To Decompose or not to Decompose: A Comparison between Different Neural Networks and GBM models for Stock Prices Predictions

Hassan Saadi

Abstract

In this work, 16 stocks are considered to predict the adjusted closing price for the next 40 trading days. The methods that were used to make predictions are: Long Short Term Memory (LSTM), Temporal Convolution Networks (TCN), decomposing the signal and applying different combinations of LSTM and TCN on the decomposed parts, and stochastic process-geometric Brownian motion. The historical data from Yahoo Finance from 2015-2018 was used to build all the models. Finally, the output of each model is compared to the actual adjusted closing price of each stock.

1 Introduction

Forecasting the stock prices accurately is a task that many companies are trying to solve and optimize on a daily basis. In the last few decades, the field of machine learning has been developing rapidly and the usage of neural networks is becoming more popular because of their success in modeling complex functions. Some types of neural networks are suited for certain tasks. For example, LSTM and TCN neural networks are good for sequences; hence, they are good for time series predictions; and convolution neural networks (CNN) are good for images. We also decomposed the time series into three components: trend, seasonality, and residual in order to study the impact of this decomposition on the performance.

Stochastic modeling is an alternative method to model a system, and it's different from a neural network approach. In finance, many people believe that the stock prices follow a geometric Brownian motion (GBM) behavior. That's to say, the returns of stock between two consecutive points in time are normally distributed and independent from each other. In this model the drift and volatility are constant, and are estimated from historical data. This assumption doesn't match in real-world time series, but future work can incorporate time dependent estimations for these parameters.

In this work, we didn't consider autoregressive models like ARIMA because the authors of this work [1] showed the superiority of LSTMs in predicting the adjusted closing prices where the dataset was large. Moreover, the prediction period in our work is 40 trading days which we considered to be not a small window of predictions, and we have 1000 training points which is not a small dataset, so a neural network can have a chance to learn it. ARIMA can be a good model if the time series is stationary or can be made stationary. But if the time series cannot be made stationary trivially, then using ARIMA models becomes not ideal. Stationarity is not required for neural networks because they can learn non-linear relationships.

To the the best of our knowledge, the literature does not contain a comparison study between LSTM, TCN, decomposed time series with LSTM and TCN, and GBM. We decided to perform these experiments on 16 stocks, and predict the adjusted closing price for the next 40 trading days in order to assess the performance of TCN and the decomposition idea. In this work, section 2 has a short literature review and related work. In section 3, has the methodology. In section 4 is the experimental setup. In section 5, the results of all the models. And finally, in section 6 conclusion and future work.

2 Short literature review and related work

In [2] work, they compared four types of neural networks (RNN,LSTM,CNN, and MLP) to predict two stock markets price. In [3] work, they also compared LSTM, CNN, MLP, and one attention-based neural network to predict the next day's index price from three different financial markets. The attention-based model performed the best in their experiments. Another group created a new neural architecture to deal with the temporal structure that a time

series has by inventing Temporal Convolution Network (TCN) [4].

3 Methodology

3.1 LSTM

Our first model is an LSTM. The formulation of LSTM is as follows

$$\begin{aligned}
\mathbf{i}_t &= \sigma_g(\mathbf{W}_i\mathbf{x}_t + \mathbf{U}_i\mathbf{h}_{t-1} + \mathbf{b}_i) && (1)\\
\mathbf{f}_t &= \sigma_g(\mathbf{W}_f\mathbf{x}_t + \mathbf{U}_f\mathbf{h}_{t-1} + \mathbf{b}_f) && (2)\\
\tilde{\mathbf{c}}_t &= \sigma_c(\mathbf{W}_c\mathbf{x}_t + \mathbf{U}_c\mathbf{h}_{t-1} + \mathbf{b}_c) && (3)\\
\mathbf{c}_t &= \mathbf{f}_t\mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t && (4)\\
\mathbf{o}_t &= \sigma_g(\mathbf{W}_o\mathbf{x}_t + \mathbf{U}_o\mathbf{h}_{t-1} + \mathbf{b}_o) && (5)\\
\mathbf{h}_t &= LSTM(\mathbf{x}_t, \mathbf{h}_{t-1}) = \mathbf{o}_t \odot \sigma_h(\mathbf{c}_t) && (6)
\end{aligned}$$

