MATH 4041 Statistical Machine Learning
New Mexico Tech
Spring 2025

Final Project:
Stock Price Prediction and Trading Simulation

Sungjun Seo

1 Introduction

A company's stock price changes in response to various factors, including material prices, seasonal effects, the national economy, and even the stock prices of other companies. Based on this information, we expect that it is possible to predict a company's stock price, in particular, Tesla's. A dataset that will be used in this project has been obtained from Kaggle and is shown in the figure below.

Date	Natural_Gas	Crude_oil	Copper	Bitcoin	Platinum	Ethereum	S&P_500	Apple	Tesla	Microsoft	Silv
7/7/2023	2.582	73.86	3.782	30,346.40	918.5	1,870.92	4,398.95	190.68	274.43	337.22	
6/7/2023	2.609	71.8	3.7345	29,913.10	909.7	1,848.36	4,411.59	191.81	276.54	341.27	
5/7/2023	2.657	71.79	3.7685	30,512.80	925	1,910.43	4,446.82	191.33	282.48	338.15	
3/7/2023	2.709	69.79	3.794	31,151.30	916	1,955.56	4,455.59	192.46	279.82	337.99	
30-06-202	2.798	70.64	3.7595	30,472.90	913.2	1,933.80	4,450.38	193.97	261.77	340.54	
29-06-202	2.701	69.86	3.699	30,445.70	906.8	1,851.92	4,396.44	189.59	257.5	335.05	
28-06-202	2.603	69.56	3.7425	30,078.60	924.7	1,827.95	4,376.86	189.25	256.24	335.85	
27-06-202	2.763	67.7	3.7885	30,689.10	934	1,889.51	4,378.41	188.06	250.21	334.57	
26-06-202	2.791	69.37	3.8045	30,267.00	932.9	1,859.00	4,328.82	185.27	241.05	328.6	
23-06-202	2.729	69.16	3.8155	30,679.40	923.7	1,891.97	4,348.33	186.68	256.6	335.02	
22-06-202	2.608	69.51	3.9005	29,890.50	926.5	1,872.32	4,381.89	187	264.61	339.71	
21-06-202	2.597	72.53	3.9105	29,996.90	949	1,889.87	4,365.69	183.96	259.46	333.56	
20-06-202	2.492	70.5	3.8935	28,307.70	968	1,791.61	4,388.71	185.01	274.45	338.05	

The dataset consists of 17 predictors, including the daily average prices of natural gas, crude oil, copper, Bitcoin, platinum, Ethereum, S&P 500, Apple, Microsoft, silver, Google, Nvidia, Berkshire Hathaway, Netflix, Amazon, Meta, and gold, recorded over 718 days. The goal of this project consists of two folds: i) to build a regression model that can reasonably forecast Tesla's stock price using the dataset ii) to conduct the trading simulation using the regression model.

2 Problem Description

2.1 Division of Dataset

This project consists of two main components: training a regression model and conducting a trading simulation. To support this, the original dataset is divided into two parts: the first 678 days are used for training the regression model, while the remaining 150 days are used for the trading simulation.

2.2 Regression Model

2.2.1 Type of Regression Model

Stock prices follow a non-Markovian process: the price on the next day depends not only on today's price but also on prices from earlier days. Because of this non-Markovian characteristic, the dataset should be reshaped so that each sample includes stock prices from multiple consecutive days, as shown in Figure. 1.

No	Variable	
1	T_0	S_0
2	T_1	S_1
...	...	...

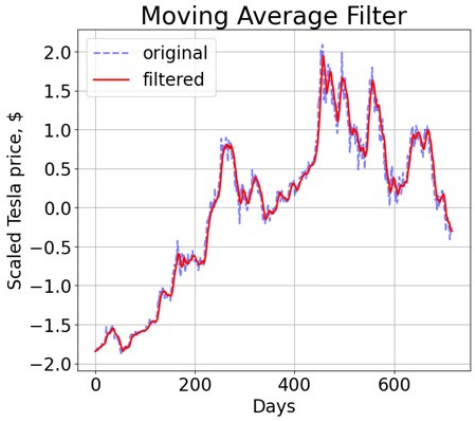
→

No	Input						Output	
1	T_0	...	T_{H-1}	S_0	...	S_{H-1}	T_H	S_H
2	T_1	...	T_H	S_1	...	S_H	T_{H+1}	S_{H+1}
...	...	...	...	...	...	...	...	...

