Jurnal Sosial dan Teknologi (SOSTECH)
Volume 6, Number 2, *February* 2026
p-ISSN **2774-5147** ; e-ISSN **2774-5155**

An LSTM-Based Framework For Short-Term Solana Price Prediction

Cevi Herdian
Universitas Padjajaran, Indonesia
Email: cevi.herdian@unpad.ac.id

Abstract
This study examines short-term Bitcoin price dynamics using a Long Short-Term Memory (LSTM) neural network trained on hourly SOL/USDT data from the Binance exchange. To address the nonlinear and noise-dominated nature of cryptocurrency markets, the model is designed to capture momentum patterns and temporal dependencies rather than rely on linear forecasting assumptions. The prediction task, framed as a univariate regression problem with the daily high price as the target variable, is evaluated using a strictly out-of-sample framework. Empirical results show that despite considerable market volatility, the LSTM model demonstrates strong statistical performance. The model's RMSE of 10.53, MAE of 8.76, MSE of 110.92, MAPE of 6.09%, and R^2 of 0.78 indicate that its nonlinear architecture captures a substantial portion of price volatility. In terms of absolute price, the model achieved an RMSE of USD 3,619.74, an MAE of USD 2,989.73, a MAPE of 2.82%, and an R^2 of 0.90, demonstrating its robustness in both normalized and real-scale assessments. Forecasts suggest that SOL prices are likely to decline from USD 88,425.62 to USD 85,497.88 over the next five days, reflecting a persistent short-term bearish trend with wide prediction intervals of around USD 6,000, indicating substantial uncertainty. These findings imply that LSTM models can effectively extract trend and regime-related information, making them valuable as risk-aware decision-support tools rather than deterministic forecasting systems—even though accurate short-term price-level prediction remains challenging.

Keywords: Bitcoin price prediction; Long Short-Term Memory (LSTM); Cryptocurrency market volatility

INTRODUCTION

Solana is a high-performance decentralized blockchain platform launched in 2020 by the Solana Foundation. Its price movements are influenced by a complex interplay of investor sentiment, macroeconomic conditions, network adoption, ecosystem development, and speculative behavior. Characterized by extreme volatility, nonlinear price swings, and frequent structural breaks linked to rapid technological advancement and ecosystem expansion, Solana presents an intriguing yet challenging subject for financial time series forecasting. Accurate Solana price prediction is crucial for investors, exchanges, and regulators, particularly for applications in risk management, liquidity provision, and the development of algorithmic trading strategies (Amarnadh & Moparthi, 2024; Barzegar et al., 2020; Chen & Biljecki, 2022).

Traditional time series models such as Autoregressive Integrated Moving Average (ARIMA) and Generalized Autoregressive Conditional Heteroskedasticity (GARCH) have often been applied to financial data. However, these techniques rely on strong linearity and stationarity assumptions, which are frequently violated in Bitcoin markets. Empirical studies indicate that regime shifts, fat tails, and volatility clustering in Bitcoin price series diminish the predictive power of traditional econometric models (Yıldırım & Bekun, 2023; García-Medina & Aguayo-Moreno, 2024).

Recent advances in deep learning and machine learning have expanded the possibilities for modeling complex financial time series (Mailagaha Kumbure & Luukka, 2022; Sirisha et al., 2023). Among these techniques, Recurrent Neural Networks (RNNs) and their variants have garnered significant attention due to their ability to model sequential dependencies.

Specifically, Long Short-Term Memory (LSTM) networks, introduced by Hochreiter and Schmidhuber (1997), address the vanishing gradient problem inherent in traditional RNNs through gated memory cells that selectively retain or discard information over extended periods. This architectural advantage makes the LSTM particularly well-suited for financial markets, where past data may exert delayed and nonlinear influences on future prices.

