MODE SYNTH SUBNET

HOW TO BUILD A SIMPLE MODEL AND START EARNING TAO

Welcome to the world of Synth Subnet – a groundbreaking project by Mode Network that leverages Bittensor's decentralized intelligence to create the world's primary source of synthetic price data for the cryptocurrency market. This technology has the potential to revolutionize how we approach decentralized finance (DeFi), from automating complex trading strategies with AI agents to forecasting future market trends for more informed investment decisions.

In this post, we will break things down into three key areas:

1. Explaining the Scoring System: What's the methodology behind how submissions are evaluated?
2. Tutorial for Miners: How to set up and submit Bitcoin price predictions using a simple Geometric Brownian Motion (GBM) model.
3. Sandboxed Competition Results: A look at how miners using similar models, like GBM, perform against each other.

Let's dive in!

MINERS' AND VALIDATORS' ROLE IN THE SYNTH SUBNET

Synth Subnet operates through a miner-validator system. Miners contribute by submitting their forecasts for future Bitcoin prices, while validators evaluate these forecasts ex-post by comparing them to the actual prices to ensure that rewards are fairly distributed.

Miners' Task: Submitting Price Predictions. Miners are asked to submit simulated Bitcoin price paths for the next 24 hours. These submissions follow a set of parameters:

- start time (t_0) : within 1 minute from the request
- time horizon (T): 24 hours
- time increment (Δt): 5 minutes
- total number of increments (N): 288 (i.e., 24 hours ÷ 5 minutes)
- number of simulated paths (N_{sim}): 100

Thus, for each time step $t_i = t_0 + i \times \Delta t$, miners will submit 100 simulated prices. A common model for generating these price paths is the Geometric Brownian Motion (GBM) model, which we will explore in more detail below.

Validators' Scoring Methodology. The Synth Subnet employs a systematic scoring methodology to evaluate miners' submissions, ensuring fairness and promoting honest, independent forecasts. Each step builds on the outputs of the previous, creating a clear, consequential flow of calculations:

1. Forecast Evaluation: After the 24-hour forecast period, the Continuous Ranked Probability Score (CRPS) is calculated for every miner at each time point t_i. For this calculation:

- The 100 simulated price paths submitted by the miner are used to compute the distribution of predicted price changes.
- These predicted price changes are compared to the single observed price change over the same interval. The CRPS score reflects two key aspects:
    - Calibration: How closely the forecasted price changes match the observed price change.
    - Sharpness: How tight or precise the miner's forecast distribution is.

This process is repeated across different intervals, including 5 minutes, 30 minutes, 3 hours, and 24 hours. The total CRPS score for a miner is the sum of the individual CRPS scores for all intervals, providing a comprehensive view of the miner's performance across short, medium, and long horizons.

2. Aggregating Scores: The CRPS scores for each interval are summed to create a total unnormalized score for each miner at the end of the forecast period. This aggregation ensures that miners are evaluated across their overall forecasting performance.

3. Normalization of CRPS scores: To ensure scores are comparable and proportional across miners, total unnormalized scores are normalized using a softmax function. This normalization brings two benefits:

    - Fair Comparisons: The softmax function scales each miner's score relative to all miners in the same prompt, ensuring proportional rewards based on comparative performance. The scaling factor (set to $\beta = 0.001$) reduces the influence of extreme raw scores, preventing any single miner from disproportionately dominating the reward distribution.
    - Encouragement of Independence: Miners who submit identical or very similar forecasts share the reward, discouraging copying and incentivizing independent strategies.

4. Leaderboard Scoring: After normalization, the scores are incorporated into the leaderboard scores, which include a memory of past performance. Specifically, an exponential moving average (EMA) of the normalized scores is used to balance short-term and long-term performance.

    - At the launch of the contest, the EMA has a half-life of 1 day, meaning recent scores contribute more heavily to the leaderboard ranking.
    - As the contest matures, the half-life will increase, rewarding miners who demonstrate sustained excellence over time while still allowing new participants to compete effectively.

