FORECASTING THE CLOSING PRICE OF AMAZON'S STOCK USING RECURRENT NEURAL NETWORKS, LONG SHORT-TERM MEMORY AND TRANSFORMERS

Lappeenranta-Lahti University of Technology LUT

BM20A6100 Advanced Data Analysis and Machine Learning

Project: Stock A1 (AMAZON)

2023

Aime Barema

Andry Andriamananjara

Socrates Waka Onyando

LIST OF ABBREVIATIONS

AMZN	Amazon
GPU	Graphics Processing Unit
LOCF	Last-Observation-Carried-Forward
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
NASDAQ	National Association of Securities Dealers Automated Quotations
R^2	Coefficient of Determination
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
TNN	Transformers

CONTENTS

1 INTRODUCTION

1.1 Context

Forecasting stock prices is a challenging task that has captivated many, ranging from financial analysts, researchers to investors alike. This complex task requires individuals to evaluate a wide array of factors, including: market trends, economic indicators, company performance and global events [1] [2]. As researchers strive to figure out the patterns influencing stock movements, advancements in technology and data analytics play an important role in enhancing the predictions accuracies [3]. In today's world, machine learning and deep learning algorithms provide significant benefits compared to traditional methods like technical and fundamental analysis [1] [3].

1.2 Company

For this project, focus is on the stock prices of Amazon (AMZN). AMZN is a global e-commerce giant whose shares are listed on the National Association of Securities Dealers Automated Quotations (NASDAQ) under the ticker symbol AMZN. The company has a diverse portfolio of businesses, including e-commerce, cloud computing, digital streaming services and artificial intelligence services [4].

1.3 Project Background and Objectives

The present project focuses on harnessing the power of machine learning to improve stock price prediction. Particularly, the aim of the project is to exploit the advantages offered by artificial neural networks such as Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM) and Transformers (TNN) to forecast the closing price of AMZN stock. By utilizing these advanced techniques on multivariate data, the objective is to enhance the accuracy and explore the time horizon for predicting the closing price of AMZN stock.

2 MATERIALS AND METHODS

The methodology is broken down into four main phases: i) the data pretreatment phase, which involves several preprocessing techniques; ii) model training; iii) sensitivity analysis and iv) model performance evaluation.

2.1 Data

The provided datasets contains stock price information for AMZN for 3019 business days, starting from 3rd January 2006 to 29th December 2017. The dataset has 6 variables: Date (the date of the trading period); Open Price (the price of the stock at the beginning of the day); High Price (the highest price reached during the day); Low Price (the lowest price reached during the day); Close Price (the price of the stock at the end of the day); and Volume (the total number of shares traded during the day). Notably, there are no missing values. The data has been visualised to show variation over time (See Appendices 1).

2.2 Data Pretreatment

The dataset comprises continuous measurements of AMZN stock prices, recorded on business days while accounting for disruptions due to US public holidays. To address irregular sampling, initial resampling to business days precedes the application of the Last-Observation-Carried-Forward (LOCF) imputation method, assuming that the market value on a holiday mirrors that of the preceding day. A distribution of 80% for training and 20% for testing is implemented. This strategy ensures learning from historical data and evaluating on unforeseen future instances. Sequences are derived from the dataset, adhering to specified length and step size parameters for subsequent sequence-based modeling. Data normalization is applied to facilitate the stability, convergence, and efficacy of the neural networks in capturing temporal dependencies within sequential data.

2.3 Model Topology

2.3.1 RNN

The RNN model architecture is detailed in Table 1. The model was implemented using the Keras framework. The RNN structure comprises a SimpleRNN layer followed by a Dense layer, a Dropout layer for regularization, and a final Dense layer with a hyperbolic tangent activation function. The

RNN model consists of 20,701 parameters, all of which are trainable.

