DOI: https://doi.org/10.31861/bmj2025.01.14

Melnyk H.V., Melnyk V.S., Vikovan V.K.

CRYPTOCURRENCY PRICE FORECASTING WITH DAUBECHIES WAVELETS AND EVOLUTIONARY FUZZY TIME SERIES

In this article we investigate the task of forecasting cryptocurrency prices based on time series using different modeling approaches. Several methods have been implemented and compared: the Dobeschi wavelet decomposition method for preliminary smoothing of data, a Random Forest machine learning model, a fuzzy logic system with automatic rule selection using a genetic algorithm, and a basic neural network. Each of the approaches was tested on the same time series, and their effectiveness was evaluated using the relevant accuracy metrics. We pay particular attention to how wavelet-based preprocessing (e.g., wavelet smoothing) impacts forecasting accuracy. The obtained results allow to draw conclusions about the strengths and weaknesses of each method in the context of cryptocurrency price forecasting, as well as the feasibility of their use depending on the characteristics of the data and the task.

Key words and phrases: Timeseries forecasting, fuzzy logic, genetic algorithms, machine learning, random forest, wavelet decomposition..

Yuriy Fedkovych Chernivtsi National University, Chernivtsi, Ukraine
e-mail: *g.melnik@chnu.edu.ua, va.melnyk@chnu.edu.ua, vikovan.valentyn@chnu.edu.ua*

Introduction

Among economic indicators, cryptocurrency markets exhibit high volatility and non-stationarity. Cryptocurrency prices fluctuate under the influence of a wide range of factors. Traditional forecasting methods, in particular statistical models, are not effective enough to accurately predict such complex processes. In this case, non-standard approaches to time series analysis deserve special attention.

In particular, we examine machine learning methods, wavelet analysis, hidden layer neural networks, and fuzzy set theory combined with genetic algorithms. These methods are compared using standard metrics such as root mean square error.

The machine learning algorithm "random forest" performed quite well in terms of error and speed. This algorithm was also combined with Daubechies wavelet decomposition. During decomposition, insignificant components were discarded, which made it possible to smooth the time series and reject noise. When working with markers such as cryptocurrency prices, it is necessary to take

УДК 519.115.1
2010 *Mathematics Subject Classification:* 05A15.

into account the high daily volatility of the market. Daubechies wavelet?based smoothing reduced daily market noise and yielded lower error metrics.

We also developed an evolutionary fuzzy-rule system that uses genetic algorithms to generate and select fuzzy logic rules for time-series forecasting. This approach takes time to execute the program code, but shows a very low error rate. The integration of mutations at specified steps of the genetic algorithm further reduces the error.

We evaluated a single-hidden-layer neural network for its forecasting accuracy.

This work addresses the need for noise-resistant and adaptive forecasting in volatile financial markets. By integrating complementary computational-intelligence techniques-wavelet preprocessing, evolutionary fuzzy logic, and neural networks-our hybrid framework demonstrates robust performance under conditions of uncertainty and complex temporal structure.

Daubechies Wavelets

The **Locally Stationary Wavelet model (LSW)** is a time series model designed to process non-stationary signals where statistical properties of data, such as mean and variance, change over time. Unlike traditional models, which assume constant statistical behavior, the LSW model reflects the evolution of the time series structure using wavelet decompositions. In this approach, the series is represented as a sum of wavelet basis functions that vary in both time and frequency, allowing the model to localize features that occur at different scales and time points.

The LSW excels at forecasting, smoothing, and identifying trends in time series data, especially when it comes to signals that exhibit changing patterns and structures over time.

Let $\{X_t\}$ be a non-stationary time series whose statistical properties (mean, variance) evolve over time. The Locally Stationary Wavelet (LSW) model represents X_t as a sum of discrete, wavelet-basis functions that vary across both time and frequency. Unlike traditional time-series models-which assume constant statistical behavior-LSW adapts to changing patterns via wavelet decomposition [3].

In the Locally Stationary Wavelet (LSW) framework, we relax the strict unit-variance Gaussian assumption and instead model the noise coefficients as a zero-mean white-noise process with arbitrary variance. Formally,

$$X_t \;=\; \sum_{j,k=-\infty}^{\infty} w_{j,k}(t)\,\psi_{j,k}(t)\,\xi_{j,k},$$

where $w_{j,k}(t)$ are time-varying wavelet coefficients, $\psi_{j,k}(t)$ are the wavelet basis functions at scale j and shift k, and

$$\xi_{j,k} \;\sim\; \mathrm{WN}(0,\sigma^2),$$

denotes a white-noise process with mean zero and variance σ^2. This generalization allows the model to accommodate different noise levels across applications and facilitates estimation of σ^2 directly from the data, thereby improving flexibility and robustness in forecasting nonstationary time series.

Daubechies wavelets offer compact support, orthogonality, and a high number of vanishing moments-properties that make them ideal for capturing the abrupt, irregular fluctuations typical of cryptocurrency prices. Both the scaling function $\varphi(t)$ and the wavelet function $\psi(t)$ satisfy recursive relationships derived from a finite-set of filter coefficients, enabling efficient decomposition and reconstruction.

We use the **Discrete Wavelet Transform (DWT)** with Daubechies wavelets, which are known for their compact support, orthogonality, and ability to capture both smooth trends and abrupt changes in data.

Given a finite realization of time series $X = \{x_1, x_2, ..., x_T\}$, DWT decomposes it into a series of approximation and detail components at multiple scales. At each level of decomposition, the time series is split into approximation coefficients A_j representing low-frequency (trend) information and detail coefficients D_j representing high-frequency (noise or short-term fluctuations). This can be written as

$$X \to \{A_j, D_J, D_{J-1}, ..., D_1\}$$

where J is the maximum level of decomposition, A_j is the approximation at the coarsest level, and D_1 is the detail at the finest level (captures the highest-frequency noise).