Figure 1: Reshape of the dataset with two variables: T_k and S_k represent the Tesla's stock price and S&P500 price at day k, respectively.

The Recurrent Neural Networks (RNNs) are well-suited for this type of problem setup. There are two main types of RNNs: the univariate RNN and the multivariate RNN. The univariate RNN uses a single variable to predict its future value, while the multivariate RNN uses multiple variables to forecast their future values simultaneously. In this project, the univariate RNN and multivariate RNN are used to train the regression model. For the multivariate RNN, two stock prices, Tesla and S&P 500, are used. In addition to these methods, the univariate RNN model trained on the filtered stock price is also investigated. Figure 2(a) shows the stock price trend that many analysts use to make trading decisions. This implies that the filtered data is one of the important indicators in stock trading. Inspired by this, the univariate RNN model with the filtered stock price is also investigated.

(a) Stock price. The blue curve represents the filtered stock price.



(b) Filtered Tesla stock price using a moving average filter.

Figure 2: Filtered stock price

Figure 2(b) shows the Tesla stock price before and after applying a moving average filter, which is one of the most well-known low-pass filters. A window size of 7 days is used for the filter.

2.2.2 The lag of the RNNs

In this project, the variable H represents the lag of the RNNs, as shown in Figure 1. The lag plays an important role in the RNNs. Setting the lag too high can increase the model's complexity, resulting in higher computational effort, whereas setting it too low may fail to capture the memory effect of past information. To find the proper value of H, the Autocorrelation Function (ACF) is used in this project. Figure 3 shows the result of the autocorrelation function. The lags from 0 to 50 exhibit strong correlations, while they fall outside the confidence bounds. Therefore, the lag H is set to 50 in this project.

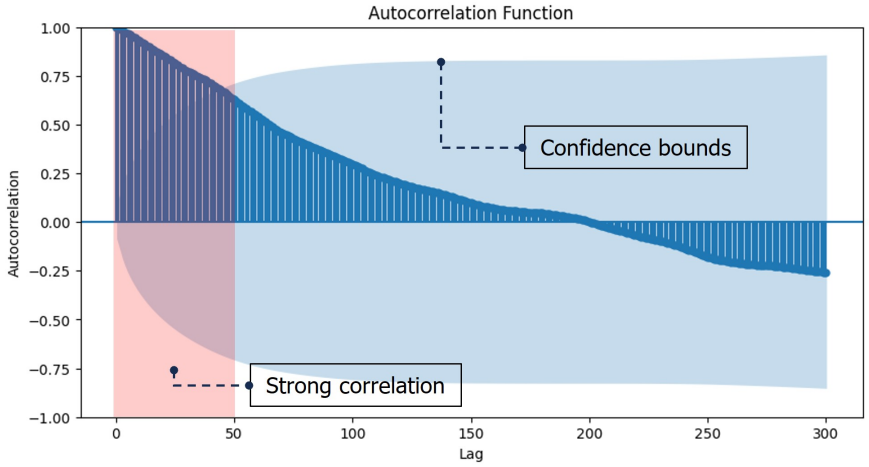


Figure 3: Autocorrelation function

2.2.3 Flowchart and parameters of RNNs

This section describes the flowchart and parameters of RNNs. Figure **??** illustrates how the data is processed. The dataset, consisting of 568 samples, is first scaled using `StandardScaler` to ensure faster and more balanced learning. Additionally, scaling ensures a fairer comparison of loss across different models. Then, the data is reshaped into consecutive sequences to apply the RNNs. The dataset is divided into 70% for training and 30% for testing. To train the model, a `SimpleRNN` is used with 50 hidden units. The `ReLU` activation function is employed to introduce nonlinearity. Backpropagation is performed over 200 epochs with a batch size of 256.

