - 0s 2ms/step - loss: 1.7528e-04
Epoch 96/100
19/19 [==============================] - 0s 2ms/step - loss: 1.7463e-04
Epoch 97/100
19/19 [==============================] - 0s 2ms/step - loss: 1.7233e-04
Epoch 98/100
19/19 [==============================] - 0s 2ms/step - loss: 1.7248e-04
Epoch 99/100
19/19 [==============================] - 0s 2ms/step - loss: 1.6964e-04
Epoch 100/100
19/19 [==============================] - 0s 2ms/step - loss: 1.6808e-04
19/19 [==============================] - 0s 7ms/step - loss: 0.0031
5/5 [==============================] - 0s 5ms/step
45/45 [==============================] - 0s 2ms/step
Mean Squared Error: 748.5039511760556


[**********************100%**********************]  1 of 1 completed
[**********************100%**********************]  1 of 1 completed

Risk of the individual companies :  [1.591392178861302, 1.9439851175631495
]
Portfolio expected return is :  0.06646236955390186
Portfolio risk is : 1.463318183630116
```

# Chapter 5

# CONCLUSION & FUTURE WORK

## 5.1 Conclusion

The first part is a Python script that calculates the risk and return for multiple portfolios consisting of different stocks. It defines a `Portfolio` class with methods to calculate individual stock returns and risks, as well as portfolio returns and risks. The code prompts the user to enter the number of portfolios and the details of each portfolio, and then calculates and stores the returns and risks in separate DataFrames.

The second part is an example of code for stock price prediction using an LSTM-based autoencoder and LSTM model. It downloads historical stock price data, preprocesses it, defines and trains the models, makes predictions, and evaluates the model's performance using mean squared error.

## 5.2 Challenges & Limitation

The combination of an Autoencoder (AE), Long Short-Term Memory (LSTM), and OMEGA (One Model for Ensemble Generation and Analysis) can be a powerful approach for time series forecasting and anomaly detection. However, there are some challenges and limitations to consider when using this combination:

**5.2.1 Complexity and computational requirements**: Training an AE-LSTM-OMEGA model can be computationally expensive, especially if you have a large dataset or complex architecture. It may require significant computational resources and time to train and tune the model effectively.

5.**2**.2 **Data preprocessing and feature engineering**: Preprocessing time series data for an AE-LSTM-OMEGA model can be challenging. Time series data often requires careful feature engineering, normalization, handling missing values, and dealing with non-stationarity or seasonality. The quality of the input data and the effectiveness of the preprocessing steps can impact the performance of the model.

5.**2**.3. **Hyperparameter tuning**: The AE-LSTM-OMEGA model has several hyperparameters that

need to be tuned, such as the number of LSTM layers, hidden units, learning rate, activation functions, and dropout rates. Finding the optimal values for these hyperparameters can be time-consuming and may require expertise and experimentation.

5.**2**.4 **Model interpretability**: Deep learning models like AE-LSTM-OMEGA are known for their black-box nature, making it challenging to interpret the learned representations and understand the reasoning behind the model's predictions. Interpretability can be crucial, especially in financial applications where decision-making and risk assessment are involved.

5.**2**.5 **Overfitting**: Deep learning models, including AE-LSTM-OMEGA, are prone to overfitting, especially when the model is complex and the training data is limited. Regularization techniques such as dropout and early stopping can help mitigate overfitting, but careful monitoring and validation are necessary.

5.**2**.6 **Lack of robustness to concept drift**: Time series data can be subject to concept drift, where the underlying patterns and relationships change over time. The AE-LSTM-OMEGA model might struggle to adapt to such changes unless it is continuously retrained or updated with new data.

5.**2**.7 **Scalability**: Scaling the AE-LSTM-OMEGA model to handle large-scale datasets can be challenging. As the dataset size increases, memory consumption and computational requirements also increase. Efficient implementation and optimization techniques may be needed to handle large-scale data effectively.

5.**2**.8. **Data availability and quality**: The performance of the AE-LSTM-OMEGA model heavily relies on the availability and quality of the training data. Insufficient or low-quality data can limit the model's ability to capture meaningful patterns and make accurate predictions.

