TDK-thesis

Nguyen Duy Chien

Bui Nguyen Kim Hai

Transformers and Multi-features Time2Vec for Financial Prediction

EÖTVÖS LORÁND UNIVERSITY

FACULTY OF INFORMATICS

DEPT. OF NUMERICAL ANALYSIS

Authors:

Nguyen Duy Chien
Computer Science BSc
II. grade

Bui Nguyen Kim Hai
Computer Science BSc
II. grade

Supervisor:

Dr. Gergő Bognár
Assistant Professor

Budapest, 2024

Abstract

Financial prediction presents an intriguing task, as numerous factors influence stock prices. Particularly, financial time series data pose a formidable obstacle to prediction due to the intricate modeling required to capture both short-term fluctuations and long-term temporal dependencies within the dataset. Transformers have remarkable success in many tasks in natural language processing and computer vision using attention mechanisms, which also triggered great interest in the time series community. The ability to capture long-range dependencies and interaction and the ability to learn and understand languages lead to the understanding of the financial market and recognize the historical pattern of prices. While the study of Transformers in stock prediction is not novel, existing research predominantly relies on historical prices of individual features for singular predictions, thus limiting the ability to comprehensively recognize or understand broader market trends. In this paper, we present the study into the different markets by selecting correlation features to enhance the accuracy in predicting multiple stock prices and minimizing the error associated with exceptional cases in stock market analysis. Combining Transformer model with Time2Vec technique as Encoder. We conducted a comparative analysis of our results with others models and suggested potential directions for advancing our research.

Contents

Chapter 1

Introduction

Time series forecasting is challenging, especially in the financial industry [1]. It involves statistically understanding complex linear, and non-linear interactions within historical data to predict the future. Traditional statistical approaches commonly adopt linear regression, exponential smoothing [2], and auto regression model [3]. With the advances in deep learning, recent works are heavily invested in ensemble models and sequence-to-sequence modeling such as RNN (recurrent neural networks), and Long Short-Term Memory [4]. However, the primary drawback of these methods is that the RNN family struggles to capture extremely long-term dependencies [5]. A well-known sequence-to-sequence model called Transformer [6] has achieved great success in NLP, especially LLM like ChatGPT, Gemini. Different from RNN-based modes, Transformer employs a multi-head self-attention mechanism to learn the relationship among different positions globally. For example, in terms of the tech industry, such as the layoff of employees in 2023, first begin with Meta, then move to Google, Microsoft, and Apple. Then the stock price of those companies is almost down or up at the same time. Or in terms of social network companies. In other industries like oil, it makes people suspect the current market and sell all stocks they have.

Chapter 2

Background and Related Work

2.1 Neural Network

A neural network (also artificial neural network or neural net, abbreviated ANN or NN) is a model inspired by the structure and function of biological neural networks in animal and human brains.

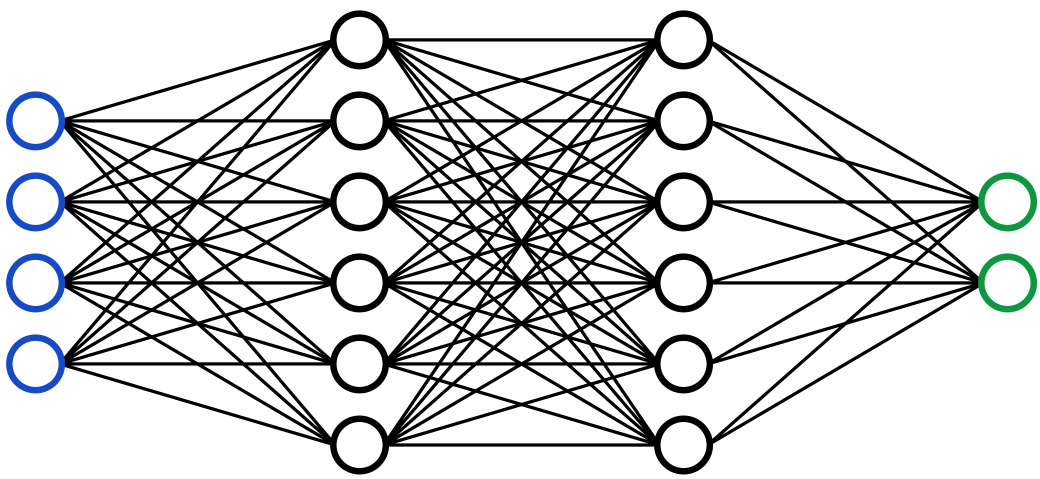

Figure 2.1: An example Neural Network. (from [7])

Neural networks consis of neurons organized into layers (see Figure 2.1), where the layers usually perform a linear transformation followed by a non-linear activation function. This allow networks to learn complex patterns and relationships in the data. There are numerous different activation functions. We can put them into 2 category: Non-Linear Activate Functions and others.

2.2 Non-Linear Activation Functions

In this section, we will walk-through the most important activation functions, and their properties.

Sigmoid:

$$f(x) = \frac{1}{1+e^{-x}} \tag{2.1}$$

It's especially useful for classification or probability prediction tasks so that it can be implemented into the training of computer vision and deep learning networks. However, vanishing gradients can make these problematic when used in hidden layers, and this can cause issues when training a model. That's why nowadays, ReLU (see Equation 2.3) is more widely used in computer vision and deep learning instead of Sigmoid.

Tanh:

$$f(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})} \tag{2.2}$$

It is a steeper gradient and also encounters the same vanishing gradient challenge as sigmoid.

ReLU:

$$f(x) = \max(0, x) \tag{2.3}$$

ReLU does not activate every neuron in sequence at the same time, making it more efficient than the tanh or sigmoid/logistic activation functions. Unfortunately, the downside of this is that some weights and biases for neurons in the network might not get updated or activated.

LeakyReLU:

$$f(x) = \max(0.1 * x, x) \tag{2.4}$$

The advantages of Leaky ReLU are same as that of ReLU, in addition to the fact that it does enable backpropagation, even for negative input values.

ParametricReLU:

$$f(x) = \max(a * x, x) \tag{2.5}$$

Parametric ReLU is another variant of ReLU that aims to solve the problem of gradient's becoming zero for the left half of the axis.

Softmax:

$$\operatorname{softmax}(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)} \tag{2.6}$$

It is most commonly used as an activation function for the last layer of the neural network in the case of multi-class classification.

2.3 ARIMA

Typical forecasting techniques in the literature utilize statistical tools, such as exponential smoothing (ETS) [2] and auto regressive integrated moving average (ARIMA) [3], on numerical time series data for making one-step-ahead predictions. These predictions are then recursively fed into the future inputs to obtain multi-step forecasts. Multi-horizon forecasting methods [8] and [9] directly generate simultaneous predictions for multiple predefined future time steps.

2.4 RNN - Recurrent Neural Network

Recurrent neural networks (RNNs) are designed to handle temporal problems, and they are widely used for sequential data and time-series analysis. The RNN family is natural choice for financial forecasting as well, see e.g. [10]. Although RNN is able to accurately characterize the contextual relationship between sequential data, this relationship weakens as the gap distance between them grows.

Figure 2.2 present the unfolded architecture of an RNN.

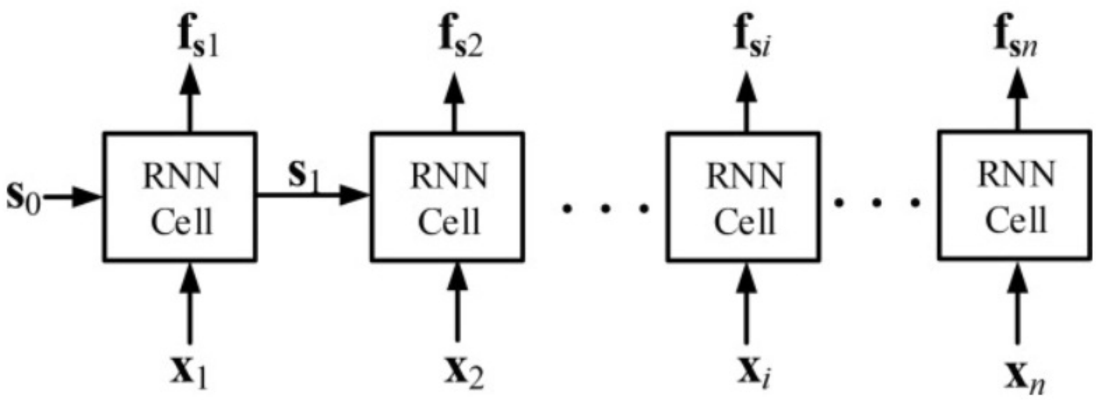

Figure 2.2: Recurrent Neural Network model. (from [11])

Moreover, an RNN model has the vanishing gradient problem for the long sequence data. However, LSTM can prevent this problem during training.

2.5 CNN - Convolutional neural network

A convolutional neural network (CNN) involves special layers that perform convolution and pooling. CNN architectures serve as the most widely used ANNs in image processing and computer vision, but they are rarely used for financial.

2.6 LSTM - Long Short-Term Memory

Long Short-Term Memory (LSTM) is a special RNN that a memory mechanism with gates, making the network capable to learn long-term dependencies and preventing the vanishing gradient problem. The structure of an LSTM cell is demonstrated in Figure 2.3.

