Investigation into the use of long-short-term memory recurrent units on the prediction of stock prices

SN:18019421

March 2022

Abstract

The task of predicting prices of stocks is very challenging amidst a volatile market. Hence, there is a need for models that are more adaptable for this ever-changing environment. Considering the time-series nature of the stock market, recurrent neural networks have become a more common tool used for forecasting. The objective of this report is to compare the performance of a stacked LSTM against a baseline RNN for the prediction of MSFT stocks. Additionally, further tests were completed for NVDA and FB stocks. The results of the investigation highlight some of the challenges in the task as well as possible modifications for future improvement. Furthermore, the report shows how the complexity of a recurrent model can impact its generalisation on time-series.

1 Introduction

The stock market can be an incredibly unpredictable ecosystem encompassing uncertainty at various aspects of the domain. This volatility can be attributed to various economic factors and market forces which have an impact on the demand, cost and accessibility of stocks. Other factors include rates associated with inflation, interest tax changes and changes within corporate industries. The process of investing in commodities is not easy and hence the before mentioned conditions make it difficult to predict future rates of stocks.

In the past 5 years, there has been significant research into the use of machine learning models for predicting prices of single assets as well as whole portfolios. The question that might be asked is what makes machine learning a suitable tool for forecasting stocks? Machine learning focuses on acquiring implicit information from the data, extracting the meaning in patterns and using it to improve on predictions through a learning process [4]. Furthermore, emphasis should be highlighted on 'learning' as an increase in data volume significantly increases this process. Furthermore, the vast availability of stock data is effective with many data hungry ML algorithms such as neural networks [6].

The processes within the methodology of traditional ML algorithms are significantly different to that used in deep learning algorithms. A key difference is that deep learning utilises neural networks, an architecture consisting of input layers, hidden layers and output layers. A neural network can be considered a form of a 'black box' system because looking at the layers or the architecture does not reveal any detail of the what the function is that its trying to estimate. This means there is no clarity in the conversion takes the inputs to the output. [10].

Additionally, Recurrent Neural Networks (RNNs) are a group of neural networks which incorporate the dependency of the output and any one input on the previous inputs. This 'memory' property allows for the network to process sequential data such as the stock time-series. Specifically, Long-Short-Term-Memory networks are RNNs that utilise a special recurrent unit (module) that are capable of learning long-term dependencies as well

short-term ones within sequences [7]. This is functionally important in the context of stock time-series because current prices of commodities might be dependent on prices and events that have happened relatively long time ago. Furthermore, it allows for the model to make more accurate long-term forecasts into the future [9][1].

Various features of the LSTM suggest it is more suitable than a simple RNN. Firstly, the presence of the cell state plays a key role in one module of an LSTM. The cell state stores the long-term dependencies which allow the present unit to receive information from inputs at time steps that occurred significantly back in time. Additionally, the derivatives of the cell state lessen the effects of vanishing and exploding gradients [8]. The vanishing gradient can be significant in simpler RNNs because the recurrent weight within the temporal loops will tend to 0 on continuous multiplication during the updates. LSTMs aim to reduce this by setting the recurrent weight to 1, given that greater values lead to exploding gradients and smaller values lead to vanishing gradients [2].

The following investigation aims to further delve into volatility and how LSTM recurrent units can be used as time series models for the prediction of stock prices. Furthermore, the performance of the LSTM model will be compared with those of a simple recurrent neural network as a baseline model.

2 Methodology

2.1 Architecture

The following section goes into the mathematical foundation of LSTMs and some of key steps underlying the proposed model. Note the features of the architecture are defined by the variables below:
Loss function E, weights vector W, forget gate layer (f_t), input gate layer (i_t), the learning rate (α), activation functions including the sigmoid function ($\sigma()$) and hyperbolic tangent function ($Tanh()$), present hidden state and the output into the next module (h_t), previous hidden state (h_{t-1}), bias for the forget gate layer (b_f), bias for the input gate layer (b_i), new data input at time t (x_t), candidate vector ($\widetilde{C_t}$), cell state (memory) at time t (C_t), output gate layer (o_t).