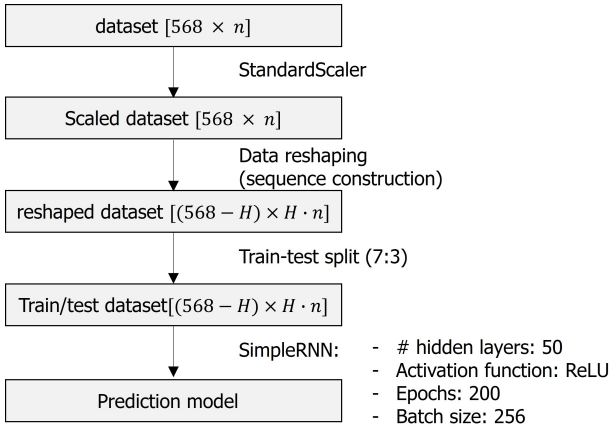


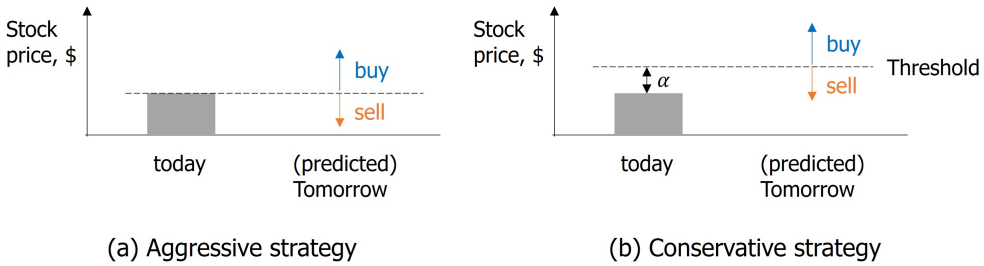
Figure 4: Flowchart and parameters of RNNs

2.3 Trading Simulation

For each regression model described in Section 2.2, a trading simulation is conducted. In the trading simulation, there are several assumptions and conditions.

- There is no trading fee.
- The stock price for each day is fixed at the daily average price given in the dataset. That is, the stock price does not fluctuate during the day.
- The initial capital is set to $ 1,000,000.

Based on the assumptions and conditions, the trading simulation is conducted. Two types of trading strategies are employed in the trading simulation: the aggressive strategy and the conservative strategy. In the aggressive strategy, the buy decision is made if the stock price is expected to increase tomorrow. However, in the conservative strategy, the buy decision is made if the stock price is expected to increase tomorrow more than the threshold, which is higher than today's stock price. This is presented in Figure. 5. Algorithms 1 and 2 show the pseudo code for both strategies.



(a) Aggressive strategy (b) Conservative strategy

Figure 5: Trading strategies

Algorithm 1 Aggressive strategy

Input: $C_0 = \$1,000,000$ (C_k :=the amount of cash at day k), T_0, ..., T_{150} (T_k :=tesla price at day k), $V_0 = 0$ (V_k :=Volume of the held Tesla stock at day k)

for k = 0,1, ..., 149 **do**
  calculate the predicted Tesla stock price, $\hat{T}_{k+1}$, using the regression model
  if $\hat{T}_{k+1} \geq T_k$ **then**
    $V_{k+1} \leftarrow V_k + C_k \bmod T_k$ # The stock volume held increases after a purchase.
    $C_{k+1} \leftarrow C_k \operatorname{div} T_k$ # The remaining cash decreases after a purchase.
  else
    $C_{k+1} \leftarrow C_k + V_k \times T_k$ # The remaining cash increases after selling the stocks.
    $V_{k+1} \leftarrow 0$ # The held stock volume becomes zero after selling all stocks.
  end if
end for