Table 1. RNN Memory Structure

Layer (type)	Output Shape	Param #
simple_rnn_1 (simpleRNN)	(None, 100)	10500
dense (Dense)	(None, 100)	10100
dropout (Dropout)	(None, 100)	0
dense_1 (Dense)	(None, 1)	101
Total params: 20701		
Trainable params: 20701		
Non-trainable params: 0		

2.3.2 LSTM

As shown in Table 2, the proposed model architecture consisted of an LSTM layer, followed by a Dense layer and a Dropout layer to mitigate overfitting. The final layer employed the hyperbolic tangent (tanh) activation function. The RMSprop optimizer with a specified learning rate was utilized, and the mean squared error was chosen as the loss function for training. The total trainable parameters amounted to 52,201. The absence of non-trainable parameters underscored the simplicity of the model architecture.

Table 2. LSTM Memory Structure

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 100)	42000
dense (Dense)	(None, 100)	10100
dropout (Dropout)	(None, 100)	0
dense_1 (Dense)	(None, 1)	101
Total params: 52201		
Trainable params: 52201		
Non-trainable params: 0		

2.3.3 Transformers

Table 3 provides an overview of the architecture of the transformer-based model. The input shape is (None, 100, 1), representing a sequence of 100 time steps with one feature. The model comprises four transformer blocks, each contributing 28672 parameters. The Global Average Pooling layer reduces the spatial dimensions, and a Multilayer Perceptron (MLP) with one layer and 128 units introduces non-linearities. The output layer produces a single value. The total trainable parameters amount to 45213, with no non-trainable parameters. This architecture is designed for sequence-to-sequence prediction tasks, utilizing transformer blocks for capturing temporal dependencies and an MLP for feature extraction and mapping to the output.

Table 3. TNN Model Structure

Layer Type	Output Shape	Param #	Additional Details
Input (input_1)	(None, 10, 5)	0	Raw input data
Lambda (tf.__operators__.add)	(None, 10, 5)	0	Element-wise addition
Layer Normalization	(None, 10, 5)	10	Normalization
Multi-Head Attention	(None, 10, 5)	23,557	4 attention heads
Dropout	(None, 10, 5)	0	Dropout regularization
Conv1D	(None, 10, 4)	24	1D Convolution
Conv1D	(None, 10, 5)	25	1D Convolution
Global Avg. Pooling1D	(None, 10)	0	Global average pooling
Dense	(None, 128)	1,408	Fully connected layer
Dropout	(None, 128)	0	Dropout regularization
Dense	(None, 1)	129	Output layer
Total params: 96,041			
Trainable params: 96,041			
Non-trainable params: 0			

2.4 Sensitivity Analysis

In order to comprehensively assess the robustness and performance of the LSTM model, a sensitivity analysis was conducted. This phase aimed to scrutinize the impact of various model parameters on their predictive capabilities. The sensitivity analysis involved systematically varying these parameters while monitoring the training and test errors. This comprehensive evaluation provided valuable insights into the model's behavior under different configurations, aiding in the selection of optimal parameters for enhanced predictive performance.

2.5 Performance Evaluation

The prediction results were evaluated by Root Mean Square Error (RMSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE) and Coefficient of Determination (R^2). In terms of interpretation, the smaller the RMSE, MAE and MAPE, the closer the predicted value to the true value. On the other hand, the closer the R^2 to 1, the better the fit of the model. The equations for the RMSE, MAE, MAPE and R^2 are illustrated in Equations 1, 2, 3 and 4 respectively.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(a_i - p_i)^2} \tag{1}$$

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|(a_i - p_i)| \tag{2}$$

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}|\frac{a_i - p_i}{a_i}| \tag{3}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(a_i - p_i)^2}{\sum_{i=1}^{N}(a_i - \overline{a_i})^2} \tag{4}$$

where a_i is the actual price of stock and p_i is the predicted price of stock on day i, N is the total number of stock trading days considered and $\overline{a_i} = \frac{1}{N}\sum_{i=1}^{N}(a_i)$.

2.6 Software and Code

The project's analyses were conducted using Python. The complete source code can be accessed on Github.

3 RESULTS AND DISCUSSION

3.1 Model Performance

The evaluation metrics RMSE, MAE, R^2 and MAPE are taken into consideration. The results of these evaluation measures are shown in Table 4 for the RNN, LSTM and Transformer models for the test set. The comparison highlights the superior performance of Transformers over LSTM and RNN across all evaluated metrics, indicating its enhanced ability to capture complex temporal dependencies.