5. Emission Allocations: To amplify the impact of high-performing miners, the leaderboard scores are raised to the power of 2. This exponentiation enhances the differences between scores, ensuring that miners with more accurate and well-optimized models receive a larger share of rewards. Over time, the power to which the leaderboard scores are raised will

increase further, enabling finer distinctions between increasingly optimized models as the contest progresses.

A more detailed explanation of the scoring mechanism can be found in the subnet's Github page.

BUILDING A SIMPLE MODEL: THE GEOMETRIC BROWNIAN MOTION (GBM)

Now that we understand how submissions are evaluated, let's look at how to build a basic forecast model using the GBM model.

The GBM Model. GBM models assume that the logarithms of price returns (log-returns) follow a stochastic process. The price change at any time point t is described by this equation:

$$dS_t = \mu S_t dt + \sigma S_t dW_t$$

where:

- μ: The drift (deterministic trend)
- σ: The volatility (random fluctuations)
- dW_t: A Wiener process representing random market movements

This model helps simulate possible price paths for Bitcoin, starting from a given initial price.

Simplified Assumptions for this Tutorial. To keep things simple for this post:

- We'll assume no drift (μ=0), meaning no specific upward or downward trend in Bitcoin's price.
- Volatility (σ) will be constant across the entire time horizon, though miners could update this based on market data.

Given these assumptions, we can generate simulated Bitcoin price paths over 24 hours using Python. The code in Appendix 1 shows the full implementation, which uses hourly volatility and divides the day into 288 increments. Figure 1 displays an example of 100 paths simulated using the GBM model.

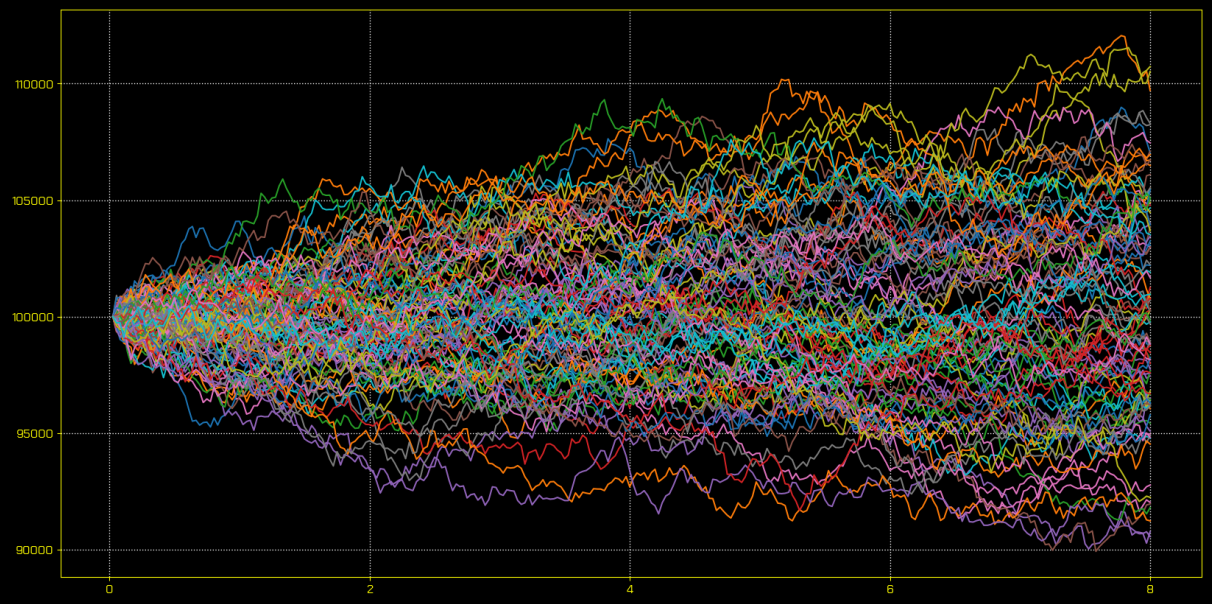

Figure 1. Example of paths generated using the GBM model. S_0 = 100,000, $\sigma = 0.01$.