The architecture proposed by the investigation consists of a stacked LSTM network. For the purpose of this study, an LSTM recurrent unit was used with 4 layers with one module of the network. 8 LSTM layers are stacked on top of each other and then connected to the one dense layer. Dropout was added after each LSTM layer. On the contrary, the baseline RNN consists of a simple recurrent layer, dropout and a dense layer.

For a simple RNN (this is shared for an LSTM):
The loss function, E chosen for training of the model was the Mean Squared Error (MSE). The MSE is defined as the mean of the squared difference between the target and the predicted values. The reason for this was because it is suitable for a regression problem like the one presented in this study. Additionally, it results in a value that is always positive, which ensures the sum is always equal to or greater than 0.
Furthermore, the effect of squaring the differences means that greater errors mean that more loss than for smaller errors. Hence, the system gets punished more for bigger errors.

$$\frac{\partial E}{\partial W} = \sum_{t=1}^{T} \frac{\partial E_t}{\partial W} \tag{1}$$

The weights are updated simply by subtracting the partial derivative of the error with respect to the weights vector W, shown by equation 1. The derivative is scaled by multiplying it with the learning rate.

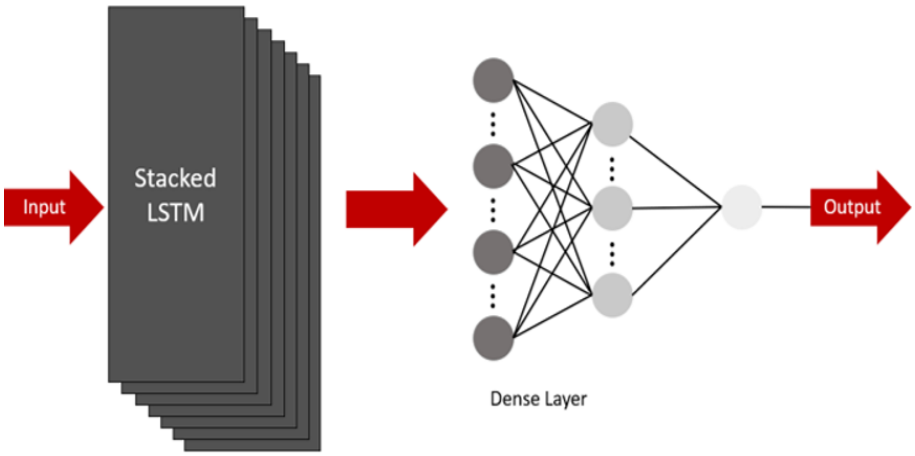

Figure 1: *ORIGINAL* Architecture of the LSTM model, consisting of 8 stacked LSTM layers and a dense layer for the output. Note this diagram was generated in Microsoft PowerPoint [3].

$$W_{t+1} = W_t - \alpha \times \frac{\partial E}{\partial W} \tag{2}$$

where α is the learning rate.