Table 4. Comparison of model results

Evaluation Metrics	Models		
	RNN	LSTM	Transformers
RMSE	100.5680	81.6031	23.3666
MAE	76.3798	59.4945	17.8814
R^2	0.6726	0.7844	0.9833
MAPE	0.0832	0.0645	0.0237

Table 5 shows the parameters that the RNN, LSTM and Transformer models used to produce the findings illustrated in Table 4.

Table 5. Model Parameters

Parameters	Model		
	RNN	LSTM	Transformers
Scaling	Minmax (0, 1)	Minmax (0, 1)	Minmax (0, 1)
Window Size	100	100	10
Learning Rate	0.001	0.001	0.001
Dropout Rate	0.25	0.25	0.25
Epochs	500	500	3
Batch Size	32	32	32
Attention Heads	–	–	4
Transformer Blocks	–	–	4
Feed-forward Dim	–	–	4

In Figure 1, the RNN model's performance is depicted when predicting one day ahead for both the training and testing sets. While the model demonstrates satisfactory performance in the training set,

it exhibits a tendency to underpredict in the test sets. Despite this underprediction, it retains the capability to accurately detect the direction of price movements.

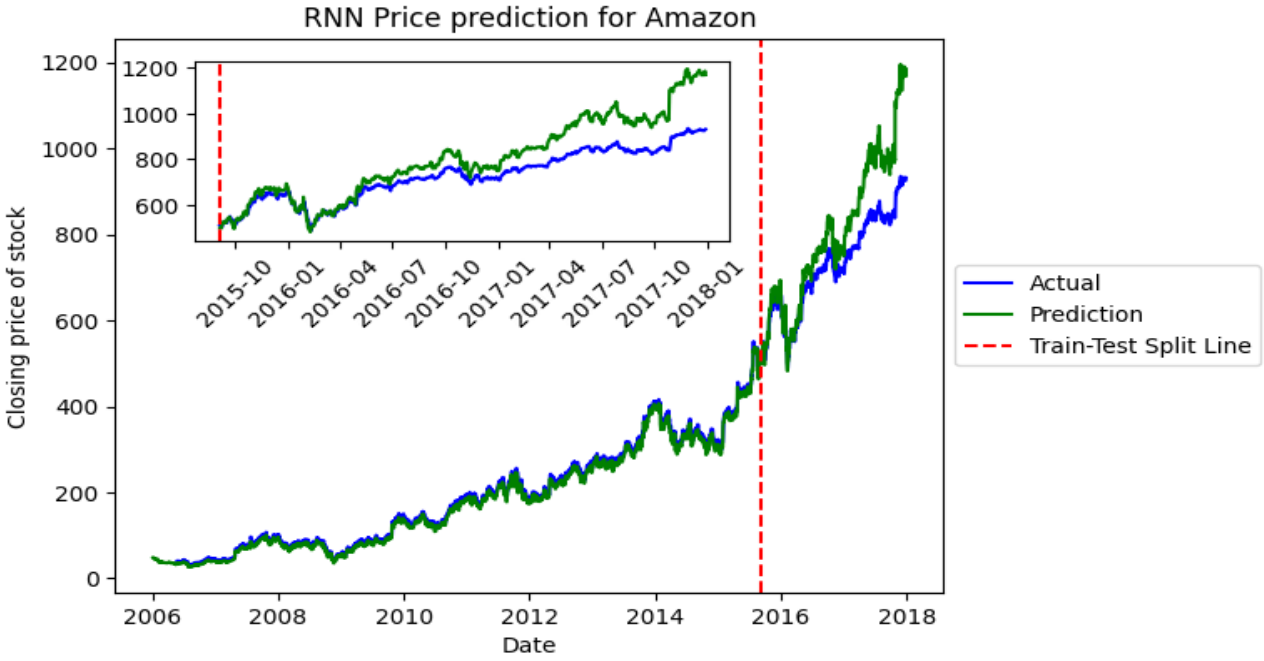

Figure 1. Actual vs RNN one-day ahead predictions for AMZN stock closing price

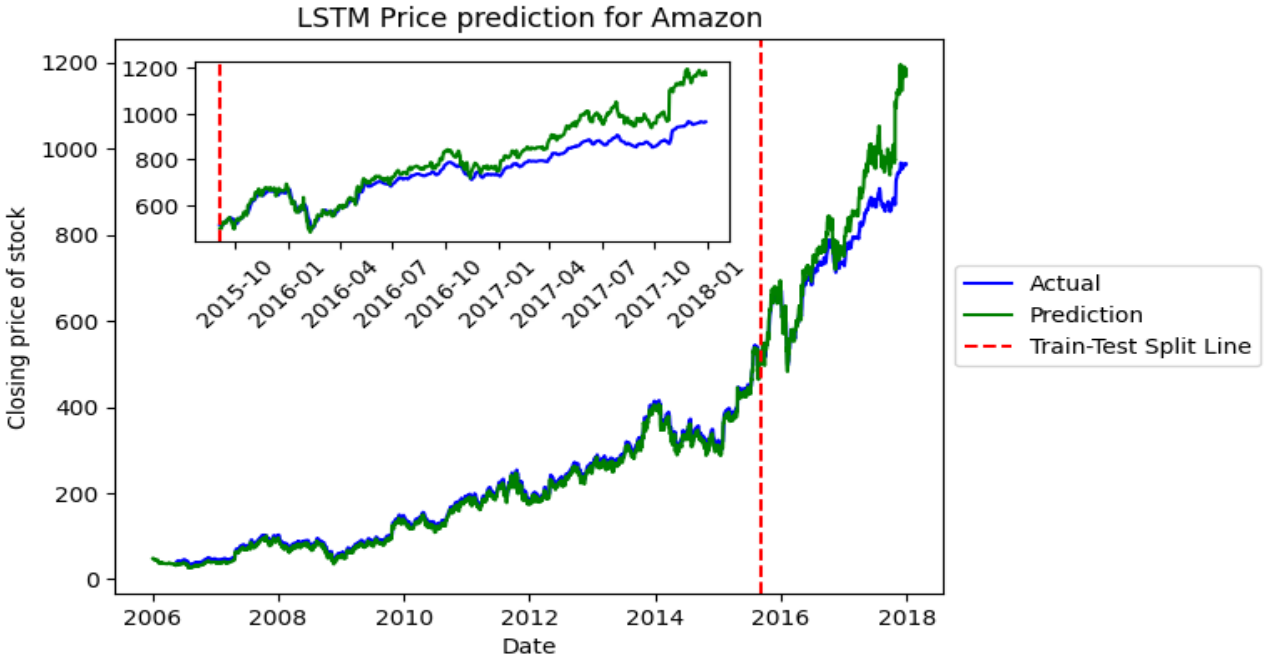

Figure 2. Actual vs LSTM one-day ahead predictions for AMZN stock closing price

The performance of the LSTM model is similar to that of the RNN model as illustrated in Figure 2. The difference in test performance of the two models is not quite clear at first. However, as already made apparent in Table 4, the predictions for the LSTM model are much closer to the true closing prices.

Unlike the RNN and LSTM models, the superior performance of the Transformer model is evident in Figure 3 with the predictions being much closer to the true closing prices as compared to the predictions in Figures 1 and 2.

