Report on the LaTeX Section

Kélian PONS
1$^{\text{st}}$ year Master student, Master Data Science
Université de Lille

September 20, 2025

Contents

List of Figures

List of Tables

List of Algorithms

Introduction

Please note that this report is written in the context of the LaTeX course, so the content and the different sections may not have a link together. The first part will give some comments, thoughts, and remarks on the review of Bao et al., 2025. This paper was published in August 2024 and explores the field of stock price prediction, especially with neural networks. The second part is here to show case the use of the algorithm2e package with some examples from the course Algorithm and their complexity.

1 Data-Driven Stock Forecasting Models

The different stock forecasting models can be split into *3 categories* depending on the types of data that are given as input:

- **Price-driven** models: These models only use the historical data of the prices of different stocks. Technical indicators can also be used as they are computed from the price data.
- **Event-driven** models: These models use other data than the stock prices to make their prediction. For example, a model can have access to financial articles or social network posts to predict market trends.
- **Hybrid** models: They are the combination of the two above.

The Table 1 shows some examples of indexes often used for training and testing models for stock forecasting.

Abbreviation	Country	Detail
CSI 100	China	China Securities 100 Index
NIKKEI 225	Japan	Nikkei 225 Index
NIFTY 50	India	National Stock Exchange of India 50 Index
FCHI	France	France CAC 40 Index
DAX	Germany	Deutscher Aktienindex
FTSE 100	UK	Financial Times Stock Exchange 100 Index
NASDAQ	USA	NASDAQ Composite Index
S&P 500	USA	Standard & Poor's 500 Index

Table 1: Stock market and exchange information

We can also differenciate the model depending on the type of network used. This report will focus on Long Short Term Memory networks and Transformer based networks.

1.1 Recurrent Neural Networks

Recurrent neural networks (RNNs) can handle time series. To do so, they process the sequence of data one element at the time. The output of the previous state is given as part of the input for the next state. However classic

RNNs are subject to gradient vanish. For long sequences, old information tends to fade throughout the iteration. This is why, Sepp and Jürgen, 1997 developed the long-short term memory (LSTM). The state of a LSTM cell is described by these equations:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{1}$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2}$$
$$g_t = \tanh(W_g \cdot [h_{t-1}, x_t] + b_g) \tag{3}$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{4}$$
$$c_t = f_t * c_{(t-1)+i_t} * g_t \tag{5}$$
$$h_t = o_t * \tanh(c_t) \tag{6}$$

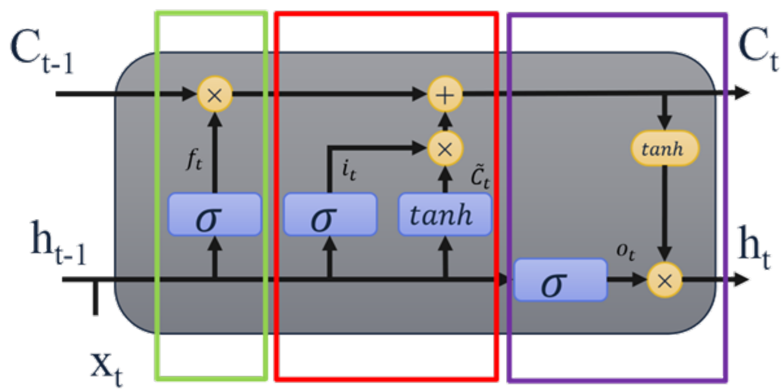

Figure 1: LSTM architecture

LSTM are very popular for time series prediction, and so for stock price forecasting. While LSTM handle long term information better then classic RNN, they still struggle if the serie is too long. In the next part we will see how transformer model can handle time serie data.

1.2 Transformer

Transformers are a model architecture develop by Vaswani et al., 2023. Transformers were first used in natural language processing (NLP) tasks such as translation. They are the base of the large language models (LLMs) such as Gpt or Bert.

Transformers are often used with text data, which can be seen as sequential data. The key process for transformer-based models is the attention mechanism, see figure 2. The first step of the attention mechanism is to compute, from the input, the 3 matrices : Q (Query), K (Key),V (Value). Then we need to compute the attention score given by this equation:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{7}$$

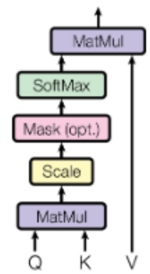

Figure 2: Attention mechanism

For more information on the attention mechanism you can see The Illustrated Transformer written by Jay Alammar from this link: https://jalammar.github.io/illustrated-transformer/

2 Algorithm feature

Algorithm 2.1: Algorithm f1

```
function f1 (a: Array of reals): Real
    input  : a: Array of Reals
    output: result: ?
    result ← 0
    for i from 0 to length(a) - 1 do
        result ← result + a[i]
    end
    result ← result / length(a)
    return result
end
```

The first algorithm f1 computes the **average** value of an array of values in $\mathbb{R}$. The complexity of this algorithm is $\Theta(n)$ with n the length of the input array.

Algorithm 2.2: Algorithm f2

```
function f2 (a: Array of reals): Real
    input  : a: Array of Reals
    output: result: ?
    m : Real
    result ← 0
    m ← f1(a)
    for i from 0 to length(a) - 1 do
        result ← result + (a[i] - m) * (a[i] - m)
    end
    result ← result / length(a)
    return result
end
```

This second algorithm f2 computes the *variance* of the input array. The complexity of the algorithm is still $\Theta(n)$ even if there are twice as many operation.

Bibliography

Bao, W., Cao, Y., Yang, Y., Che, H., Huang, J., & Wen, S. (2025). Data-driven stock forecasting models based on neural networks: A review. *Information Fusion*, *113*, 102616. https://doi.org/10.1016/j.inffus.2024.102616

Sepp, H., & Jürgen, S. (1997). Long Short-Term Memory. *Neural Computation*, *9*, 1735–1780.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2023, August). Attention Is All You Need. https://doi.org/10.48550/arXiv.1706.03762