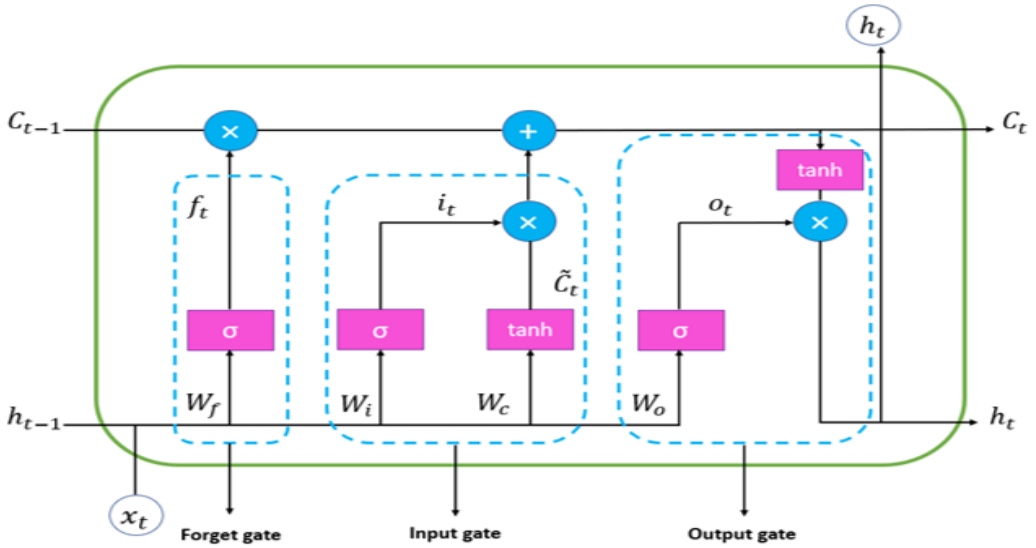

Figure 2: *ORIGINAL* Diagram showing module of an LSTM network, where 4 C_t is the cell state at time t and h_t is the hidden state at time t. Note the diagram has been generated using Microsoft PowerPoint [5].

The structure of an LSTM is a lot more complex than that of an RNN. Each of the LSTM modules contains a forget gate layer, input gate layer, candidate vector layer and a final layer. The calculations that utilised in each layer can be found below.

The forget gate layer:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{3}$$

Note the input for the network is the $W_f(h_{t-1,x_t})$. This takes into account the hidden state from the previous module, h_{t-1} and new input data, x_t.

Input gate layer:

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{4}$$

A layer which creates new candidate vectors:

$$\widetilde{C}_t = Tanh(W_c[h_{t-1}, x_t] + b_c) \tag{5}$$

The $Tanh()$ activation function is used here to ....

The update process for the cell state from C_{t-1} to C_t involves the multiplication of the of the old cell state C_{t-1} by the output of the function f_t from the forget gate layer. Following this, the candidate vector, $\widetilde{C_t}$ is scaled by multiplying it with the function output of the input gate layer, i_t. The scaled candidate vector, $i_t \times \widetilde{C_t}$ is then added to the cell state. The cell state at present time, t:

$$C_t = f_t \times C_{t-1} + i_t \times \widetilde{C}_t \tag{6}$$

The final layer of module (output):

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{7}$$

Hidden state output at time t:

$$h_t = o_t \times Tanh(C_t) \tag{8}$$

2.2 Hyperparameters

Dropout was added after each LSTM layer in the network for the purpose removing some neurons randomly and hence making the model less prone to overfitting, by reducing co-adaptation. Co-adaptation is a phenomena when different hidden units exhibit similar behaviour, which can lead to overfitting during the training phase.

2.3 Data

The data used in this study has been obtained and imported from the Yahoo Finance website, by using the pandas datareader. Three sets of data were used to assess the performance of the model. This included historical data from the Microsoft corporation, Meta corporation and NVIDIA corporation. The data used for the study ranged from 01-01-2010 to 28-02-2022.

The data was split manually into training and testing sets by changing the dates in the pandas data reader. The train set ranged from 01-01-2010 to 14-01-2022 and the test set ranged form 14-01-2022 to 13-02-22.

2.3.1 Data cleaning and pre-processing

One of the key steps in the data pre-processing was the creation of arrays for the inputs and arrays for the target values. The inputs array was reshaped into a 3 dimensional tensor. This is crucial to ensure the input shape for the network is correct.