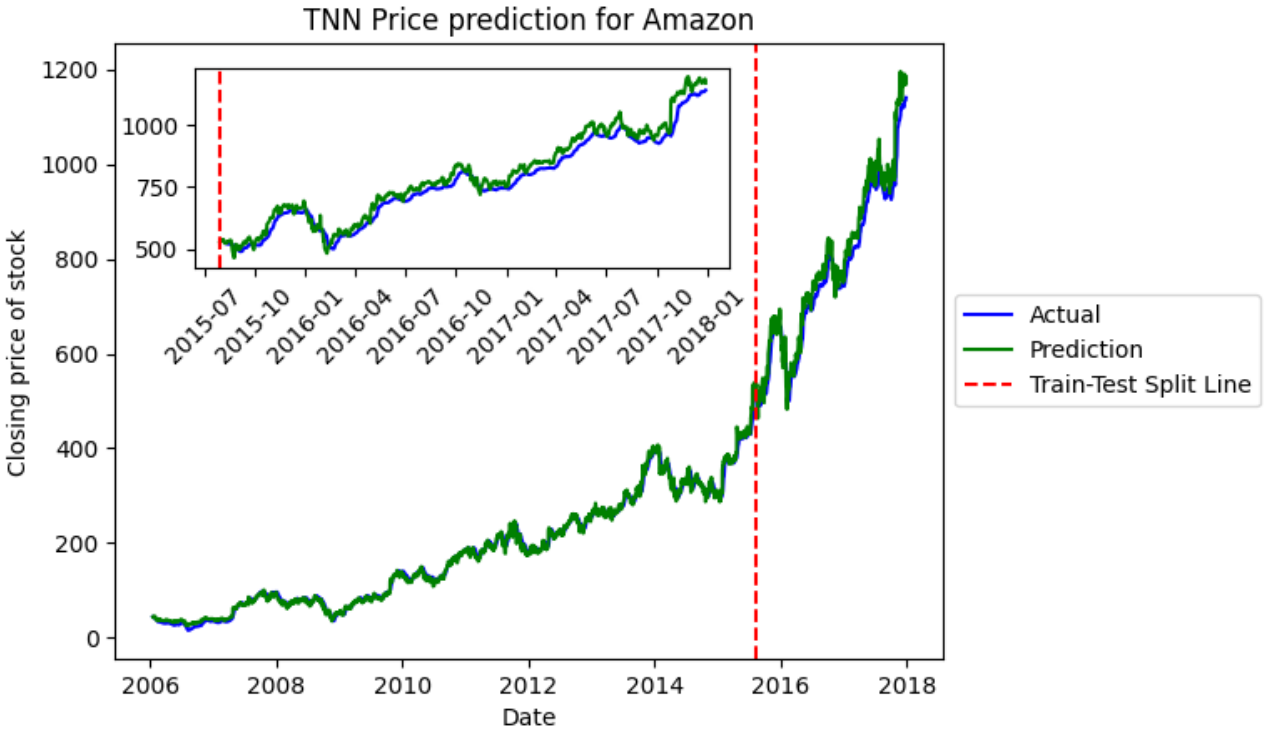

Figure 3. Actual vs Transformer one-day ahead predictions for AMZN stock closing price

3.2 Sensitivity Analysis

3.2.1 RNN

Due to computational time constraints and difficulty to speed up the training of the RNN model, a smaller hyperparameter search space was considered for the sensitivity analysis. The parameters considered for analysis include Window Size, Learning Rate, and Dropout. The corresponding results for both Training Error and Testing Error are presented in Appendix 2. It is evident that adjustments in the window size do not significantly impact the model's performance. The errors remain relatively constant across different window sizes, suggesting robustness to variations in this parameter. On the

other hand, a decrease in the learning rate from 0.1 to 0.001 results in a notable reduction in both training and testing errors. Lower learning rates contribute to improved convergence and generalization. Lastly, dropout rates of 0.25 and 0.4 result in similar errors, with a slight increase in testing error observed at a dropout rate of 0.4. It can therefore be concluded that the model exhibits a degree of resilience to moderate dropout variations.

3.2.2 LSTM

The comprehensive results of the sensitivity analysis for the LSTM model are shown in Appendix 3. First of all, the window size of the input sequence exhibits consistent results, indicating that variations in this parameter do not significantly impact model performance. The errors remain stable across different window sizes. Of interest is that a decrease in the learning rate from 0.1 to 0.001 results in a notable reduction in both training and testing errors. This suggests that lower learning rates contribute to improved convergence and generalization. The dropout parameter, controlling the regularization in the model, demonstrates minimal influence on errors. Even with varying dropout rates (0.00, 0.25, 0.40), errors remain relatively constant, suggesting robustness to dropout variations. Generally, based on the analysis, a learning rate of 0.001 with a window size of 100 appears to yield lower errors during both training and testing.