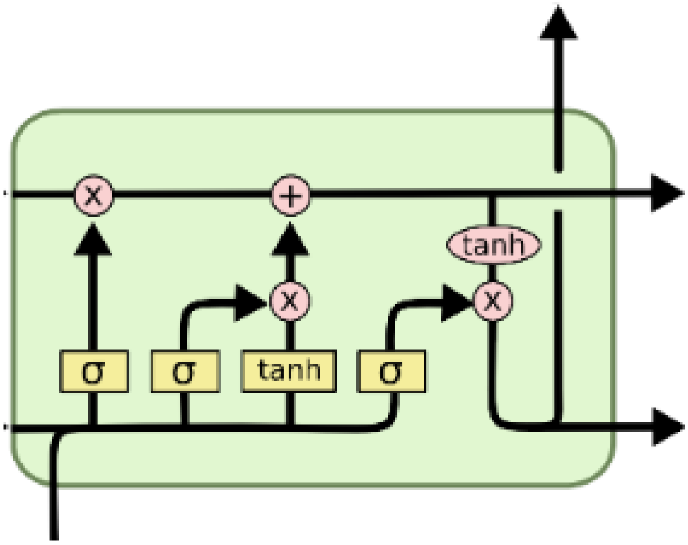

Figure 2.3: LSTM - Long Short-Term Memory architecture. (from [12])

LSTM [4] is widely used for financial time series prediction, particularly in forecasting stock prices. There are also many variables of LSTM such as BiLSTM, Conv-LSTM [13] , Attentive – LSTM [14] with an attention mechanism to predict stock price movement. However, [5] points out that LSTM can only distinguish 50 positions nearby with an effective context size of about 200. That means that LSTM-based models suffer from the difficulty in capturing extremely long- term dependencies in time series

2.7 Time2Vec

Time2Vec [15] is an approach providing a model – agnostic vector representation for time. Time decomposition technique that encodes a temporal signal into a set of frequencies.

$$t2v(\tau)[i] = \begin{cases} w_i\tau + \varphi_i & \text{if } i = 0 \\ \sin(w_i\tau + \varphi_i), & \text{if } 1 \leq i \leq k \end{cases} \tag{2.7}$$

This design have three important properties:

1. Capturing both periodic and non periodic patterns.

2. Being invariant to time re-scaling.

3. Being simple enough so it can be combined with many models.

The sin activation function is inspired parts of positional encoding from [6] which can be combined with transformers model.

2.8 Transformer

The innovation of Transformer in deep learning [6] has brought great interests recently due to its excellent performances in natural language processing (NLP) [16]. Over the past few years, numerous Transformers have been proposed to advance the state-of-the-art performances of various tasks "significantly" Transformers have shown great modeling ability for long- range dependencies and interactions in sequential data and thus are appealing to time series modeling. Many variants of Transformer have been proposed to address special challenges in time series modeling and have been successfully applied to various time series tasks, such as forecasting anomaly detection and classification [17].

Figure 2.4 shows the architecture of Transformer.

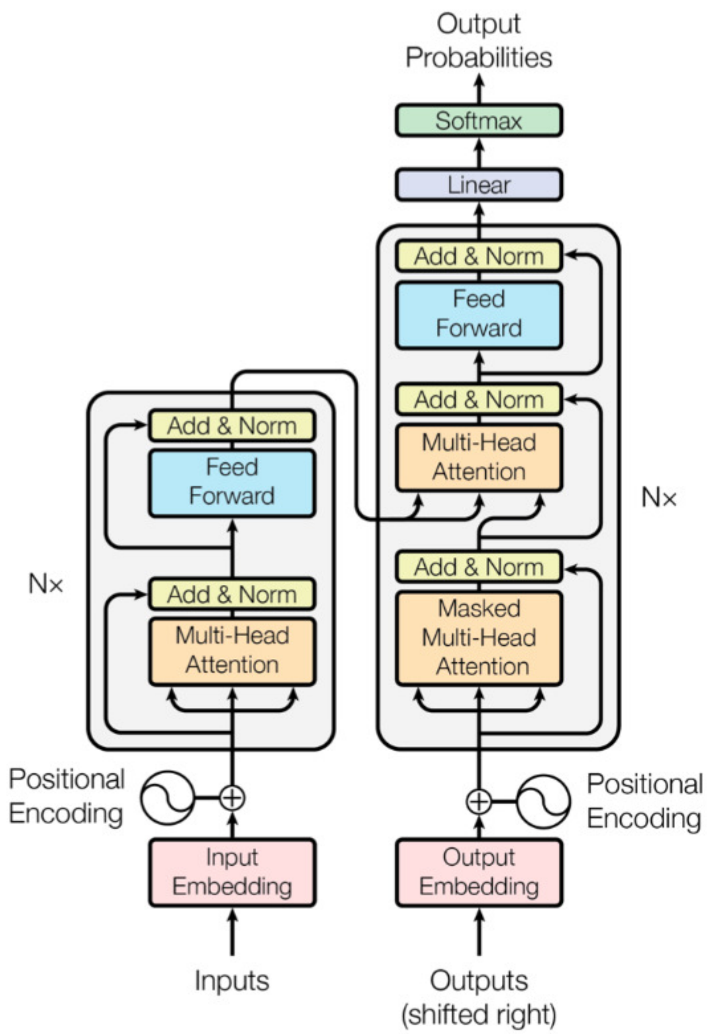

Figure 2.4: The Transformer - model architecture. (from [18])

2.9 Transformer and Time-Embedding

Unlike natural language data where positional encoding suffices to capture word order, processing sequential data with Transformers necessitates a nuanced approach to extract temporal dependencies. This gap is bridged by Time Embedding, which imbue the model with an understanding of chronological order, preventing nonsensical predictions where distant historical prices hold equal sway as recent ones. To address the temporal challenges inherent in Transformers, the Time2Vec methodology is adopted, offering a model-agnostic vector representation of time. This approach encapsulates both periodic and non-periodic patterns while maintaining invariant to time re-scaling, ensuring the model's ability to comprehend and utilize temporal features effectively in predicting stock prices.

2.10 Optimizer

In supervised learning, the training of ANNs are usually carried out as a data-driven numerical optimization o a lost function. In feedforward NNs, where the neurons are organized in layers, backpropagation and a stochastic gradient descent optimizer are mainly used. There are an enormous optimizer out there, but in this thesis, we are using Adam Optimizer.

2.10.1 Adaptive Moment Estimation

Adaptive Moment Estimation is an algorithm for optimization technique for gradient descent. The method is really efficient when working with large problem involving a lot of data or parameters. It requires less memory and is efficient. Intuitively, it is a combination of the ‘gradient descent with momentum’ algorithm and the ‘RMSP’ algorithm. Adam Optimizer inherits the strengths or the positive attributes of the two methods and builds upon them to give a more optimized gradient descent.

2.10.2 Momentum

This algorithm is used to accelerate the gradient descent algorithm by taking into consideration the ‘exponentially weighted average’ of the gradients. Using averages makes the algorithm converge towards the minima in a faster pace.

2.10.3 Root Mean Square Propagation - RMSP

Root mean square prop or RMSprop [19] is an adaptive learning algorithm that tries to improve AdaGrad [20]. Instead of taking the cumulative sum of squared gradients like in AdaGrad, it takes the ‘exponential moving average’.

2.11 Evaluation metrics

In this section we overview the most important metrics that are used to evaluate the performance of the predictions of a forecasting method.

2.11.1 MAPE - Mean Absolute Percentage Error

Also known as mean absolute percentage deviation (MAPD), is a measure of prediction accuracy of a forecasting method in statistics. It usually expresses the accuracy as a ratio defined by the formula.

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{Y_i - \hat{Y}_i}{Y_i}\right| \tag{2.8}$$

Y_i: Actual value.
$\hat{Y}_i$: Predicted value.
n: number of values.

2.11.2 MAE - Mean Absolute Error

Mean absolute error (MAE) is a measure of errors between paired observations expressing the same phenomenon.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|Y_i - \hat{Y}_i\right| \tag{2.9}$$

Y_i: Actual value.
$\hat{Y}_i$: Predicted value.
n: number of values.

2.11.3 RMSE - Root Mean Square Error

Root mean square error (RMSE) is the residuals' standard deviation, or the average difference between the projected and actual values produced by a statistical model.

$$\text{RMSE} = \sqrt{\sum_{i=1}^{n}\frac{\left(Y_i - \hat{Y}_i\right)^2}{n}} \tag{2.10}$$

Y_i: Actual value.
$\hat{Y}_i$: Predicted value.
n: number of values.

2.11.4 MSE - Mean Square Error

Mean squared error (MSE), the average squared difference between the value observed in a statistical study and the values predicted from a model.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2. \tag{2.11}$$

Y_i: Actual value.

$\hat{Y}_i$: Predicted value.

n: number of values.

2.11.5 R2 - R-Square

R-Squared value shows how well the model predicts the outcome of the dependent variable. R-Squared values range from 0 to 1. Squared value of 0 means that the model explains or predicts 0% of the relationship between the dependent and independent variables.

$$R^2 = 1 - \frac{\sum\left(Y_i - \hat{Y}_i\right)^2}{\sum\left(Y_i - \bar{Y}\right)^2}. \tag{2.12}$$

Y_i: Actual value.

$\hat{Y}_i$: Predicted value.

$\bar{Y}$: Mean of all values.

n: number of values.

Chapter 3

Our Work

3.1 Motivation

The forecasting finance area is currently under intense research focus: many researchers and specialists are trying to combine Time2Vec with CNN, RNN, LSTM, and Attention mechanism. Moreover, not only finance but also other fields as well like Aeroengine Risk Assessment [21] and Predicting Production in Shale and Sandstone Gas Reservoirs [22].

They usually use only one feature to predict things (for instance, only one stock price to predict its prices). However, using techniques that rely solely on a single feature can be quite challenging when it comes to predicting unexpected financial events, which are often influenced by external factors beyond our control. On the flip side, in the world of finance, it's common practice to focus on identifying and analyzing correlations between specific features and target prices when making investment decisions.

Building upon previous challenges, we were motivated to explore the integration of correlations from multiple features into the Transformer model. We represents a new approaches over existing methods by developing a neural network architecture that consists of Time2Vec, residual, multiple attention layers, pooling, and a dense fully-connected part. Given the limitations inherent in financial data often sparse and messy we sought to address these issues by selecting multiple stocks exhibiting similar behavior and interconnectedness. The subsequent sections of this paper will detail our approach to time series modeling.

3.2 Methodology

The performance of a stock market predictor heavily depends on the correlations between historical data for training and the current input for prediction. The results show below here in Figure 3.1 and Figure 3.2 we use Exxon Mobil and NASDAQ stock prices as base market.

