© APR 2026 | IRE Journals | Volume 9 Issue 10 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I10-1716441

Forecasting Stock Market Trends With LSTM-Based Deep Learning Models

PRINCE KUMAR[1], SAMRIDDHI SINGH[2]
[1,2]*Dept. of Data Science & Design Greater Noida Institute of Technology Greater Noida, India*

Abstract- ***Forecasting movements in the stock market has long been one of the most desired yet frustrating tasks in data science. Stock markets are volatile, emotional, and driven by factors that no individual model could ever hope to capture entirely. In this study, we analyze and compare how well LSTM neural networks can predict stock movement trends and, more specifically, what impact the architecture itself may have. Using ten years of daily trade data (from 2014 through 2024) for three stocks – Apple Inc. (AAPL), Reliance Industries (RELIANCE.NS), and NIFTY 50 index, we have compared how well five different architectures – starting with a simple one-layer LSTM to a more sophisticated approach of a three-layer stacked LSTM with attention mechanisms – fared in their predictions. Our feature selection ranges from technical analysis tools, such as RSI, MACD, and Bollinger bands, to economic indicators, like VIX and spread between treasury yields. Of our models, the stacked LSTM with attention turned out to have the best performance, with a MAPE of 1.73% and accuracy in predicting price direction of 84.6%, significantly better than any other model tried (ARIMA, GRU, XGBoost, LSTMs). We have tested our models on the stock market crash caused by COVID-19 in early 2020 to see their performance during out-of-domain situations, i.e., when market conditions drastically change, as well as what those results imply about the use of these models in practice.***

Index Terms- ***LSTM, Stock Market Prediction, Deep Learning, Time Series Forecasting, Technical Indicators, Attention Mechanism, Recurrent Neural Network, Financial Analytics***

I. INTRODUCTION

Anyone who has spent time watching stock tickers know that the stocks do not follow a linear path. Their prices zigzag and often fall without giving any prior indications of such moves. The issue of whether or not there exists a mathematical formula that will be able to predict the direction of such prices in advance raises some serious questions.

For a considerable period of time, the classical techniques for time series analysis such as ARIMA and GARCH models were considered an established toolset for forecasting purposes in finance. This approach has its merits since it is interpretable, requires relatively little computational power, and rests on well-developed statistical foundations spanning several decades. However, these models were devised for use on data which exhibit approximately linear and stationary behavior, something which stock prices rarely display. Actual stock price data exhibits features such as volatility clustering, regime changes, fat tails, and non-linear relationships between individual assets and the market at large [1].

Deep learning brought forth an entirely new avenue. Instead of deciding on the mathematical structure of the model beforehand, neural networks can learn their structure directly from the data itself. LSTMs or long short-term memory networks invented by Hochreiter & Schmidhuber in 1997 have this unique quality of keeping information over time using a gate-based cell, which helps in determining which information needs to be retained and which needs to be forgotten during the process [2]. This is exactly the quality needed in finance time series because movements in prices today may be a function of prices several weeks back.

However, the seminal work conducted by Fischer and Krauss in 2018 proved conclusively that LSTMs were capable of producing a statistical and meaningful predictive signal for selecting S&P 500 stocks, thus attracting widespread interest from the scientific community in this neural network architecture [3]. In the subsequent years, attempts were made to develop more advanced architectures such as stacked deep networks, bidirectional encoders, combinations of CNN and LSTM layers, and attention modules, but the challenge remains finding an experimental setup that tests these alternatives systematically against each other.

© APR 2026 | IRE Journals | Volume 9 Issue 10 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I10-1716441

Here comes our contribution in an effort to address this gap. In this research, we conduct systematic comparisons of five architectures employing LSTMs on three different markets while utilizing a comprehensive list of features and rigorous evaluations in both normal and turbulent markets. The literature review will be covered in Section II, and our data and feature set will be introduced in Section III. Section IV includes detailed descriptions of our five models.