Submitting the Paths as a Miner. Detailed instructions for setting up and creating a miner for the Synth Subnet competition can be found in this link. Miners are free to use their own forecast-generating functions, such as the GBM model presented in Appendix A, by modifying the corresponding code in the Synth Subnet repository.

BACKTESTING: HOW GBM PERFORMS WITH REAL DATA

To test the Synth Subnet and its scoring system, we performed two key analyses: a Monte Carlo simulation and a backtest using real Bitcoin data. Both are designed to demonstrate the performance of GBM models under different conditions.

Monte Carlo Simulation. To explore the Synth Subnet's potential behavior, we conducted a Monte Carlo simulation study. In this experiment, we replaced real Bitcoin prices with prices simulated using a Geometric Brownian Motion (GBM) model. This allowed us to evaluate how effectively miners using their own GBM models performed under a fully specified data-generating process.

Simulation Setup. In this experiment, we generated 100 simulated paths for Bitcoin prices over a period of 60 days using the GBM model. Each simulated path was based on the same hourly volatility parameter, σ=0.01, which was also the parameter used by our Oracle miner. This "Oracle" miner served as a benchmark, as it utilized the exact volatility to produce the most accurate forecasts.

We included miners with varying levels of volatility estimates to assess their performance relative to the Oracle:

- Baseline Miner: σ=0.01
- Miners with Adjusted Volatility:
    - 0.9σ: σ=0.009 (-10%)

- 0.99σ: σ=0.0099 (-1%)
- 0.999σ: σ=0.00999 (-0.1%)
- 1.001σ: σ=0.01001 (+0.1%)
- 1.01σ: σ=0.0101 (+1%)
- 1.1σ: σ=0.011 (+10%)

This configuration allowed us to assess how different volatility assumptions affected the miners' forecast accuracy and overall performance in the backtest.

Results and Insights. Figure 2 displays the distribution of CRPS score differences between each non-Oracle miner and the Oracle miner. Since lower CRPS scores indicate better forecasts, the portion of the distribution below 0 shows how often each miner outperformed the baseline, while the portion above 0 shows how often the baseline performed better. The results indicated that miners using volatility estimates close to the true value (up to ±1% of the original volatility) achieved scores similar to the Oracle, with differences largely attributable to random simulation noise. In contrast, miners who adjusted their volatility estimates by ±10% experienced significantly worse performance.