where $\odot$ means element-wise multiplication, σ_g is the logistic sigmoid function, σ_c is the hyperbolic tangent function, and σ_h is the hyperbolic tangent function which are used as recurrent activation functions. LSTM architecture has memory cells to store and output information. This property aids a recurrent neural network to find long-range temporal dependencies. $\mathbf{i}$, $\mathbf{f}$, $\mathbf{c}$, and $\mathbf{o}$ are the input gate, forget gate, cell activation, and output gate respectively.

3.2 TCN

TCN has two main ideas: The convolutions are causal, i.e., no information is leaked from future to past; and the architecture can have a sequence of any length as input and map it to an output sequence that has the same length. The first idea is accomplished by making convolutions where an output at time t is convolved only with elements from time t and in the past in the previous layer. The second idea is accomplished by utilizing a $1D$ fully convolutional network architecture such that each hidden layer has the same size as the input layer. Dilated convolutions are also used here to enable a larger receptive field to capture dependencies at larger scales. For an input sequence $\mathbf{x} \in \mathbb{R}^n$ and a filter $h : \{0, ..., k-1\} \to \mathbb{R}$, the dilated convolutional

operation F on element s on the sequence is

$$F(s) = (\mathbf{x} *_d h)(s) = \sum_{i=0}^{k-1} h(i) \cdot \mathbf{x}_{s-d \cdot i} \tag{7}$$

where d is the dilation factor, k is the size of the filter, and $s - d \cdot i$ captures the direction in the past.

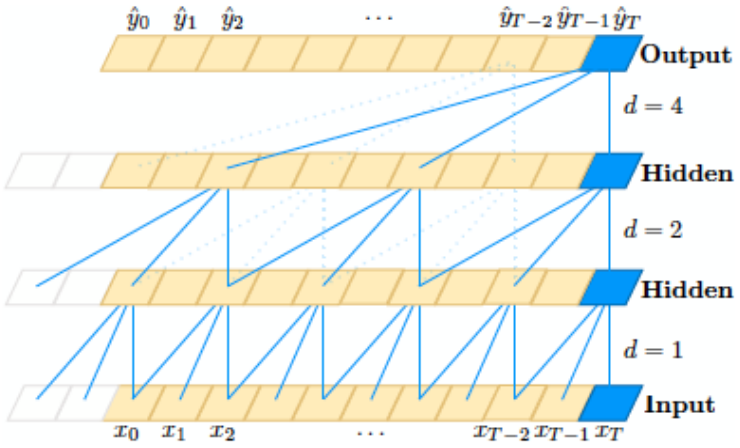

Figure 1: A dilated Causal convolution with dilation factors $d = 1, 2, 4$ and filter size $k = 3$ [5].

3.3 Decomposition

A time series can be decomposed into three parts: trend, seasonality, and residual. In this work, the "*stat_decompose*" library in python was used in order to the decomposition. Note that, the trend is obtained by using convolution filter as a moving average of the last 10 days. After decomposition, I applied different combinations of LSTM(s) and TCN(s) to these components. In total, there is 8 models that incorporate decomposition: (L,L,L), (L,L,T), (L,T,L), (L,T,T), (T,L,L), (T,L,T), (T,T,L), and (T,T,T); where L and T mean LSTM and TCN were used respectively;, and the first, second, and third entries in each model correspond to trend, seasonality, and residual part of the time series respectively. So for example, (L,L,T) model means that two LSTMs are used to model trend and seasonality separately and one

TCN is use to model the residual.