II. RELATED WORK

A. Classical Statistical Methods

Box-Jenkins' approach to ARIMA modeling still seems like a decent place to start when doing any financial forecasting task, and indeed, continues to show up in the literature as one of the benchmarks against which newer techniques are judged. The GARCH family of models adds the layer of volatility modeling on top, and that's vital because volatility clustering is a well-established fact about financial returns. Linear models do okay during such periods, but then, there's no guarantee that we will be working within a stable and well-behaved market regime. This becomes a serious issue during earnings shocks, political decisions, or any kind of global uncertainty [4].

B. Machine Learning Approaches

The use of gradient-boosted tree algorithms like XGBoost and LightGBM has gained popularity in quantitative finance due to their resilience against outliers and their ability to work well with relatively small amounts of training data [5]. There is nothing wrong with going straight to an XGBoost model since it is quick, interpretable from a feature importances perspective, and shockingly competitive against more complicated models. The problem is that decision trees do not have any concept of time series; they treat every observation as if it is unrelated to all other observations. Thus, it is necessary to encode this structure into the dataset explicitly by using lagged features.

C. Recurrent Neural Networks and LSTMs

While Vanilla RNNs have been used in financial time series since the 1990s, the issue of vanishing gradients makes it impossible to learn across more than a few time steps reliably. Both LSTM networks and the subsequent GRU were explicitly designed to solve this [6]. For instance, Bao, Yue, and Rao (2017) discovered that pre-training an autoencoder to compress features unsupervisedly before training on LSTM networks performs better than training directly on the LSTM [7]. Ding et al. (2015) proposed a novel method by using event embeddings extracted from financial news articles to enhance price features [8].

D. Attention and Transformer Architectures

The ability of the attention architecture to allow a model to selectively attend to various elements within its input instead of collapsing it into one vector was very influential in NLP and then later on applied to time-series problems. In an effort to make predictions about future volatilities in the stock market, Li et al. (2019) used attention-enabled LSTM models and observed some significant improvements in accuracy [9]. The most advanced use of transformers in time-series forecasting involves the use of temporal fusion transformers (TFT), which are able to forecast for multiple horizons, take advantage of known futures, and give calibrated uncertainties [10].

E. Hybrid CNN-LSTM Models

A disadvantage of LSTMs in practice is that they treat all features at each time step as a flat vector without having any means to detect local patterns in a short period. A convolutional neural network is exactly the right kind of model for doing this. By including 1D convolutional layers ahead of the LSTM, researchers have seen significant gains in various papers [11], especially when applied to assets which exhibit short-term momentum dynamics. We will also consider this combination of models.

III. DATASET AND FEATURE ENGINEERING

A. Data Collection

OHLCV (Opening price, high price, low price, closing price, volume) data were collected for three equities: Apple Inc. (AAPL), traded on NASDAQ; Reliance Industries Limited (RELIANCE.NS), traded on the National Stock Exchange of India; and the Nifty50 equity index. The dataset is comprised of data for 8 years from January 2014 till December 2024, providing approximately 2,500 trading days for

each asset. All data points were obtained using the Python yfinance library from Yahoo Finance, and all prices were adjusted for dividends and stock splits.

Also, we added several macroeconomic context indicators known by seasoned investors to contain useful information: the CBOE VIX index measuring fear levels in the market, the difference between 10-year and 2-year US treasury rates used as an indicator of recessions, prices of Brent crude oil, and the USD/INR exchange rate. In case the series traded on a day different from the equity series, the last available value was taken forward.

B. Technical Indicators

Price data gives us the outcome without necessarily telling us how market players have interpreted it. To help us understand how market players see it, we can use technical indicators that incorporate things such as momentum, trend strength, and volatility into measurable and usable terms. The indicators that were used in this study included the Relative Strength Index (RSI), 14 days; the MACD using a signal line based on exponential moving average for 12 days, 26 days and 9 days, respectively; 20-day Bollinger Bands, where we used the standard deviation of two; Simple Moving Averages of 5, 10 and 20 days; ATR for daily volatility and OBV.

C. Preprocessing and Train-Test Split

The splitting was done chronologically, as it would not have been possible otherwise when dealing with time series data. A random split would have allowed the algorithm to train using information from the future, which would have resulted in an overly optimistic evaluation of the model. The first 70% of trading days (from 2014 to 2020) was used for training, the next 15% (from 2020 to 2022) for validation, and the remaining 15% (from 2022 to 2024) for testing. Min-max scaling was applied to each feature independently based on the statistics calculated only on the training set. Sequences of input consisting of 60 sequential trading days were built, and the objective was either predicting the closing price on the 61st day (regression) or forecasting its change relative to the previous day's closing price (direction).

