Coin Metrics Subnet Design Spec

Overview

Subnet Task

The Coin Metrics subnet will initially be based on short-term price prediction. This is a high-variance problem, but the high frequency of the scoring intervals will allow for rapid feedback and many iterations over which Miner participants can demonstrate an edge in their methodology. Initially this will be a prediction of Bitcoin price, although in principle it should be trivial to extend to other assets with reasonable liquidity.

The Miners will be tasked with two outputs:

1. The price of Bitcoin 1-hour from the prediction time (Point Prediction)
2. An price range equal to the min and max value of the price during the 1-hour following prediction time

I propose emissions be equally split between these two tasks. Miner's will be compensated for each task independently, according to the performance they have demonstrated on that task in the past. In other words, the Miner's accuracy in previous Point Predictions will be used to determine their share of emissions for new Point Predictions, and likewise for Interval Predictions.

Subnet Assets

Initially we propose running this only for Bitcoin as the most well established and obviously interesting asset. However, I believe there may be interesting advantages expanding this idea to less-liquid assets for which the price is more volatile, options markets are less robust, and overall higher uncertainty gives Miners with more intelligent strategies room for a larger edge.

Subnet Frequency

Every 5 Minutes:

1. The Validators will evaluate the absolute error of Predictions made 1 hour ago
2. The Validators will calculate a rolling average of the last *N* errors evaluated for each Miner, and rank the Miners from least to greatest average error. (N=12 for full hour?)
    a. This rank determines the Miners demonstrated quality and sets their compensation for the tasks performed in steps 3 & 4.
3. The Miners submit a Point Prediction for the price at one hour in the future
4. The Miners will submit an Interval Prediction for the price over the next hour

Ranking

Ranking will be done separately for the two prediction types. Miners will be compensated separately based on their rank for each prediction type (Point and Interval). Ranking will determine emissions for subsequent tasks, with higher emissions to Miners which have demonstrated the most accuracy on previous tasks.

Point

For the Point Prediction, each Miner has submitted a value for the price one hour before the target time. The Validator will pull the Coin Metrics Real Time Reference Rate for at the time of the scoring, and each Miner's error is evaluated as the absolute value of the difference between the Miner's Prediction and the Reference Rate, as a percentage of the Reference Rate, at the target time. So the error for miner *i* would be:

$$e_i = |p_i - p_{CM}| / p_{CM}$$

The last N-errors **(N to be determined?)** for each miner will then be averaged together, this rolling average can referred to as the delta-factor (Δ):

$$\Delta = \frac{1}{N} \sum_{n=0}^{N} e_n$$

and the Miners will be sorted according to this rolling average. In the case of a tie in the value of Δ, the tied Miners will be given the same rank.

Interval

The Interval Prediction is intended to provide an avenue for Miners to accurately estimate the price range during varying periods of volatility. The optimal score will be given to Miners who predict the exact minimum and maximum price reached during the next hour. The score of the prediction will decrease if the interval predicted extends larger than the maximum price observed, lower than the minimum price observed, or does not include any observed price values during the 1-hour window.

The Interval Prediction consists of two prices, the predicted minimum price ($p_{p,min}$) and the predicted maximum price ($p_{p,max}$). The score of each Miner's interval prediction will consist of two parts: a width-factor (f_w) and an inclusion-factor (f_i). The former will penalize intervals which are excessively wide, while the latter will penalize intervals that do not fully contain the range of prices.

To determine the width-factor we use what we call the effective bottom (*b*) and effective top (*t*) of the Interval Prediction by limiting the bottom and top of the interval according to the minimum ($p_{o,min}$) and maximum ($p_{o,max}$) of the observed price (p_{CM}):

$$p_{o,min} = min(p_{CM}(t))$$
$$p_{o,max} = max(p_{CM}(t))$$

$$b = max(p_{p,min}, p_{o,min})$$

$$t = min(p_{p,max}, p_{o,max})$$

Then, the width-factor is determined by the proportion of Interval Prediction captured by the effective bottom and top:

$$f_w = \frac{t-b}{p_{p,max} - p_{p,min}}$$

The width-factor has a maximum of 1.0 when the predicted minimum and maximum are equal to or completely between the range of the observed minimum and maximum price. It has a minimum of 0.0 as the width of the Interval Prediction goes to infinity.