To smooth the time series, we remove the highest-frequency detail coefficients, typically D_1 and sometimes D_2. These components are most sensitive to minor, rapid changes in the data and are often associated with noise.

The smoothed signal $\widetilde{X}$ is reconstructed by applying the inverse DWT (IDWT) using only the remaining components:

$$\widetilde{X} = IDWT(A_j, D_J, D_{J-1}, ..., D_1), \text{ for some } k > 1$$

That is, we discard $D_1, D_2, ..., D_{k-1}$ retaining only the larger-scale features from level k and above. This process effectively filters out high-frequency noise while preserving the main trends of the series.

By applying Daubechies wavelet-based smoothing, we can localize transient features at multiple scales and reject high-frequency noise before forecasting.

Wavelet-Smoothed Random Forest Forecasting

Random Forest builds and combines the results of many decision trees to improve prediction. Each tree in the forest is trained on a randomly selected subset of the training data using a technique called bootstrap sampling, and only a random subset of features is considered at each split within the tree. This double randomness-in the data and in the features - reduces overfitting and increases the robustness of the model.

When used for regression tasks, the Random Forest regressor works by averaging the predictions from all the individual trees. Instead of relying on a single model that may be overfitted or sensitive to noise, Random Forest uses the collective output of many diverse models. This "wisdom of the crowd" effect helps the model produce smoother, more stable predictions, especially when dealing with complex, nonlinear relationships in the data. This is particularly useful in cases where traditional linear models fail, thus it is fit for cryptocurrency forecasting, financial modeling, and other regression applications.

After exploring traditional time series methods, a machine learning solution using random forest regression was implemented. This algorithm offers a different perspective on time series forecasting by treating the forecasting problem as a supervised learning task. The random forest approach transforms temporal dependencies in data into a feature engineering task, where past values serve as predictors of future values.

Building on the strengths of wavelet decomposition and Random Forest regression, we developed a hybrid approach combining both methods. This integration aims to leverage the noise reduction capabilities of wavelets and the pattern recognition capabilities of Random Forest, potentially achieving better prediction accuracy than either method alone.

The implementation begins with applying wavelet decomposition to smooth Bitcoin price data before feeding it into the Random Forest model. This preprocessing step removes high-frequency noise while preserving underlying price trends and patterns.

To transform the time series into a supervised learning problem, we construct feature vectors using a lag embedding of the smoothed series. For a given lag window size L, the input vector and target are defined as:

$$\mathbf{x}_t = [\tilde{x}_{t-1}, \tilde{x}_{t-2}, \ldots, \tilde{x}_{t-L}]$$

$$y_t = x_t$$

Note that the input is based on the smoothed series $\tilde{X}$, while the target is the original (unsmoothed) value from X. This allows the model to learn from denoised data while still being evaluated on real-world values.

A Random Forest regressor is trained on dataset $\mathcal{D} = \{(\mathbf{x}_t, y_t)\}_{t=L+1}^{T}$ to learn the mapping from smoothed lagged inputs to the next-step target value. Random Forest is an ensemble of decision trees:

$$\hat{y}_t = \frac{1}{B} \sum_{b=1}^{B} T_b(\mathbf{x}_t)$$

where T_b is the b-th regression tree trained on a bootstrap sample of the data, B is the number of trees in the forest.

Each tree partitions the input space to minimize variance within leaf nodes, resulting in a robust, nonlinear predictor that handles noisy data and feature interactions well.

Evolutionary Fuzzy-Rule Optimization

Evolutionary Algorithms (EA) are a family of optimization methods based on the mechanisms of natural evolution. EA solve complex optimization and search problems by reproducing the evolutionary behavior observed in nature. The basic principle is to maintain a population of potential solutions that evolves over time. By imitating evolutionary processes such as reproduction, mutation, and selection, the algorithm gradually improves the quality of solutions over successive generations.

Genetic Algorithms (GA) are an important subclass of the broader category of evolutionary algorithms. In this class, each possible solution is encoded in a structure known as a chromosome, which consists of individual units called genes. These genes correspond to specific elements or variables of the solution being optimized. A chromosome is essentially a vector or array that encodes a full set of parameters or decision variables needed to describe a candidate solution.

A group of such chromosomes forms a so-called population. Typically, this population is initialized randomly, providing a diverse starting point for the search process. Each solution is evaluated using a function that measures its effectiveness in solving the problem at hand. Depending on whether the goal is to maximize or minimize this value, the algorithm selects the most promising

individuals for reproduction. During the reproduction phase, the genetic material of the selected parents is combined through a process similar to biological crossbreeding, resulting in new offspring. Random changes, known as mutations, are introduced from time to time to preserve genetic diversity and prevent premature convergence. These changes allow the algorithm to explore new areas in the solution space.

The iterative process continues until a predefined termination condition is met. This can be a fixed number of iterations, a time limit, or the achievement of a solution that meets the required performance criteria [5].

Fuzzy logic (FL) is a method of reasoning that mimics the way humans make decisions. Unlike traditional logic used in computers, which relies on precise input data to produce a specific result of "True" or "False," similar to a strict "YES" or "NO," FL allows for a range of values that represent degrees of truth. This means that it can cope with the uncertainty and nuances that are often present in real-world scenarios, where decisions are not always black and white.

FL rules usually have an "if-then" structure, for example: if temperature is high, then fan speed is fast. However, since there are no clearly defined values in fuzzy logic, all rules apply simultaneously to varying degrees. For example, if the temperature is somewhere between "moderate" and "high," the fan speed will also be somewhere between "medium" and "fast." This means that the system evaluates how well each rule fits the current situation and uses this to form a final decision.

As a result, FL rules allow you to create flexible, adaptive control systems that behave more like human reasoning than rigid traditional logic.