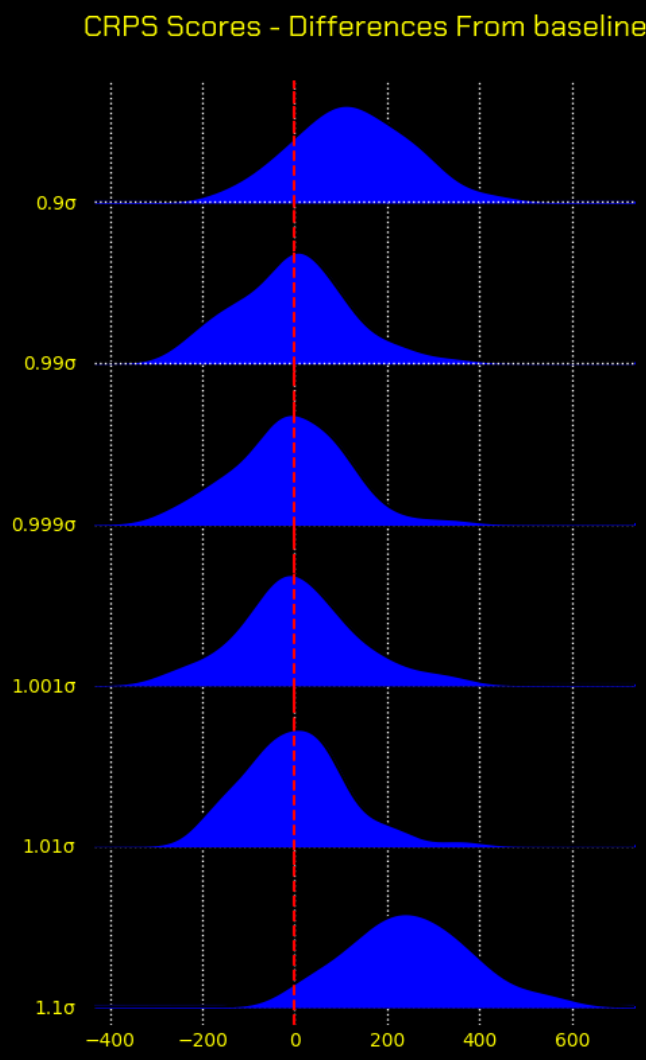

Figure 2. Distribution of differences between CRPS scores of non-Oracle miners and the Oracle miner.

The averaged leaderboard scores in Figure 3 reveal a similar trend. Miners with volatility estimates closely aligned with the Oracle's tended to stay near the top of the leaderboard throughout the simulation period. However, those miners who inflated or deflated their volatility (like the +10% miner) faced a noticeable decline in scores, illustrating how crucial accurate volatility estimation is for reliable forecasting.

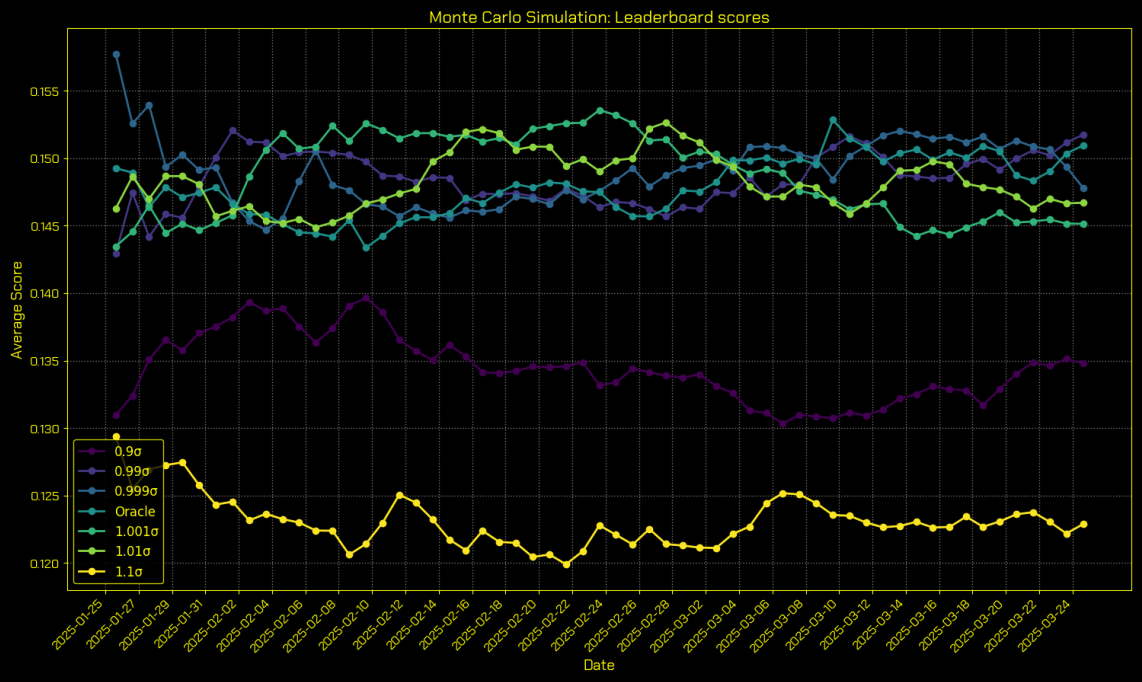

Figure 3. Average leaderboard scores of the 7 miners in the Monte Carlo simulation study over 60 simulated days.

Figure 4 further emphasizes this point by showing confidence intervals around the average leaderboard scores. While the miners near the Oracle had overlapping intervals, indicating similar performance, the miners with more extreme volatility adjustments had significantly lower average scores with no overlap. This stark contrast highlights how inaccuracies in volatility can lead to substantial penalties in performance.