2.3.2 Net returns and feature scaling

Feature scaling is the normalisation of data. This is important in machine learning because the magnitude of one data point may have a significant impact on the representation of relatively small points. Furthermore, normalisation was used instead of standardisation because the output layer used the sigmoid activation function. In regard to deep learning, the normalisation of data is crucial to allow for the same update rate of the weights at

each time step during gradient descent. Normalization was achieved using the standard normalization equation:

$$X_{normalized} = \frac{(X - X_{minimum})}{(X_{maximum} - X_{minimum})} \tag{9}$$

In the context of stock time series, net return were calculated for each close price at time t, P_t. This was done at the start to first initially analyses the data, of which figures can be found in the results section below. The return on an asset R_t between times $t-1$ and t is given by:

$$R_t = \frac{(P_t - P_{t-1})}{P_{t-1}} = \frac{P_t}{P_{t-1}} - 1 \tag{10}$$

In the following the figure the returns were calculated by letting the P_t be the close price and P_{t-1} be the open price.

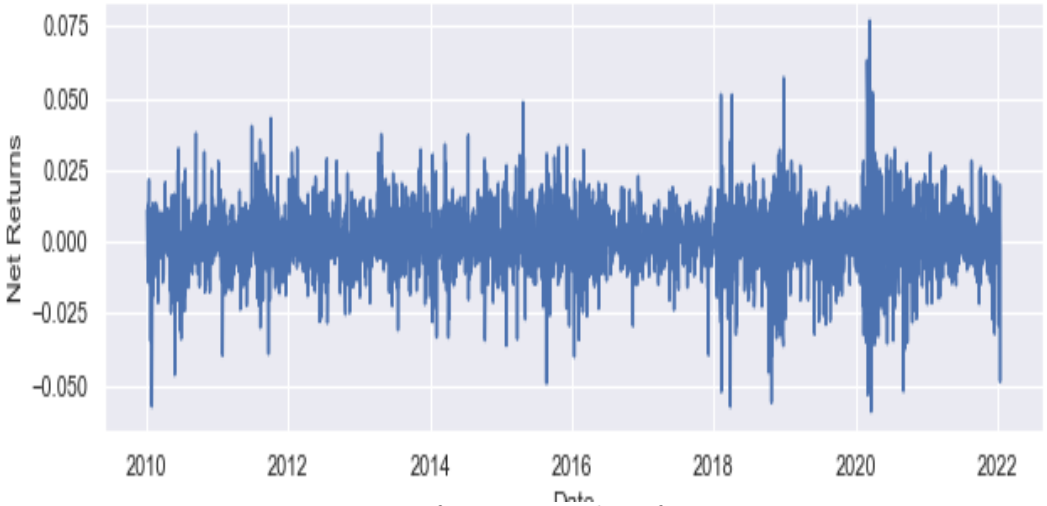

Figure 3: Net returns for training data from 2010 to 2022.

3 Results

Training was performed on the LSTM complex model and baseline RNN model using the MSFT data (with 150 epochs). Furthermore, the same LSTM model was trained also trained on NVDA data (with 150 epochs) and on FB data (with 80 epochs). This section presents the analysis of these experiments.