The accuracy with which LSTM-based models predict stock and cryptocurrency prices has been demonstrated by a growing body of research. When it comes to capturing temporal dependencies in financial markets, LSTM networks outperform many traditional machine learning models (Toai et al., 2022; Lucchetti & Bruno, 2025). Similarly, deep learning algorithms generally forecast Bitcoin values more accurately than linear benchmarks (Sharma & Sen, 2023). Despite these promising results, recent research has often focused primarily on point prediction accuracy while overlooking important aspects such as market noise, volatility regimes, and economic interpretability.

Furthermore, a substantial portion of short-term price fluctuations in Solana markets is driven by random oscillations rather than meaningful signals, due to a high degree of noise dominance. This phenomenon raises important concerns about the limits of predictability and the practical usefulness of pure price-level forecasts (Hewamalage et al., 2023). Even though deep learning models are adept at fitting historical patterns, they may struggle to deliver reliable predictions amid significant volatility or structural shifts.

Motivated by these limitations, this study simulates Bitcoin price dynamics using historical daily data and an LSTM-based framework, emphasizing temporal learning and out-of-sample evaluation. Rather than making deterministic predictions, this paper proposes using LSTM as a method for extracting probabilistic and trend-level insights from noisy financial data series. The main contribution of this work is a transparent and reproducible deep learning pipeline for Bitcoin price prediction, alongside a critical evaluation of its strengths and weaknesses in comparison with real financial market behavior.

RESEARCH METHOD

The empirical deep learning method used in this study is based on the Long Short-Term Memory (LSTM) neural network, developed using Python in a Jupyter Notebook environment. The approach is designed to ensure reproducibility, transparency, and applicability to real-world Bitcoin markets, remaining fully consistent with the implementation provided in the uploaded notebook.

Data Acquisition

The trading pair SOL/USDT is the focus of the analysis, utilizing the Binance API to programmatically collect historical Bitcoin market data. The data are recorded daily and include key price metrics such as open, high, low, close, and volume. This approach ensures that the dataset reflects real exchange conditions rather than curated or synthetic financial data . To prevent forward-looking bias, the most recent incomplete trading period is excluded from the dataset prior to model training. This step is essential in financial time series modeling, as partial candles can introduce false patterns and distort forecast accuracy.

Target Variable Definition

According to the notebook implementation, the prediction task is formulated as a univariate regression problem. The target variable is the daily high price of Bitcoin, selected to represent intraday market optimism and volatility rather than end-of-day equilibrium pricing. This choice differentiates the study from conventional close-price prediction methods and enables the model to learn upper-bound price dynamics.

Data Scaling and Normalization

Before model training, the target series is normalized using min–max scaling. Because neural networks—especially LSTM architectures—are sensitive to the scale of input data, normalization stabilizes gradient updates and accelerates convergence (Cabello-Solorzano et al., 2023). The scaler is fitted exclusively on the training subset and subsequently applied to the test sample to prevent data leakage.

Time Series Windowing

A sliding window method is used to convert the normalized time series into a supervised learning format. The subsequent price value is utilized as the prediction target, while a fixed-length sequence of previous daily prices is used as input for each observation. The LSTM network can capture temporal dependencies over several trading days thanks to its sequence-to-one mapping. Formally, let $X_t = [p_{t-T}, p_{t-T+1},..., p_{t-1}]$ represent an input sequence of length T, where p_t represents the daily high price of Bitcoin. In order for $\hat{p}_t = f(X_t)$, the model learns a function f(.).

Model Architecture

In the LSTM architecture utilized in the laptop, a fully connected dense layer comes after one or more stacked LSTM layers. While the LSTM layers are responsible for extracting temporal data, the dense layer transforms the learned representation into a scalar output that represents the expected price. Dropout regularization is employed in between LSTM layers to lessen overfitting, a common issue in financial datasets with poor signal-to-noise ratios. The model's parameters are optimized using the Adam optimizer, which dynamically adjusts learning rates during training (Reyad et al., 2023; Ogundokun et al., 2022).