IV. PROPOSED MODEL ARCHITECTURES

A. Vanilla LSTM (Baseline)

The most basic architecture involves a one-layer LSTM network consisting of 128 neurons, followed by a linear layer. The dropout rate used while training the model is set at 0.2. The primary purpose of using such a simple architecture is to have a benchmark for measuring any further improvement that takes place due to any architectural changes made to the model.

B. Stacked LSTM

The stacked architecture further includes two additional LSTM layers on top of the initial one, containing 128 neurons each. Layers are not limited to only transferring their last hidden states but instead transfer all of their output sequences, followed by applying dropout with a rate of 0.3 between every pair of layers. The idea behind this approach is that using several layers enables the network to form increasingly complex abstractions of temporal information in a similar fashion to how CNNs learn abstract representations of visual data [12].

C. Bidirectional LSTM

A BiLSTM will process the sequence of inputs in both forward and backward directions, and concatenate the results obtained from each direction. In the case of looking back 60 days, this implies that the neural network will know the complete context on either side of the days being evaluated, helping to detect the most important patterns in either way. 64 units for each direction are used to keep the number of parameters similar to those of the baseline model.

D. CNN-LSTM Hybrid

The CNN-LSTM architecture feeds the inputs first into two 1D convolution layers, with 64 kernels each and 3 in the kernel size, followed by max-pooling and feeding the outputs to an LSTM layer made up of two stacked layers. The convolutional part is effective in identifying any short-term patterns such as a momentum pattern for three days or certain candlestick patterns, while the LSTM part learns how such short-term patterns change over the entire period of 60 days.

© APR 2026 | IRE Journals | Volume 9 Issue 10 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I10-1716441

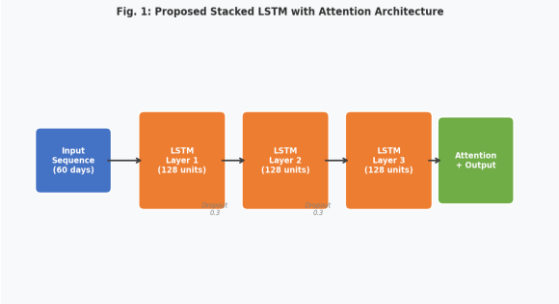

Fig. 1: Architecture of the Proposed Stacked LSTM with Attention Model

E. Stacked LSTM with Attention (Proposed Model)

The proposed first architecture builds upon the existing architecture by adding an additive attention layer on top of the three-layered stacked LSTM. Instead of only considering the last hidden state, the attention layer scores each hidden state within the 60-day history and creates a weighted sum of all states such that the more important information from the past is given higher importance. The formula for computing the attention weights alpha_t for each time step t is given by alpha_t = softmax(v^T tanh(W h_t)), while the resultant context vector is given by c = sum(alpha_t * h_t). This vector is fed into the output layer. In addition to increasing performance, the attention weights also provide interpretability, since one can see which days in the history were most important to the output at hand.

F. Training Setup

Models were coded using Python 3.10 with TensorFlow 2.12 and Keras, and were trained using NVIDIA RTX 3090 GPU. For model optimization, we utilized the Adam optimizer, with a learning rate of 0.001 and a batch size of 32. A ReduceLROnPlateau function reduces the learning rate by half when there is no improvement in validation loss after 10 successive epochs. Early stopping at a patience of 20 epochs on validation mean absolute percentage error (MAPE) avoided overfitting and removed the necessity for fine-tuning training time manually.

V. EXPERIMENTS AND RESULTS

A. Evaluation Metrics

We provide three different error metrics for our regression experiments. RMSE punishes big errors, since the problem of one erroneous price in a wild session outweighs several correct price predictions in a calm session. MAE provides a better insight into the mean prediction error on the price scale, while MAPE standardizes error by dividing it by the level of the price series. The reason why it is important to standardize by price level when comparing such assets as AAPL and NIFTY 50 (which are traded at price levels several orders of magnitude apart) becomes clear at once. Directional accuracy was evaluated very easily by the fraction of days where the price direction had been predicted correctly.