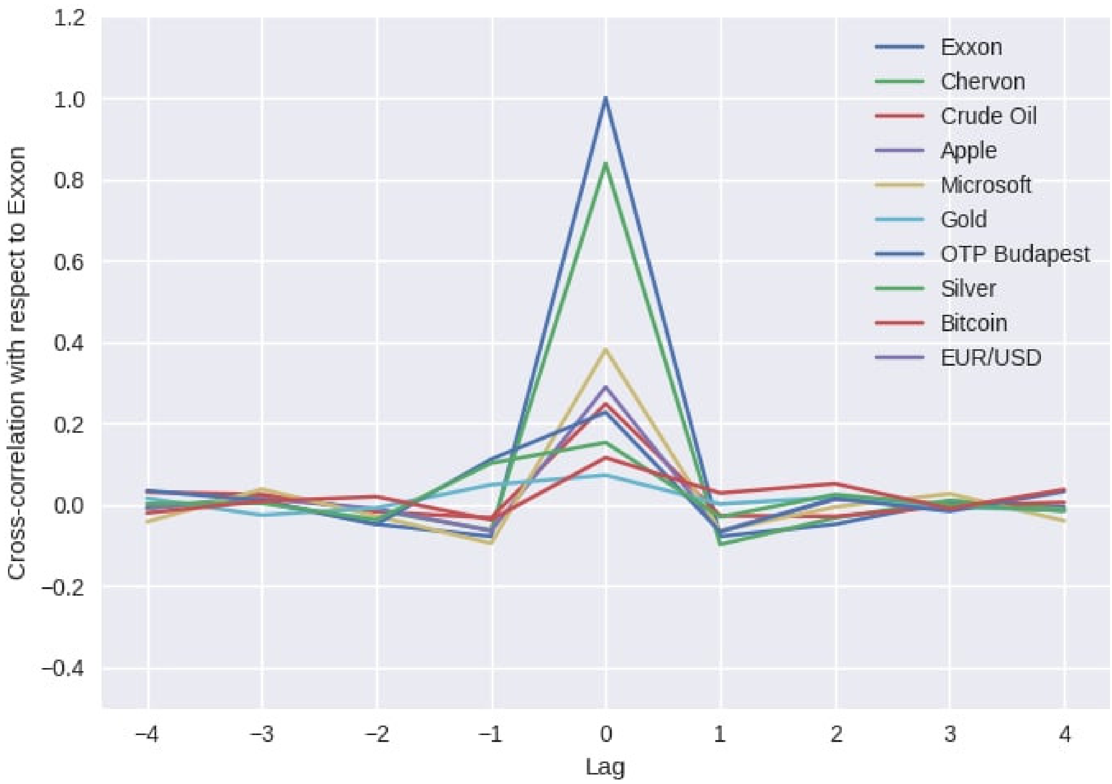

Figure 3.1: Auto-correlation and cross-correlation of markets trend using Exxon Mobil as a based market.

The graph (Figure 3.1) reveals that base markets' auto-correlation is solely non-zero at the origin. This observation suggests that the daily trend of Exxon stock approximates a Markov process [23]. Consequently, historical data offers limited insight into its future movements. However, data source such as Chevron is a promising feature for our approach.

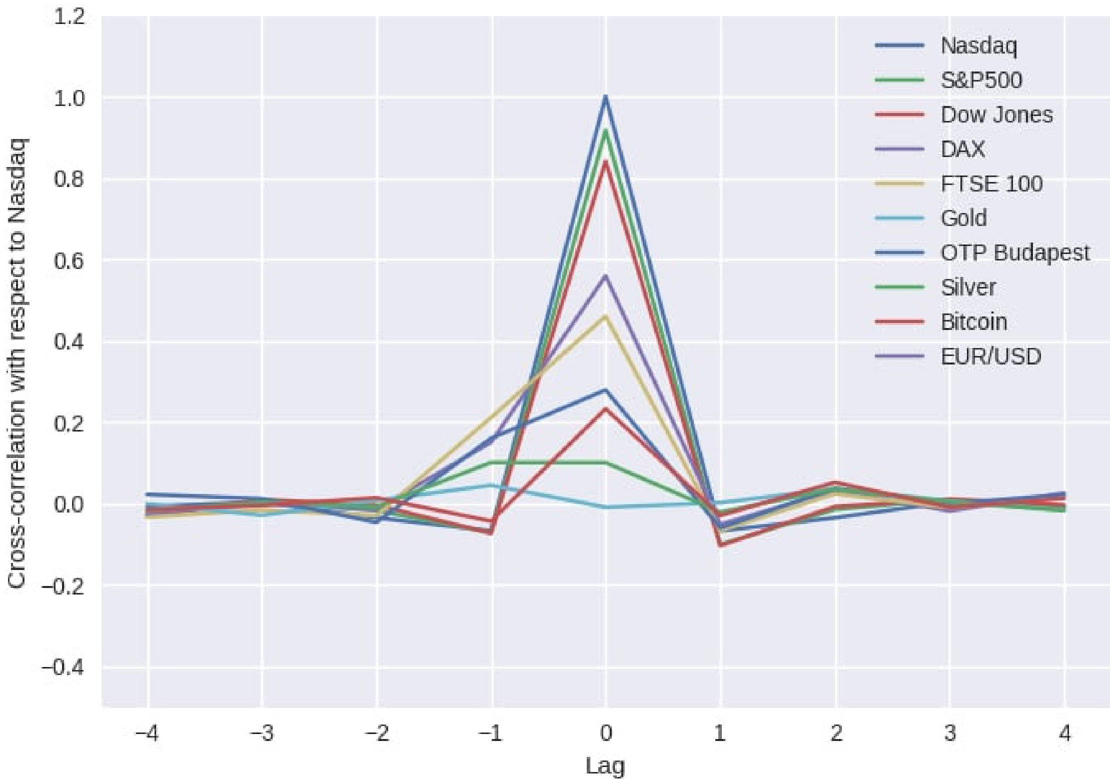

Figure 3.2: Auto-correlation and cross-correlation of markets trend using NASDAQ as a based market.

From Figure 3.2 we get same conclusion for NASDAQ, S&P500 will be a good data to pair with. Moreover, we will try to evaluate will it be better if we put more relatable data (for example: NASDAQ, S&P500, Dow Jones, and DAX into the model) than only 2 features (NASDAQ and S&P500).

3.3 Mean Not NaN

Each time series might have missing data points represented as NaN values, and we have to use a proper combination approach that handles this issue. In this paper, we considered Geometry Mean Not NaN - GMNN (Figure 3.3) and Arithmetic Mean Not NaN - AMNN (Figure 3.4).

3.3.1 Geometry Mean Not NaN - GMNN

GMNN is the technique we find out to combine multiple normalized data into one normalized data. This technique assure that the transformed data will be between 0 and 1 – which is the attribute the combined data must have.

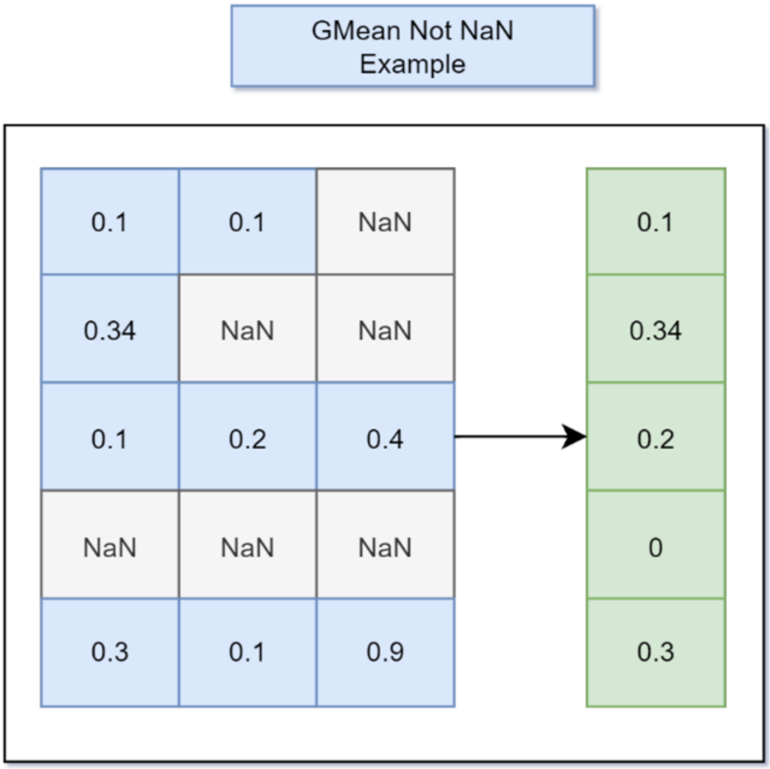

Figure 3.3: Simple illustration with 1-Dimensional input pass through GMNN step.

How do GMNN work? – GMNN iterate by row and take the Geometry Mean of real numbers in each row and assign it as a output value for that row.

3.3.2 Arithmetic Mean Not NaN - AMNN

AMNN follows the same idea with GMNN that is combining multiple normalized data into one normalized data.

AMNN iterate by row and take the Arithmetic Mean of real numbers in each row and assign it as a output value for that row.