3.4 GBM

A Wiener process (Brownian Motion) is a Markov stochastic process with zero drift and variance rate of one; $dW = \epsilon\sqrt{dt}$ where ϵ is sampled from a normal distribution. It has the following properties:

1. $W_0 = 0$
2. $W_{s+t} - W_t \in \mathcal{N}(0, s)$ for $0 \leq s \leq t \leq T$
3. $\{W_t\}_{t\geq 0}$ has stationary and independent non-overlapping increments.

A generalized Wiener process has a drift term and has the form $dX(t) = \mu dt + \sigma dW(t)$. If the stochastic process is defined as $X(t) = \log(S(t))$ because the prices are assumed to follow a lognormal distribution since prices can't be negative, then we obtain the geometric Brownian motion as

$$\frac{dS(t)}{S(t)} = \mu dt + \sigma dW(t) \tag{8}$$

This can be solved for $t > 0$ by integrating both sides as

$$S(t) = S(0) + \mu \int_0^t S(\tau)d\tau + \sigma \int_0^t S(\tau)dW(\tau) \tag{9}$$

By using Ito's formula we obtain

$$d\log(S(t)) = (\mu - \frac{\sigma^2}{2})dt + \sigma dW(t) \tag{10}$$

Hence, after integrating both sides

$$S(t) = S(0)e^{(\mu - \frac{1}{2}\sigma^2)t + \sigma W(t)} = S(0)e^{X(t)} \tag{11}$$

Since we are working with discrete data, we can make progressive predictions from time t_0 incrementally as $t_0 < t_1 < t_2 < ... < t_N$ by generating N iid $\mathcal{N}(0, 1)$ random variables $z_1, ..., z_N$. Note that this can be done because for any $0 \leq s < t$ we have $S(t) = S(0)\frac{S(s)}{S(0)}\frac{S(t)}{S(s)} = S(0)e^{X(s)} \times e^{X(s)-X(t)}$, and we already know that $X(s)$ is independent of the increment $X(t) - X(s)$;

hence, $\frac{S(s)}{S(0)}$ and $\frac{S(t)}{S(s)}$ are independent lognormals. Generalizing the idea, let $Y_i = \frac{S_{t_i}}{S(t_{i-1})}$, we have $S(t_k) = S(t_{k-1})Y_k = S_0 Y_1 Y_2 ... Y_k$, with $Y_i = e^{\sigma z_i + \mu}$ because $t_i - t_{i-1} = 1$ in our case because we are interested in the daily adjusted price of a stock. Therefore, we obtain

$$\begin{aligned} S_{t_k} &= S_0 \prod_{i=1}^{k} e^{\mu - \frac{1}{2}\sigma^2 + \sigma z_i} \\ &= S_0 e^{(\mu - \frac{1}{2}\sigma^2)k + \sigma \sum_{i=1}^{k} z_i} \\ &= S_0 e^{(\mu - \frac{1}{2}\sigma^2)t_k + \sigma W_k} \end{aligned} \tag{12}$$

where we replaced k with t_k, and $W_k = \sum_{i=1}^{k} z_i$.

4 Experimental Section

4.1 Datasets

All experiments are performed on 16 stocks from the S&P 500 list: Apple (AAPL), IBM (IBM), Tesla Inc. (TSLA), Microsoft Corp. (MSFT), Facebook (FB), Google Inc. (GOOGL), Procter & Gamble (PG), JPMorgan Chase & Co. (JPM), Netflix Inc. (NFLX), Intel Corp. (INTC), Adobe Inc. (ADBE), Johnson & Johnson (JNJ), Goldman Sachs Group (GS), Morgan Stanley (MS), (NDAQ), General Motors (GM). The historical data was taken from Yahoo Finance from 2015-2018, and the goal is to predict the adjusted close price for the next 40 trading days; i.e. the months of January and February of 2019.