Developing FL rules manually can be a tedious and time-consuming task, so evolutionary algorithms can be used to automate this process. In this approach, each candidate rule base is treated as a separate individual in a population that evolves over time. First, a large number of fuzzy "if-then" rule sets are randomly generated, each embodying its own hypothesis about how input data should affect output. Their effectiveness is evaluated using a fitness index that rewards accurate predictions and penalizes overly complex rule sets. Then, the best ones are selected for reproduction, exchanging and recombining their rules. Random mutations create new variations of rules, preventing the search from getting stuck in a narrow range of possibilities. Over successive generations, this evolutionary cycle refines the rule bases and hones a compact, highly efficient set of fuzzy rules. The end result is a rule base that balances interpretability with optimized control behavior, with all rules discovered automatically rather than created manually.

Fuzzy logic has been investigated as a powerful tool for timeseries forecasting in [1, 2]. Usage of genetic algorithms for generation of fuzzy logic rulesets has been proposed in [4].

Evolutionary Fuzzy-Rule Selection for Time-Series Forecasting

We present a GA-driven approach to optimize fuzzy-logic rules for Bitcoin price prediction. This method marries the linguistic modeling capabilities of fuzzy systems with genetic algorithms to automatically discover high-performance rule bases.

The implementation begins with the definition of the main components of the fuzzy system. Membership functions are defined for different levels of price changes and a population of fuzzy rule bases is created, which will evolve through genetic optimization.

The genetic algorithm component initializes a population of fuzzy rule bases, each of which contains a set of rules that map input lag characteristics to output predictions. The rules are structured using linguistic variables that describe price movements in human-understandable terms.

A critically important component of the system is the development of appropriate membership functions. Triangular membership functions have been implemented for seven different categories of price changes, ranging from extreme declines to extreme increases. Such detailed categorization allows the system to capture the slightest market movements.

The fuzzy logic system processes input data with a shift using an advanced rule base to generate forecasts. The system includes a complex syntactic rule analyzer that interprets rules based on strings and builds a corresponding fuzzy control system.

Suppose we use a lag window of size L, then a rules has a following form:

$$\text{IF } x_{t-1} \text{ is } A_1 \text{ AND } x_{t-2} \text{ is } A_2 \text{ AND ... AND } x_{t-L} \text{ is } A_L \text{ THEN } x_t \text{ is B}$$

where x_{t-k} are lagged values of the smoothed time series, A_k and B are fuzzy sets (e.g., "Increase", "Const", "Decrease High") defined over the universe of discourse via membership functions $\mu_{A_k}(x)$.

An example of fuzzy logic rulebase:

```
[
"lag_0 const and lag_1 const and lag_2 const and lag_3
   const and lag_4 const then const",
"lag_0 decrease_high and lag_4 increase then increase",
"lag_0 increase and lag_4 const then decrease",
"lag_1 increase_high and lag_3 decrease then
   increase_high",
"lag_0 increase and lag_2 const then increase_high",
"lag_3 increase_high and lag_0 decrease then
   increase_high",
"lag_1 decrease and lag_2 const then decrease_high",
"lag_0 increase_high and lag_0 increase_high then
   decrease_high",
"lag_0 increase_high and lag_2 const then increase_high"
]
```

In this example, `lag_i` is timeseries data from $i+1$ days before the predicted: $x_{t-(i+1)}$.

In the genetic algorithm, each individual (or chromosome) represents a complete fuzzy rule set:

$$Chromosome = [r_1, ..., r_M]$$

where each gene r_i is a fuzzy rule encoded as a tuple of linguistic variables:

$$r_i = (A_1, ..., A_L, B)$$

The number of possible rules grows exponentially with the number of fuzzy sets M and lag size L, i.e., up to M^L possible antecedents.

Given a set of fuzzy rules and a fuzzy partition of the input space, we forecast the next value $\widehat{x_t}$ using a defuzzification of the weighted output values from matching rules:

$$\hat{x}_t = \frac{\sum\limits_{i=1}^{N} w_i \cdot c_i}{\sum\limits_{i=1}^{N} w_i}$$

where w_i is the firing strength of rule r_i, defined as

$$w_i = \prod_{k=1}^{L} \mu_{A_k^{(i)}}(x_{t-k}),$$

c_i is the centroid (or representative value) of the consequent fuzzy set $B^{(i)}$.

The genetic algorithm uses tournament selection (100 generations), crossover, and mutation operations to evolve a population of fuzzy rule bases. The evolution process includes an adaptive mutation rate, which increases at regular intervals to maintain genetic diversity and avoid local optima. The mutation rate is maintained at 1% throughout the execution of the algorithm and at specially defined generations, this rate is increased to 10%.

We use real valued mutation operator, where real number representation of the value of linguistic variable in fuzzy rule changes with predefined step. An example of real valued mutation:

$$r_i = (\text{Decrease High}, \text{Const}, \text{Const}) => r_i' = (\text{Decrease High}, \text{Increase}, \text{Const})$$

We use standard uniform crossover, where each gene of each of the two parents has 50-percent chance of being selected for offspring, ensuring that genetic material is combined in a balanced and non-biased manner. An example crossover for parents $p^{(1)} = [r_1^{(1)}, r_2^{(1)}, r_3^{(1)}, r_4^{(1)}, r_5^{(1)}]$, $p^{(2)} = [r_1^{(2)}, r_2^{(2)}, r_3^{(2)}, r_4^{(2)}, r_5^{(2)}]$ and a random mask $[0, 1, 1, 0, 1]$ the offspring will be $c = [r_1^{(1)}, r_2^{(2)}, r_3^{(2)}, r_4^{(1)}, r_5^{(2)}]$, i.e. each position is selected based on the mask of random decisions.