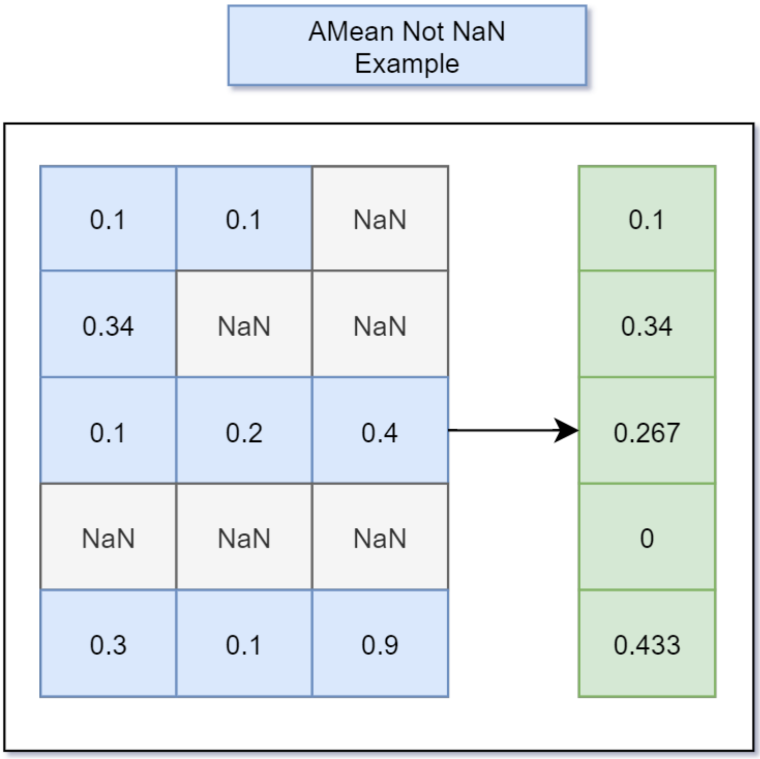

Figure 3.4: Simple illustration with 1-Dimensional input pass through AMNN step.

3.3.3 AMNN or GMNN?

Both AMNN and GMNN can resolve the task. However, by experiment several times, we can conclude that GMNN yields better result in predicting stock prices than AMNN, so that we will choose GMNN as a method to represent our result in this thesis.

Chapter 4

Experiment

4.1 Data Collection

To evaluate proposed method. We use two group of stocks:

1. Group 1: NASDAQ, S&P500, DJI, DAX.

2. Group 2: Exxon Mobil, Chervon.

In each group, group members are highly correlated with each other (see Figure 3.2 for group 1, Figure 3.1 for group 2).

We collect the daily quote data of those stocks from Yahoo Finance [24].

Table 4.1 shows an example for data download from Yahoo.

Table 4.1: NASDAQ data download from Yahoo Finance.

Date	Open	High	Low	Close	Volume
1971-02-05	100.0000	100.0000	100.0000	100.0000	0
1971-02-08	100.8399	100.8399	100.8399	100.8399	0
1971-02-09	100.7600	100.7600	100.7600	100.7600	0
1971-02-10	100.6900	100.6900	100.6900	100.6900	0
1971-02-11	101.4499	101.4499	101.4499	101.4499	0
...	...	...	...	...	...
2024-05-01	15646.0898	15926.2197	15557.6396	15605.4804	5277790000
2024-05-02	15758.1103	15862.7900	15604.7304	15840.9599	4901610000
2024-05-03	16147.4804	16204.7099	16068.3398	16156.3300	4887310000
2024-05-06	16208.5400	16350.0800	16197.8603	16349.2500	4460130000
2024-05-07	16208.5000	16396.4589	16326.2109	16373.5625	2693650000

Date: The specific day when the trading occurred.

Open: The price of the stock index at the beginning of the trading day (USD).

High: The highest price reached by the stock index during the trading day (USD).

Low: The lowest price reached by the stock index during the trading day (USD).

Close: The price of the stock index at the end of the trading day (USD).

Volume: The total number of shares traded during the day (share).

4.2 Data Pre-processing

First, we fill forward all missing dates in the range from starting date to current date. To smooth out the ups and downs in stock prices, we start by averaging the values over a **14-day** period (see Table 4.2). Then, we calculate the percentage change for the next day (see Table 4.3). When building our model, we make sure to scale these percentage changes so they fall between 0 and 1, using a standard min-max scaling method we get the result is Table 4.4 for NASDAQ and Table 4.5 for S&P500.

$$x_{\text{standarlized}} = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{4.1}$$

Finally, we push all the processed data (Table 4.4 and Table 4.5) to the Geometry Mean Not NaN block (Figure 3.3), which will be the final input to feed for the model (see Table 4.6).

Table 4.2: NASDAQ data after filling missing dates and applying Moving Average.

Date	Open	High	Low	Close	Volume
1971-02-18	101.3850	101.3850	101.3850	101.3850	6.2860e+7
1971-02-19	101.4350	101.4350	101.4350	101.4350	6.2860e+7
1971-02-20	101.3521	101.3521	101.3521	101.3521	6.2860e+7
1971-02-21	101.2692	101.2692	101.2692	101.2692	6.2860e+7
1971-02-22	101.1864	101.1864	101.1864	101.1864	6.2860e+7
...	...	...	...	...	...
2024-05-03	15729.2649	15846.3864	15614.3349	15750.9149	4.870674e+9
2024-05-04	15787.2942	15904.3207	15680.9206	15815.0535	4.859489e+9
2024-05-05	15845.3235	15962.2550	15747.5064	15879.1921	4.848303e+9
2024-05-06	15903.3528	16020.1893	15814.0921	15943.3307	4.837117e+9
2024-05-07	15952.1349	16067.8352	15870.7528	15988.7533	4.815941e+9

Table 4.3: NASDAQ data after computing percentage change.

Date	Open	High	Low	Close	Volume
1971-02-19	0.000493	0.000493	0.000493	0.000493	0
1971-02-20	-0.000817	-0.000817	-0.000817	-0.000817	0
1971-02-21	-0.000818	-0.000818	-0.000818	-0.000818	0
1971-02-22	-0.000818	-0.000818	-0.000818	-0.000818	0
1971-02-23	-0.000734	-0.000734	-0.000734	-0.000734	0
...	...	...	...	...	...
2024-05-03	0.002734	0.002839	0.003883	0.003981	-0.006248
2024-05-04	0.003689	0.003656	0.004264	0.004072	-0.002297
2024-05-05	0.003676	0.003643	0.004246	0.004056	-0.002302
2024-05-06	0.003662	0.003629	0.004228	0.004039	-0.002307
2024-05-07	0.003067	0.002974	0.003583	0.002849	-0.004378

Table 4.4: NASDAQ data after normalizing.

Date	Open	High	Low	Close	Volume
1971-02-19	0.635564	0.635564	0.635564	0.635564	0.481983
1971-02-20	0.606946	0.606946	0.606946	0.606946	0.481983
1971-02-21	0.606932	0.606932	0.606932	0.606932	0.481983
1971-02-22	0.606917	0.606917	0.606917	0.606917	0.481983
1971-02-23	0.608753	0.608753	0.608753	0.608753	0.481983
...	...	...	...	...	...
2024-05-03	0.684514	0.686807	0.709619	0.711752	0.449510
2024-05-04	0.705385	0.704658	0.717949	0.713747	0.470046
2024-05-05	0.705089	0.704367	0.717554	0.713387	0.470019
2024-05-06	0.704795	0.704079	0.717161	0.713029	0.469991
2024-05-07	0.691800	0.689762	0.703062	0.687029	0.459228

Table 4.5: S&P500 data after normalizing.

Date	Open	High	Low	Close	Volume
1928-01-13	0.521913	0.521913	0.521913	0.521913	0.458965
1928-01-14	0.490008	0.490008	0.490008	0.490008	0.458965
1928-01-15	0.489934	0.489934	0.489934	0.489934	0.458965
1928-01-16	0.489860	0.489860	0.489860	0.489860	0.458965
1928-01-17	0.490609	0.490609	0.490609	0.490609	0.458965
...	...	...	...	...	...
2024-05-03	0.561882	0.562470	0.570750	0.574165	0.460139
2024-05-04	0.572563	0.568649	0.577909	0.576782	0.455490
2024-05-05	0.572467	0.568569	0.577788	0.576667	0.455481
2024-05-06	0.572371	0.568490	0.577668	0.576552	0.455472
2024-05-07	0.573196	0.563337	0.571300	0.561493	0.421249

Table 4.6: Combine NASDAQ and S&P500 using GMNN block.

Date	Open	High	Low	Close	Volume
1928-01-13	0.521913	0.521913	0.521913	0.521913	0.458965
1928-01-14	0.490008	0.490008	0.490008	0.490008	0.458965
1928-01-15	0.489934	0.489934	0.489934	0.489934	0.458965
1928-01-16	0.489860	0.489860	0.489860	0.489860	0.458965
1928-01-17	0.490609	0.490609	0.490609	0.490609	0.458965
...	...	...	...	...	...
2024-05-03	0.620174	0.621537	0.636408	0.639268	0.454793
2024-05-04	0.635514	0.633011	0.644135	0.641621	0.462711
2024-05-05	0.635327	0.632836	0.643890	0.641394	0.462693
2024-05-06	0.635141	0.632662	0.643647	0.641169	0.462675
2024-05-07	0.629712	0.623353	0.633766	0.621098	0.439829

We use the technique moving average that used sequence previous days movements to predict the next day change, which is our target. This approach takes into account the short-term trends in stock prices and uses time series indicators as features to classify the trend.

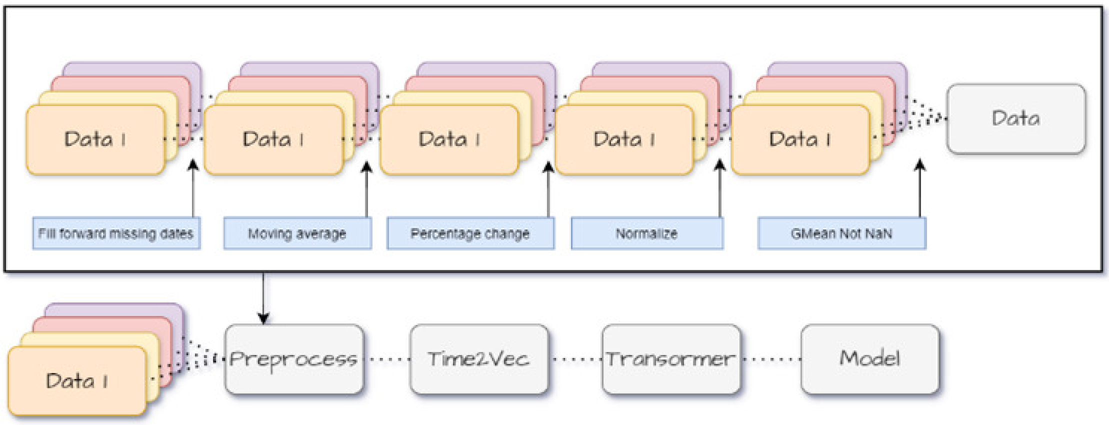

Figure 4.1: Pre-processing data pipeline.

