Academic Journal of International University of Erbil

Journal Homepage: https://ojs.cihanrtv.com/index.php/public

PRINT-ISSN: 2519-6928

Online-ISSN: 3080-7174

Hyperparameter Tuning of the Prophet Model Using Particle Swarm Optimization: A Case Study on Ethereum

Kwestan Ahmed Ismael[1], Heshu Othman Faqe[2], Mohammed Hussein Abdalla[3], Hindreen A. Taher[4,5]*

[1]Department of Mathematics, College of Basic Education, University of Raparin, Sulaimani, IRAQ

[2]Department of Statistics and Informatics, College of Administration and Economic, University of Sulaimani, Sulaimani, IRAQ

[3]Department of Computer Science, College of Basic Education, University of Raparin, Sulaimani, IRAQ

[4]Department of Information Technology, College of Commerce, University of Sulaimani, Sulaimani, IRAQ

[5]Department of Software Engineering, Faculty of Engineering & Computer Science, Qaiwan International University Sulaymaniyah, Kurdistan Region-IRAQ.

DOI: https://doi.org/10.63841/31651

Received 13 Aug 2025; Accepted 104 Sep 2025; Available online 20 Jan 2026

ABSTRACT:

In this work we use historical market data from Bitget to predict weekly open prices of Ethereum (ETH) for a 96-week period with the Prophet forecast model trained by using Particle Swarm Optimization (PSO) algorithm. Because of this, the research delves into automated hyperparameter tuning for Prophet in order to improve forecast performance on cryptocurrency markets where volatility, structural breaks and irregular trading patterns pose a significant challenge to time series prediction. The PSO algorithm is a good method to explore the high dimensional parameter space in which it can strike between the global analysis and local exploitation for detecting minimal forecast errors. Based on evaluating model performance for which we used accuracy metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) in training, test holdout & full-fit situations. PSO-optimized Prophet: The results show a great in-sample fitting and fast convergence behaviour, as the best CV RMSE is slightly higher than the lowest one should have obtained if used only 10 iterations. Although forecasts exhibit stability and track long-term trends well, the model does not predict short-term fluctuations in the holdout set with high accuracy (wider forecast uncertainty intervals). Our results shed light on the utility of PSO to improve Prophet-based price prediction in cryptocurrencies, reinforce the relevance of uncertainty quantification in asset markets and inform risk-aware decisions of financial agents dealing with unstable assets

Keywords: Ethereum, Prophet model, Particle Swarm Optimizer, Seasonality, Time series.

1 INTRODUCTION

The rapidly growing digitalization and the increasing acceptance of cryptocurrencies have led to forecasting their prices as an important research area. The latter the DeFi ecosystem — and Ethereum (ETH), in particular, which has become the second-largest cryptocurrency by market cap as of this writing largely on the strength of its foundation for countless decentralized financing applications, non-fungible tokens (NFTs) and other token-based/asset offerings [1]. Precise Ethereum price predictions are important for traders, investors and to the researcher which in turn can allow them to make better decisions regarding their portfolio and risk management. However, cryptocurrencies are also very volatile [2], have frequent structural breaks, and exhibit different types of trading irregularities which makes the process especially hard. Prophet, a procedure for forecasting time series data based on an additive model where non-linear trends

*Corresponding author: beston.muhammed@chu.edu.iq

are fit with yearly, weekly, and daily seasonality, plus holiday effects developed by Facebook's Core Data Science team for modeling complex seasonalities with multiple. The Prophet model: Prophet hadditive regression modeluristicforest is only good for seasonal data. It does not have the attribute to remove seasonality on its own so we will the package decomposes a time series into trend, seasonality, and holiday effects. It can handle linear and logistic growth trends, multiple seasonal patterns via Fourier series and is robust to missing data and outliers. It also auto-detects changepoints from the trend, which is wonderful for financial time series data as markets states are known to shift abruptly. Another feature that is important from finance perspective where you might rely on financial decision making then the uncertainty intervals of forecasts which are highly valuable which Prophet does provide commenting and explaining each function unit.