Algorithm 2 Conservative strategy

Input: $C_0 = \$1,000,000$ (C_k :=the amount of cash at day k), T_0, ..., T_{150} (T_k :=tesla price at day k), $V_0 = 0$ (V_k :=Volume of the held Tesla stock at day k), α ($0 < \alpha < 0.3$)

for k = 0,1, ..., 149 **do**
  calculate the predicted Tesla stock price, $\hat{T}_{k+1}$, using the regression model
  if $\hat{T}_{k+1} \geq T_k \times (1+\alpha)$ **then**
    $V_{k+1} \leftarrow V_k + C_k \bmod T_k$ # The stock volume held increases after a purchase.
    $C_{k+1} \leftarrow C_k \operatorname{div} T_k$ # The remaining cash decreases after a purchase.
  else
    $C_{k+1} \leftarrow C_k + V_k \times T_k$ # The remaining cash increases after selling the stocks.
    $V_{k+1} \leftarrow 0$ # The held stock volume becomes zero after selling all stocks.
  end if
end for

After running the simulation, the total profit or loss is calculated as follows:

$$\text{Total profit or loss} = C_{150} + T_{150} \times V_{150} - \$1,000,000 \tag{1}$$

Figure 6 presents the summary of the problem description. A total of six trading simulations are conducted

using three different regression models. The performance of each model is evaluated based on the total profit at the end of the trading simulation.

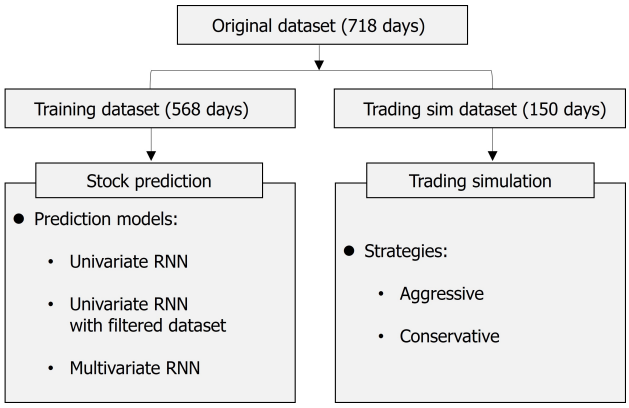


Figure 6: Summary of the problem description

3 Simulations

Trading simulations are conducted for each regression model: the univariate RNN, the multivariate RNN, and the univariate RNN with filtered data. For each model, 4 types of results are presented as follows:

- The loss curves for training and validation are presented to demonstrate that the regression models are sufficiently trained.
- The scatter plot comparing the true and predicted values for the *training dataset* (568 days), along with the Mean Absolute Error (MAE), illustrates how well the regression model fits the data.
- The net asset trend graph shows how effectively the regression model performs in the trading simulation.
- The scatter plot comparing the true and predicted values for the trading dataset (150 days), along with the Mean Absolute Error (MAE), demonstrates how accurately the regression model forecasts future stock prices.

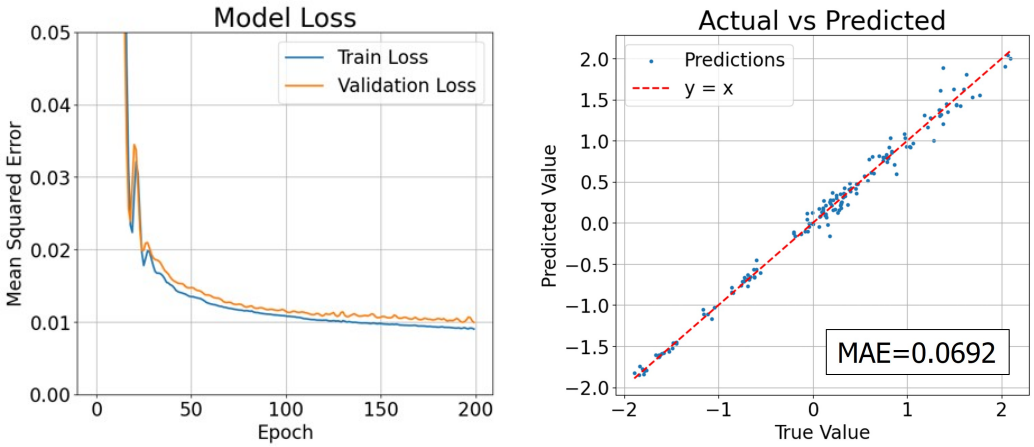
The net asset refers to the combined tangible and intangible assets, as defined by