4.2 Evaluation Metrics

Two metrics are adopted in this study, $RMSE$ and MAE:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \hat{x}_i)^2}{N}} \tag{13}$$

$$MAE = \frac{\sum_{i=1}^{N}|x_i - \hat{x}_i|}{N} \tag{14}$$

where x_i is the true value, $\hat{x}_i$ is the predicted value, N is the total number of test data points.

4.3 Hyperparameters

If only one LSTM is used for all stocks, then its properties are: One layer, 300 neurons, 100 epochs, 64 batch size, time step of 10, mean squared error loss, and adam optimizer. The number of trainable parameters is 362701.

In all experiments, the TCNs properties are: filter size of 2, 100 epochs, time step 10, dilations of $\{1, 2, 4, 8\}$,mean squared error loss, and adam optimizer. The number of trainable parameters is 87937.

If an LSTM is used on the decomposed components then its properties are: One layer, 100 neurons, 100 epochs, 64 batch size, time step of 10, mean squared error loss, and adam optimizer. The number of trainable parameters is 40901. Note that the number of neurons in this LSTM is one-third of the LSTM described earlier.

4.4 Machine Specs

All experiments were run on a laptop with 1.6 GHz Dual-Core Intel Core i5 and 8GB of main memory.

5 Results

In table (1), the average RMSE and MAE over all 16 stocks, and the number of stocks that each model predicted the best according to RMSE and MAE are listed. The best model is shown in bold and colored with blue.

Model	*Number of Parameters*	*Avg. RMSE*	*Avg. MAE*	*number of best RMSE*	*number of best MAE*
LSTM	362701	**4.02**	**3.31**	**15**	**13**
TCN	**87937**	9.79	8.76	0	0
GBM	0	13.06	11.80	0	0
(L,L,L)	122703	5.51	4.48	0	0
(L,L,T)	169739	6.78	5.52	0	1
(L,T,L)	169739	5.51	4.48	1	0
(L,T,T)	216775	6.78	5.52	0	0
(T,L,L)	169739	13.22	12.15	0	0
(T,L,T)	216775	14.07	12.77	0	0
(T,T,L)	216775	13.22	12.16	0	2
(T,T,T)	263811	14.08	12.77	0	1

Table 1: Average RMSE and MAE for 18 stocks

The best model is LSTM. Below are boxplots for the RMSE and MAE values of each model where the mean of each model is shown as a green full circle. The data contains 16 stocks, and doesn't exhibit normality as we can see from the boxplots and after performing QQ plots, Kolmogorov-Smirnov, and Shapiro tests. The whiskers in the boxplots represent the minimum and maximum values of the dataset (model) range and not the 1.5 IQR; i.e., we are not showing outliers.