While these features make Prophet extremely easy to use and have contributed significantly to its popularity, the predictive performance of a Prophet model is strongly influenced by the specific choice of hyperparameters such as the changepoint prior scale, seasonality prior scale, mode (additive or multiplicative) and number of changepoints. Here we discussed how to avoid underfitting and overfitting, both improvement points will be helpful in this model tuning techniques. Most practitioners are inclined to tune the algorithm manually which can be computationally expensive and not exhaustive enough for exploring complex, high-dimensional parameter spaces — A grid search. This limitation can be handled by using Particle Swarm Optimization (PSO) as efficient and adaptable search tool for tuning the hyperparameters of Prophet. PSO is a population-based metaheuristics algorithm firstly proposed by [3], inspired from the social behavior of bird flocks and fish schools. In PSO, while searching for a solution, a group of particles-(candidate solutions) changes its positions and velocities according to the best-known position and global-best-known-position found by the swarm as shown in figure 1. This trade-off between exploration (exploring widely) and exploitation (zoning in on good solutions) is what makes PSO appropriate for complex and multimodal problems This approach considers each particle as particular set of training hyperparameters in the context of Prophet. Models with those parameters are then fit to evaluate the performance of each particle and therefore error metrics (examples include RMSE) calculated through rolling-origin cross-validation. PSO is an iterative method that refines the positions of the swarm to find a set of hyperparameters which minimizes forecasting error. It avoids all the brute-force search and eliminates time wastage, but also it might find the parameter combination even human cannot tune during manual tuning notifications.

1.1 PROBLEM STATEMENT

Ethereum's high volatility, structural breaks, and irregular trading patterns make accurate forecasting challenging. While the Prophet model can capture complex seasonality and trends, its performance depends heavily on optimal hyperparameters, which are difficult to tune manually. This study addresses that gap by integrating Particle Swarm Optimization (PSO) to efficiently search the parameter space and improve Prophet's predictive accuracy for Ethereum's weekly open prices.

1.2 LITERATURE REVIEW

Nowadays, hybrid models based on a fusion of time-series forecasting techniques with metaheuristic optimization methods are becoming popular in the context of financial analytics. Moreover, a novel additive model called Prophet by Taylor & Letham [4] aimed to automatically decompose time series data by trend, seasonality, and certain holidays in the form of misspecifications (i.e. gaps) or irregularities; however such methods were shown to be highly dependent on hyperparameter tuning leading to inconsistent performance in other literature. In 1995, Kennedy & Eberhart [3] proposed Particle Swarm Optimization (PSO) as a bio-inspired algorithm for automating the tuning of these parameters; it conducts population-based searches that are capable of thoroughly exploring high-dimensional parameter spaces because particles move in cohesive groups while balancing exploration and exploitation to better navigate tough optimization landscapes. The use of a variety of techniques in the context of cryptocurrency was conducted by Ma & Mahmoudinia [5] as they compared Prophet with three instances: SARIMAX, Time2Vec, and Temporal Fusion Transformer for predicting Bitcoin transaction fees; this study demonstrates Prophet to be a viable tool to utilize under crypto circumstances since, with scarce data available during cross-validation stage, Prophet still had great performance. Complementing this, Choi et al. [6] introduced a reinforcement learning approach (based on Proximal Policy Optimization, PPO hereafter) for dynamic tuning of hyperparameters in cryptocurrency scam detection pipelines to reveal that automation-based optimization strategies can maintain model characteristics over changing data patterns.

2 METHODOLOGY

2.1 PROPHET'S ADDITIVE MODEL

Prophet is a robust time series forecasting tool developed by Facebook's Core Data Science team. It is designed to handle time series data that exhibit strong seasonal effects, trends, and holidays. Prophet is built on an additive model that allows for flexibility in modeling complex seasonality and trend changes, making it suitable for a variety of forecasting tasks in different domains, such as finance, retail, and healthcare [4].

Prophet is an additive model, meaning that it decomposes the time series into several components: trend, seasonal, and holiday effects. It assumes that the observed time series y_t at time t can be modelled as a sum of three components:

$$y_t = g(t) + s(t) + h(t) + \epsilon_t \quad (1)$$

where:

$g(t)$ represents the trend component, which models the underlying trend of the data.

$s(t)$ represents the seasonal component, which captures the periodic fluctuations due to seasonality.