3.1 Experiments with MSFT stocks

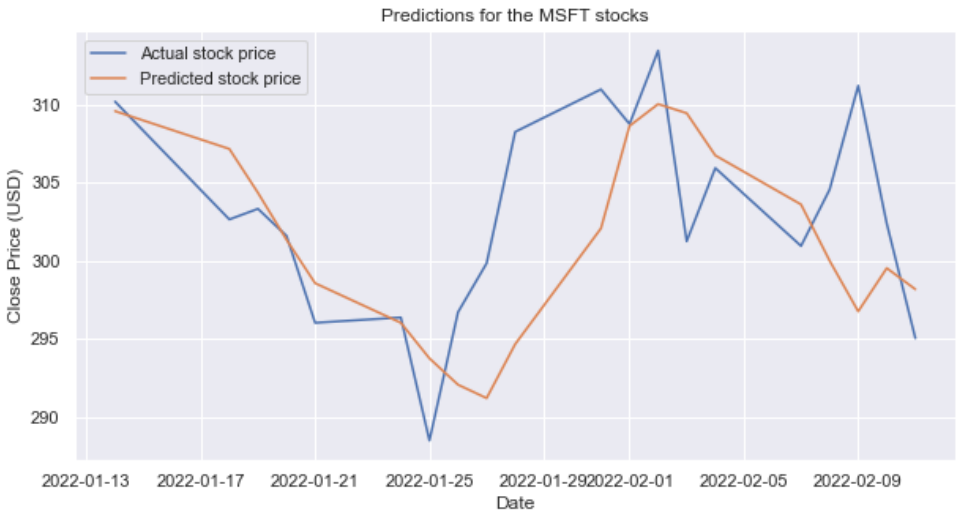

Figure 4: Line plots for the performance of the LSTM complex model on the testing data ranging from mid January 2022 to mid February 2022 for the MSFT stock. Note the yellow line represents the predicted stock price and the blue line represents the actual stock price at each respective date. The training of the model used 150 epochs.

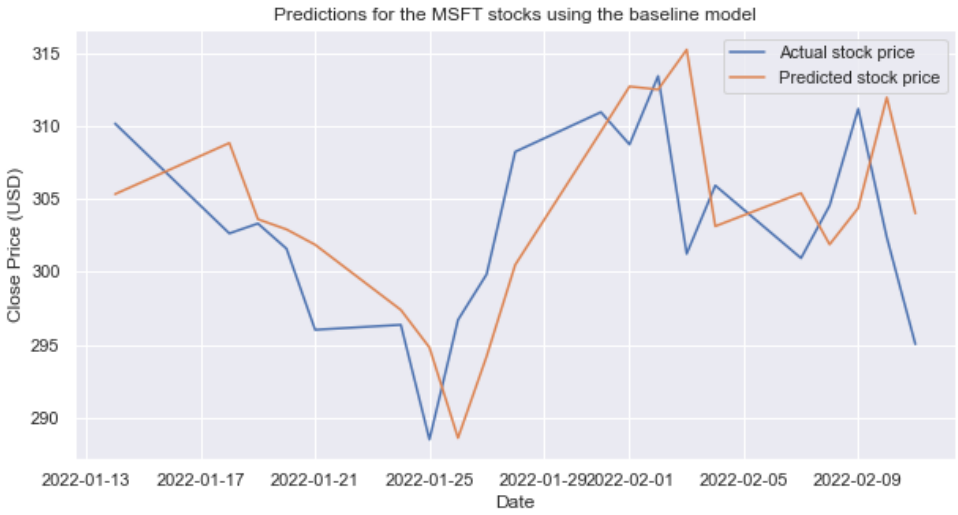

Figure 5: Line plots for the performance of the baseline RNN model on the testing data ranging from mid January 2022 to mid February 2022 for the MSFT stock. Note the yellow line represents the predicted stock price and the blue line represents the actual stock price at each respective date. The training of this model used 150 epochs.

From figure 4, it can be inferred visually that the predicted prices follow the general trend of the actual stock prices, however there is a clear time gap visible between the two lines. This gap varies quite significantly for different dates, however from looking at the time positions of the peaks it could be estimated that the time period is between 1 to 2 days. Similarly in figure 5, the predictions of the baseline model are also 1 to 2 days behind of the actual prices. However, the baseline model has a lot sharper peaks compared to the LSTM model which has a lot more broader peaks. Comparing the predicted and actual prices in figure 5, one can infer that the predicted peaks are typically higher than the actual price peaks. On the other hand, figure 4 shows that the predicted peaks are normally lower than the respective peaks for the actual prices. This further highlights that the LSTM model makes more general predictions.