The inclusion-factor meanwhile is determined by looking at the percentage of time points when the observed price lies within the Interval Prediction. This is a simpler to express in code than in math, if `p[t]` represents an array of the Coin Metrics Reference Rate at each second, indexed on the timestamp `t`, then the inclusion-factor is given by:

```python
f_i = sum((p > pred_low) & (p < pred_max)) / len(p)
```

Like before, the inclusion-factor has a maximum of 1.0 and a minimum value of 0.0. It reaches 1.0 when the Interval Prediction encompasses all observed prices. It reaches a value of 0.0 if the interval does not include any prices.

Finally, the Miner's score (*r*) for the interval prediction is given quite simply by the product:

$$r = f_w \cdot f_i$$

See drawn figures in **Details** below for visual examples.

Emissions

Closely related to scoring is the distribution of emissions. We propose emissions be evenly split between the Point and Interval Predictions, as both are interesting and useful for different kinds of use-cases. For the Point Predictions the proportion of emissions each Miner gets is based on their weight, which is determined by the rank according to lowest error. For the Interval Predictions the Miner's score, determined by their width- and inclusion-factors, will determine the rank which in turn determines the weight. This weight determines the share of emissions the Miner will receive for producing a new prediction of the future.

Rank

Before a new prediction, each Miner is given a rank which will determine their compensation for the task. The rank is determined by the performance on previous predictions.

Point Prediction
For Point Prediction tasks, the Miners are sorted starting with the lowest *delta-factor* (best) and ascending.

Interval Prediction
For Interval Prediction tasks the Miners are sorted starting with the highest score (best) and descending. Each rank is independent, as are the weights and share of emissions a Miner will receive for each prediction type.

Weights from Rank Algorithm:

- The Miners are ranked from best to worst, depending on type of prediction.
- The first Miner is given an emission *weight* of 1.0: $w_0 = 1.0$. Then for each Miner in sequence:
- If the Miner's error is larger than the one before them (usual case), their weight is 90% of the previous weight: $w_{i+1} = 0.9 * w_i$
- If the Miner's error is tied with the one before them (edge case), their weight is equal: $w_{i+1} = w_i$
- Once all the Miners have been assigned a weight their *share* of the emissions is given by the proportion of the their weight relative to all the weights assigned: $s_i = w_i/(\sum_n w_n)$

The motivations and consequences of this algorithm are explored in more detail below, but one thing to note is that for a very small number of Miners it allocates emissions in a very flat way, incentivizing new participants to join. For a large number of Miners, the emissions skew sharply towards the top ranks incentivizing good performance.

Time Slot Bounty

We do not currently have a satisfying mechanism for this, but in the future we may explore ideas for increasing the value predictions at certain times more than others, based on whether the real price moved significantly during the predicted time.

Data Input - Coin Metrics

Coin Metrics can provide data available through our free Community API. We are currently clarifying and compiling exactly which data is available, but at a minimum it should include the previous 24 hours and 59 minutes of Reference Rates at 1-second frequency. The Community API also provides all market and asset candles down to 1-minute frequency for the past

24h59m. Beyond 24 hours of history both Reference Rates and Candles are available at 1-day frequency for the full history of available data.

It is possible that we can provide an “enhanced” version of the Community permissions, which will extend Reference Rates, Candles, and possibly other essential data feeds. This will allow Miners to tune and test their models. It should be much easier to justify this Bittensor-specific permission profile if we require proof of enrollment in the subnet.