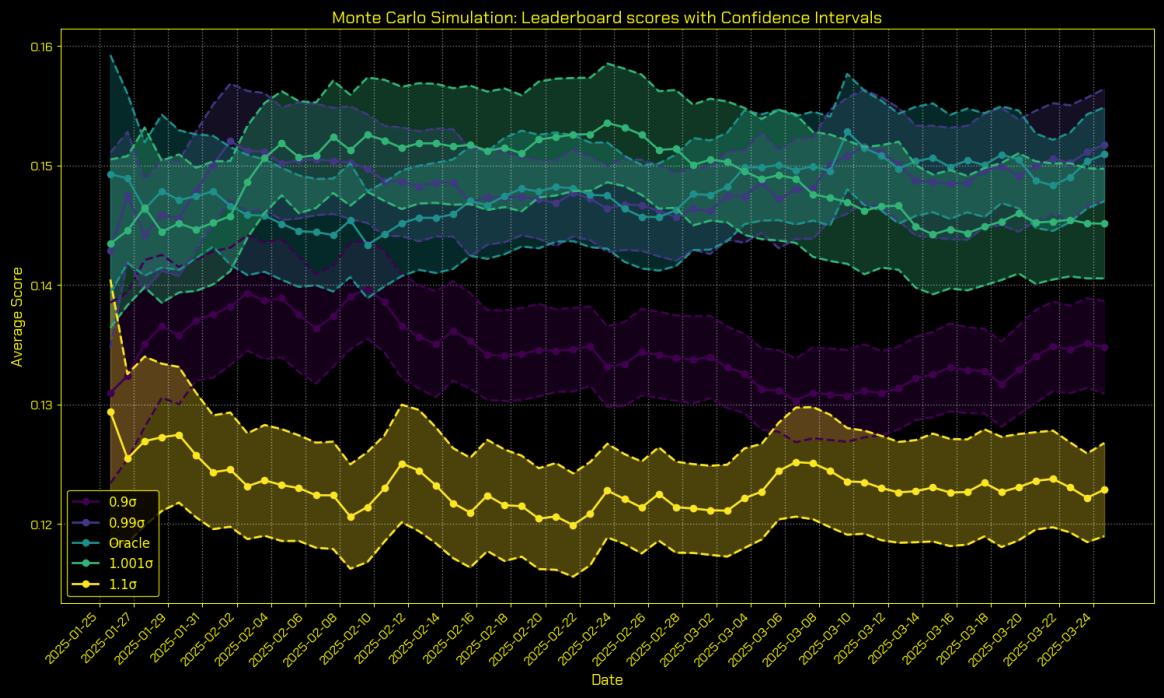

Figure 4. Confidence intervals for average leaderboard scores of the Oracle miner and miners using 0.9σ, 0.99σ, 1.001σ, and 1.001σ.

Conclusion of the Monte Carlo Study. The findings from the Monte Carlo simulation underline the importance of precise volatility estimations in the Synth Subnet contest. Miners who can accurately forecast volatility are better positioned to produce competitive submissions and secure higher rewards. These results serve as a powerful reminder of the need for miners to develop robust forecasting models that adapt to market dynamics effectively.

Backtesting: Performance of GBM with Real Bitcoin Data. To evaluate the effectiveness of the Synth Subnet's scoring mechanism in real-world scenarios, we conducted a backtest using actual Bitcoin price data from January 1 to April 15, 2024. This period spanned approximately 105 days and encompassed different volatility regimes, allowing us to assess how the scoring mechanism adapted to the market's behavior. Figure 5 presents the log-returns of Bitcoin prices and the corresponding daily rolling volatility throughout the tested period. During this time, we identified two distinct volatility regimes: a low-volatility phase from February 1 to February 26 and a high-volatility phase from February 27 to March 23. This variability in market conditions tested each miner's forecasting capabilities.