3.2 Experiment with NVDA stocks

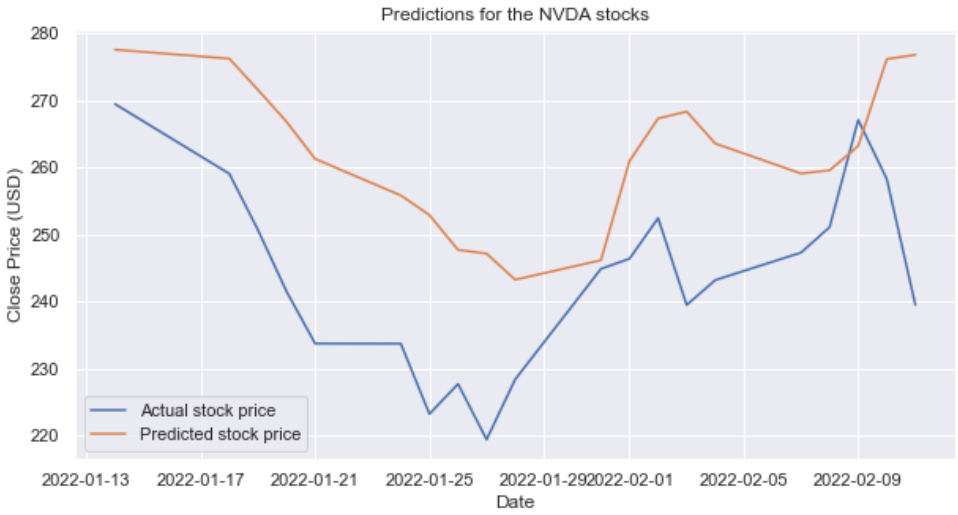

Figure 6: Line plots for the performance of the LSTM model on the testing data ranging from mid January 2022 to mid February 2022 for NVDA stock. Note the yellow line represents the predicted stock price and the blue line represents the actual stock price at each respective date. The training of this model used 150 epochs.

The LSTM model was tested to determine if its predictive behaviour and performance would be reproducible on a different dataset. For this experiment, the number of epochs and the amount of data was maintained constant as control variables, and the dataset was the independent variable that was changed. In contrast to the MSFT data in figure 4, the LSTM model continuously predicts higher above the respective actual stock prices. Another feature found in figure 6 is that the shape of the predicted stocks matches more closely to the shape of the curve for the actual prices. Furthermore, the predicted stock prices are no longer behind as much as for the MSFT data, and the time points for peaks and respective curve features seem to match a lot more. This indicates that model has a predisposition to predict the NVDA stocks more favourably than for MSFT.

3.3 Experiment with FB stocks

The above LSTM model was trained and tested on different data with less epochs. As shown in figure 7, the predictions maintain a very general trend that follows the direction of the actual stock prices. Nonetheless, the model does make any predictions that are significantly different to the previous prediction. As a result, the predictions curve does have many sharp spikes and or high gradients. This behaviour is concurrent with results in figures 4 and 6.

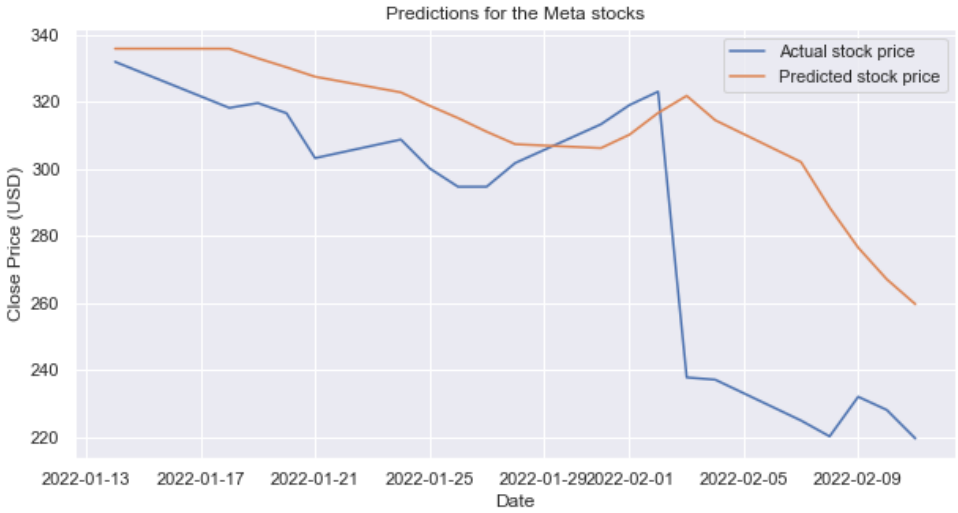

Figure 7: Line plots for the performance of the LSTM model on the testing data ranging from mid January 2022 to mid February 2022 for FB stock. Note the yellow line represents the predicted stock price and the blue line represents the actual stock price at each respective date. The training of this model used 80 epochs.