The fitness function for this genetic algorithm is defined as an inverse of the root mean squared error:

$$f(\widehat{x}) = \frac{1}{\sqrt{\sum\limits_{i=1}^{N} \frac{(\widehat{x_i^2} - x_i^2)}{N}} + \varepsilon}$$

where $x = (x_i)$ — observed values of time-series, $\widehat{x} = (\widehat{x_i})$ — values, predicted by fuzzy rule system, N — number of samples, $f(\widehat{x})$ — fitness function of that prediction, $\varepsilon > 0$ is a small constant to avoid division by zero.

The main evolution cycle runs for 100 generations, evaluating the performance of each rule base on training data and selecting the best ones for reproduction.

After 100 generations of evolution, the system determines the most effective set of fuzzy rules and applies it to the test data. The evolutionary rule base demonstrates the system's ability to automatically detect meaningful patterns in price data.

The evolution of average mean squared error of the population is presented on Figure 1. You can see the stages at which the probability of mutation increased (from 1% to 10%), at which the error value rapidly increased and fell below the previous value at subsequent stages.

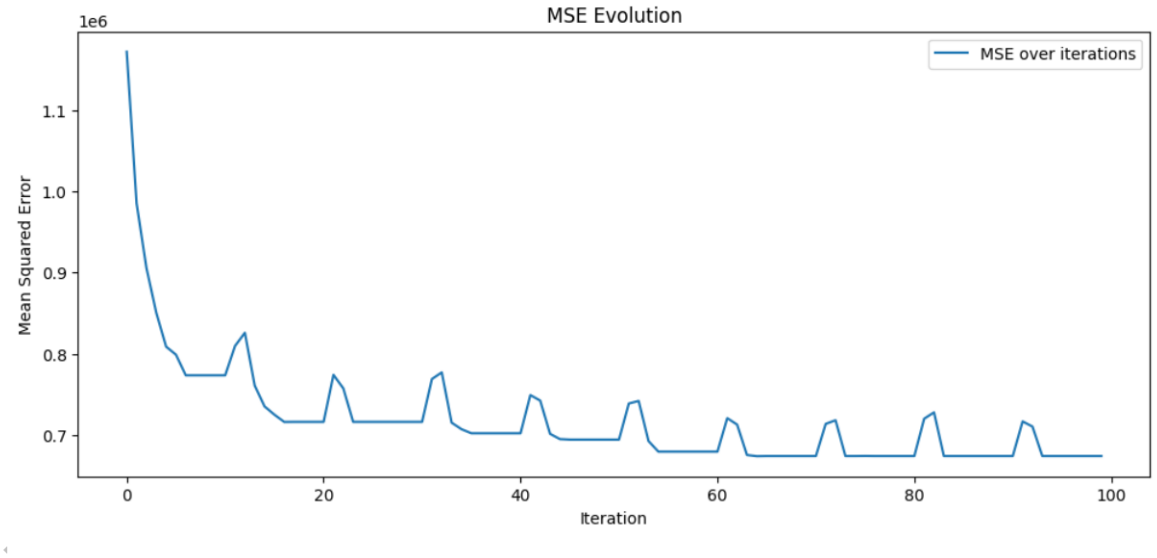

Figure 1: Evolution of Mean Squared Error over Genetic Algorithm Iterations

FEEDFORWARD NEURAL NETWORK FORECASTING

The application of neural networks with hidden layer for financial markers predictions are described in [6, 7]. In these works it has been concluded, that the most effective architecture is that with input neurons divisible by 5, and in range from 5 to 20, which is explained with stock exchange trading week being equal to 5 days. The hidden layer in investigated architectures is 15 neurons. Simple RNNs are used to capture short-term temporal dependencies in price series [8, 9]. 1D-CNNs extract local temporal features by applying convolutional filters over lagged price inputs [10, 11, 15]. LSTMs employ gating mechanisms to learn long-range dependencies and mitigate vanishing gradients in financial time series [12, 13]. Self-attention architectures (e.g. Stockformer) model global sequence relationships without recurrence, improving parallelism [14, 16]. Combine convolutional feature extraction with LSTM memory cells to leverage both local pattern recognition and sequential modeling [17].

A three-layer feedforward neural network was selected as the baseline forecasting model due to its well-established capacity to approximate complex nonlinear mappings with limited computational overhead. Under the Universal Approximation Theorem, a single hidden layer suffices to represent any continuous function on a compact domain, which justifies the choice of this architecture for modeling the nonlinear dynamics of cryptocurrency time series. Unlike recurrent or convolutional architectures, the feedforward network eschews temporal feedback loops and extensive parameterization, thereby reducing the risk of overfitting on relatively short financial datasets. Furthermore, the use of lagged inputs and ReLU activations enables the network to capture autoregressive dependencies efficiently, facilitating rapid training and clear attribution of performance gains to subsequent wavelet preprocessing and fuzzy-GA rule optimization.

We implement a three-layer feedforward network in PyTorch to benchmark forecasting performance. Neural networks capture complex nonlinear relationships in time-series data thanks to their multilayer architecture and nonlinear activation functions. The implementation uses PyTorch to build a three-layer feedforward neural network specifically designed to handle lag functions. The network architecture consists of an input layer that accepts five lag features (cryptocurrency price

values for the last 5 days), a hidden layer with 15 neurons, and a single output neuron for price prediction.

The architecture uses Rectified Linear Unit (ReLU) activation functions between layers to introduce nonlinearity, allowing the network to model complex relationships in Bitcoin price data. The feedforward method determines how data passes through the network, transforming the input lag features into a single predicted value.

The data preparation process follows a similar approach to previous methods, creating lag functions based on the Bitcoin price time series. A systematic function engineering pipeline is implemented, which generates five lag functions representing the previous five time steps. We split the data into training and test sets in an 80-20 ratio, providing sufficient data for both training and evaluating the model. We scale lag features to zero mean and unit variance before training.