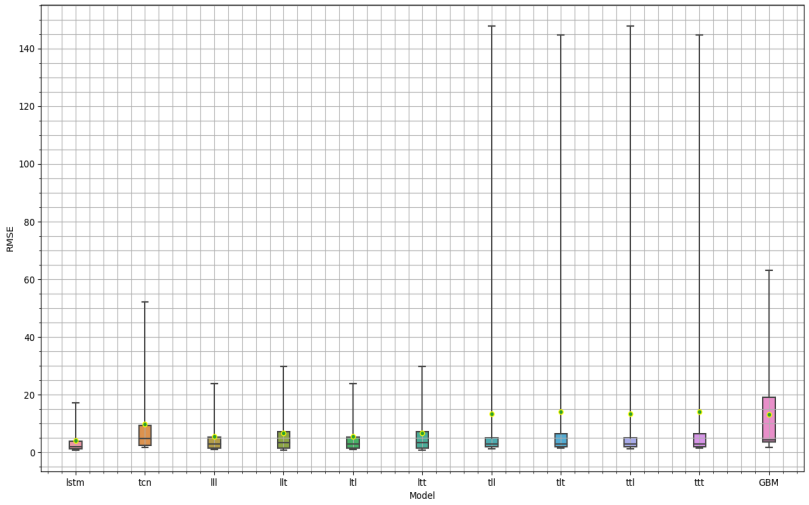

Figure 2: RMSE values for each model

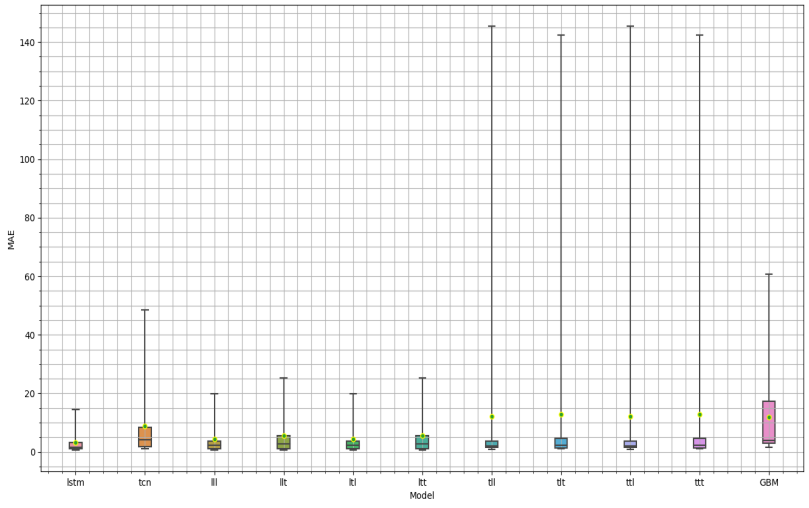

Figure 3: MAE values for each model

The heat map of RMSE for each stock is listed. The GBM model in this heat map was removed in order to compare different neural networks architectures. In the heat maps, green means low values of z-score, and red means high values of z-score. The tables (2) and (3) of the RMSE and MAE for each stock is listed at the appendix were the best model is shown in bold and colored with blue.

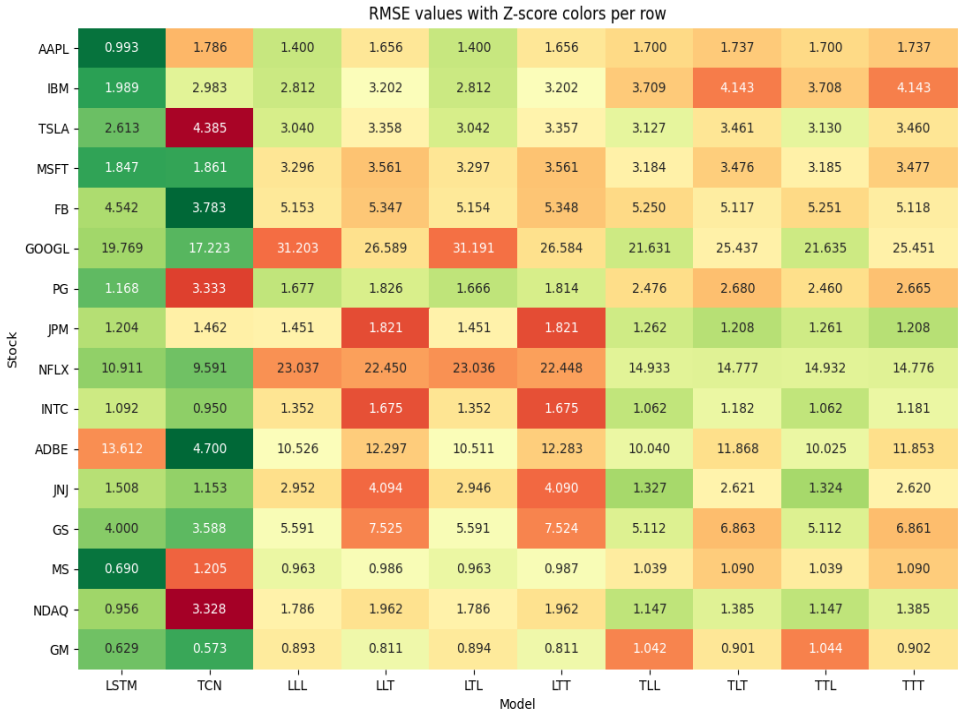

Figure 4: RMSE values with Z-score colors per row for 16 stocks and all models excluding GBM.