$h(t)$ represents the holiday component, which captures the effects of special events such as holidays or promotions.

ϵ_t is the noise or error term at time t, assumed to be normally distributed with zero mean and constant variance.

1. Trend Component $g(t)$

The trend component of Prophet is modelled using piecewise linear or logistic growth curves. The choice of growth type depends on whether the time series exhibits a linear trend or has a saturating growth pattern [4].

a. Linear Trend:

For linear growth, the model assumes that the trend at time t, $g(t)$, follows a linear function of time:

$$g(t) = \beta_0 + \beta_1\, tg(t) \quad (2)$$

where:

β_0 is the intercept (the starting value of the time series).

β_1 is the slope of the trend (the rate of change in the time series over time).

t is the time.

b. Logistic Growth:

For logistic growth, the model assumes that the trend follows a logistic curve, which is particularly useful for modelling processes that exhibit saturation:

$$g(t) = \frac{C}{1 + exp\big(-k(t - t_0)\big)} \quad (3)$$

where:

C is the carrying capacity (the maximum value that the series can reach).

k is the growth rate.

t_0 is the inflection point (the time at which the growth rate is maximized)?

In practice, Prophet automatically selects between these two growth types based on the input data [4].

2. Seasonal Component $s(t)$

The seasonal component models the periodic fluctuations in the time series. Prophet supports both yearly and weekly seasonality's, and it allows the user to specify custom seasonality's if needed. Prophet uses Fourier series to capture the periodic patterns in the data.

a. Yearly Seasonality:

The yearly seasonal component is modeled using a Fourier series expansion:

$$s(t) = \sum_{i=1}^{k} \left[a_i cos\left(\frac{2\pi it}{365}\right) + b_i sin\left(\frac{2\pi it}{365}\right)\right] \quad (4)$$

where:

a_i and b_i are the Fourier coefficients that determine the amplitude of the seasonal fluctuations.

k is the number of Fourier terms used (determines the complexity of the seasonal pattern).

the factor $\frac{2\pi it}{365}$ normalizes the time to account for yearly periodicity, assuming there are 365 days in a year [4].

b. Weekly Seasonality:

Similarly, weekly seasonality can be modelled using a Fourier series expansion with a period of 7 days [4]:

$$s(t) = \sum_{i=1}^{k} \left[a_i cos\left(\frac{2\pi it}{7}\right) + b_i sin\left(\frac{2\pi it}{7}\right)\right]$$

3. Holiday component $h(t)$

The holiday component models the effect of special events, such as holidays or specific promotions, on the time series. The holiday effect is modelled as a set of binary indicator variables that indicate whether a holiday occurs at time t:

$$h(t) = \sum_{i=1}^{m} \delta_i . 1(t \in Holiday\ i) \qquad (5)$$

where:

δ_i is the impact of holiday i on the time series.

$1(t \in Holiday\ i)$ is an indicator function that takes the value 1 if t corresponds to holiday i, and 0 otherwise?

Users can specify a list of holidays and their corresponding effects, and Prophet will automatically incorporate these effects into the model [4].

4. error term ϵ_t

The error term ϵ_t represents the unexplained variance in the data after accounting for the trend, seasonality, and holidays. It is assumed to be independently and identically distributed (i.i.d.) with zero mean and constant variance (Yaequb, S, 2022):

$$\epsilon_t \sim N(0, \sigma^2)$$

$$\epsilon_t \sim N(0, \sigma^2)$$

2.2 MODEL ESTIMATION AND FITTING

Prophet uses a Bayesian approach to estimate the parameters of the model, allowing for uncertainty quantification. The parameters include the trend growth rates, seasonalities, and holiday effects. The model's parameters are estimated by maximizing the posterior distribution of the model, using Markov Chain Monte Carlo (MCMC) or optimization algorithms.

The process involves:

1. Defining the likelihood function for the time series based on the model structure.
2. Estimating the parameters using MCMC methods or optimization techniques.
3. Generating forecasts with uncertainty intervals.

$$\hat{y}_t = g(t) + s(t) + h(t) \qquad (6)$$

where the future values of $g(t)$, $s(t)$ and $h(t)$ are predicted based on the historical data and the estimated model parameters [4].