In the training process we use PyTorch's automatic differentiation capabilities to optimize network weights through backpropagation. It uses the Adam optimizer, which adapts the learning rate for each parameter, and a mean squared error loss function, which is suitable for regression tasks.

The training cycle lasts 1,000 epochs, with each epoch processing the entire training dataset in batches of 10 samples. This mini-batch approach provides a balance between computational efficiency and gradient estimation accuracy.

Results and Discussion

In this research we used available history of bitcoin price data from Kaggle [9]. This dataset contais historic data for bitoin price over the daterange from 2010-07-18 to 2022-08-23. For timeseries forecast we used different offsets from the end date.

To ensure a reliable comparison between the standard Random Forest and the wavelet-enhanced version, we conducted an extensive evaluation at various time intervals and configurations of parameters. This comprehensive study helps to understand the consistency and reliability of performance improvements.

To assess consistency and reliability, we compared standard and wavelet-enhanced Random Forest across multiple parameter settings. The comparison structure includes two functions: one for the standard Random Forest and another that includes wavelet smoothing. Both functions accept parameters for data selection, time window size, and lag configuration, allowing for flexible experimentation.

The evaluation process involves running both methods on 200 different random time windows from the Bitcoin dataset, testing with 5 and 7 lags to assess the impact of different time dependencies.

The comparison revealed several important conclusions. In 200 different time windows, the wavelet-smoothed Random Forest consistently outperformed the standard approach. The average RMSE values show that the wavelet-smoothed version with 5 lags achieved an average RMSE of 2891.671, compared to 3137.581 for the standard approach - an improvement of approximately 7.8%. That method shows also improvements for other metrics: for mean absolute percentage error (MAPE) from 10.73 to 10.06, for symmetric mean absolute percentage error (SMAPE) from 10.265 to 9.694, for coefficient of determination (R2) increase from -3.8 to -3.2.

Interestingly, the results also show that using 7 lags instead of 5 does not necessarily improve performance, as both the standard and smoothed versions show slightly higher RMSE values with a larger number of lags. This suggests that after a certain point, additional historical information may introduce more noise than useful patterns.

Consistent gains across time windows confirm the reliability of the hybrid approach. Even during periods of high volatility, such as significant price declines, as seen in the data, wavelet processing helps the Random Forest model maintain better forecasting accuracy. This stability makes the combined approach particularly suitable for cryptocurrency markets, which are characterized by sharp price fluctuations and high volatility. As shown in Figure 2, the wavelet-smoothed Random Forest model closely tracks real Bitcoin price movements.

In this combined plot, the blue line shows actual Bitcoin prices, the orange line shows predictions from the wavelet-smoothed Random Forest, and the green line shows forecasts from the GA-optimized fuzzy-logic model. Presenting both predicted series together on the same time axis allows immediate visual comparison of their relative accuracy and responsiveness to market movements.

This overlay confirms that the fuzzy-evolutionary approach (green) more closely tracks sharp price spikes and dips compared to the wavelet-smoothed Random Forest (orange), demonstrating the benefit of rule-based adaptivity in high-volatility periods.

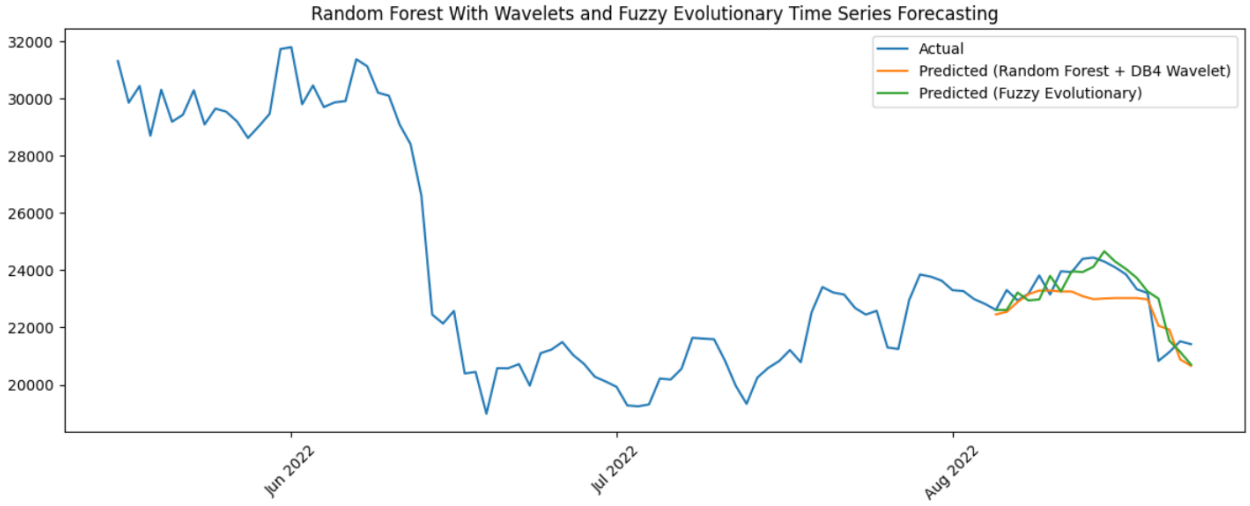

Figure 2: Actual vs. Predicted Bitcoin Prices Using Wavelet-Smoothed Random Forest and Fuzzy Evolutionary Algorithm

The genetic fuzzy approach achieved an average root mean square error (RMSE) of 1011.034, as illustrated in Figure 2, which is a significant improvement compared to both the standard Random Forest (3137.581) and the wavelet-extended Random Forest (2891.671), demonstrating the effectiveness of combining genetic fuzzy and wavelet approaches for predicting cryptocurrency prices. There is also significant improvement for other metrics, for MAPE 3.17, for SMAPE 3.141, for MASE 1.255 and for R2: 0.462.