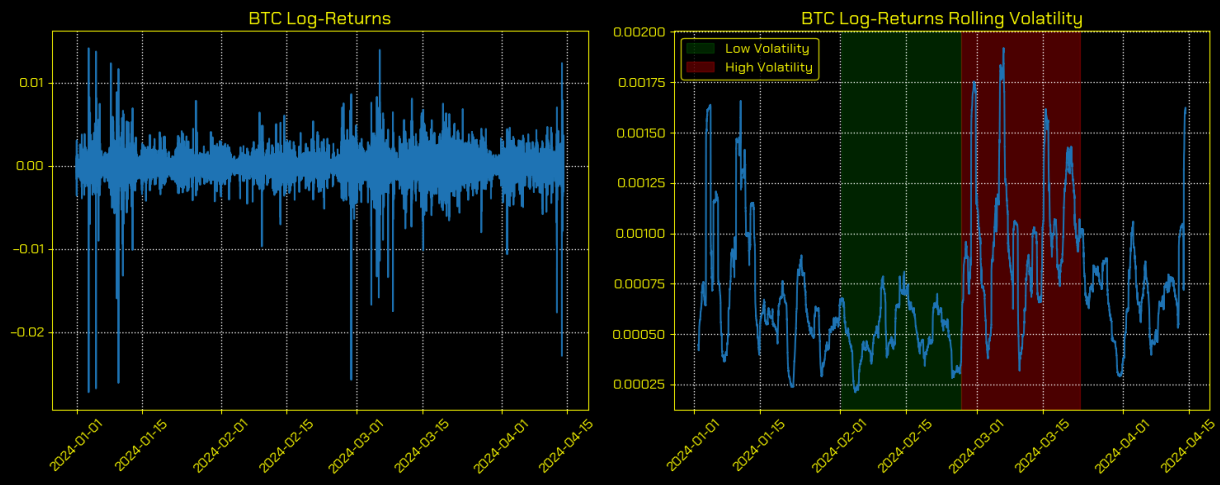

Figure 5. Log-returns of Bitcoin prices (1-minute frequency) and corresponding daily rolling volatility from January 1 to April 15, 2024.

Backtest Setup. As done in the previous analysis, we compared 7 different miners, each using a fixed volatility parameter, σ. Because in this case the true data generating mechanism was not known, we estimated the log-returns' volatility across the whole period (a sort of 'average' volatility parameter) and used that as a reference. This led to a baseline miner which used a GBM model with $\sigma \approx 0.005$ for its forecasts. The other 6 miners were set up in a similar fashion to the Monte Carlo study, with volatility adjustments equal to ±0.1%, ±1%, and ±10%.

Results and Insights. In Figure 6, we examine the distribution of CRPS score differences between each miner and the baseline. Miners with predicted volatility that closely aligned with actual market conditions generally performed better. For instance, during the high-volatility period, the -10% miner struggled significantly, as its conservative estimates led to consistently lower scores. This is reflected in the long right tail of its score distribution compared to the baseline. However, because periods of stability were more frequent than those of high volatility, this miner still achieved better scores than the baseline most of the time.

Conversely, miners who inflated their volatility estimates, particularly the +10% miner, excelled during the high-volatility phase but lagged during stable periods. This is evident in the long left tail of its score distribution, where it outperformed the baseline miner. Given the smaller proportion of high-volatility periods, the majority of the +10% miner's score differences ended up above 0, indicating that the baseline miner outperformed it more frequently.