It's important to keep these challenges and limitations in mind when implementing an AE-LSTM-OMEGA model and consider them while designing your solution and addressing the specific requirements and constraints of your project.

## 5.3 Future Work

Incorporating AE (Autoencoder), LSTM (Long Short-Term Memory), and OMEGA (One-class Matrix Factorization) in a single framework can be an interesting approach for anomaly detection in time series data. Here's a potential outline for future work involving AE+LSTM+OMEGA:

5.3.1. Data Preparation:

Gather the time series data that you want to analyze. Ensure that the data is properly formatted and preprocessed for further steps.

5.3.2. Autoencoder (AE) Training:

Train an AE model to learn the underlying patterns and representations in the time series data. The AE will aim to compress the input data into a lower-dimensional latent space and then reconstruct the original data from the compressed representation.

Use a suitable loss function such as mean squared error (MSE) to measure the reconstruction error between the input and output of the AE.

Experiment with different architectures, layer sizes, and hyperparameters to optimize the performance of the AE.

5.3.3 LSTM Model Integration:

Incorporate LSTM layers into the AE architecture to capture temporal dependencies in the time series data. This allows the model to analyze the sequential nature of the data and learn long-term dependencies.

Connect the output of the AE's encoder part to the LSTM layers. The encoder will provide the compressed representation of the data, which can be fed into the LSTM for further processing.

Fine-tune the combined AE+LSTM model using a suitable objective, such as minimizing the reconstruction error while preserving temporal dependencies.

5.3.4. OMEGA for Anomaly Detection:

Integrate OMEGA, which is a one-class matrix factorization method, into the AE+LSTM framework. OMEGA aims to factorize the observed data matrix into low-rank and sparse components, where the low-rank component represents normal patterns, and the sparse component captures anomalies.

Train OMEGA on the latent representations obtained from the AE+LSTM model. The objective is to identify the anomalies in the data based on the reconstructed error or residual between the original data and the reconstructed data.

5.3.5. Anomaly Detection and Evaluation:

Apply the trained AE+LSTM+OMEGA model to detect anomalies in new unseen time series data.

Define a suitable anomaly threshold or scoring mechanism based on the reconstruction error or residual obtained from the OMEGA component.

Evaluate the performance of the model using appropriate metrics such as precision, recall, F1-score, or area under the receiver operating characteristic curve (AUC-ROC).

Fine-tune and optimize the parameters of the AE+LSTM+OMEGA model to improve its anomaly detection capabilities.

5.3.6. Experimentation and Comparison:

Conduct experiments and compare the performance of the AE+LSTM+OMEGA model with other existing anomaly detection techniques for time series data.

Explore different variations of the model, such as incorporating attention mechanisms, varying the architecture, or combining with other unsupervised learning approaches to further enhance anomaly detection performance.

for future work on AE+LSTM+OMEGA for anomaly detection in time series data. The specific implementation details and adjustments may vary depending on your dataset, problem domain, and research objectives.

# REFERENCES


1. https://www.simplilearn.com/tutorials/deep-learning-tutorial/what-are-autoencoders-in-deep-learning
2. https://www.wallstreetmojo.com/portfolio-optimization/
3. https://cleartax.in/s/stock-market-analysis
4. https://www.tensorflow.org/api_docs/python/tf/keras/layers/LSTM
5. https://towardsdatascience.com/omega-ml-deploying-data-machine-learning-pipelines-the-easy-way-a3d281569666
6. https://finance.yahoo.com/?guccounter=1&guce_referrer=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8&guce_referrer_sig=AQAAAML8ULHkJD-unt1DB-e_yQRPH9yDdUXEVoHXlGosyGZzCi4M2GkZ0GRzeuYsRstQc5JfgLvalwdqQPU6gu7GL1iZQ0u3SJgKzIDqLTG2zTBuLhbqpmZoanW9083L-C51hIR19sZT2vcSa4XMJreSnOi73PhHhDbtyRdks7J9XVjX