4.3 Our Model

We will introduce a novel network architecture, presented on Figure 4.2 that combines the Time2Vec and the Transformer model.

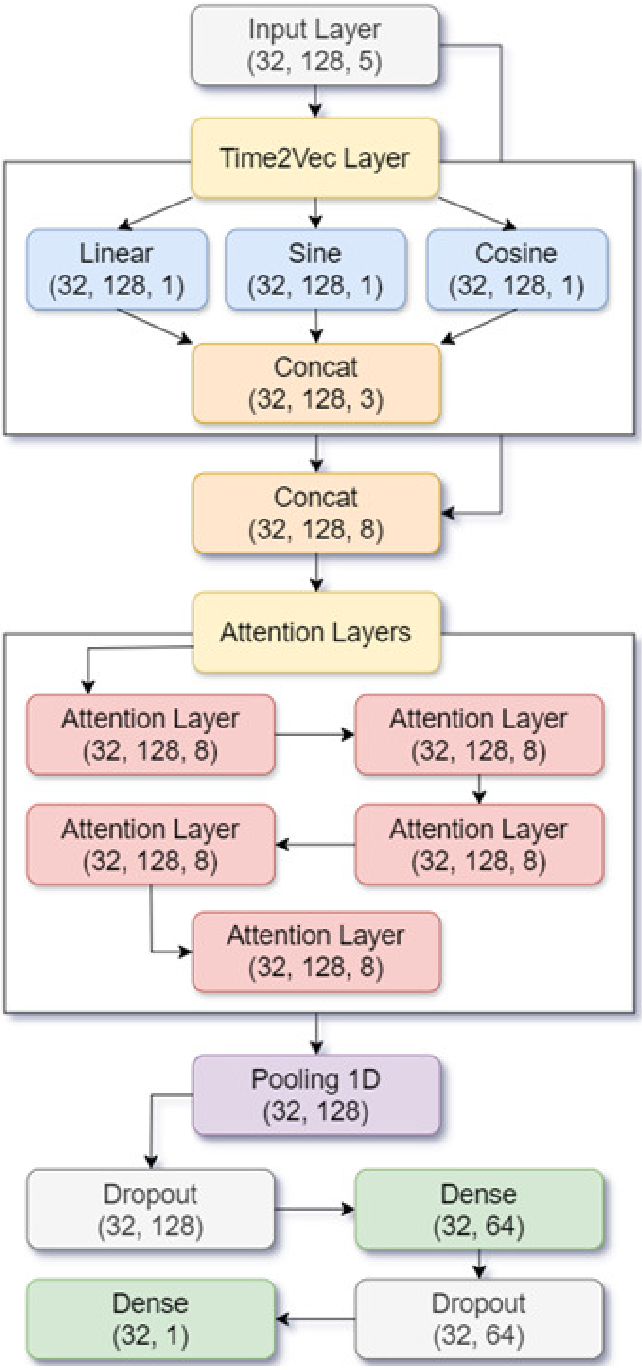

Figure 4.2: Propose model.

Each colored rectangle in the Figure 4.2 represents a layer.

Each layer have a tuple of numbers represent the output size after the input pass through it.

1. **Input Layer**: Take the processed data.

2. **Time2Vec Layer**: This layer contains 4 sub-layers

   (a) **Linear**

- This layer is used for capturing linear trends include steady increase and decrease in prices overtime - which might not be periodic but still play an important role in time-series.

(b) **Sine** and **Cosine**

- These two layers are really important for our model. Their first task is encoding positions which we can consider as time - represents continuous time instead of discrete positions. They also capture the periodic behaviors of prices.

- In the Time2Vec paper, they use only one periodic function that is sine Equation 2.7. But after experiment, we realize that using an additional periodic function will yield a better result in this case.

(c) **Concat**

- Use for concatenating above three layers, preparing for next steps.

3. **Concat**: This layer take Input layer and combine it with the output from Time2Vec layer. We make a residual connections - as known as ResNet [25]. This action significantly improve the behavior of the network which makes the Time2Vec layer more valuable and powerful to our architecture and our model.

4. **Attention Layers**: As known as Multi-head Attention layer, this group contains 5 single consecutive attention layers. This structure allow our model to jointly attend to values from different aspects at different positions ensure that the model can study the trend as clear as possible.

5. **Pooling 1D**: This layer is responsible for reducing the spatial dimensions of the output from Multi-head Attention layers, in terms of width and height, while retaining the most important information.

6. **Dropout**: We need to prevent over-fitting the model. Chosen dropout rate is 0.1.

7. **Dense**: Dense is a fully-connected neural network layer. The first Dense applies the activate functions ReLU (see Equation 2.3) and the other applies linear activation function. We use it for decreasing the data dimension.

4.4 Why it must be this model?

We choose Time2Vec to catch continuous attribute of time in the data, the Attention to get a deep understanding of the movement of the trend, the Dropout to prevent over-fitting. We believe that each layer play an important role in the architecture.

4.5 Result and Evaluation

For evaluation, we will split input into 3 parts before actually put them into training process. Train data will be 80% of the input. Validation data will be the next 10% of the input. Test data will be 10% left of the input.

4.5.1 Result of Group 1: NASDAQ, S&P500, DJI, DAX

We have trained 5 models for this group.

Note: Multi-feature models are M1_1, and M1_4.

Note: One feature models are M1_2 and M1_3.

1. M1_1: nas_sp_dji_dax (Table 4.7a)

   Features: NASDAQ, S&P500, DJI, and DAX

2. M1_2: nas (Table 4.7b)

   Features: NASDAQ

3. M1_3: sp (Table 4.7c)

   Features: S&P500

4. M1_4: nas_sp (Table 4.7d)

   Features: NASDAQ, S&P500

(a) M1_1

Train MAE	0.0127
Train MAPE	2.3391
Train loss	0.0004
Val MAE	0.0131
Val MAPE	2.1929
Val loss	0.0004
Test MAE	0.0147
Test MAPE	3.3804
Test loss	0.0006

(b) M1_2

Train MAE	0.0137
Train MAPE	2.2923
Train loss	0.0004
Val MAE	0.0124
Val MAPE	2.0052
Val loss	0.0003
Test MAE	0.0192
Test MAPE	3.2285
Test loss	0.0008

(c) M1_3

Train MAE	0.0106
Train MAPE	2.1629
Train loss	0.0003
Val MAE	0.0119
Val MAPE	2.4171
Val loss	0.0004
Test MAE	0.0108
Test MAPE	2.1768
Test loss	0.0003

(d) M1_4

Train MAE	0.0105
Train MAPE	2.0188
Train loss	0.0003
Val MAE	0.0128
Val MAPE	2.3027
Val loss	0.0004
Test MAE	0.0119
Test MAPE	2.15
Test loss	0.0003

Table 4.7: Metrics for models in Group 1

4.5.2 Evaluation for Group 1

We can see that M1_4 (Table 4.7d) yields better result than M1_1 (Table 4.7a) which means that 4 features does not lead to a better model.

We can also point out that M1_4 (Table 4.7d) have better results than M1_2 (Table 4.7b) and M1_3 (Table 4.7c). This means that the multi-feature models **may** work better than the single-feature models.

4.5.3 Result of Group 2: Exxon Mobil, Chervon

We have trained 3 models for this group.

Note: Multi-feature model is M2_3.

Note: One feature models are M2_1 and M2_2.

1. M2_1: exxon (Table 4.8a)

   Features: Exxon

2. M2_2: chervon (Table 4.8b)

Features: Chervon

3. M2_3: exxon_chervon (Table 4.8c)

   Features: Exxon, Chervon

(a) M2_1

Train MAE	0.0138
Train MAPE	2.1378
Train loss	0.0003
Val MAE	0.012
Val MAPE	1.8924
Val loss	0.0003
Test MAE	0.0303
Test MAPE	6.14
Test loss	0.0028

(b) M2_2

Train MAE	0.0114
Train MAPE	1.9157
Train loss	0.0002
Val MAE	0.0099
Val MAPE	1.6689
Val loss	0.0002
Test MAE	0.0163
Test MAPE	3.7077
Test loss	0.0009

(c) M2_3

Train MAE	0.011
Train MAPE	1.7753
Train loss	0.0002
Val MAE	0.0096
Val MAPE	1.5592
Val loss	0.0002
Test MAE	0.018
Test MAPE	3.8575
Test loss	0.0011

Table 4.8: Metrics for models in Group 2

4.5.4 Evaluation for Group 2

The results in subsection 4.5.2 show similar findings here.

Multi-feature models outperform single-feature model, as evidenced by M2_3 (Table 4.8c) being superior to M2_1 (Table 4.8a). However, model M2_2 slightly outperforms the multi-feature models, as indicated in Table 4.8b.

This shows that although multi-feature model outperforms one-feature models under certain circumstances, special one-feature model may still have better performance.

4.6 Further evaluation

From now on, we will not split the data into 3 parts (train, validation, and test) to evaluate since we have the models (as trained above).

Moreover, we will use RMSE (from subsection 2.11.3), MSE (from subsection 2.11.4), and R2 (from subsection 2.11.5) as additional metrics with MAPE (from subsection 2.11.1) and MAE (from subsection 2.11.2) to have a clearly evaluation for models.

We also use Accuracy, Precision (Pre), Recall, and F1-score (F1) to evaluate our trend prediction (classification problem).

Note: Accuracy, Pre, Recall, F1-score will evaluate on 2 labels: Decreasing, and Non-decreasing. We will se there is only one value for Accuracy but two for the others (the left value will be for label 0, the right value will be for label1).