The Community API does *not* include most pre-computed metrics, but we can write the functions in the Base Miner Client to query our API for some of some potentially interesting metrics, like Trusted Volume and Rolling Average Realized Volatility. This will allow participants to frictionlessly incorporate new data series once they provide a paid API key.

The Validators should only need the freely available Reference Rate for the previous 24 hours, which is included in our free Community tier.

We also provide a suite of Network Data and Metrics including Numbers of Transactions, Transaction Volumes, Percentage of Addresses with a Given Supply, as well as fine grained address-level transaction information. We are in the process of determining exactly what is available through the Community API, but most derived metrics and account-level information will require a paid API key.

Parameters

Several parameters are mentioned above in terms of specific values. I’ll try to collect them all here along with a tentative name. This should help if we want to try other time frames in the future:

- Prediction Horizon (1 hour) - How far in the future the prediction will be evaluated
- Prediction Frequency (5 Minutes) - How frequently predictions are submitted by Miners
- Evaluation Frequency (5 Minutes) - How frequently the Validators rank miners to determine their share of emissions for new predictions.
- Error Rolling Average Size (12 ?) - Number of evaluation errors used in rolling average to determine delta-factor
- Miner Rank Prediction Ratio (0.9) - The ratio of weight between each Miner and the more highly ranked Miner previous to them.

Motivation

While predicting markets is hardly original there are several aspects that make this implementation unique and valuable:

Unique Market: Cryptocurrency markets and prices involve many considerations distinct from traditional assets. The environment is constantly changing, and given their short history it's

likely even the fundamentals are still being discovered in the market. All this means it is difficult to know exactly which models apply, which in turn makes these markets an ideal arena to open to novel prediction methods. Multitudes of different ideas can be tested against each other continuously with the best performers eventually being compensated for their performance.

Clarity: The subnet task requires the Miners to digest market information and output the most actionable measures users are interested in: What is the Price Going to Be? How Confident Are We or How Much Will the Price Move?

While some spaces for expressing knowledge of price predictions already exist, in the form of option and futures markets, interpreting these as human-readable quantities requires mathematical transformation: Options may provide an implied volatility for example, but if a user wants to know the probability the price stays within some band, they'll need to significant work. Assumptions need to be made about the shape of the probability distribution, that distribution needs to be integrated, and that's without even considering how to combine volatility values from different strike prices.

On this subnet, all of those calculations and approximations are offloaded to the individual miners to implement, if they consider them significant.

High Frequency Evaluation: BitTensor enables a regime of prediction parameters never before seen, on the order of minutes and hours. Although derivatives markets exist, the shortest expiration dates of these contracts is typically 1-day. Publishing and settling contracts on the hour would likely be an impractical amount of overhead for a centralized exchange. However this does not pose a challenge to an open community of permanently connected Miners and Validators.

The result is predictions with much finer time resolution and the ability to leverage different strategies and market knowledge than is currency possible.

Volatility: While a highly volatile emission schedule can have downsides, it brings some benefits as well. Variance is a central concept in game design, and there are several real world examples where uncertainty can make a challenge more exciting. Having many emissions for short-term predictions means the Miners will get ample opportunities to make up for bad luck, which is sure to happen in a stochastic environment like trading markets.

Variety: An attractive feature of this design is the variety it provides participants to find a problem that appeals to their preferences. The two types of prediction can be calculated using different models and input data: Interval Predictions will likely incorporate the price volatility of the assets, while Point Predictions may be more interested in filtering and smoothing the historical signals.

The emission profiles are also distinct, although the details of the environment will have to determine which of the Point or Interval Predictions have sharper or flatter overall emissions

Details

Emissions

The algorithm above sets the weight, but because there can be a variable number of Miners participating, it is helpful to look at how this weight translates into the share of emissions earned by Miners in different scenarios.