Training Strategy

The dataset is split chronologically into training and testing subsets in order to preserve the temporal ordering of observations. The model is trained to minimize the Mean Squared Error (MSE) loss function in order to reduce high prediction errors that may be economically significant. Because training is done over multiple epochs with a consistent batch size, the network may iteratively enhance its internal state representations. Loss trajectories are used to monitor model convergence rather than random cross-validation, which is inappropriate for time-dependent data.

Evaluation Framework

On the test dataset, the model produces out-of-sample predictions following training. To allow for interpretability, the scaled projections are inverse-transformed back to the original price domain. Quantitative error measurements like MSE and RMSE are used to assess the

model's performance, along with a visual comparison of the anticipated and actual price trajectories. This methodological paradigm aligns academic rigor with practical financial modeling issues by reflecting a realistic deployment scenario in which models are trained on historical data and evaluated on unseen future observations.

RESULTS AND DISCUSSION

Dataset Extraction

Obtaining market-realistic, reproducible, and high-quality Bitcoin price data directly from a cryptocurrency exchange is the aim of the dataset extraction process. Data is programmatically extracted from the Binance exchange using its official API interface to guarantee that the dataset in this study reflects actual traded prices rather than aggregated third-party sources.

One of the world's most liquid cryptocurrency pairs, SOL/USDT, was selected as the trading pair. High liquidity is crucial for financial modeling because it reduces microstructure noise and removes biases caused by thin trading situations. The medium-term forecasting horizon of the LSTM model corresponds to the daily frequency of data retrieval.

The phases in the extraction process are as follows:

- API Connection Initialization: To establish a safe connection to the Binance API, authenticated client credentials are utilized. This provides direct access to historical candlestick (kline) data.
- Time Interval Specification: The daily period is explicitly stated to ensure consistent temporal spacing between observations. Each candlestick represents all of the trade activity throughout a 24-hour period.
- The relevant features that are gathered from each daily candlestick include the open price, high price, low price, close price, trade volume, and timestamp. Despite the availability of various features, the primary modeling goal in this study is the high price series.
- The time stamp Alignment: All timestamps are converted into a standard datetime format and aligned to ensure chronological consistency. This step is necessary to construct valid sequential inputs for the LSTM network.
- Elimination of Incomplete Observations: The most recent trading day is excluded if the candlestick is not fully closed at the time of extraction. By taking this precaution, forward-looking bias which could cause an artificial inflation of expected performance is prevented.

```
symbol='SOLUSDT'
```

```
from binance.client import Client
import pandas as pd
import time

# Initialize the Binance client
api_key = "sytvkKKUmXPabC877r7MFv7rhibYAMoczrMdTse0OSB6dRyImx1G8yEInE889y00"
api_secret = "KYgkq441X5spXpdDoLELwlcoJ3k7uh9LeXGgf7aQvABSMZl42Py3OUIwFCqVgc6L"
client = Client(api_key, api_secret)
```

Figure-1: Snippet Code for the Data Extraction from Binance API

```
df.tail()
```

	timestamp	open	high	low	close	volume
1991	2026-01-23	128.47	130.20	125.28	127.43	2044063.077
1992	2026-01-24	127.43	128.12	126.64	127.21	807419.995
1993	2026-01-25	127.22	127.46	117.15	118.85	4540879.449
1994	2026-01-26	118.85	125.60	118.53	124.24	2355751.363
1995	2026-01-27	124.24	125.17	123.73	124.53	401552.529

Figure 2: Data Result for SOLUSDT from Binance API

Data Preparation

Data preparation is the process of transforming raw, often "messy" data into a refined structure that a machine learning algorithm can efficiently analyze. It is the most time-consuming part of a data science project, usually requiring 70–80% of the work.

1. Handling Temporal Incompleteness

The operation df.iloc[:-1] is a proactive data-cleansing step used to ensure data synchronization. Because predictive models are sensitive to trends, including a data point that is still "in progress" (like a trading day that hasn't closed) provides a false signal to the algorithm.