3.2.3 TNN

The sensitivity analysis results presented in Table 3 give insights of the performances for the TNN model with varying sequence length, number of attention heads, number of transformer blocks, learning rate, dropout rate, and the number of training epochs. By varying window size to small and larger (10 and 90), smaller window sizes tend to perform better than larger ones while predicting the stock price. In the terms of the TNN model training, a small leaning rate of 0.001 encourages the model to converge better and provide lower testing errors compared to large learning rate of 0.1. In terms of model complexity, we observe that randomly reducing neurons in the model architecture for instance by a dropout rate of 0.25 is frequently better performance in comparison to a model with with 0.0 dropout rate. The influence of the number of attention heads when it varies does not show significant difference in the performance. On the other hand, there is an improvement in the model's performance with an increase in the number of transformer blocks. In summary, a good combination involves a smaller window size of 10, a learning rate of 0.001, a dropout rate of 0.25, 2 attention heads, and 4 transformer blocks.

3.3 Forecasting Power

In the analysis of forecasting power, focus is on comparing the models across different prediction horizons using the RMSE. The results are presented in Table 6. Notably, only a small number of prediction horizons were considered due to the computational constraints of some of the models.

Table 6. Forecasting power of the three models based on RMSE

Days Ahead	RNN	LSTM	Transformer
1	99.0669	81.8738	41.6892
5	113.6339	94.3103	46.5708
10	121.6127	100.4834	68.2511
15	122.7781	101.4239	53.3563
20	139.6170	109.0333	55.0760

Across all prediction horizons, the Transformer consistently outperformed RNN and LSTM in terms of lower RMSE values. The results suggest that the Transformer is more effective in capturing and predicting stock price movements, even as the forecasting horizon is extended.

3.4 Running Times

The RNN model, with its sequential learning capability, demonstrated the least efficiency in terms of running time. For instance, to fit a model for one-day-ahead predictions, it required 1432 seconds to fit the model. LSTM, which is also a sequential learner, showed significant improvements with a similar model requiring 220 seconds. Transformers had the best efficiency overall relative to performance with the model requiring only 21 seconds. However, it is worth pointing out that the models were trained over different number of epochs, with the transformer requiring much less epochs to get good results. When accounting for this and evaluating the results per epoch, the LSTM model was the most efficient with a training time of 2.2 seconds, followed by the Transformer (6.85 seconds) and finally the RNN model (14.32 seconds). Nonetheless, the results between the Transformer and the other two models should be interpreted cautiously in this case given that the LSTM and RNN models were trained using Graphics Processing Unit (GPU). Unlike the Transformer, these two models required GPU acceleration to get results within a reasonable time period. This further highlights the efficiency of the Transformer architecture in making stock price predictions from a computational point of perspective.

3.5 Model Strengths and Limitations

3.5.1 RNN

A major strength of the RNN is its ability to handle sequential data and capture temporal dependencies between elements in a sequence. This enables it to handle problems such as stock price prediction, as has already been demonstrated. However, RNNs, whilst revolutionary at one point in time, lag far behind newer models. Two areas of limitations that are evident in this case is in its accuracy performance as well as computational efficiency. The RNN model was the slowest to train, even with GPU acceleration, and provided the least accurate results.

3.5.2 LSTM

The same strengths and limitations as the RNN model apply to the LSTM model, with the difference being that the LSTM is much better than the RNN. The inclusion of a memory cell in the LSTM architecture enables it to retain information over extended periods thereby facilitating the capture of long-term dependencies. This enables the LSTM to give better predictions than the RNN. Even though it is computationally more efficient than the RNN, it still lacks compared to the Transformer.

3.5.3 Transformer

The Transformer model is state of the art in terms of performance and computational efficiency; in regards to which it has demonstrated superior performance than the other models as highlighted in Sub-sections 3.1 and 3.4. Regardless, its architecture is much more complex than the other two models requiring a substantial amount of time to comprehend its operations and implement it correctly.