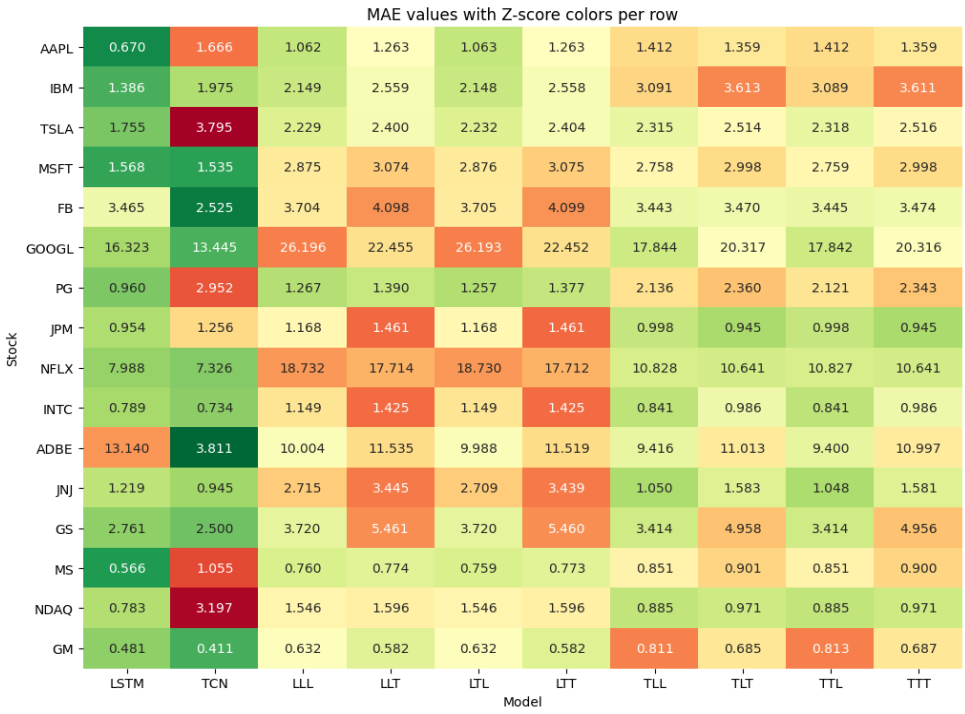

Figure 5: MAE values with Z-score colors per row for 16 stocks and all models excluding GBM.

Moreover, we evaluated and plotted the logarithm of the standard deviation on the x-axis and the Hurst exponent on the y-axis of our training set for each stock. In plots (6) and (7). The colors in these plots denote the best neural network that performed on the written stock according to RMSE or MAE.

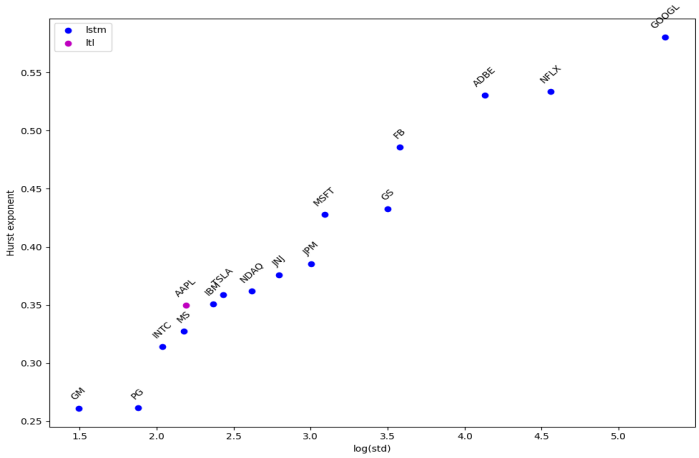

Figure 6: log(std) vs Hurst Exponent, best neural networks performance according to RMSE