4.6.1 Method for decoding predicted output to prices

We need to follow the pipeline how we pre-processed the data to decode it. Figure 4.3 describes the decoding pipeline.

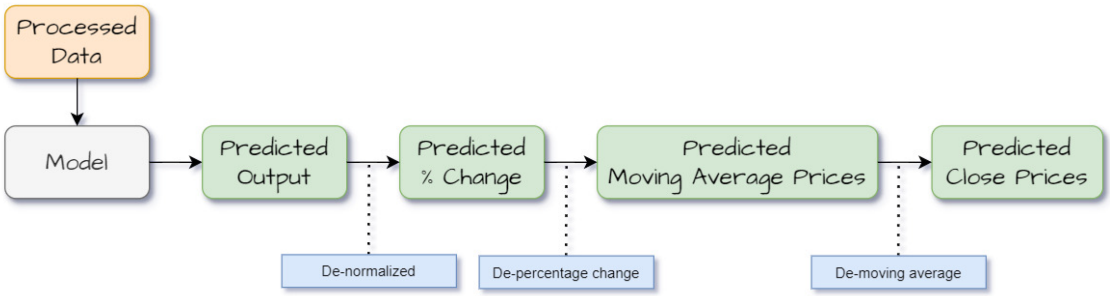

Figure 4.3: Decode predicted output pipeline.

Walk-through for each step in the pipeline.

Note that: x is used for predicted values, v is use for real values.

1. **De-normalized step**: This step include 2 small steps in it.

   (a) Making sure that the output is between 0 and 1, if there exist values such that $x_i > 1$ we will take its reciprocal instead. Cases that $x_i < 0$ seem to not occur in experiment.

   (b) Apply min-max de-normalized formula:

   $$x_{pct} = x_{nor} \times (max - min) + min \tag{4.2}$$

   $x_{predictnor}$: vector of output from model (predicted normalized values).

   max, min: max and min value from Equation 4.1.

   x_{pct}: vector of predicted percentage change values.

   Note that: max and min is max and min of open, high, low, and close.

2. **De-percentage change step**:

$$x_{mva} = \begin{cases} v_{pct}, i = 0 \\ v_{pct} \times (1 + x_{pct}), otherwise \end{cases} \tag{4.3}$$

v_{pct}: vector of real percentage change values.

x_{pct}: vector of predicted percentage change values.

v_{mva}: vector of predicted moving average prices.

3. **De-moving average step**:

$$x_{close} = \begin{cases} v_{close} & , i \leq s-1 \\ x_{mva} \times s & -\sum_{k=i-s}^{i-1} v_{close}, i \geq s \end{cases} \tag{4.4}$$

x_{mva}: vector of predicted moving average prices.

v_{close}: vector of real close prices.

x_{close}: vector of predicted close prices.

s: moving average step

In this thesis, we follow moving average strategy every fortnight which means our step will be 14. So that, Equation 4.4 can also be written as

$$x_{close} = \begin{cases} v_{close} & , i \leq 13 \\ x_{mva} \times 14 & -\sum_{k=i-14}^{i-1} v_{close}, i \geq 14 \end{cases} \tag{4.5}$$

4.6.2 Deep evaluation with respect to NASDAQ

In terms of NASDAQ, we can use M1_1, M1_2, and M1_4. These model contains NASDAQ as a feature in them.

(a) M1_1

RMSE	0.025553224
MSE	0.000652967
MAPE	2.83452573
MAE	0.016752568
R2	0.836478961

(b) M1_2

RMSE	0.01870842
MSE	0.000350005
MAPE	2.006188642
MAE	0.011983401
R2	0.91235242

(c) M1_4

RMSE	0.02125059
MSE	0.000451588
MAPE	2.408059498
MAE	0.014652739
R2	0.886914298

Table 4.9: Predicting normalized values metrics of the NASDAQ dataset.

(a) M1_1

RMSE	5.73749947
MSE	32.91890017
MAPE	0.076206824
MAE	2.399314816
R2	0.999997427

(b) M1_2

RMSE	4.56276691
MSE	20.81884188
MAPE	0.054502324
MAE	1.727510579
R2	0.999998373

(c) M1_4

RMSE	4.774897561
MSE	22.79964672
MAPE	0.066603007
MAE	2.009664443
R2	0.999998218

Table 4.10: Predicting moving average values of the NASDAQ dataset.

(a) M1_1

RMSE	80.29814013
MSE	6447.791308
MAPE	1.069284617
MAE	33.56795279
R2	0.999498387

(b) M1_2

RMSE	63.85738223
MSE	4077.765265
MAPE	0.764418767
MAE	24.16898073
R2	0.999682766

(c) M1_4

RMSE	66.82621854
MSE	4465.743484
MAPE	0.929676058
MAE	28.11649422
R2	0.999652583

Table 4.11: Predicting close price metrics of the NASDAQ dataset.

(a) M1_1

Acc	0.8921	
Pre	0.85623003	0.91801682
Recall	0.88347914	0.89922817
F1	0.86964119	0.90852537

(b) M1_2

Acc	0.9026	
Pre	0.87157555	0.92468766
Recall	0.89146697	0.91027196
F1	0.88140905	0.91742318

(c) M1_4

Acc	0.912	
Pre	0.89040576	0.92683562
Recall	0.89311525	0.92490906
F1	0.89175845	0.92587134

Table 4.12: Predicting trend metrics of the NASDAQ dataset.

Base on Table 4.9, Table 4.10, Table 4.11, and Table 4.12. We can proudly say that, with respect to NASDAQ, model M1_2 (one-feature) and model M1_4 (two-feature) outperform model M1_1 (four-feature).

In terms of comparing between M1_2 and M1_4, we can say that the multi-feature one (M1_5) is better in the real world - when people usually look for the trend of a stock rather than at its real prices.

However, both play very good predictions, their trade-offs are tiny, so that we can pick any of them to use.

4.6.3 Deep evaluation with respect to S&P500

In terms of S&P500, we can use M1_1, M1_3, M1_4. These model contains S&P500 as a feature in them.

(a) M1_1

RMSE	0.018034203
MSE	0.000325232
MAPE	2.242464144
MAE	0.011113481
R2	0.883783973

(b) M1_3

RMSE	0.017224297
MSE	0.000296676
MAPE	2.208480955
MAE	0.01112233
R2	0.893987027

(c) M1_4

RMSE	0.016504522
MSE	0.000272399
MAPE	1.960718136
MAE	0.009845696
R2	0.902662108

Table 4.13: Predicting normalized values metrics of the S&P500 dataset.

(a) M1_1

RMSE	1.020056917
MSE	1.040516114
MAPE	0.055009158
MAE	0.34066933
R2	0.99999897

(b) M1_3

RMSE	1.016457032
MSE	1.033184897
MAPE	0.055034722
MAE	0.347453224
R2	0.999998977

(c) M1_4

RMSE	0.922520875
MSE	0.851044764
MAPE	0.048713757
MAE	0.303149401
R2	0.999999157

Table 4.14: Predicting moving average values metrics of the S&P500 dataset.

(a) M1_1

RMSE	14.27815877
MSE	203.8658177
MAPE	0.773167971
MAE	4.767608709
R2	0.999798843

(b) M1_3

RMSE	14.22776968
MSE	202.4294301
MAPE	0.771584123
MAE	4.86254813
R2	0.99980026

(c) M1_4

RMSE	12.91290642
MSE	166.7431522
MAPE	0.683400608
MAE	4.242523745
R2	0.999835472

Table 4.15: Predicting close price metrics of the S&P500 dataset.

(a) M1_1

Acc	0.8911	
Pre	0.86583744	0.90993943
Recall	0.88077678	0.90356932
F1	0.87324322	0.90674319

(b) M1_3

Acc	0.8914	
Pre	0.86586099	0.9103421
Recall	0.87746126	0.90150512
F1	0.87162253	0.90590206

(c) M1_4

Acc	0.8975	
Pre	0.87868519	0.91116987
Recall	0.87725827	0.912242
F1	0.87797115	0.91170562

Table 4.16: Predicting trend metrics of the S&P500 dataset.

Base on Table 4.13, Table 4.14, Table 4.15, and Table 4.16. We can proudly say that, with respect to S&P500, model M1_5 outperform the others.

4.6.4 Deep evaluation with respect to Exxon Mobil

In terms of Exxon Mobil, we can use M2_1 and M2_3. These model contains Exxon Mobil as a feature in them.

(a) M2_1

RMSE	0.023179541
MSE	0.000537291
MAPE	2.290878355
MAE	0.013910819
R2	0.801418014

(b) M2_3

RMSE	0.02010469
MSE	0.000404199
MAPE	2.158960848
MAE	0.01338959
R2	0.85060855

Table 4.17: Predicting normalized values metrics of the Exxon dataset.

(a) M2_1

RMSE	0.048242004
MSE	0.002327291
MAPE	0.075914309
MAE	0.01831591
R2	0.999996625

(b) M2_3

RMSE	0.03920276
MSE	0.001536856
MAPE	0.073085054
MAE	0.016113275
R2	0.999997771

Table 4.18: Predicting moving average values metrics of the Exxon dataset.

(a) M2_1

RMSE	0.675197252
MSE	0.455891329
MAPE	1.061601097
MAE	0.256277127
R2	0.999341621

(b) M2_3

RMSE	0.548681928
MSE	0.301051858
MAPE	1.020793064
MAE	0.225457042
R2	0.999565233

Table 4.19: Predicting close price metrics of the Exxon dataset.

(a) M2_1

Acc	0.8794	
Pre	0.85236517	0.90044542
Recall	0.86948059	0.88686216
F1	0.86083781	0.89360217

(b) M2_3

Acc	0.8746	
Pre	0.84454201	0.89820924
Recall	0.86721311	0.88009851
F1	0.85572743	0.88906165

Table 4.20: Predicting trend metrics of the Exxon dataset.