2.3 UNCERTAINTY AND FORECAST INTERVALS

One of the key features of Prophet is the ability to generate uncertainty intervals for the forecasts. The uncertainty is derived from the model parameters, which are estimated with a Bayesian approach. Prophet typically uses a 95% confidence interval for its forecasts, providing a range of possible future values [7].

The uncertainty intervals are generated by sampling from the posterior distribution of the model parameters, and the forecast at time tt is represented as:

$$\hat{y}_t \pm 1.96 \, . \sigma_{\hat{y}_t} \tag{7}$$

where $\sigma_{\hat{y}_t}$ is the standard deviation of the forecast at time t.

2.4 VISUALIZING PROPHET FORECASTS

Prophet provides a built-in visualization tool that allows users to visualize the individual components of the forecast (trend, seasonality, and holiday effects), along with the overall forecast and uncertainty intervals. These visualizations help in understanding the contributions of different components to the final forecast [4].

2.5 PARTICLE SWARM OPTIMIZATION (PSO)

Particle Swarm Optimization (PSO) is a population-based, nature-inspired optimization technique introduced by Kennedy and Eberhart (1995) [3], drawing inspiration from the collective foraging strategies of bird flocks and fish schools. In PSO, a swarm of N particles, each representing a potential solution in a d-dimensional search space, navigates the landscape by adjusting its position and velocity according to both individual experience and social interaction [3]. At iteration t, the velocity vi(t) and position $y_i(t)$ of the i-th particle are updated as follows:

$$v_{i(t+1)} = \omega v_{i(t)} + c_1 r_1 \, [p_i - y_{i(t)}] + c_2 r_2 \, [g - y_{i(t)}] \tag{8}$$

$$y_{i(t+1)} = y_{i(t)} + v_{i(t+1)} \tag{9}$$

where ω is the inertia weight to control explore and exploit abilities, c1, c2 are two sets of acceleration coefficients (cognitive coefficient personal best,social coefficient global best), r1,r2∼U(0,1) are random value from uniform distribution (j.r = N.rand()),pi is the best position visited by particle i and g represent the best found by whole swarm_shi1998. The way PSO converges depends greatly on the choice of it control parameters [8][9]. Larger values of the inertia weight (ω > 0.8) enhance global exploration, so as to avoid premature convergence; smaller values (ω < 0.5), in turn, promote local exploitation by helping close-in on near-optimal solutions [10]. Likewise, the acceleration coefficients c1 and c2 determine the rate of learning and social sharing respectively; balanced values (usually c1=c2 = 2.0) are often used to ensure convergence with stability [9][11]. The literature also presents constriction factors and adaptive strategy for parameter [10] [11], in order to increase the convergence reliability and avoid stagnation. Because of the use of a metaheuristic without derivatives, and simple computation (mostly algebraic fuzzes), as well as suitable performance on multidimensional continuous optimisation problems, PSO has been widely applied in machine learning and forecasting areas. The parameter optimization method proposed in this article uses PSO to achieve hyperparameter optimization for the Prophet forecasting model, and the advantage of using PSO is that feature have global capability, fast exploration of solution search space and improved prediction performance without gradient information citation [4][12][13].

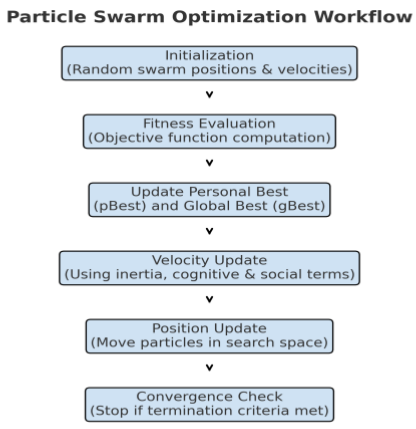

FIGURE 1. Workflow of the Particle Swarm Optimization process.

2.6 EVALUATING PRECISION OF FORECASTING MODELS

To test the accuracy and the performance of the proposed model used, some statistical tests and measurements, including, mean square error, root of mean square error, Akaike information criteria, and Bayesian information criteria.