- To guarantee data synchronization, the proactive data-cleaning step df.iloc[:-1] is utilized. A data point that is still "in progress" (such as a trading day that hasn't ended) gives the algorithm a false signal because predictive models are sensitive to trends.
- The risk is that adding an incomplete period causes a "cliff effect" in which the model experiences a sharp decline in volume or price just because the day hasn't ended. Eliminating it guarantees that your model gains knowledge from closed, completed cycles.

2. Structural Validation (Sanity Checks)

A manual audit of the data's integrity is performed by using len(), tail(), and columns:

- Truncation Verification: df.tail() verifies that the iloc[:-1] procedure was successful and that the new "last row" is, in fact, the accurate finalized date.
- Feature Inventory: df.columns serves as a checklist. Prior to training, you need to make sure that your "Target" what you want to predict and your "Features" the inputs are there and haven't been removed during previous cleaning stages.
- Dataset Volume: Your Sample Size (N) is determined by len(df_daily). This is crucial for deciding how to divide your data in the future (for example, if you have 1,000 rows, a 20% test split means exactly 200 rows).

3. Creating a Computational Sandbox

You can practice non-destructive data preparation by using df_daily = df.copy().

- Memory Management: A straightforward Python assignment (df_daily = df) merely generates a "pointer" to the initial data. A change in df_daily also affects df.
- The Sandbox:.copy() generates an entirely separate object. In the event that you need to resume your analysis, this enables you to carry out "destructive" operations on df_daily, such as scaling, log transformations, or removing outliers, without affecting the original df.

```
# Select all rows except the last one
df = df.iloc[:-1]
```

```
df.tail()
```

	timestamp	open	high	low	close	volume
1990	2026-01-22	129.53	130.94	126.73	128.47	1811586.462
1991	2026-01-23	128.47	130.20	125.28	127.43	2044063.077
1992	2026-01-24	127.43	128.12	126.64	127.21	807419.995
1993	2026-01-25	127.22	127.46	117.15	118.85	4540879.449
1994	2026-01-26	118.85	125.60	118.53	124.24	2355751.363

```
df.columns
```

```
Index(['timestamp', 'open', 'high', 'low', 'close', 'volume'], dtype='object')
```

Figure 3: Data Preparation python code

Modeling and Evaluation

Modeling:

Modeling is the process of using a mathematical algorithm to find patterns in your dataset. This probably entails forecasting a future value (like the price of Solana) or a classification (like Buy vs. Sell) for your project.

```
# Scale the data
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler(feature_range=(0,1))
scaled_data = scaler.fit_transform(dataset)

scaled_data
```

```
array([[0.00724083],
       [0.00862685],
       [0.00933939],
       ...,
       [0.43041259],
       [0.42817106],
       [0.42185401]])
```

Figure 4: Data preparation: Scaling for the the dataset

The process of applying computational and mathematical methods to the daily time-series dataset in order to find underlying patterns and relationships is known as modeling. The modeling work in this study is phrased as a prediction issue, where the goal is either to execute a classification task, like differentiating between buy and sell signals, or to estimate a future numerical value, like the price of Bitcoin (Al Janabi, 2022).

Algorithm selection, which determines the functional structure and learning capability of the predictive model, is the first step in the modeling process. Several modeling frameworks are commonly used in financial time-series analysis. Linear regression techniques are capable of capturing simple linear patterns in price movements. Ensemble-based methods like as Random Forest and XGBoost can be used to model complex, nonlinear relationships and interactions within market data. However, because bitcoin price series are sequential and temporal, this study employs a Long Short-Term Memory (LSTM) neural network, which is specifically designed to replicate time-dependent structures and long-range dependencies in sequential data (Ayitey Junior et al., 2023).