Base on Table 4.17, Table 4.18, Table 4.19, and Table 4.20. We can proudly say that, with respect to Exxon Mobil, model M2_3 outperform the other one. Despite the fact that M2_1 has a slightly better in predicting trend, the difference between M2_1 and M2_3 in that area is less than 1% which is not considerable. So that our multi-feature model work well in this case.

4.6.5 Deep evaluation with respect to Chervon

In terms of Chervon, we can use M2_2 and M2_3. These model contains Chervon as a feature in them.

(a) M2_2

RMSE	0.01682611
MSE	0.000283118
MAPE	1.929007573
MAE	0.010853351
R2	0.887236384

(b) M2_3

RMSE	0.017909977
MSE	0.000320767
MAPE	2.090819573
MAE	0.011799609
R2	0.872240963

Table 4.21: Predicting normalized values metrics of the Chervon dataset.

(a) M2_2

RMSE	0.062136657
MSE	0.003860964
MAPE	0.073510284
MAE	0.021515203
R2	0.999997359

(b) M2_3

RMSE	0.061813397
MSE	0.003820896
MAPE	0.07992873
MAE	0.022514326
R2	0.999997386

Table 4.22: Predicting moving average values metrics of the Chervon dataset.

(a) M2_2

RMSE	0.869664813
MSE	0.756316888
MAPE	1.031343171
MAE	0.301040856
R2	0.999484581

(b) M2_3

RMSE	0.865140471
MSE	0.748468035
MAPE	1.123356882
MAE	0.315020592
R2	0.99948993

Table 4.23: Predicting close price metrics of the Chervon dataset.

(a) M2_2

Acc	0.8841	
Pre	0.85960136	0.90432272
Recall	0.88110425	0.88647799
F1	0.87021999	0.89531145

(b) M2_3

Acc	0.8798	
Pre	0.85623658	0.89910457
Recall	0.87424016	0.88418901
F1	0.86514472	0.89158441

Table 4.24: Predicting trend metrics of the Chervon dataset.

Base on Table 4.21, Table 4.22, Table 4.23, and Table 4.24. We can proudly say that, with respect to Chervon, M2_2 and M2_3 only has a little bit better in total, our multi-feature model is less than the other one at most about 1% - that is not significant in real life problems.

4.7 Comparing to SOTA models

We will follow all the pre-processing steps before putting the data into LSTM model and RNN model as well as follow all the decoding steps to have the most fair competition between models.

4.7.1 With respect to NASDAQ

(a) LSTM

RMSE	0.022196892
MSE	0.000492702
MAPE	2.385642716
MAE	0.014343534
R2	0.876628911

(b) RNN

RMSE	0.047336384
MSE	0.002240733
MAPE	6.698210951
MAE	0.041594085
R2	0.438909067

Table 4.25: Predicting normalized values metrics of the NASDAQ dataset.

(a) LSTM

RMSE	5.125561613
MSE	26.27138185
MAPE	0.065215182
MAE	2.026053739
R2	0.999997954

(b) RNN

RMSE	9.676503936
MSE	93.63472843
MAPE	0.188923519
MAE	5.049059684
R2	0.999992717

Table 4.26: Predicting moving average values of the NASDAQ dataset.

(a) LSTM

RMSE	71.73388027
MSE	5145.749579
MAPE	0.912244328
MAE	28.34579584
R2	0.999601079

(b) RNN

RMSE	135.4257838
MSE	18340.1429
MAPE	2.624377079
MAE	70.63959959
R2	0.998580178

Table 4.27: Predicting close price metrics of the NASDAQ dataset.

(a) LSTM

Acc	0.891	
Pre	0.85934969	0.91332924
Recall	0.87460378	0.90225954
F1	0.86690964	0.90776065

(b) RNN

Acc	0.9027	
Pre	0.86621869	0.92918703
Recall	0.89894764	0.90521942
F1	0.88227974	0.91704665

Table 4.28: Predicting trend metrics of the NASDAQ dataset.

Base on result from our model (Table 4.9, Table 4.10, Table 4.11, and Table 4.12) and SOTA models (Table 4.25, Table 4.26, Table 4.27, and Table 4.28).

We can point out that our multi-feature model (M1_4) completely outperforms LSTM model and RNN model.

4.7.2 With respect to Exxon Mobil

(a) LSTM

RMSE	0.021515314
MSE	0.000462909
MAPE	2.475415234
MAE	0.015313505
R2	0.828910244

(b) RNN

RMSE	0.044859558
MSE	0.00201238
MAPE	6.108091144
MAE	0.039855796
R2	0.256180494

Table 4.29: Predicting normalized values metrics of the Exxon dataset.

(a) LSTM

RMSE	0.042892362
MSE	0.001839755
MAPE	0.083627915
MAE	0.018872521
R2	0.99999734

(b) RNN

RMSE	0.083857919
MSE	0.007032151
MAPE	0.217348967
MAE	0.045318649
R2	0.999989843

Table 4.30: Predicting moving average values of the Exxon dataset.

(a) LSTM

RMSE	0.600321653
MSE	0.360386087
MAPE	1.174699108
MAE	0.264064471
R2	0.999480945

(b) RNN

RMSE	1.173675761
MSE	1.377514792
MAPE	3.018434554
MAE	0.634098928
R2	0.998018214

Table 4.31: Predicting close price metrics of the Exxon dataset.

(a) LSTM

Acc	0.8586	
Pre	0.82496029	0.88531568
Recall	0.8509832	0.86433244
F1	0.83776971	0.87469823

(b) RNN

Acc	0.8728	
Pre	0.84051938	0.89851641
Recall	0.86831866	0.87618315
F1	0.8541929	0.88720926

Table 4.32: Predicting trend metrics of the Exxon dataset.

Base on result from our model (Table 4.17, Table 4.18, Table 4.19, and Table 4.20) and SOTA models (Table 4.29, Table 4.30, Table 4.31, and Table 4.32).

Again, the same conclusion goes here that our multi-feature model (M2_3) utterly surpass LSTM model and RNN model.

4.7.3 Visualization

Note: In Figure 4.4 and Figure 4.5, Accuracy and R2-score are performance-related metrics, and others (MAPE, MAE, RMSE, and MSE) are error-related metrics.

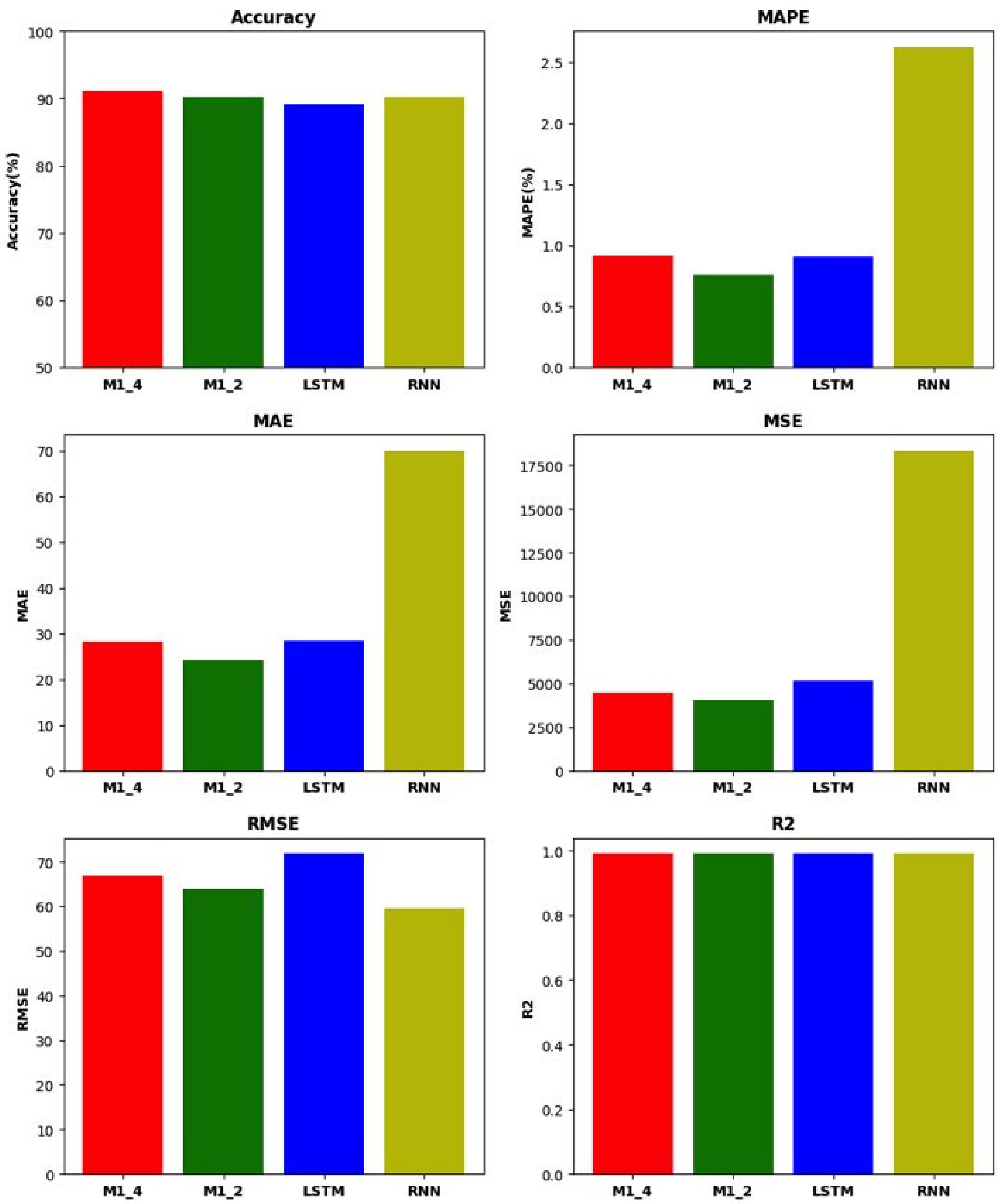

Figure 4.4: Comparing metrics among Multi-feature model, Single-feature model, LSTM model, and RNN model using NASDAQ as a target.