2.6.1 MEAN SQUARE ERROR (MSE)

Mean Squared Error (MSE) is a widely used metric for assessing the accuracy of a predictive model. It measures the average of the squares of the errors that is, the average squared difference between the actual values and the values predicted by the model [14].

$$\text{MSE} = \frac{1}{n}\sum_{t=1}^{n} (y_t - \hat{y}_t)^2 \qquad (10)$$

where:

n is the number of observations.

yt is the actual value.

$\hat{y}_t$ is the predicted value.

A lower MSE indicates a better fit of the model to the data, as it suggests that the predictions are closer to the actual values. However, MSE can be sensitive to outliers because it squares the errors, which can disproportionately affect the overall score.

2.6.2 SQUARE ROOT OF MEAN SQUARE ERROR (RMSE)

RMSE stands for Root Mean Square Error, and it tends to be the default statistic for measuring how well a model predicts. The RMSE is basically a nice measure of prediction error, the square root of the average squared errors of predicted values from real output. According to Ahmed [15], it is more interpretable than Mean Squared Error because it's reported in the same units as the real measurement.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n} (y_t - \hat{y}_t)^2} \qquad (11)$$

2.6.3 AKAIKE INFORMATION CRITERIA (AIC)

Akaike Information Criterion (AIC) is a relative goodness-of-fit statistic that indicates how well the model represents the data, compensating for the number of parameters in the model. The AIC calculated is presented by Hirotugu Akaike to assess which fit is the best in comparison to ease of simplicity in selection.

$$\text{AIC} = n \cdot \text{Ln}(\text{MSE}) + 2k \qquad (12)$$

Where:

n: is the number of observations.

MSE: is the mean square error.

k: is the number of explanatory variables in the model.

2.6.4 BAYESIAN INFORMATION CRITERIA (BIC)

Bayesian Information Criterion (BIC), or Schwarz Information Criterion (SIC), is a criterion used for model selection from a finite array of models. It's a penalty for selecting an array of goodness-of-fit measures, which include a penalty for the number of generated parameters [16].

$$\text{BIC} = n \cdot \text{Ln}(\text{MSE}) + k \cdot \text{Ln}(n) \qquad (13)$$

Where:

n: is the number of observations.

MSE: is the mean square error.

k: is the number of explanatory variables in the model.

In conclusion the lower MSE, RMSE, AIC and BIC indicates a better fit of the model to the data, as it suggests that the predictions are closer to the actual values.

3 APPLICATIONS

3.1 DATA DESCRIPTION

The dataset contains weekly open prices of the Ethereum (ETH) cryptocurrency, expressed in U.S. dollars (USD), spanning a continuous period of 96 weeks and concluding with the week of 2025-08-03 to 2025-08-10. Each record consists of two variables: (1) Date, labeled sequentially from week1 to week96, representing the week index in the dataset, and (2) Ethereum weekly open price, a floating-point value denoting the market's opening price at the start of the respective week. The dataset captures medium-term price trends in Ethereum by using weekly resolution, which helps smooth out short-term volatility and noise while preserving significant market movements. Over the recorded period, the dataset exhibits substantial variability, reflecting the inherent volatility of cryptocurrency markets, with prices ranging from approximately $ 1,808.70 USD in week 1 to $ 3,914.11 USD in week 96. Therefore, for time series comparison, prediction, and modeling, it is important to assess what possible trends may be found in cryptocurrency.

The data was gathered from Bitget's Historical Ethereum Price Data (https://www.bitget.com/price/ethereum/historical-data) so it's consistent and trustworthy with prices assessed within the actual trading market. This is a great sample to validate/testing predictive modeling based on the Prophet algorithm with Particle Swarm Optimization (PSO) for improved prediction accuracy in volatile asset markets.

3.2 RESULTS AND DISCUSSIONS

This section presents results and projections from forecasting Ethereum's weekly opening price using the Prophet method with a PSO tuning challenge. Results are discussed based on projections from various fitting scenarios training, test holdout, and one-time full fit, as well as the final results from convergence of the tuned output, in-sample fit, multi-period prediction accuracy, and holdout evaluation. MSE, RMSE, AIC, BIC, along with graphical representations (Figures 2-5) not only reflect goodness of fit but also the robustness of the method relative to the volatile nature of cryptocurrency trading.