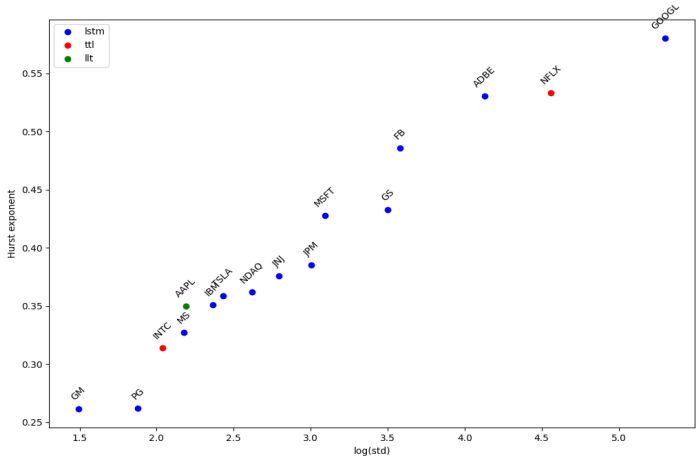

Figure 7: log(std) vs Hurst Exponent, best neural networks performance according to MAE

6 Conclusion and Future Work

In this work, we compared multiple neural network architectures with and without decomposition and a simple GBM model. LSTM has the best results overall on 16 stocks to predict the adjusted closing price for the next 40 trading days. According to our results, additive decomposition did not give better results on average; however, it gave some significant results on a few stocks. As future work, one can investigate different methods to improve the performance of these results and study their generalization capabilities. This can be done through building ensembles of these models and using other ways of decomposing the time series such as singular spectrum analysis or STL. Moreover, it would informative to investigate the relationship between time series properties and neural network architectures and decomposition methods.

References

[1] S. Siami-Namini, N. Tavakoli, and A. S. Namin, "A comparison of arima and lstm in forecasting time series," in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2018, pp. 1394–1401.

[2] M. Hiransha, E. A. Gopalakrishnan, V. K. Menon, and K. Soman, "Nse stock market prediction using deep-learning models," *Procedia computer science*, vol. 132, pp. 1351–1362, 2018.

[3] P. Gao, R. Zhang, and X. Yang, "The application of stock index price prediction with neural network," *Mathematical and Computational Applications*, vol. 25, no. 3, p. 53, 2020.

[4] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks for action segmentation and detection," in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 156–165.

[5] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.

7 Appendix

Model / Stock	LSTM	TCN	GBM	(L,L,L)	(L,L,T)	(L,T,L)	(L,T,T)	(T,L,L)	(T,L,T)	(T,T,L)	(T,T,T)
AAPL	3.7732	13.798	2.3101	**1.4085**	1.4225	1.4085	1.4231	1.9425	2.0317	1.9425	2.0322
IBM	**1.8703**	2.3723	16.400	2.9161	2.4318	2.9170	2.4325	2.5856	2.1448	2.5864	2.1454
TSLA	**2.5574**	3.4833	3.7576	3.2674	4.3991	3.2639	4.3900	3.3520	4.5929	3.3444	4.5812
MSFT	**1.8002**	2.0502	3.5986	3.1065	3.7175	3.1065	3.7175	2.9282	2.9010	2.9282	2.9011
FB	**4.4810**	7.9291	19.818	5.1439	6.9630	5.1452	6.9635	6.0286	7.8579	6.0296	7.8583
GOOGL	**17.098**	52.038	34.764	21.169	22.564	21.171	22.566	147.83	144.65	147.88	144.70
PG	**1.1298**	1.9705	5.2331	1.6126	1.4151	1.6236	1.4113	1.8622	2.2325	1.8648	2.2243
JPM	**1.1847**	1.5927	3.2384	1.4118	1.2243	1.4109	1.2238	1.7457	1.6014	1.7451	1.6012
NFLX	**16.584**	25.582	63.185	23.821	29.810	23.821	29.810	19.130	24.862	19.130	24.862
INTC	**1.2396**	2.6455	2.5388	1.3841	1.6631	1.3837	1.6626	1.2442	1.3881	1.2441	1.3879
ADBE	**3.9097**	7.8236	18.950	10.519	15.875	10.515	15.872	9.4549	14.006	9.4521	14.004
JNJ	**1.3145**	2.7775	3.9341	2.6616	4.4166	2.6598	4.4166	3.5664	5.7523	3.5653	5.7525
GS	**3.8214**	6.2563	21.124	5.5424	7.9968	5.5424	7.9967	4.5451	5.8167	4.5453	5.8167
MS	**0.8915**	18.076	1.6694	0.9834	0.9644	0.9822	0.9654	1.8804	1.3262	1.8784	1.3248
NDAQ	**1.9477**	2.2029	4.1382	2.3314	2.9055	2.3324	2.9065	2.3572	2.6653	2.3573	2.6656
GM	**0.7346**	6.1905	4.3646	0.8941	0.8240	0.8949	0.8240	1.1409	1.4324	1.1396	1.4308