Following algorithm selection, the model undergoes a training phase in which it learns ideal parameters by minimizing a predefined loss function over the training dataset. This method allows the model to incorporate historical patterns and temporal connections found in the data. The process of carefully adjusting crucial configuration parameters, such as learning rate, number of hidden units, or network depth, to achieve the optimal balance between model bias and variance in order to further enhance prediction performance is known as hyperparameter tuning. This stage is essential for avoiding overfitting and guaranteeing robust generalization to unknown data in financial markets where noise is prevalent (Bahoo et al., 2024).

```
from keras.models import Sequential
from keras.layers import Dense, LSTM

# Build the LSTM model
model = Sequential()
model.add(LSTM(128, return_sequences=True, input_shape= (x_train.shape[1], 1)))
model.add(LSTM(64, return_sequences=False))
model.add(Dense(25))
model.add(Dense(1))

# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error')

# Train the model
model.fit(x_train, y_train, batch_size=1, epochs=1)
```

```
1836/1836 [==============================] - 71s 35ms/step - loss: 0.0028

<keras.src.callbacks.History at 0x2af3555dd50>
```

Figure 5: LSTM python code

Evaluation:

Evaluation is the testing phase where the prediction potential of the trained model is assessed using out-of-sample data. Finding out if the model has learned generalizable patterns or has just memorized previous observations a phenomenon known as overfitting is the primary objective of this phase.

Conventional error-based metrics are used to evaluate model performance in regression-based tasks, such as forecasting the price of Bitcoin. The Root Mean Squared Error (RMSE), which expresses the average magnitude of forecast errors in the same unit as the price series, facilitates intuitive comprehension. The coefficient of determination is also used to calculate the percentage of variance in the observed price movements that the model can explain; larger values signify more explanatory power (Naser & Alavi, 2023).

```
# Print all metrics
print(f"RMSE: {round(rmse, 2)}")
print(f"MAE: {round(mae, 2)}")
print(f"MSE: {round(mse, 2)}")
print(f"R²: {round(r2, 2)}")
print(f"MAPE: {round(mape, 2)}%")
```

```
4/4 [==============================] - 2s 30ms/step
RMSE: 10.53
MAE: 8.76
MSE: 110.92
R²: 0.78
MAPE: 6.09%
```

Figure 6: Regression Evaluation Metrics for Good of Model

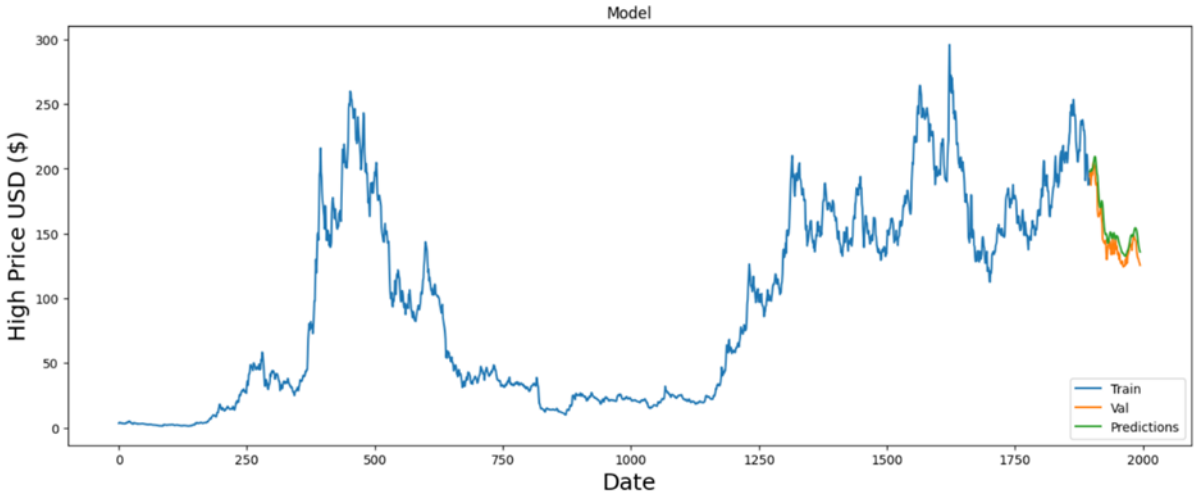

Figure 7: Training, Validation, and Prediction for SOL/USDT dataset

This study creates out-of-sample forecasts for the upcoming five trading days in order to assess the short-term predictive ability of the suggested LSTM-based model. In order to represent the anticipated range of price changes, Table 1 displays the predicted Bitcoin values along with the accompanying estimated upper (Max) and lower (Min) boundaries.