Table 1. Crepresents the accuracy measurements of Prophet (PSO)

Fitting Type	MSE	RMSE	AIC	BIC
Prophet (PSO) – Train fit	4,948.79	70.3476	922.607	1,138.14
Prophet (PSO) – Test holdout	348,013.54	589.9267	1226.97	1229.53
Prophet (PSO) – Full fit	4,240.64	65.1202	914.8371	1,115.45

According to Table (1), the accuracy of the Prophet algorithm with PSO for the various fit options are as follows. For the train fit, which means the model is fit with the training set exclusively, the MSE is 4,948.79 (RMSE=70.35, AIC=922.61, BIC=1,138.14) suggesting decent in-sample performance. For the test holdout, the model's performance on unseen data, the errors increase considerably, with an MSE of 348,013.54 and RMSE of 589.93, while AIC and BIC values rise to 1,226.97 and 1,229.53, respectively, reflecting the greater challenge of predicting out-of-sample data. In the full fit, where the model is trained on the entire dataset, the MSE and RMSE drop to 4,240.64 and 65.12, respectively, and the AIC and BIC improve to 914.84 and 1,115.45, suggesting that the model fits well when given all available historical data, but the increase in test errors highlights the importance of careful validation to avoid overfitting.

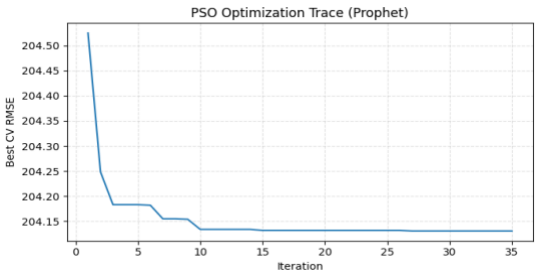

FIGURE 2. Convergence Curve of PSO-Optimized Prophet Model for Ethereum Price Forecasting

The above figure illustrates the PSO Optimization Trace for the Prophet model, showing how the best cross-validation Root Mean Squared Error (CV RMSE) evolves over 35 iterations of the Particle Swarm Optimization process. Initially, at iteration 0, the best CV RMSE is approximately 204.52, indicating the baseline performance before optimization. In the first few iterations (up to around iteration 5), there is a sharp decline in CV RMSE, reflecting the rapid improvements as the swarm explores the parameter space and quickly converges towards better Prophet hyperparameters. After iteration 10, the curve flattens out, with only minimal improvements observed, suggesting that the PSO algorithm has effectively

converged and found a near-optimal solution for minimizing forecast error. This convergence pattern demonstrates PSO's efficiency in fine-tuning Prophet parameters, achieving the final best CV RMSE of about 204.15, which represents a notable improvement compared to the initial configuration.

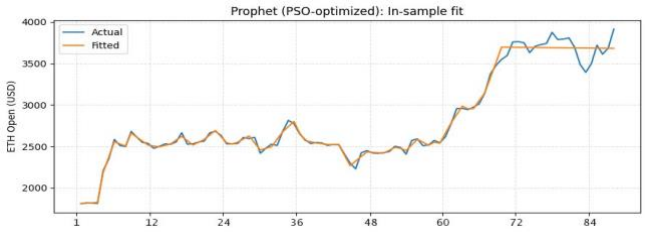

FIGURE 3. In-Sample Fit of the Prophet Model Optimized via Particle Swarm Optimization for Ethereum Weekly Open Prices

The above figure presents the in-sample fit of the Prophet model, optimized using Particle Swarm Optimization (PSO), for forecasting Ethereum's weekly open prices. The blue line represents the actual observed values of ETH open prices, while the orange line depicts the fitted values predicted by the optimized model during the training period. The proximity of the two lines over the 96-week training period suggests a strong model for predicting both incremental shifts and overall increases on a moving average basis. Moreover, the model also captures the ETH price spike at week 65 and remains accurate during dips, implying that the PSO-tweaked hyperparameters effectively helped Prophet compensate for both trend adjustments and seasonal variations in a time series fraught with financial volatility.