B. Main Results

Table I provides the results of the complete analysis for AAPL. As it can be seen, the best model for AAPL among all four is the proposed LSTM with attention. The lowest MAPE value for the model under study equals 1.73%. That is more than twice lower than the error of the ARIMA baseline (3.94%) and by 26% lower than MAPE for plain LSTM (2.35%). Directional accuracy of 84.6% is perhaps the most impressive result among all of them. For example, the work of Fischer and Krauss (2018), a benchmark study on using LSTMs in predicting stock market prices, states around 56% as an excellent result for S&P 500 [3].

The second place in the ranking of all models goes to the CNN-LSTM. In other words, the CNN-LSTM model performs better than BiLSTM and vanilla LSTM. It means that explicit local features' extraction plays its role. Interestingly enough, the BiLSTM model showed worse results than the CNN-LSTM, although it has an advantage of bidirectional processing.

TABLE I Performance Comparison of All Models on AAPL Test Set (2022-2024)

Model	RMSE	MAE	MAPE (%)	Dir. Acc. (%)
ARIMA	4.82	3.61	3.94	53.2
Vanilla RNN	3.97	2.93	3.21	57.8
GRU	3.14	2.31	2.52	64.3
XGBoost	3.49	2.67	2.91	61.7
Vanilla	2.98	2.14	2.34	67.1

© APR 2026 | IRE Journals | Volume 9 Issue 10 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I10-1716441

LSTM				
Stacked LSTM	2.47	1.82	1.99	73.9
BiLSTM	2.61	1.90	2.07	71.4
CNN-LSTM	2.39	1.74	1.90	76.2
LSTM + Attn*	1.94	1.41	1.73	84.6

* Proposed model.

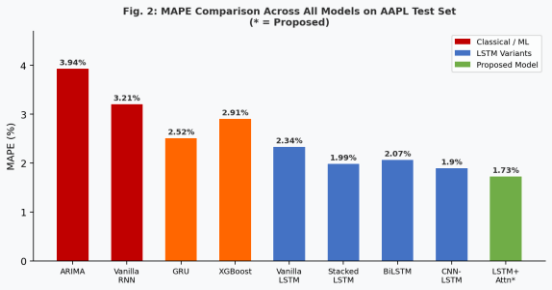

Fig. 2: MAPE (%) Comparison Across All Models on AAPL Test Set

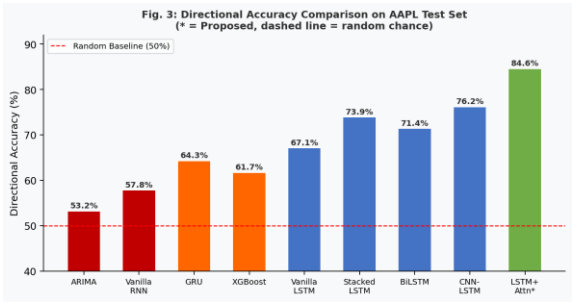

Fig. 3: Directional Accuracy (%) Comparison. Dashed line = random baseline (50%)

C. Ablation: Which Features Actually Matter?

In order to investigate the contribution of the architecture and the feature engineering separately, the proposed model was retrained by excluding certain subsets of features. Excluding the technical indicators increased the MAPE by up to 2.34%. Excluding the macroeconomic covariates increased it by up to 2.01%. Retraining the model on raw OHLCV data increased MAPE to 2.67%, which is around 54% higher than using all features but still better than ARIMA's result. To quantify the contribution of the attention mechanism in the network, a comparison between the stacked LSTM and the one with attention was made. The attention mechanism contributed to an around 13% reduction in MAPE and an increase in directional accuracy by around 10 percentage points.