$$\text{net asset} = \text{remaining cash} + \text{value of held stock} = C_k + V_k \times T_k. \tag{2}$$

3.1 Simulation: The univariate RNN model

3.1.1 model training

Figures 7(a) and (b) illustrate the model's training performance. In Figure 7(a), both the training and validation losses converge to approximately 0.01, indicating that the model was sufficiently trained. In Figure 7(b), the blue points represent the predicted versus true values of the output for each sample in the *training dataset*. The set of points is well aligned with the $y = x$ line, indicating a good fit of the regression model. The MAE is calculated as 0.0692.

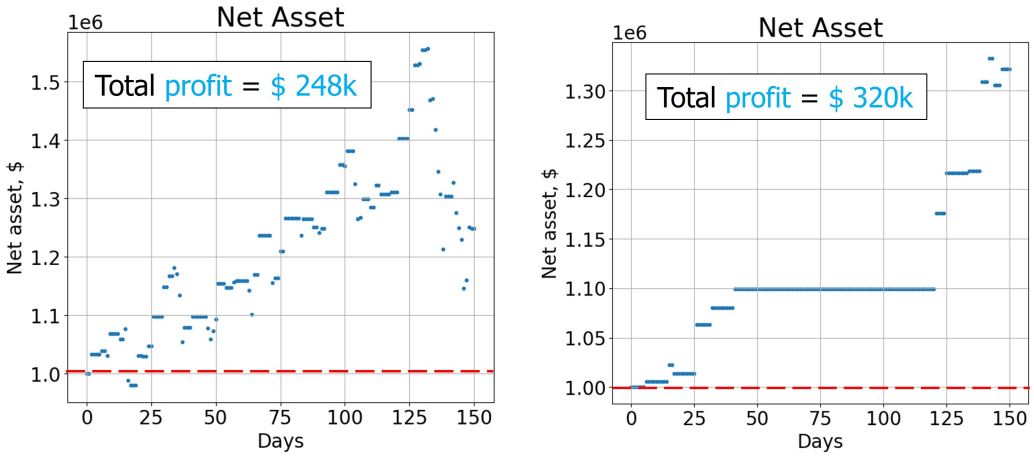


(a) Loss curve (b) True vs. Predicted plot

Figure 7: Performance of the model training

3.1.2 trading simulation

The trained regression model is used to run the trading simulation. Figure 8 presents the performance of the simulation for both the aggressive and conservative strategies. The algorithms for these strategies are described in Algorithms 1 and 2, respectively. The parameter α for the conservative strategy is set to 0.03. The total profit is calculated using Equation (1). Both strategies yielded a profit, with the conservative strategy generating a higher return.



(a) Aggressive strategy (b) Conservative strategy

Figure 8: Performance of the trading simulation

Figure 9 demonstrates the predicted versus true values of the output for each sample in the *trading dataset*. The points are closely clustered around the $y = x$ line, indicating good performance in forecasting the stock price. The MAE is calculated as \$8.37, indicating that the predicted stock price deviates from the actual value by \$8.37 on average. Given that the average stock price in the *trading dataset* is approximately \$250, this deviation corresponds to about

$$\text{deviation } (\%) = 8.37/250 \times 100 = 3.34. \tag{3}$$

Among all proposed regression models, this univariate model yielded the highest profit due to its lowest prediction error.