Each miner's share of the emissions can be derived from the equation of the weight function. We stated each Miner gets 0.9 times the weight of the previous Miner. For ease of writing we refer to the ratio of sequential miners' weights as *r* (*r = 0.9*). Then, Miner *i*'s share (s_i) of the emissions when there are *N* total Miners is given by

$$s_{i,N} = \frac{r^i}{\sum_{n=0}^{N-1} r^n} = r^i \frac{1-r}{1-r^N}$$

Where for the last step we used the formula for a Partial Geometric Sum to simplify. This lets us easily calculate some benchmarks for the Miner's emissions based on their rank and the total number of Miners. Note that the best miner has an index of *i=0*, the second best is *i=1,* and so on.

When r = 0.9 (this proposal)

Total Miners (N)	Best Miner (i=0)	Second Best	Tenth Best	Worst (i=N-1)
2	52.63%	47.37%	N/A	47.37%
3	36.90%	33.21%	N/A	29.89%
10	15.35%	13.82%	5.95%	5.95%
11	14.57%	13.12%	5.65%	5.08%
100	10.00%	9.00%	3.87%	0.00%

An appealing feature is that for small numbers of Miners the incentives are almost equally distributed. If there are only two Miners, a third participant is guaranteed at least 29.9% of the emissions for joining regardless of their accuracy. If there's 10 Miners a new participant is guaranteed at least 5.08% of the emissions.

However as the number of participants grows the low-performing predictions keep a very small share. Note that at 100 participants the top two Miners get 10% and 9% respectively, but the last one gets less than 0.005%. With 250 total Miners, everyone past rank 100 makes less than 0.00027%, and the last Miner makes less than 1e-8 of the total emissions.

We believe this is a reasonable ratio to incentivize both participation and competition. Still, for

reference we can compare what happens to emissions if we make the ratio of each successive weight flatter (r=0.95) or sharper (r=0.8):

r = 0.95 (flatter emissions)

Total Miners (N)	Best Miner (i=0)	Second Best	Tenth Best	Worst (i=N-1)
2	51.28%	48.72%	NA	48.72%
3	35.06%	33.30%	NA	31.64%
10	12.46%	11.84%	7.85%	7.85%
11	11.60%	11.02%	7.31%	6.94%
100	5.03%	4.78%	3.17%	0.03%

We see here that at small numbers of participants the results are very similar, but at 100 participants the top Miners are getting almost half of what they receive in *r = 0.9* case.

r = 0.8 (sharper emissions)

Total Miners (N)	Best Miner (i=0)	Second Best	Tenth Best	Worst (i=N-1)
2	55.56%	44.44%	NA	44.44%
3	40.98%	32.79%	NA	26.23%
10	22.41%	17.92%	3.01%	3.01%
11	21.88%	17.50%	2.94%	2.35%
100	20.00%	16.00%	2.68%	0.00%

Conversely, in this case we see when the third Miner first joins they are guaranteed only 26% rather than almost 30% under the original parameters. By the time we have 100 participants, the 10th ranked Miner is receiving roughly one tenth the top Miner's emissions, as opposed to about a third in the proposed (*r=0.9*) scenario.

Interval Ranking Visualization

The process used to score and rank the Interval Predictions may sound convoluted at first but is relatively straightforward if observed on a graph. Below we've drawn three examples of the weight allocated to different Interval Predictions based on a hypothetical price movement.