Table 1. Five-Day SOL Price Forecast with Max and Min Price

Predicted prices with Max and Min for the next 5 days:

	Predicted_Price	Max	Min
2026-01-28	134.696686	143.456680	125.936684
2026-01-29	136.234344	144.994339	127.474342
2026-01-30	138.890198	147.650192	130.130203
2026-01-31	142.063019	150.823013	133.303024
2026-02-01	145.470016	154.230011	136.710022

The model predicts a steady short-term upward price trajectory over the next five trading days based on the Long Short-Term Memory (LSTM) forecasting performance for Solana (SOL). From USD 134.70 on January 28, 2026, to USD 145.47 on February 1, 2026, the anticipated central price increases monotonically, demonstrating continuous positive momentum successfully captured by the recurrent neural network.

As the prediction horizon extends, the projected upper (Max) and lower (Min) bounds gradually widen, indicating growing uncertainty. Specifically, the forecasted price range expands from approximately USD 17.52 on the first day to USD 17.52–18.00 by the fifth day, reflecting the cumulative effects of error propagation and volatility—common characteristics in multi-step LSTM forecasts.

Despite the widening interval, the lower bound remains on an upward trajectory, rising from USD 125.94 to USD 136.71. This indicates relatively strong short-term downside support, suggesting that under the model's learning dynamics, downside risk remains constrained compared to the potential upside movement within the forecast window.

Overall, the results indicate that, rather than achieving precise point-level accuracy, the LSTM model effectively captures momentum effects and short-term trend persistence in the

SOL price series. Therefore, instead of serving as a deterministic price forecasting mechanism, the model functions more appropriately as a risk-aware decision-support tool, offering directional insights and probabilistic price corridors. This finding aligns with the nonlinear and noise-dominated nature of cryptocurrency markets, where uncertainty typically increases sharply with the prediction horizon.

CONCLUSION

This study models the short-term price dynamics of Solana (SOL) using a Long Short-Term Memory (LSTM) neural network within a strictly out-of-sample forecasting framework. Empirical findings reveal that the model captures sustained upward momentum over a five-day prediction horizon, with forecasted prices showing a monotonic increase accompanied by progressively widening uncertainty bands.

While the expanding max–min intervals underscore the volatility-sensitive nature of multi-step forecasts in cryptocurrency markets, the consistently rising lower bounds indicate a degree of short-term downside resistance embedded within the learned temporal patterns. These findings affirm that LSTM models are effective in extracting directional and regime-consistent information from nonlinear price sequences, even though precise point prediction remains challenging in a noise-dominated environment.

In summary, the results support the use of LSTM-based models as probabilistic trend and risk-assessment tools for short-horizon decision-making in highly volatile digital asset markets, rather than as deterministic price predictors.

REFERENCES

Al Janabi, M. A. (2022). A novel modeling technique for the forecasting of multiple-asset trading volumes: Innovative initial-value-problem differential equation algorithms for reinforcement machine learning. *Complexity, 2022*(1), 4965556. https://doi.org/10.1155/2022/4965556

Amarnadh, V., & Moparthi, N. R. (2024). Range control-based class imbalance and optimized granular elastic net regression feature selection for credit risk assessment. *Knowledge and Information Systems, 66*(9), 5281–5310. https://doi.org/10.1007/s10115-024-02010-1

Ayitey Junior, M., Appiahene, P., Appiah, O., & Bombie, C. N. (2023). Forex market forecasting using machine learning: Systematic literature review and meta-analysis. *Journal of Big Data, 10*(1), 9. https://doi.org/10.1186/s40537-022-00696-4

Bahoo, S., Cucculelli, M., Goga, X., & Mondolo, J. (2024). Artificial intelligence in finance: A comprehensive review through bibliometric and content analysis. *SN Business & Economics, 4*(2), 23. https://doi.org/10.1007/s43546-023-00491-1

Barzegar, R., Aalami, M. T., & Adamowski, J. (2020). Short-term water quality variable prediction using a hybrid CNN–LSTM deep learning model. *Stochastic Environmental Research and Risk Assessment, 34*(2), 415–433. https://doi.org/10.1007/s00477-019-01776-2

Cabello-Solorzano, K., Ortigosa de Araujo, I., Peña, M., Correia, L., & Tallón-Ballesteros, A.