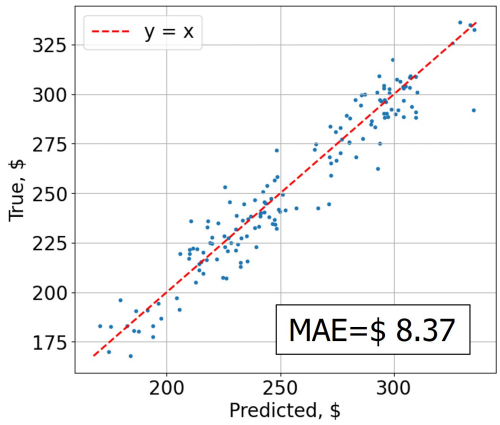
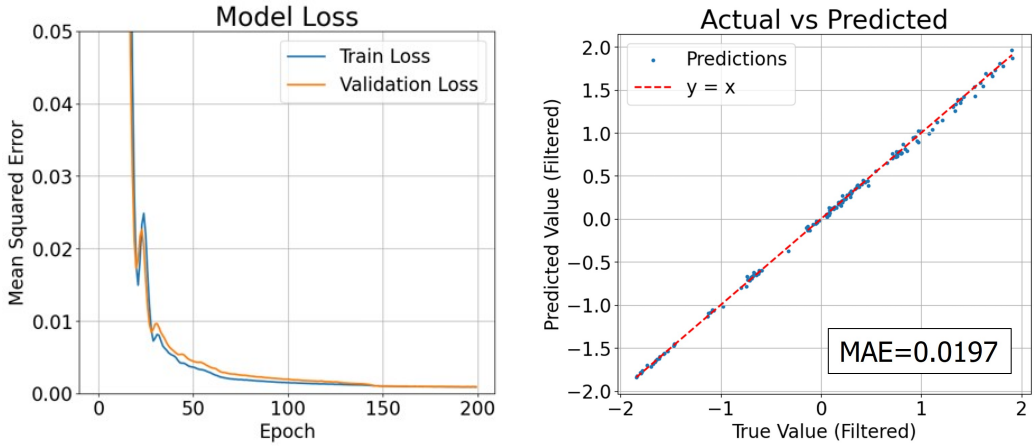


Figure 9: True vs. Predicted plot for trading dataset

3.2 Simulation: The univariate RNN model with filtered data

3.2.1 model training

Being different from the univariate RNN model, this model takes the filtered stock price using the moving average filter, as shown in Figure 2(b). Due to the nature of the low-pass filter, the filtered stock price exhibits a smoother curve, eliminating high-frequency fluctuations. The smoothness of the dataset resulted in a better fit of the regression model, as shown in Figure 10(b). Compared to the MAE of the simple univariate model discussed in Section 3.1.1, the MAE of this model is significantly lower, indicating a better fit.



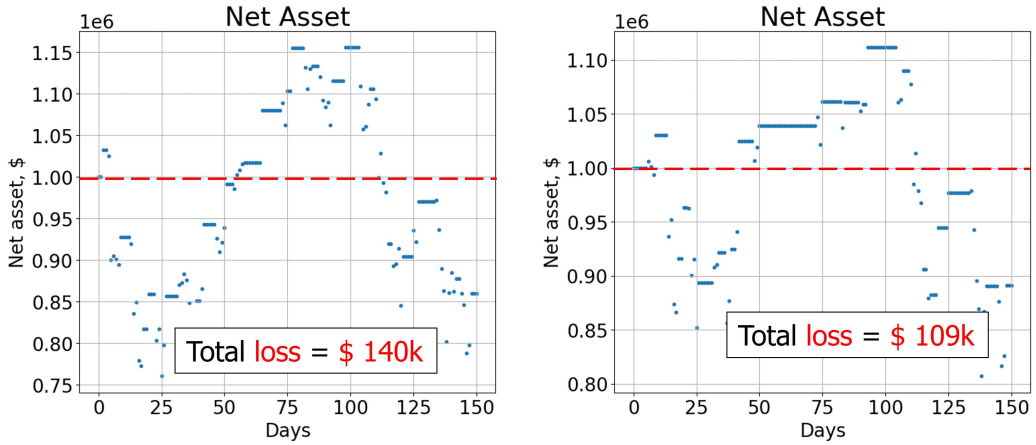
(a) Loss curve

(b) True vs. Predicted plot

Figure 10: Performance of the model training

3.2.2 trading simulation

In contrast to the promising result of the model training, the trading simulation showed poor performance. Both strategies resulted in a loss, with the conservative strategy incurring a smaller loss. Figure 12 explains the reason for this poor performance.