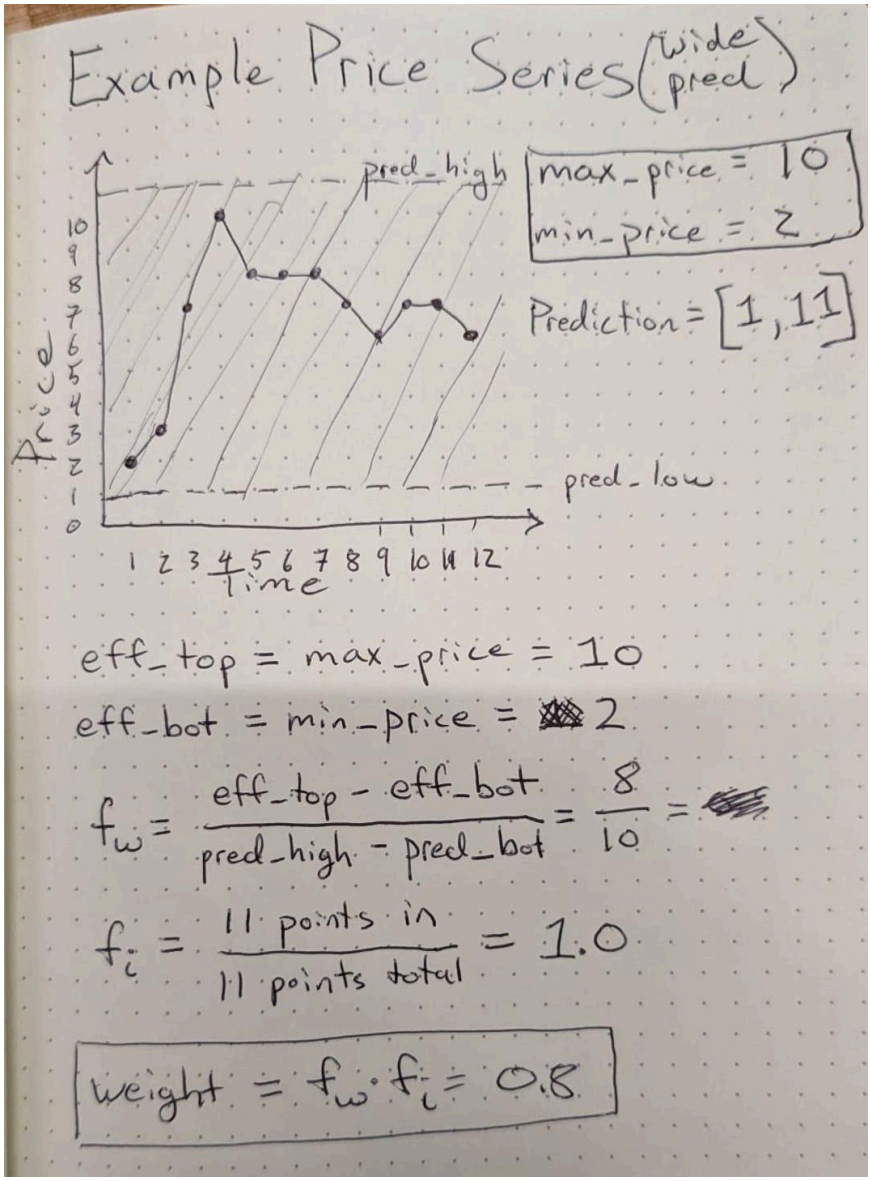
The first example shows a case where the Interval is wider than necessary to capture the full range of price movement. The second example shows the score for an interval which is too narrow to capture the full range of prices. Finally, the third example shows how we evaluate an Interval which is the correct width, but is offset from where the price movements happened. In

other words, for the third case the top of the interval was too high, but the bottom of the interval was too conservative and failed to capture some of the price movements.

Excessive Width:

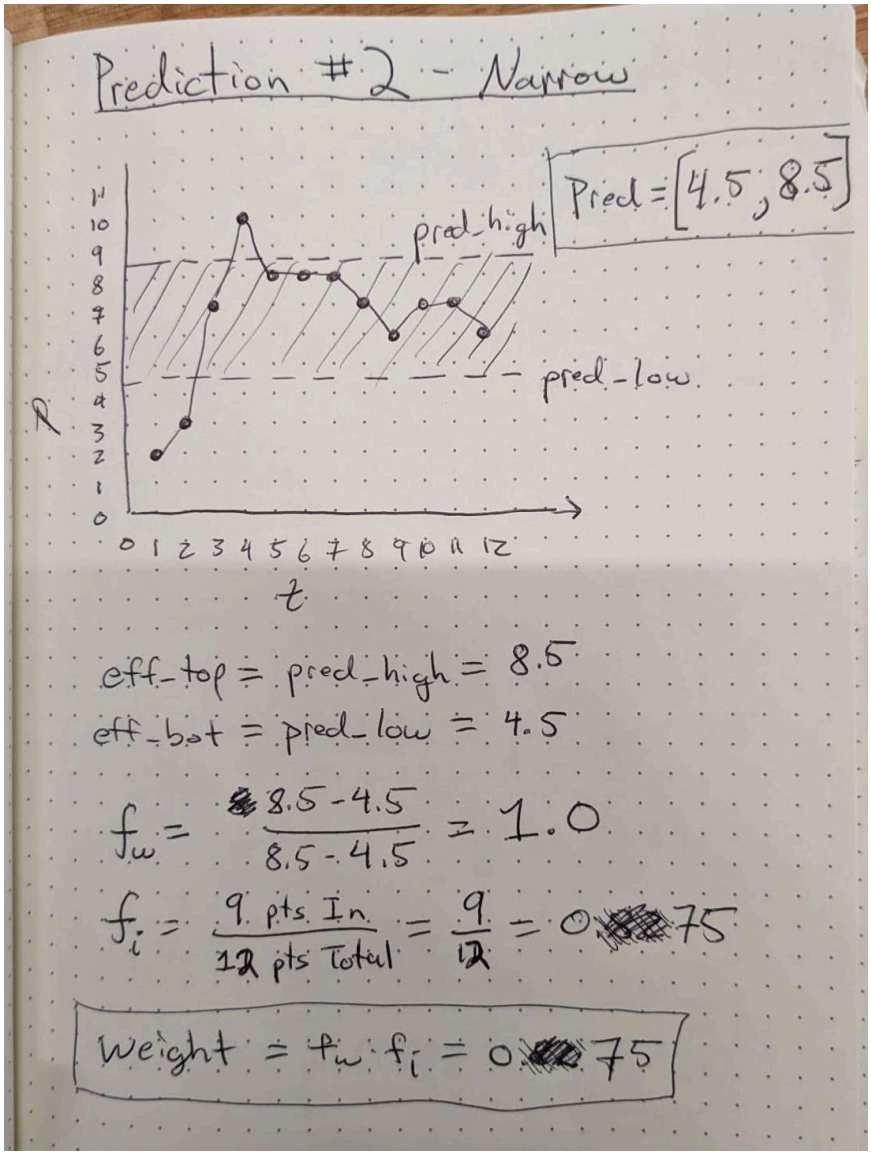
In this example, the top and bottom of the prediction (*pred_high*, *pred_bot*) are respectively above and below the range of the observed price. Therefore the *effective* top and bottom of the interval are limited by the max and min price points. This results in a width-factor less than 1, because the interval was excessively wide. The inclusion-factor is still 1 because it did nevertheless capture 100% of the observed prices in the prediction.

Small typo - f_i should count **12** points in and **12** total, doesn't change calculation



Interval Not Wide Enough:

In this example the interval is too narrow, and as a result the prediction does not correctly capture the full range of the price movement. The effective top and bottom of the interval are set by the prediction itself, which produces a width-factor of 1.0. However, the inclusion factor is imperfect, and only captures 75% of the price data points.



Interval Correct Width but Offset:

The last example shows what happens if one side of the Interval Prediction is wider than necessary, while the other fails to capture all the price data. In this case, because the top of the

Interval was wider than necessary, the effective top is set by the max price; which results in a width-factor less than 1.0. At the same time, the interval failed to include all of the price movements so it is not even able to fall back on that consolation, and the interval-factor is also less than 1.0. This combination gives a final score less than either of the two previous cases.