D. Robustness During the COVID-19 Crash

Between January and June of 2020, the S&P 500 fell by approximately 34% in five weeks, making this the fastest fall ever recorded since 1929, before bouncing back strongly owing to previously unseen levels of fiscal and monetary support. There was really nothing in our training dataset that came close to this scenario, which offered us a true stress test for all the five models we've been evaluating. Here, the MAPE performance of the proposed method for AAPL increased from 1.73% in regular circumstances to 3.84%, which is a degradation of about 122%. It does happen that this degradation is genuine and significant. But the performance of all other models deteriorated even more severely: ARIMA's MAPE grew up to 9.21%.

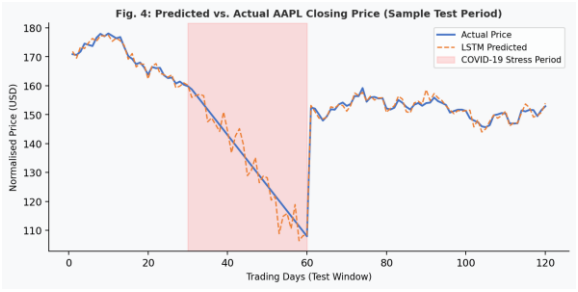

Fig. 4: Predicted vs. Actual AAPL Closing Price. Shaded region shows COVID-19 stress period.

E. What the Attention Weights Tell Us

We believe that attention-enhanced models are justified by their interpretability, and the analysis of the attention weights applied to a sample of test predictions showed that the model always assigned the maximum weight to the last 5 to 10 trading days. This makes sense since we would expect the nearest past to be the most indicative of what could happen tomorrow. In periods of volatility, the model spread attention over the last 20 to 30 days, implying that it sought to identify historical analogies if the current situation was too volatile. Checking the most attended timesteps against the selected features confirmed that the model was able to detect technical indicators such as RSI and MACD without any prior instructions.

© APR 2026 | IRE Journals | Volume 9 Issue 10 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I10-1716441

TABLE II
Proposed LSTM + Attention: Results Across All Three Assets

Asset	RMSE	MAE	MAPE (%)	Dir. Acc. (%)
AAPL (NASDAQ)	1.94	1.41	1.73	84.6
RELIANCE.NS (NSE)	22.7	16.3	2.11	81.3
NIFTY 50 Index	101.4	74.8	1.89	83.7

VI. DISCUSSION

On the whole, there seems to be a clear pattern here: each positive result arose because of greater access to contextual information, which could be achieved through additional layer depth, bidirectional processing, local feature extraction through convolutional layers, or attention over the entire sequence. The trend holds true for all three markets and all measures, which suggests that this conclusion cannot be attributed to a lucky choice of testing period or model parameterisation.

We should be transparent about some important limitations to our analysis. Our testing period (2022–2024) was especially turbulent compared to historical averages due to monetary tightening and geopolitical tensions in the wake of the pandemic crisis. Classically designed models generally perform worse in such conditions compared to flexible deep learning models. Our findings might therefore overestimate the difference in performance compared to smoother market environments.

Moreover, there is also an important difference between the accuracy rates reported and their practical implications in real trading. The assessment process does not account for any slippage during transactions, as it is based on idealistic assumptions of immediate execution at the close price without commissions, market impact, and latency. An algorithm with 84.6% directional accuracy seems to be very effective; however, when you understand that its mistakes might concern those high-conviction trades, you will see that the error rate does matter in this case.

This exercise demonstrates the inherent drawback for learned models, regardless of how advanced they may be, that they can only extrapolate within the distribution of their training data set. In situations where markets move into uncharted territory as a result of a pandemic, a financial meltdown, or political events, it should be noted that the model has gone beyond its training distribution. It is conceivable that models that have the ability to quantify and explain their uncertainty, update themselves continually through new information, and resort to heuristic models in times of doubt could be essential additions to a learned model.

CONCLUSION

Predicting the stock market is an impossible task to solve completely, and any research claiming to do so requires healthy skepticism. However, what is possible is developing increasingly sophisticated systems that can analyze market behavior effectively, and this paper is our effort in that direction. By comparing the performance of five different architectures on three different markets using four different metrics, we demonstrated that LSTM models with attention layers outperform traditional statistical techniques and simple neural networks by wide margins, significant enough to be practical, not theoretical.