(a) Aggressive strategy (b) Conservative strategy

Figure 11: Performance of the model training

In Figure 12, the scatter plot exhibits a wider spread and a higher MAE compared to the distribution in Figure 9. Even though the regression model was well trained on the filtered data, the data itself lost its high-frequency characteristics. This may lead to the underfitting of the regression model with respect to the actual (unfiltered) data, resulting in poor performance.

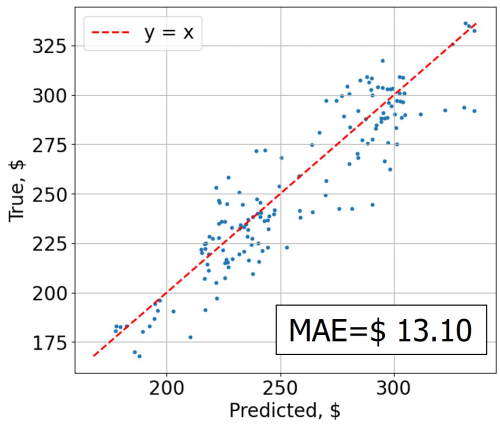
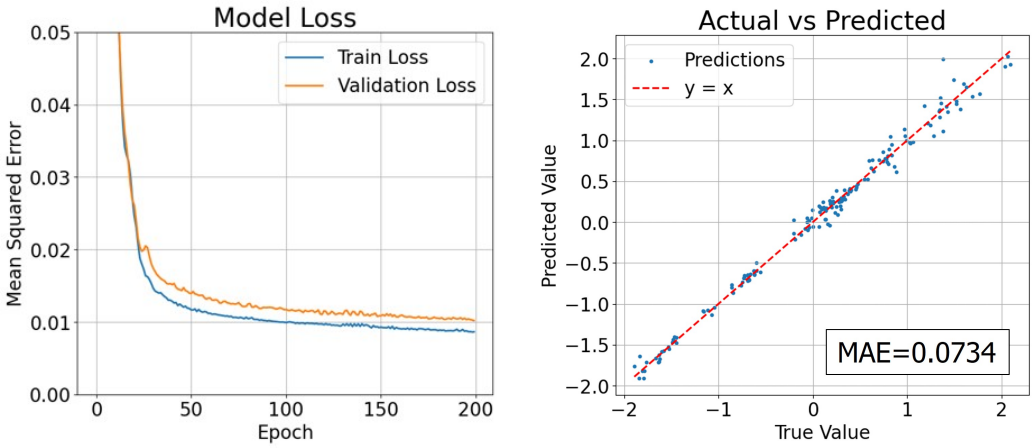


Figure 12: True vs. Predicted plot for trading dataset

3.3 Simulation: The multivariate RNN model

3.3.1 model training

In the multivariate RNN model, two stock prices are used for the dataset: Tesla stock price and S&P 500 price. Relative to the univariate RNN, the MAE shows a slight increase. This increase is attributed to the training of multiple variables. The multivariate RNN model is trained not only on the Tesla stock price but also on the S&P 500 price simultaneously. This introduces a trade-off resulting from the additional variable.