3.6 Model Interpretation

In the evaluation of the three models—RNN, LSTM, and Transformers—on the test set, a noticeable trend emerges where the performance begins promisingly but experiences a degree of divergence as time progresses. This phenomenon is more pronounced in the RNN and LSTM models. Whilst propositional at this level of analysis, this can be explained by the sequential nature of the RNN and LSTM models. Given that they are sequential, both models suffer from the vanishing gradient problem, with these being more evident in RNNs. This can hinder their ability to adapt to evolving patterns and result in a decline in performance over time. Moreover, the limited contextual memory of RNNs and LSTMs may contribute to their divergence over time. As the models continue to make

predictions in the test set, their ability to retain and utilize relevant historical information diminishes, impacting their predictive accuracy.

3.7 Model Improvements

A major hindrance, especially in the case of the RNN and LSTM models was the inherent computational inadequacy of both models. The models required GPUs which came at the cost. This limited the exploration of the hyperparameter space; the full exploration of which could have yielded better parameters with much better overall performance. This is therefore a possible means of improving their performance. In addition to this, another measure that can improve the performance of all three models is transfer learning in which case case pre-trained model weights, embeddings or representations from models developed in a related task or domain can be used. This can accelerate the convergence of the models and potentially improve performance for the models.

4 CONCLUSION

To sum up, our work focused on Amazon stock price prediction using RNN, LSTM and TNN. These three models have the ability to deal with sequential data. Amongst the three models, the TNN ran faster and was more efficient than RNN and LSTM. In terms of performance among the given neural networks, TNN has the better performance across all the evaluation metrics considered. In terms of sensitivity, RNN and LSTM are more sensitive to the learning rates and dropout rates parameters while TNN is affected by small window size, dropout rate and transformer blocks.

REFERENCES

[1] Gaurang Sonkavde, Deepak Sudhakar Dharrao, Anupkumar M. Bongale, Sarika T. Deokate, Deepak Doreswamy, and Subraya Krishna Bhat. Forecasting stock market prices using machine learning and deep learning models: A systematic review, performance analysis and discussion of implications. *International Journal of Financial Studies*, 11(3), 2023.

[2] Bouteska Ahmed. Understanding the impact of investor sentiment on the price formation process: A review of the conduct of american stock markets. *The Journal of Economic Asymmetries*, 22:e00172, 2020.

[3] Nusrat Rouf, Majid Bashir Malik, Tasleem Arif, Sparsh Sharma, Saurabh Singh, Satyabrata Aich, and Hee-Cheol Kim. Stock market prediction using machine learning techniques: A decade survey on methodologies, recent developments, and future directions. *Electronics*, 10(21), 2021.

[4] V. Siahaan and R.H. Sianipar. *Amazon Stock Price: Visualisation, Forecasting, and Prediction Using Machine Learning with Python GUI*. Balige Publishing, 2023.

Appendix 1. Data Visualization

Figure A1.1. Stock Price Analysis of Amazon (AMZN): Opening, Closing, Highest, and Lowest Prices, along with Trading Volume Over Time

Appendix 2. Sensitivity Analysis Results- RNN

Window Size	Learn Rate	Dropout	Training Error	Testing Error
100	0.100	0.00	0.7582	0.1364
100	0.100	0.25	0.7582	0.1364
100	0.100	0.40	0.7582	0.1688
100	0.010	0.00	0.7582	0.1363
100	0.010	0.25	0.7582	0.1364
100	0.010	0.40	0.7582	0.1364
100	0.001	0.00	0.000027	0.0091
100	0.001	0.25	0.0001	0.0072
100	0.001	0.40	0.000065	0.0076