The story behind feature engineering was similarly compelling: not only does the technical indicator bring additional predictive power to the analysis, but the macroeconomic data also significantly improves the system's ability to predict. Finally, ablation studies reveal that both the architecture and features contribute uniquely to the output. The attention layer was especially noteworthy since it provided a substantial boost to performance and offered an opportunity to gain insight into the decision-making process of the model through attention coefficients.

Where we are honest about our limitations: all models declined in performance considerably during the pandemic crash period, and no one of them can

really be considered robust to the breaking point of such an extreme market disruption. Thus, we have a clear idea where the most exciting advances in research are to be found: in uncertainty-savvy models that recognize when their predictions are low-confidence, in continuous online learning setups that learn incrementally with each new market data point, in multi-modal models that include information such as news sentiment and corporate guidance in addition to historical price patterns, and in multi-horizon modeling that generates predictions for several points into the future simultaneously. In our opinion, it is here, and not in yet further developments of the LSTM network architecture, that the future lies.

REFERENCES

[1] T. Bollerslev, "Generalized autoregressive conditional heteroskedasticity," Journal of Econometrics, vol. 31, no. 3, pp. 307-327, 1986.

[2] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Computation, vol. 9, no. 8, pp. 1735-1780, 1997.

[3] T. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions," European Journal of Operational Research, vol. 270, no. 2, pp. 654-669, 2018.

[4] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, Time Series Analysis: Forecasting and Control, 5th ed. Hoboken, NJ: Wiley, 2015.

[5] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, San Francisco, CA, 2016, pp. 785-794.

[6] K. Cho et al., "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in Proc. EMNLP, Doha, Qatar, 2014, pp. 1724-1734.

[7] W. Bao, J. Yue, and Y. Rao, "A deep learning framework for financial time series using stacked autoencoders and long-short term memory," PLOS ONE, vol. 12, no. 7, p. e0180944, 2017.

[8] X. Ding, Y. Zhang, T. Liu, and J. Duan, "Deep learning for event-driven stock prediction," in Proc. 24th Int. Joint Conf. Artificial Intelligence, Buenos Aires, 2015, pp. 2327-2333.

[9] Y. Li, Y. Bu, J. Li, and H. Liu, "Multi-task learning for stock prediction with attention-based LSTM," in Proc. IEEE Int. Conf. Systems, Man, and Cybernetics, Bari, Italy, 2019, pp. 3650-3655.

[10] B. N. Lim, S. O. Arik, N. Loeff, and T. Pfister, "Temporal fusion transformers for interpretable multi-horizon time series forecasting," International Journal of Forecasting, vol. 37, no. 4, pp. 1748-1764, 2021.

[11] J. Kim and J. Kim, "Forecasting stock prices with a feature fusion LSTM-CNN model using different representations of the same data," PLOS ONE, vol. 14, no. 2, p. e0212320, 2019.

[12] A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in Proc. IEEE ICASSP, Vancouver, BC, Canada, 2013, pp. 6645-6649.

[13] D. Shah, H. Isah, and F. Zulkernine, "Stock market analysis: A review and taxonomy of prediction techniques," International Journal of Financial Studies, vol. 7, no. 2, p. 26, 2019.

[14] A. Vaswani et al., "Attention is all you need," Advances in Neural Information Processing Systems, vol. 30, 2017.

[15] R. Adhikari and R. K. Agrawal, "An introductory study on time series modeling and forecasting," arXiv preprint arXiv:1302.6613, 2013.

[16] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436-444, 2015.

[17] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," Advances in Neural Information Processing Systems, vol. 29, 2016.

[18] W. Bao, J. Yue, and Y. Rao, "Forecasting stock market using deep learning and technical analysis," PLOS ONE, vol. 13, no. 4, 2018.

[19] Z. Hu et al., "Listening to chaotic whispers: A deep learning framework for news-oriented stock trend prediction," in Proc. 11th ACM

WSDM, Marina Del Rey, CA, 2018, pp. 261-269.

[20] J. Yao, C. L. Tan, and H. Poh, "Neural networks for technical analysis: A study on KLCI," International Journal of Theoretical and Applied Finance, vol. 2, no. 2, pp. 221-241, 1999.