FIGURE 4. Prophet Model Optimized via Particle Swarm Optimization: 12-Week Ethereum Price Forecast with 95% Confidence Interval

Figure 4 illustrates the 12-week forecast of Ethereum's weekly open prices using the Prophet model optimized through Particle Swarm Optimization (PSO). The blue line to the left fits the price series observed, whereas the orange line refers to the model's expectation of mean value during each subsequent time period. The blue shaded region is the 95% prediction interval. Thus, the model's expectation is that the price movement for ETH (in USD) will remain constant throughout the prediction timeframe, and although the prediction intervals expand at the end which might seem disconcerting, this is natural over a prediction interval as uncertainty inevitably increases especially within such a volatile market. Thus, being able to predict not only ETH price movements but also price intervals suggests that this model is useful for not only point estimate forecasting but probability-based forecasts as well.

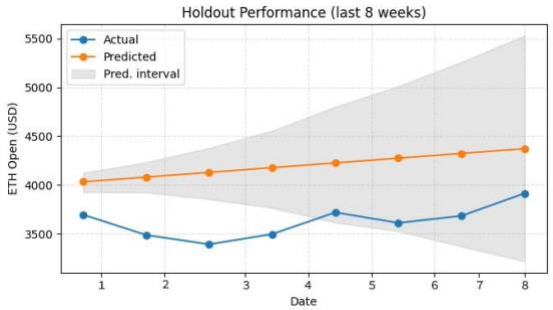

FIGURE 5. Holdout Set Performance of Prophet (PSO-Optimized) Model over the Last 8 Weeks

This figure depicts the outcome of the Prophet model adjusted by PSO with the testing set composed of Ethereum weekly open prices over an 8-week stretch. In the above image, the blue line represents the actual weekly opens, and the orange line represents the predicted weekly opens, while the gray shaded area represents the predicted confidence interval. In this instance, the prediction trend for the value remains constant and then slowly increases over time, while the market price increases and decreases more erratically week by week with spikes above and dips below the predicted value range. This gap between actual and predicted values, especially in the later weeks, reflects the model's limitations in fully capturing short-term fluctuations in such a volatile market. The widening confidence interval over time also emphasizes the increasing uncertainty of forecasts further into the horizon.

Table 1. Twelve-Week Ethereum Price Forecast with 95% Confidence Intervals

Forecast the next 12 weeks			
Weeks	Forecasted	95% Interval	
		hi	low
1	3,681.94	3,585.46	3,781.56
2	3,681.25	3,533.98	3,832.70
3	3,680.57	3,421.21	3,963.31
4	3,679.88	3,298.64	4,128.53
5	3,679.20	3,144.48	4,296.94
6	3,678.51	2,997.04	4,493.34
7	3,677.83	2,794.49	4,740.01
8	3,677.14	2,573.68	5,001.02
9	3,676.46	2,374.47	5,282.26
10	3,675.77	2,159.46	5,487.85
11	3,675.09	1,867.89	5,741.11
12	3,674.40	1,639.50	6,058.11

The above table 12 weeks forecast for Ethereum (ETH) weekly opening prices are beyond the observed value and the prediction with a 95% confidence interval shows that prices remain relatively stable, going from 3,681.94 to 3,674.40, indicating a consistent if low growth atmosphere as it decreases over time. The confidence interval, however, increases over time starting from 3,585.46–3,781.56 to 1,639.50–6,058.11. This likely reflects the algorithm's uncertainty in projecting values over time for something that is inherently unpredictable with changing prices and probabilities.

CONCLUSION

Final Summary The final study determined that the use of Particle Swarm Optimization to the Prophet forecasting model objectively improved forecasting accuracy for weekly opening prices for Ethereum and even more so over a 96-week period. PSO hyper-parameterized Prophet to reduce in-sample error with fast convergence compared to the baseline configuration where the first ten iterations produced the most drastic adjustment. The hypothesized configuration maintained excellent accuracy for trend capture and seasonality expected for the testing set over extended periods, but less so for shorter. This compounded how deviations and confidence intervals were wider in the holdout set. Ultimately PSO is found advantageous in time-series forecasting for financial markets of this nature. Implications Yet it implies that cryptocurrency is still a high-risk investment with a lot of uncertainty, leading to overfitting problems.