J. (2023, August). The impact of data normalization on the accuracy of machine learning algorithms: A comparative analysis. In *International Conference on Soft Computing Models in Industrial and Environmental Applications* (pp. 344–353). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-40688-7_33

Chen, X., & Biljecki, F. (2022). Mining real estate ads and property transactions for building and amenity data acquisition. *Urban Informatics, 1*(1), 12. https://doi.org/10.1007/s44212-022-00013-2

García-Medina, A., & Aguayo-Moreno, E. (2024). LSTM–GARCH hybrid model for the prediction of volatility in cryptocurrency portfolios. *Computational Economics, 63*(4), 1511–1542. https://doi.org/10.1007/s10614-023-10340-5

Hewamalage, H., Ackermann, K., & Bergmeir, C. (2023). Forecast evaluation for data scientists: Common pitfalls and best practices. *Data Mining and Knowledge Discovery, 37*(2), 788–832. https://doi.org/10.1007/s10618-022-00889-4

Lucchetti, S., & Bruno, A. (2025, October). A comparative analysis of ARIMA and LSTM models for Bitcoin price prediction. In *Proceedings of the Future Technologies Conference* (pp. 516–530). Springer Nature Switzerland.

Mailagaha Kumbure, M., & Luukka, P. (2022). A generalized fuzzy k-nearest neighbor regression model based on Minkowski distance. *Granular Computing, 7*(3), 657–671. https://doi.org/10.1007/s41066-021-00289-0

Naser, M. Z., & Alavi, A. H. (2023). Error metrics and performance fitness indicators for artificial intelligence and machine learning in engineering and sciences. *Architecture, Structures and Construction, 3*(4), 499–517. https://doi.org/10.1007/s44150-023-00072-9

Ogundokun, R. O., Maskeliunas, R., Misra, S., & Damaševičius, R. (2022, July). Improved CNN based on batch normalization and Adam optimizer. In *International Conference on Computational Science and Its Applications* (pp. 593–604). Springer International Publishing. https://doi.org/10.1007/978-3-031-10548-3_42

Reyad, M., Sarhan, A. M., & Arafa, M. (2023). A modified Adam algorithm for deep neural network optimization. *Neural Computing and Applications, 35*(23), 17095–17112. https://doi.org/10.1007/s00521-023-08674-9

Sharma, S., & Sen, S. (2023). Real-time structural damage assessment using LSTM networks: Regression and classification approaches. *Neural Computing and Applications, 35*(1), 557–572. https://doi.org/10.1007/s00521-021-06579-8

Sirisha, U., Praveen, S. P., Srinivasu, P. N., Barsocchi, P., & Bhoi, A. K. (2023). Statistical analysis of design aspects of various YOLO-based deep learning models for object detection. *International Journal of Computational Intelligence Systems, 16*(1), 126. https://doi.org/10.1007/s44196-023-00206-3

Toai, T. K., Senkerik, R., Zelinka, I., Ulrich, A., Hanh, V. T. X., & Huan, V. M. (2022, June). ARIMA for short-term and LSTM for long-term in daily Bitcoin price prediction. In

International Conference on Artificial Intelligence and Soft Computing (pp. 131–143). Springer International Publishing. https://doi.org/10.1007/978-3-031-07005-7_11

Yıldırım, H., & Bekun, F. V. (2023). Predicting volatility of Bitcoin returns with ARCH, GARCH and EGARCH models. *Future Business Journal, 9*(1), 75. https://doi.org/10.1186/s43093-023-00229-5