The neural network achieves an average root mean square error (RMSE) of 1579.28, asillustrated in Figure 3, which is lower than the wavelet-based Random Forest model, and demonstrates competitive performance. The training process shows a consistent decrease in losses, starting at 36,485,368 in epoch 50 and decreasing to 35,014,856 in epoch 1000, indicating successful training. Analysis of the forecast graph shows that the neural network captures the general trend of Bitcoin price movement, although it tends to give smoother forecasts compared to actual volatile price data.

This smoothing effect is characteristic of neural networks trained with RMSE loss, as they tend to predict average values to minimize quadratic error.

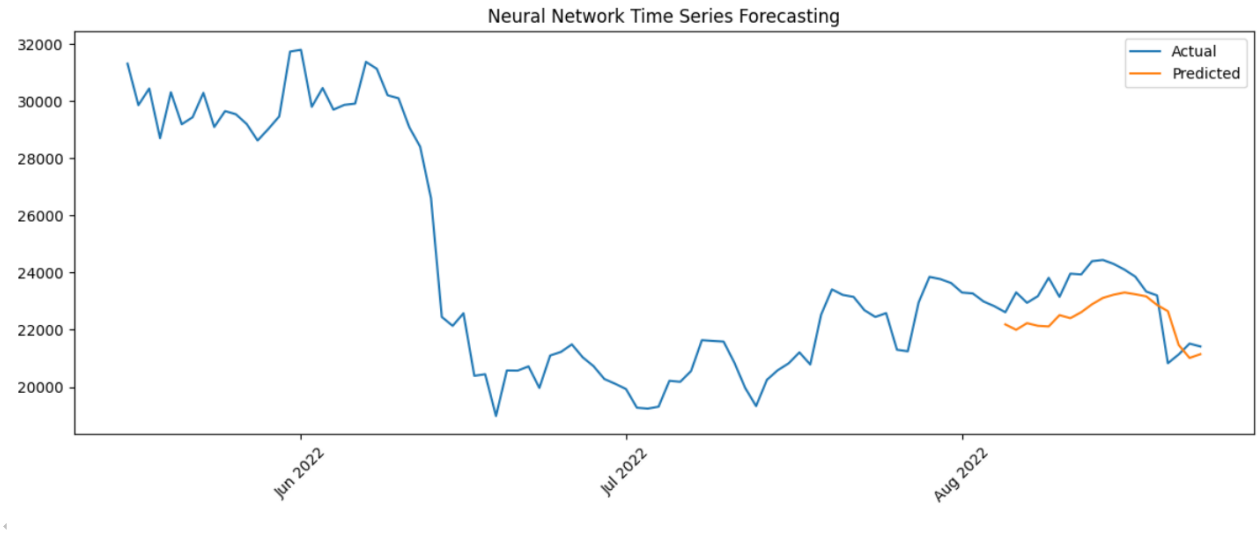

Figure 3: Actual vs. Predicted Bitcoin Prices Using Feedforward Neural Network

Root mean square errors for 10 random ranges out of 200 tested are presented in the table below. Random forest methods are denoted RF, smoothed with Daubechies wavelet decomposition are denoted with D, amount of lags window are in brackets. RMSE for evolutionary fuzzy algorithm is denoted EA FL, feedforward neural network with NN and convolutional neural network with CNN.

offset	RF (lag 5)	RF (7)	RF D (5)	RF D (7)	EA FL	NN	CNN
906.0	292.2	320.8	298.9	322.3	**239.8**	489.7	304.6
619.0	702.2	**691.1**	695.3	704.5	774.7	2386.0	722.2
771.0	291.2	208.8	179.1	**134.9**	148.0	150.1	157.7
387.0	2451.5	2401.2	1909.8	2037.4	1213.0	3782.7	**1212.0**
297.0	8032.3	7729.2	6812.4	6670.5	2056.3	**2032.9**	2069.9
728.0	272.3	335.1	472.5	509.6	344.0	290.9	**218.9**
641.0	2379.7	2414.7	2375.8	2395.0	**513.3**	917.1	539.1
875.0	1654.5	1636.5	1703.7	1694.8	880.1	889.9	**826.2**
55.0	9172.4	9274.6	8268.3	8192.6	**876.7**	2275.4	1553.2
941.0	424.0	421.6	232.4	**154.0**	258.3	443.2	259.5

Table 1: RMSE values for different models and offsets

offset	RF (lag 5)	RF (7)	RF D (5)	RF D (7)	EA FL	NN	CNN
906.0	2.3969	2.6504	2.4343	2.6500	**1.8618**	4.0330	2.6297
619.0	3.0485	**3.0125**	3.0232	3.1036	3.3410	11.3930	3.0361
771.0	2.8457	1.8894	1.6657	1.2137	1.1769	**1.1610**	1.2923
387.0	6.2980	6.2291	4.9782	5.3722	2.9622	7.5544	**2.9226**
297.0	12.4421	11.9160	10.3099	10.0586	2.8124	2.8846	**2.6325**
728.0	1.8757	2.4284	3.6788	4.0164	2.4655	2.0009	**1.3204**
641.0	12.4847	12.7135	12.4432	12.5702	**2.5601**	4.2789	2.5885
875.0	26.9740	26.2824	28.1973	27.9536	9.5756	10.3183	**8.6701**
55.0	41.0920	41.4883	37.0658	36.7361	**2.7172**	12.8298	5.1310
941.0	3.6157	3.6403	2.3623	**1.5362**	2.3755	3.8619	2.3990