Appendix 3. Sensitivity Analysis Results- LSTM

Window Size	Learn Rate	Dropout	Training Error	Testing Error
100	0.100	0.00	0.758198	0.136375
100	0.100	0.25	0.758198	0.136375
100	0.100	0.40	0.758198	0.136375
100	0.010	0.00	0.000020	0.008027
100	0.010	0.25	0.000120	0.004698
100	0.010	0.40	0.000218	0.004956
100	0.001	0.00	0.000026	0.005227
100	0.001	0.25	0.000076	0.004498
100	0.001	0.40	0.000062	0.005166
150	0.100	0.00	0.751648	0.133051
150	0.100	0.25	0.751648	0.133051
150	0.100	0.40	0.751648	0.133051
150	0.010	0.00	0.000116	0.005079
150	0.010	0.25	0.000284	0.018977
150	0.010	0.40	0.000077	0.006852
150	0.001	0.00	0.000027	0.004759
150	0.001	0.25	0.000027	0.005150
150	0.001	0.40	0.000025	0.005302
250	0.100	0.00	0.737706	0.126564
250	0.100	0.25	0.737706	0.126564
250	0.100	0.40	0.737706	0.126564
250	0.010	0.00	1.327532	2.825587
250	0.010	0.25	0.737706	0.126564
250	0.010	0.40	0.737706	0.126564
250	0.001	0.00	0.000024	0.004743
250	0.001	0.25	0.000025	0.004833
250	0.001	0.40	0.000028	0.004582

Appendix 4. Sensitivity Analysis Results- TNN

Window Size	Learn Rate	Dropout	Attention Head	TNN Block	Training Error	Testing Error
10	0.100	0.00	2	2	0.041194	0.019672
10	0.001	0.00	2	2	0.042378	0.025597
10	0.100	0.00	2	4	0.035639	0.021323
10	0.001	0.00	2	4	0.036313	0.025101
10	0.100	0.00	4	2	0.059043	0.039264
10	0.001	0.00	4	2	0.038451	0.022579
10	0.100	0.00	4	4	0.062785	0.042937
10	0.001	0.00	4	4	0.040180	0.026262
10	0.100	0.00	8	2	0.036140	0.021784
10	0.001	0.00	8	2	0.036408	0.022450
10	0.100	0.00	8	4	0.094780	0.177273
10	0.001	0.00	8	4	0.057754	0.039394
10	0.100	0.25	2	2	0.052992	0.022657
10	0.001	0.25	2	2	0.046762	0.028005
10	0.100	0.25	2	4	0.074962	0.031233
10	0.001	0.25	2	4	0.036656	0.031847
10	0.100	0.25	4	2	0.039193	0.030098
10	0.001	0.25	4	2	0.041108	0.042894
10	0.100	0.25	4	4	0.100472	0.113232
10	0.001	0.25	4	4	0.060432	0.045414
10	0.100	0.25	8	2	1.434399	0.806956
10	0.001	0.25	8	2	0.051791	0.028416
10	0.100	0.25	8	4	1.113436	0.770162
10	0.001	0.25	8	4	0.083023	0.058588
90	0.100	0.00	2	2	0.137618	0.493672
90	0.001	0.00	2	2	0.088248	0.069757
90	0.100	0.00	2	4	0.670392	0.876407
90	0.001	0.00	2	4	0.074621	0.067274
90	0.100	0.00	4	2	0.713124	19.406458
90	0.001	0.00	4	2	0.090439	0.056791
90	0.100	0.00	4	4	0.092346	0.074456
90	0.001	0.00	4	4	0.091134	0.068844
90	0.100	0.00	8	2	0.429248	5.649458
90	0.001	0.00	8	2	0.095043	0.059193
90	0.100	0.00	8	4	2.024102	0.564864
90	0.001	0.00	8	4	0.082349	0.067584
90	0.100	0.25	2	2	0.171668	0.052328
90	0.001	0.25	2	2	0.082533	0.058679
90	0.100	0.25	2	4	1.237342	0.891990
90	0.001	0.25	2	4	0.101153	0.050390
90	0.100	0.25	4	2	0.692812	0.224322
90	0.001	0.25	4	2	0.106479	0.059011
90	0.100	0.25	4	4	0.977572	0.996718
90	0.001	0.25	4	4	0.083866	0.054511
90	0.100	0.25	8	2	2.817894	1.580573
90	0.001	0.25	8	2	0.079760	0.066910
90	0.100	0.25	8	4	1.048132	0.190505
90	0.001	0.25	8	4	0.080769	0.059059