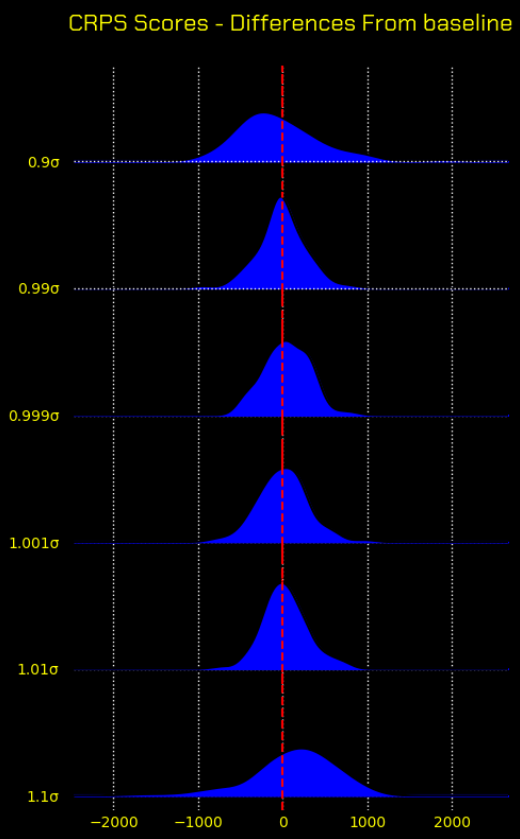

Figure 6. Distribution of differences in CRPS scores between miners and the baseline miner.

Figure 7 highlights the leaderboard scores assigned to each miner throughout the backtest. The green and red backgrounds represent the same low- and high-volatility periods shown in Figure 5. To emphasize significant score variations, the plot also displays the ±1 daily standard deviation of the scores from their mean.

In the figure, miners with volatility estimates closest to actual conditions consistently ranked higher. During the low-volatility period, the -10% miner had a distinct advantage, topping the leaderboard due to its ability to accurately predict stable price movements. However, as the market transitioned to a high-volatility environment, this miner fell behind, while those who used a larger value for σ (in particular the 1.1σ miner) performed better, demonstrating suitability to the changing market conditions. The figure also illustrates how the half-life scoring mechanism smooths scores over time, balancing short-term and long-term performance.

Conclusion of the backtest analysis. The backtest results underscore the critical importance of accurate volatility estimation in successful forecasting within the Synth Subnet. Miners who continuously adapt their models to reflect real-time market conditions are more likely to achieve competitive scores and earn rewards. These findings reinforce the need for flexibility in forecasting approaches, especially in a volatile asset class like Bitcoin, where market dynamics can change rapidly.