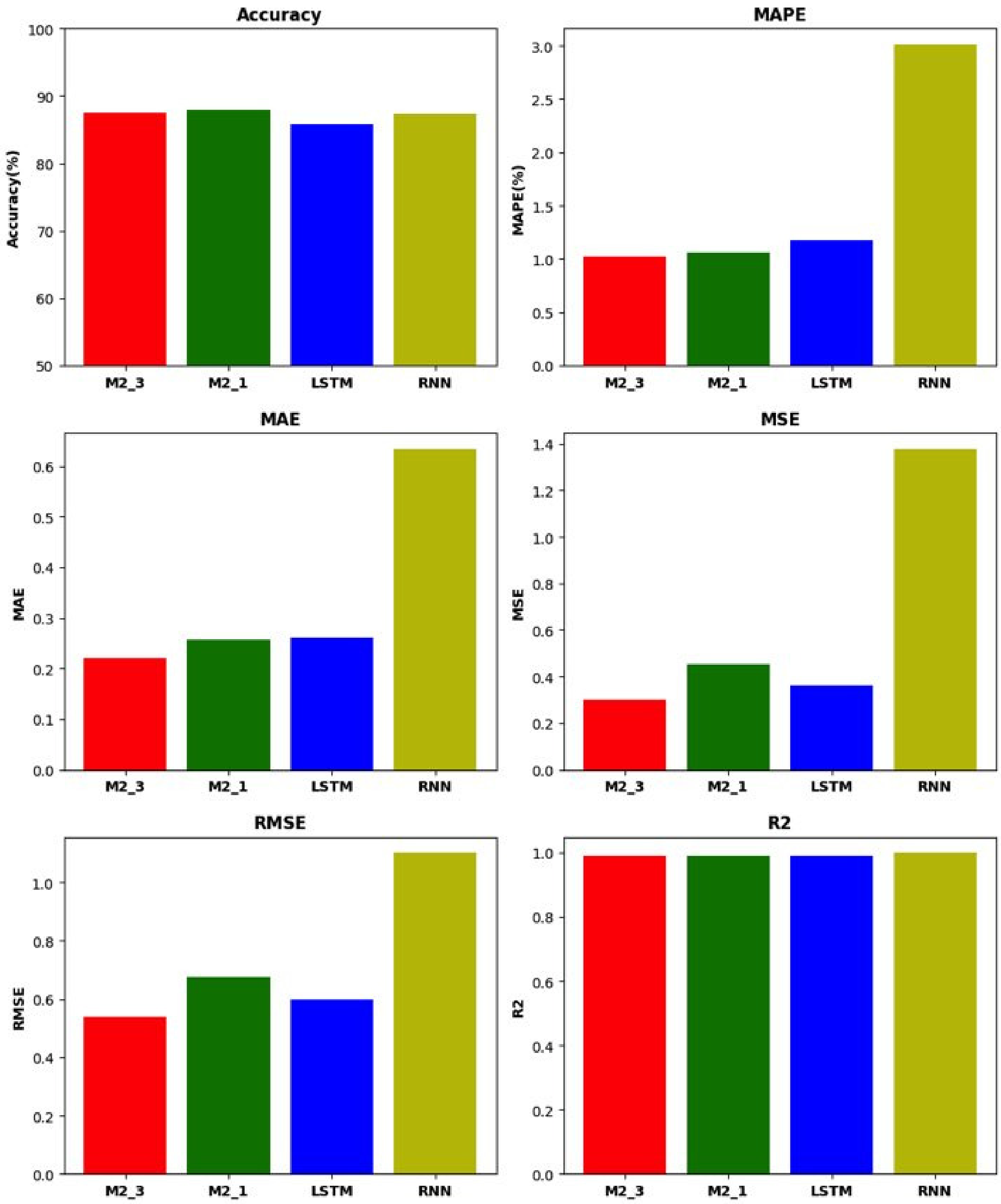

Figure 4.5: Comparing metrics among Multi-feature model, Single-feature model, LSTM model, and RNN model using Exxon Mobil as a target.

Chapter 5

Summary

In this thesis, we investigated the application of deep learning for stock price prediction, which is a challenging time series forecasting problem in the financial industry. We studied different markets by selecting correlation features to enhance the accuracy in predicting multiple stock prices, and we developed a novel neural network architecture that combines the Transformer model with Time2Vec technique as Encoder.

As we delved into our investigation, we made a surprising discovery about how well our model works. Through careful testing and analysis, we found that our model performs exceptionally well, better than simpler models and standard ones like LSTM and RNN and better than one-features model. This success is because our model looks at the bigger picture, using a mix of different features that work together seamlessly, setting a new standard for how good predictions can be in our field.

These findings show that using advanced methods for predicting can make a big difference. They also remind us how important it is to think about things like how complicated our model is and which features we use. By paying attention to these details, our study shows us a way to make predictions that work well in real-life situations. This can help people make smarter decisions and gain valuable insights in the financial world and beyond.

In summary, our thesis contributes to the ongoing discourse on the application of deep learning in finance, showcasing the effectiveness of correlation-based features and innovative neural network architectures in improving the accuracy and robustness of stock price prediction models. These advancements hold the promise of facilitating more informed investment decisions and enhancing risk management practices in the ever-evolving landscape of financial markets.

Bibliography

[1] Lasse Heje Pedersen. *Efficiently Inefficient: How Smart Money Invests and Market Prices Are Determined*. Princeton University Press, 2019.

[2] Jr. Gardner Everette S. and Ed McKenzie. "Forecasting Trends in Time Series". In: *Management Science* 31.10 (Oct. 1985), pp. 1237–1246. URL: http://www.jstor.org/stable/2631439.

[3] Artemios-Anargyros Semenoglou et al. "Investigating the accuracy of cross-learning time series forecasting methods". In: *International Journal of Forecasting* 37.3 (2021), pp. 1072–1084. ISSN: 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2020.11.009. URL: https://www.sciencedirect.com/science/article/pii/S0169207020301850.

[4] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

[5] Shiyang Li et al. "Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting". In: *CoRR* abs/1907.00235 (2019). arXiv: 1907.00235. URL: http://arxiv.org/abs/1907.00235.

[6] Ashish Vaswani et al. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[7] Accessed: 2024-05-10. URL: https://victorzhou.com/series/neural-networks-from-scratch/.

[8] Souhaib Ben Taieb, Antti Sorjamaa, and Gianluca Bontempi. "Multiple-output modeling for multi-step-ahead time series forecasting". In: *Neurocomputing* 73 (June 2010), pp. 1950–1957. DOI: 10.1016/j.neucom.2009.11.030.

[9] Massimiliano Marcellino, James H. Stock, and Mark W. Watson. "A Comparison of Direct and Iterated Multistep AR Methods for Forecasting Macroeconomic Time Series". In: *Journal of Econometrics* 135.1-2 (2006), pp. 499–526. DOI: 10.1016/j.jeconom.2005.07.013.

[10] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323.6088 (1986), pp. 533–536. ISSN: 1476-4687. DOI: 10.1038/323533a0. URL: https://doi.org/10.1038/323533a0.

[11] Tianyu Wang et al. "From model-driven to data-driven: A review of hysteresis modeling in structural and mechanical systems". In: *Mechanical Systems and Signal Processing* 204 (2023), p. 110785. ISSN: 0888-3270. DOI: https://doi.org/10.1016/j.ymssp.2023.110785. URL: https://www.sciencedirect.com/science/article/pii/S0888327023006933.

[12] Accessed: 2024-05-10. URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[13] Jia Wang et al. "CLVSA: A Convolutional LSTM Based Variational Sequence-to-Sequence Model with Attention for Predicting Trends of Financial Markets". In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. IJCAI-2019. International Joint Conferences on Artificial Intelligence Organization, Aug. 2019. DOI: 10.24963/ijcai.2019/514. URL: http://dx.doi.org/10.24963/ijcai.2019/514.

[14] Fuli Feng et al. *Enhancing Stock Movement Prediction with Adversarial Training*. 2019. arXiv: 1810.09936 [q-fin.TR].

[15] Seyed Mehran Kazemi et al. "Time2Vec: Learning a Vector Representation of Time". In: *CoRR* abs/1907.05321 (2019). arXiv: 1907.05321. URL: http://arxiv.org/abs/1907.05321.

[16] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Minneapolis, Minnesota: Association for

Computational Linguistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423.

[17] Qingsong Wen et al. *Transformers in Time Series: A Survey*. 2023. arXiv: 2202.07125 [cs.LG].

[18] Accessed: 2024-05-10. URL: https://blog.stackademic.com/what-i-learned-from-creating-a-large-language-model-from-scratch-dd9a4dc6a73d.

[19] Accessed: 2024-05-10. URL: https://optimization.cbe.cornell.edu/index.php?title=RMSProp.

[20] Accessed: 2024-05-10. URL: https://optimization.cbe.cornell.edu/index.php?title=AdaGrad.

[21] Jian-Hang Li et al. "Multi-Head Attention-Based Hybrid Deep Neural Network for Aeroengine Risk Assessment". In: *IEEE Access* 11 (2023), pp. 113376–113389. DOI: 10.1109/ACCESS.2023.3323843.

[22] Mandella Ali M. Fargalla et al. "TimeNet: Time2Vec attention-based CNN-BiGRU neural network for predicting production in shale and sandstone gas reservoirs". In: *Energy* 290 (2024), p. 130184. ISSN: 0360-5442. DOI: https://doi.org/10.1016/j.energy.2023.130184. URL: https://www.sciencedirect.com/science/article/pii/S0360544223035788.

[23] Shunrong Shen, Haomiao Jiang, and Tongda Zhang. "Stock Market Forecasting Using Machine Learning Algorithms". In: 2012. URL: https://api.semanticscholar.org/CorpusID:16643114.

[24] Accessed: 2024-05-05. URL: https://www.finance.yahoo.com.

[25] Accessed: 2024-05-08. URL: https://wikipedia.org/wiki/Residual_neural_network.

List of Figures

List of Tables