Table 2: RMSE for 16 stocks

Model / Stock	LSTM	TCN	GBM	(L,L,L)	(L,L,T)	(L,T,L)	(L,T,T)	(T,L,L)	(T,L,T)	(T,T,L)	(T,T,T)
AAPL	3.6503	13.709	2.0978	1.0678	**1.0592**	1.0682	1.0597	1.3580	1.4251	1.3583	1.4253
IBM	**1.1992**	1.6910	14.509	2.2621	1.7077	2.2632	1.7083	1.8512	1.4632	1.8519	1.4639
TSLA	**1.7042**	2.8907	3.1476	2.2831	3.3891	2.2823	3.3808	2.4782	3.8120	2.4711	3.7953
MSFT	**1.5085**	1.5817	2.9642	2.7042	2.9035	2.7042	2.9036	2.3198	2.3081	2.3200	2.3082
FB	**3.4225**	6.5911	17.240	3.6793	5.3109	3.6796	5.3105	4.6189	6.2645	4.6182	6.2648
GOOGL	**14.382**	48.502	30.560	16.743	17.173	16.753	17.165	145.45	142.35	145.50	142.40
PG	**0.9209**	1.6908	4.1822	1.1934	1.1387	1.2048	1.1356	1.6909	1.8091	1.6935	1.8071
JPM	**0.9307**	1.1974	2.9672	1.1382	0.9672	1.1376	0.9665	1.4149	1.2571	1.4144	1.2568
NFLX	14.265	19.539	60.671	19.824	25.333	19.824	25.334	13.592	18.129	**13.592**	18.128
INTC	1.0944	2.4763	1.9854	1.1761	1.4280	1.1755	1.4275	0.9734	1.1606	**0.9730**	1.1600
ADBE	**3.0786**	6.6752	17.305	10.008	15.287	10.004	15.282	7.9345	12.677	7.9308	12.672
JNJ	**1.0399**	2.4654	3.1015	2.3932	2.5420	2.3914	2.5406	3.0080	3.2733	3.0066	3.2719
GS	**2.7245**	5.2393	19.063	3.7432	6.0801	3.7433	6.0800	3.4518	4.0840	3.4521	4.0840
MS	**0.7218**	18.061	1.5041	0.7798	0.7592	0.7783	0.7604	1.6543	1.0804	1.6523	1.0791
NDAQ	**1.7228**	1.7599	3.5573	2.0877	2.7283	2.0886	2.7294	1.8008	2.0176	1.8012	2.0182
GM	**0.6062**	6.1459	4.0517	0.6324	0.6239	0.6331	0.6235	0.9339	1.2014	0.9325	1.1995

Table 3: MAE for 16 stocks