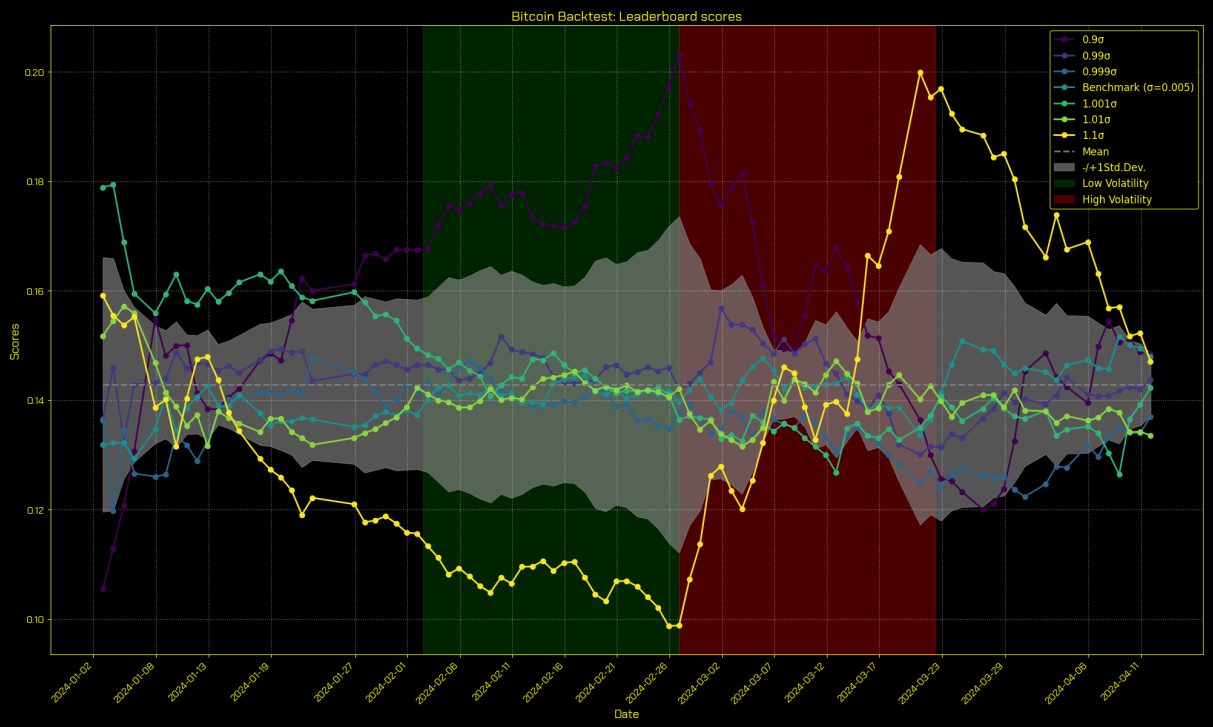

Figure 7. Leaderboard scores of miners during the backtesting period.

CONCLUSIONS

This post introduced the Synth Subnet, an innovative platform designed to generate synthetic price data for the cryptocurrency space. We explored the mechanics of the contest, focusing on how miners can participate by submitting Bitcoin price forecasts using models like the Geometric Brownian Motion (GBM). The Synth Subnet scoring mechanism, powered by the Continuous Ranked Probability Score (CRPS), ensures fair and dynamic reward allocation based on the accuracy and reliability of the forecasts.

We also demonstrated the performance of GBM models in two settings: a controlled Monte Carlo simulation and a backtest using real Bitcoin data. These analyses illustrated how the scoring system rewards miners who adapt their models to the true market conditions.

Here are the key lessons learned:

- Volatility Estimation is Crucial: Accurately estimating volatility is essential for achieving high scores. Even small deviations from the true volatility can significantly impact a miner's performance.

- Dynamic Scoring Rewards Adaptability: The Synth Subnet's scoring mechanism is sensitive to shifts in market conditions. Miners who dynamically update their models in response to real-time data perform better than those relying on static parameters.

- Avoid Copying the Winning Miner: While it may be tempting to replicate the approach of a top-performing miner, this strategy can backfire. Market conditions are constantly evolving,

and models that excel today might deteriorate tomorrow. Instead, miners should focus on developing flexible and robust forecasting models tailored to their own data and insights.

- Embrace Model Evolution: The results show that miners who adapt their strategies to account for changes in volatility regimes are rewarded. Relying on a fixed model, such as GBM with constant volatility, may work in the short term but will likely underperform over longer periods or during significant market shifts.

- Precision and Robustness of the Scoring System: The Synth Subnet's scoring system fairly discerns small differences in model accuracy and minimizes the impact of short-term luck. This is due to the scores smoothing over long time periods using a moving average, which ensures that rewards are distributed based on consistent performance rather than chance.

By using a foundational model like GBM, miners can enter the Synth Subnet contest and begin contributing to synthetic price forecasts. However, the true challenge lies in refining and evolving these models to stay competitive in an ever-changing market. To improve performance, miners could consider using models that update their volatility in real time, like GARCH models, or take into account daily and weekly cycles in volatility, or even explore machine learning techniques that can adapt quickly to new market information.

APPENDIX: PYTHON CODE FOR GBM MODELS SIMULATION

```
import numpy as np

def simulate_gbm(current_price, time_increment, time_length, num_simulations, sigma):
        '''
        Simulates num_simulations prices paths using a GBM model.

        Args:
                current_price: The latest available Bitcoin price.
                time_increment: The time increment in seconds.
                time_length: The time horizon in seconds.
                num_simulations: The number of paths to simulate.
                sigma: The volatility parameter of the GBM model.

        Returns:
                np.array: A numpy array where each row corresponds to a simulated path.
        '''

        num_steps = int(time_length / time_increment)
        dt = time_increment / 3600
        simulated_prices = np.zeros((num_simulations, num_steps + 1))
        simulated_prices[:,0] = current_price
        dW = np.random.normal(0, 1, (num_simulations, num_steps))
        dS = np.exp(((-0.5 * sigma**2) * dt) + (sigma * np.sqrt(dt) * dW))

        for t in range(1, num_steps + 1):
                simulated_prices[:,t] = simulated_prices[:,t-1] * dS[:,t-1]

        return simulated_prices
```