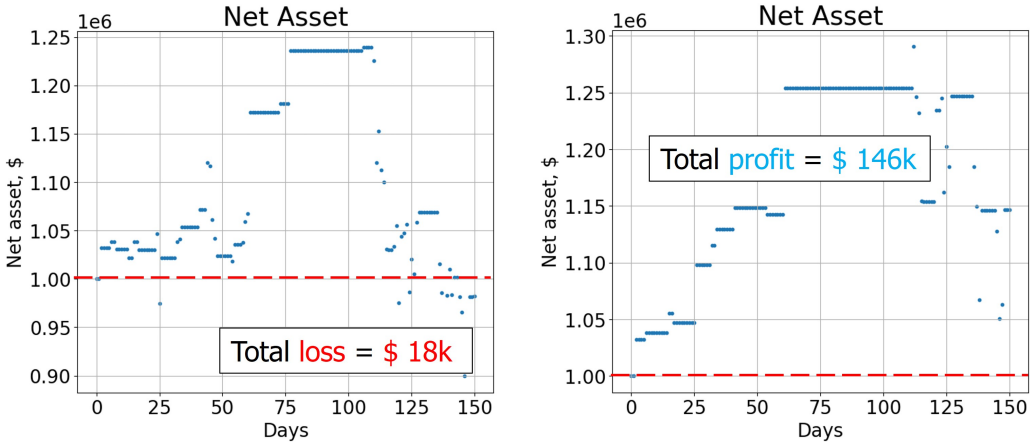


(a) Loss curve (b) True vs. Predicted plot

Figure 13: Performance of the model training

3.3.2 trading simulation

The trading simulation showed decent performance. Although the aggressive strategy resulted in a small loss, the conservative strategy generated a profit.



(a) Aggressive strategy (b) Conservative strategy

Figure 14: Performance of the model training

In Figure 15, the scatter plot exhibits a wider spread and higher MAE compared to the univariate RNN case. Fitting multiple variables simultaneously in the multivariate RNN model may cause deviation in the predictions.

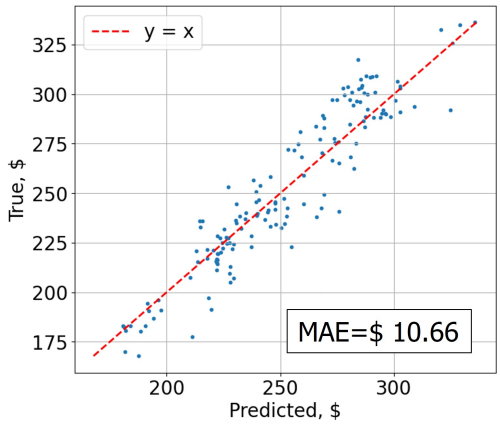


Figure 15: True vs. Predicted plot for trading dataset

4 Conclusion

In this project, stock market prediction and trading simulation were performed using RNNs. Three different regression models based on RNNs were proposed: the univariate RNN model, the univariate RNN model with filtered data, and the multivariate RNN model. For each model, trading simulations were conducted using two different strategies: an aggressive strategy and a conservative strategy. Among the proposed models, the univariate RNN model achieved the best performance, yielding the highest total profit in the trading simulation. In all cases, the conservative strategy outperformed the aggressive strategy.
However, the result of the best-performing model, the univariate RNN, did not produce a reliable outcome. The prediction deviation, calculated as 3.34% using MAE in Equation (3), may lead to incorrect trading decisions, considering that the stock price typically fluctuates by 2–3%. Therefore, a more precise regression model should be investigated to apply it to the real-world stock market.
Furthermore, in this approach, the dataset is divided into an older subset and a newer subset. The regression model is trained on the older subset and then used to predict the output of the newer subset. This approach is less appropriate, as the model is trained on outdated patterns and applied to data that may follow new patterns. To address this issue, the model should rely more on recent data. This can be achieved by training the regression model in real time.

5 Description of Attached Files

The following files are included to support the project results:

- `Mod_US Stock Market Dataset_2.csv` – Dataset used for training and testing the model.
- `Project_Stock_Univ.ipynb` – Python script containing the training and the simulations for the univariate RNN model.
- `Project_Stock_univ_filtered.ipynb` – Python script containing the training and the simulations for the univariate RNN model with filtered data.
- `Project_Stock_Mult.ipynb` – Python script containing the training and the simulations for the multivariate RNN model.