LIMITATIONS

Yet the study itself limited it to 96 weeks of univariate predicted thresholds for Ethereum pricing which does not forecast every market regime or windows of profitability. In addition, exogenous variables like trading prices or macroeconomic dynamics were not considered. Finally, although PSO was the most time-efficient study long-term/hybrid application, it can be computationally intensive for larger data/population/time-periods.

FUTURE RESEARCH

Future directions should include adding exogenous predictors or applying it to higher frequencies when applicable or testing the hybrid integration with deep learning to see response and accuracy in low-volatility markets.

REFERENCES

[1] Y. Chen and C. Bellavitis, "Blockchain disruption and decentralized finance: The rise of decentralized business models," J. Bus. Venturing Insights, vol. 13, p. e00151, 2020.

[2] D. Shen, A. Urquhart, and P. Wang, "Forecasting the volatility of Bitcoin: The importance of jumps and structural breaks," Eur. Financ. Manag., vol. 25, no. 2, pp. 289–315, 2019.

[3] J. Kennedy and R. Eberhart, "Particle swarm optimization," in Proc. ICNN'95 - Int. Conf. Neural Netw., vol. 4, pp. 1942–1948, 1995.

[4] S. J. Taylor and B. Letham, "Forecasting at scale," Amer. Statist., vol. 72, no. 1, pp. 37–45, 2018.

[5] J. Ma and E. Mahmoudinia, "Comprehensive modeling approaches for forecasting Bitcoin transaction fees: A comparative study," arXiv preprint arXiv:2502.01029, 2025. [Online]. Available: https://arxiv.org/abs/2502.01029.

[6] S.-H. Choi, S.-M. Choi, and S.-J. Buu, "Proximal policy-guided hyperparameter optimization for mitigating model decay in cryptocurrency scam detection," Electronics, vol. 14, no. 6, p. 1192, 2025. [Online]. Available: https://www.mdpi.com/2079-9292/14/6/1192.

[7] A. A. Aziz, B. M. Shafeeq, R. A. Ahmed, and H. A. Taher, "Employing recurrent neural networks to forecast the dollar exchange rate in the parallel market of Iraq," Tikrit J. Admin. Econ. Sci., vol. 19, no. 62, p. 2, 2023.

[8] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer," in Proc. IEEE Int. Conf. Evol. Comput., pp. 69–73, 1998.

[9] R. C. Eberhart and Y. Shi, "Particle swarm optimization: Developments, applications and resources," in Proc. Congr. Evol. Comput., vol. 1, pp. 81–86, 2001.

[10] M. Clerc and J. Kennedy, "The particle swarm—explosion, stability, and convergence in a multidimensional complex space," IEEE Trans. Evol. Comput., vol. 6, no. 1, pp. 58–73, 2002.

[11] A. Ratnaweera, S. K. Halgamuge, and H. C. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," IEEE Trans. Evol. Comput., vol. 8, no. 3, pp. 240–255, 2004.

[12] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization: An overview," Swarm Intell., vol. 1, pp. 33–57, 2007.

[13] M. Zambrano-Bigiarini, M. Clerc, and R. Rojas, "Standard particle swarm optimisation 2011 at CEC-2013: A baseline for future PSO improvements," in Proc. IEEE CEC, pp. 2337–2344, 2013.

[14] A. Asraa, W. Rodeen, and H. Tahir, "Forecasting the impact of waste on environmental pollution," Int. J. Sustain. Develop. Sci., vol. 1, no. 1, pp. 1–12, 2018.

[15] B. K. Ahmed, S. A. Rahim, B. B. Maaroof, and H. A. Taher, "Comparison between ARIMA and Fourier ARIMA model to forecast the demand of electricity in Sulaimani Governorate," Qalaai Zanist J., vol. 5, no. 3, pp. 908–940, 2020.

[16] S. T. Ahmed and H. A. Tahir, "Comparison between GM (1, 1) and FOGM (1, 1) models for forecasting the rate of precipitation in Sulaimani," J. Kirkuk Univ. Admin. Econ. Sci., vol. 11, no. 1, pp. 301–314, 2021.