Table 2: MAPE values for different models and offsets

offset	RF (lag 5)	RF (7)	RF D (5)	RF D (7)	EA FL	NN	CNN
906.0	2.3779	2.6385	2.4130	2.6336	**1.8518**	3.9128	2.5890
619.0	3.0615	3.0215	3.0231	3.1069	3.3452	10.2849	**3.0047**
771.0	2.7971	1.8663	1.6542	1.2131	**1.1790**	1.2640	1.2799
387.0	6.2577	6.1349	4.9305	5.2971	2.9911	9.1067	**2.9441**
297.0	13.3444	12.7482	10.9483	10.6694	2.8383	2.9161	**2.6283**
728.0	1.9007	2.4682	3.7596	4.1107	2.4691	1.7099	**1.3088**
641.0	13.5974	13.8622	13.5518	13.6985	**2.5972**	4.4111	2.6172
875.0	23.0450	22.4075	24.0891	23.8722	8.8537	10.9930	**7.7821**
55.0	33.2223	33.4599	30.5444	30.3248	**2.6668**	7.2578	4.9327
941.0	3.7450	3.7642	2.4025	**1.5517**	2.3957	4.2425	2.4140

Table 3: SMAPE values for different models and offsets

offset	RF (lag 5)	RF (7)	RF D (5)	RF D (7)	EA FL	NN	CNN
906.0	**1.5006**	1.6743	2.1311	2.3443	0.2832	2.4072	0.3970
619.0	2.5110	2.4429	3.3173	3.3674	0.9744	8.8418	**0.8787**
771.0	1.2332	0.8202	1.0652	0.7726	**0.1726**	0.4847	0.1876
387.0	1.4022	**1.3948**	1.6431	1.7933	1.6236	2.0247	1.6014
297.0	5.9219	5.5560	6.8317	6.6664	2.6405	**1.3977**	2.5349
728.0	1.3081	1.7084	4.2629	4.7568	0.4535	1.3067	**0.2441**
641.0	10.6493	10.9550	15.1982	15.3594	**0.6180**	4.0826	0.6445
875.0	8.0983	7.7152	10.7872	10.4473	0.8578	3.2363	**0.7495**
55.0	9.3326	9.5212	14.0025	14.2791	**0.9928**	1.7420	1.7955
941.0	1.7572	1.8062	1.7745	1.1621	**0.3061**	1.9729	0.3172

Table 4: MASE values for different models and offsets

offset	RF (lag 5)	RF (7)	RF D (5)	RF D (7)	EA FL	NN	CNN
906	0.8077	0.7525	0.8022	0.7501	**0.8672**	0.4149	0.7814
619	0.3583	0.3564	0.2984	0.2928	0.3306	**0.3622**	0.1268
771	0.4790	0.5368	0.6892	**0.6982**	0.6875	0.2421	0.2180
387	0.9100	0.9077	0.9448	0.9492	**0.9519**	0.8826	0.8996
297	0.3136	0.3136	0.1523	0.1523	**0.7702**	0.6938	0.6304
728	0.0398	0.0172	0.0955	0.0609	**0.1603**	0.0009	0.1594
641	0.3645	0.3211	0.0000	0.1609	**0.9259**	0.9106	0.8957
875	0.2131	0.1392	0.0194	0.2523	**0.3879**	0.0714	0.0978
55	0.1232	0.1661	0.0603	0.1690	**0.9698**	0.8177	0.8686
941	0.8240	0.8387	**0.9577**	0.9568	0.8528	0.7605	0.7766

Table 5: Pearson correlation coefficient r values

Here Pearson correlation coefficient r is calculated with the following formula:

$$r = \frac{\sum_{i=1}^{N}(y_i - \overline{y})(\hat{y}_i - \overline{\hat{y}})}{\sqrt{\sum_{i=1}^{N}(y_i - \overline{y})^2 \ \sum_{i=1}^{N}(\hat{y}_i - \overline{\hat{y}})^2}}.$$

Average values for 200 tested ranges.

Model	RMSE	MAPE	SMAPE	MASE	r
mse_rf_5	3137.581	10.730	10.265	4.558	0.225407
mse_rf_7	3124.796	10.723	10.212	4.548	0.187698
mse_rf_smooth_5	2891.671	10.063	9.694	6.361	0.356471
mse_rf_smooth_7	2905.612	10.071	9.659	6.398	0.358103
evolutionary_plain	**1011.034**	**3.170**	**3.141**	**1.255**	**0.690407**
nn_plain	1579.283	5.487	5.481	4.981	0.428941

Table 6: Mean values for each metric

Conclusions

The conducted research demonstrates the effectiveness of combining modern computational intelligence techniques for the task of forecasting cryptocurrency prices. In particular, the application of Daubechies wavelet decomposition as a preprocessing step notably improves forecasting accuracy by filtering high-frequency noise while preserving the underlying trends. When integrated with Random Forest regression, wavelet smoothing resulted in consistently lower RMSE, MAPE, SMAPE and MASE values compared to the standard Random Forest model across multiple experimental runs. However, increasing the lag window size from 5 to 7 did not lead to further improvements and, in some cases, slightly degraded performance, suggesting that beyond a certain point, additional historical information may introduce more noise than useful signal.

Among all the models tested, the evolutionary fuzzy logic system showed the most promising results, achieving the lowest average RMSE, MAPE, SMAPE and MASE, and outperforming both Random Forest variants and the neural network model. High Pearson r values confirm strong agreement between model forecasts and observed prices, reinforcing the superiority of the evolutionary

fuzzy approach. This approach, which relies on a genetic algorithm to automatically discover and optimize fuzzy rule sets, effectively captures subtle patterns in highly volatile time series. It balances accuracy with interpretability and adapts well to the non-stationary nature of financial data. Meanwhile, the neural network model demonstrated competitive performance, with its architecture capable of modeling nonlinear dependencies. However, due to the nature of the loss functions, it produced smoother forecasts that tended to underrepresent abrupt fluctuations, which are common in cryptocurrency markets.