4 Discussion

The presence of broader peaks in the curve for the LSTM model is indicative of the model making more general predictions relative to the simple model. This behaviour could be attributed due to the LSTM net having more dropout in its architecture, compared to the simple model which only has 1 dropout. This indicates the more complex model tends to generalise its predictions rather more than making definite predictions which may be far from the general trend of the data. On the contrary, the simple model appears to be more likely to make more extreme predictions, as evident by the sharp peaks.

A possible reason for the predicted prices lagging behind the actual prices could be that the models detect changes in direction and hence tries to make future predictions based on those past observations. However, due to the volatility of stocks, as previously mentioned the direction of the actual prices can change rapidly from one day to another. The model may not be capable of making predictions at the same rate at which stock market prices change. A possible way of improving this process could be to use live stock data that is changing during the day.

In conclusion, the investigation has clearly shown evidence of the volatility of stocks and its effect on the ability of models to make accurate predictions. Therefore, models must be designed with consideration for the data that will be used by them for processing. As presented by figures 4 and 5, increasing a model's complexity has the potential to increase its ability to generalise and focus more on the trend and direction of the data rather than specific values.

One of the limitations of the investigation was that the differences between the actual and predicted stock prices were not further interpreted numerically. Future work could involve determining if the difference between the two curves is statistically significant. This could be done through the use of hypothesis testing techniques with a null hypothesis that there is no significant difference between the predicted and actual stock prices. Furthermore, standard errors could be calculated for predictions. Additionally, a wider range of experiments could be carried out such as the use of the same LSTM model with the same data but with different numbers of epochs to determine epoch effects on training. Also, models could be tested for longer time periods as well as tested for greater price ranges.

References

[1] Manuel Camargo, Marlon Dumas, and Oscar González-Rojas. Learning accurate lstm models of business processes. In *International Conference on Business Process Management*, pages 286–302. Springer, 2019.

[2] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[3] Ioannis E Livieris, Emmanuel Pintelas, and Panagiotis Pintelas. A cnn–lstm model for gold price time-series forecasting. *Neural computing and applications*, 32(23):17351–17360, 2020.

[4] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. Statistical and machine learning forecasting methods: Concerns and ways forward. *PloS one*, 13(3):e0194889, 2018.

[5] Pulkit Mehndiratta and Devpriya Soni. Identification of sarcasm in textual data: A comparative study. *Journal of Data and Information Science*, 4(4):56–83, 2019.

[6] Adil Moghar and Mhamed Hamiche. Stock market prediction using lstm recurrent neural network. *Procedia Computer Science*, 170:1168–1173, 2020.

[7] Nguyet Nguyen and Mohammad Islam. Comparison of financial models for stock price prediction. In *2021 Joint Mathematics Meetings (JMM)*. AMS, 2021.

[8] Khandu Om, Spyros Boukoros, Anupiya Nugaliyadde, Tanya McGill, Michael Dixon, Polychronis Koutsakis, and Kok Wai Wong. Modelling email traffic workloads with rnn and lstm models. *Human-centric Computing and Information Sciences*, 10(1):1–16, 2020.

[9] Jiayu Qiu, Bin Wang, and Changjun Zhou. Forecasting stock prices with long-short term memory neural network based on attention mechanism. *PloS one*, 15(1):e0227222, 2020.

[10] Yi-han Sheu. Illuminating the black box: Interpreting deep neural network models for psychiatric research. *Frontiers in Psychiatry*, page 1091, 2020.