Overall, the findings confirm that hybrid models-particularly those combining wavelet-based noise reduction with advanced learning techniques such as evolutionary fuzzy systems?offer a robust solution for forecasting in high-volatility environments. The choice of forecasting method should ultimately depend on the specific characteristics of the data and the desired balance between interpretability, computational efficiency, and prediction accuracy.

References

[1] Matviychuk A. Fuzzy logic approach to identification and forecasting of financial time series using Elliott wave theory // Fuzzy Economic Review. - 2006. - Vol. 11, no. 02. - P. 45-54. URL: https://doi.org/10.25102/fer.2006.02.04.

[2] Bielinskyi A., Ivanov B., Petrenko C. Fuzzy time series forecasting using semantic artificial intelligence tools // Neuro-Fuzzy Modeling Techniques in Economics. - 2022. - No. 11. - P. 157-198. URL: https://doi.org/10.33111/nfmte.2022.157.

[3] Rhif M., Hleli H., Smara Y. Wavelet transform application for non-stationary time-series analysis: A Review // Applied Sciences. - 2019. - Vol. 9, no. 7. - P. 1345. URL: https://doi.org/10.3390/app9071345.

[4] Surmann H., Selenschtschikow A. Automatic generation of fuzzy logic rule bases: Examples I // Proceedings of the First International ICSC Conference on Neuro-Fuzzy Technologies (16-19 January 2002). - Vienna, 2002. - P. 75-81.

[5] Glybovets M., Gulaeva N. Evolutionary algorithms. - Kyiv: NaUKMA, 2013. - 828 p.

[6] Matviychuk A. V. Artificial Intelligence in Economics: Neural Networks, Fuzzy Logic: Monograph. - Kyiv: KNEU, 2011. - 439 p.

[7] Matviychuk A. V. Research on the dependence of the quality of forecasting security prices by neural networks on the form of input data presentation // Collection of scientific papers of Cherkasy State Technological University. Series: Economic Sciences. - 2003. - Issue 8. - P. 147-156.

[8] Tayib H., Abdulazeez M. A. A Review of Bitcoin Price Prediction Based on Deep Learning Algorithms // The Indonesian Journal of Computer Science. - 2024. - Vol. 13, no. 2. - P. 45-58. URL: https://doi.org/10.33022/ijcs.v13i2.3858.

[9] Top 10 Cryptocurrencies Historical Dataset [Електронний ресурс] // Kaggle. URL: https://www.kaggle.com/datasets/kaushiksuresh147/top-10-cryptocurrencies-historical-dataset.

[10] Zhang X., Huang Z., Wu Y., Lu X., Qi E., Chen Y., Xue Z., Wang Q., Wang P., Wang W. Multi-period Learning for Financial Time Series Forecasting // Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD -25). - New York; ACM, 2025. - P. 2848?2859. URL: https://doi.org/10.1145/3690624.3709422.

[11] Badar W., Ramzan S., Raza A., Fitriyani N. L., Syafrudin M., Lee S. W. Enhanced Interpretable Forecasting of Cryptocurrency Prices Using Autoencoder Features and a Hybrid CNN-LSTM Model // Mathematics. - 2025. - Vol. 13, no. 12. - Art. 1908. URL: https://doi.org/10.3390/math13121908.

[12] Zeng Z., Kaur R., Siddagangappa S., Rahimi S., Balch T., Veloso M. Financial Time Series Forecasting using CNN and Transformer // arXiv preprint arXiv:2304.04912. - 2023. URL: https://arxiv.org/abs/2304.04912).

[13] Kong X., Chen Z., Liu W. et al. Deep learning for time series forecasting: a survey // Int. J. Mach. Learn. & Cybern. - 2025. URL: https://doi.org/10.1007/s13042-025-02560-w.

[14] Kilic D. K., Ugur O. Hybrid wavelet-neural network models for time series // Applied Soft Computing. - 2023. - Vol. 144. - Art. 110469. URL: [https://doi.org/10.1016/j.asoc.223.110469](https://doi.org/10.1016/j.asoc.2023.110469) .

[15] Li Y., Dai W. Bitcoin price forecasting method based on CNN-LSTM hybrid neural network model // The Journal of Engineering. - 2020. - P. 344-347. URL: https://doi.org/10.1049/joe.2019.1203.

[16] Wu J., Zhang X., Huang F., Zhou H., Chandra R. Review of deep learning models for crypto price prediction: Implementation and evaluation // arXiv preprint arXiv:2405.11431. - 2024. URL: https://arxiv.org/abs/2405.11431.

[17] Ahmed B., Abedin M. Z., Hajek P., Yuan K. Cryptocurrency price forecasting - A comparative analysis of ensemble learning and deep learning methods // International Review of Financial Analysis. - 2024. - Vol. 92. - Art. 103055. URL: https://doi.org/10.1016/j.irfa.2023.103055.

Received 22.06.2025

Вікован В.К., Мельник Г.В., Мельник В.С. *Прогноз цін на криптовалюту через нечіткі часові ряди та розклад в вейвлети Добеші.* // Буковинський матем. журнал — 2025. — Т.13, №1. — С. 157–172.

У цій статті ми досліджуємо задачу прогнозування цін на криптовалюту за допомогою часових рядів і порівнюємо чотири підходи: вейвлет-декомпозиція Добеші для згладжування даних; Random Forest — ансамблеву модель машинного навчання; еволюційно нечітку систему, що автоматично генерує й відбирає правила за допомогою генетичного алгоритму; базову нейронну мережу. Кожний метод протестовано на однакових часових рядах, а їх ефективність оцінили за стандартними метриками точності. Особливу увагу приділено впливу вейвлет-згладжування на якість прогнозів. Отримані результати дозволяють виявити сильні та слабкі сторони кожного підходу та обґрунтувати їх використання залежно від характеристик даних і постановки задачі.