OK Party People, here it is. An open source, modular system design, to implement decentralized portfolios based on the Numerai crypto meta model, but the generic oracles could really be based on anything.

For all the most up-to-date code and examples, check out the github project:GitHub - jefferythewind/defi_crypto_pm: Open Source Project for Defi Crypto Portfolio Management

There is also a decent PDF document that explains main concepts and also has example code here.

This has been a pet project for just a short time, and I realized there is nothing really holding us back at the moment from having completely decentralized portfolios running right on our own laptop. Idealism is working, at the moment. I've designed a simple and modular concept that makes the problem as simple as it needs to be. 2.

Oracles

The first concept is the oracle. This an object that always knows the optimal portfolio. Of course, in practice, this is up to the person implementing the code, what portfolios are returned from the oracle. The the first example, i've implemented a very simple version of the TB N

portfolio. This portfolio simply returns an equally weighted portfolio with N long and N short positions, based on the largest and smallest ranks given by Numerai's meta model.

I've made the oracle receive a tradable universe

argument, which should come from the DEX. The oracle here returns the best portfolio given the intersection of the tradable universe and Numerai's meta model.

[We see here that we only require two methods from the oracle

](https://github.com/jefferythewind/defi_crypto_pm/blob/main/oracle_interfaces.py):

1. Fetch Portfolio Weights
2. Validate Weights

DEX Interfaces

Once we know an optimal portfolio, all we need is to set our exposure to match this portfolio in the real market. Our interface with the DEX should be this simple.

We see here that we only require two methods from the DEX interface

1. Get Universe
2. Set Portfolio Weights

Portfolio Manager

With the generic Oracle and DEX interfaces, a portfolio manager can do its job in straight-forward manner. It will get the portfolio weights from the oracle, given the tradable universe of the dex. It will then check the weights, and assuming they are good, it will send the weights to the DEX interface for setting. The portfolio manager only needs to implement the manage_portfolio

method.

Portfolio Manager

class PortfolioManager: def **init**(self, oracle: OracleInterface, dex: DEXInterface): """Initialize the Portfolio Manager with an Oracle implementation.""" self.oracle = oracle self.dex = dex

```
def manage_portfolio(self, timestamp = datetime.now(timezone.utc) ):
    """Fetch weights from the Oracle and print them."""
    print(f"[PortfolioManager] Requesting portfolio weights for {timestamp}.")
    tradable_universe = dex.get_universe()
    # tradable_universe = ['BTC','ETH','AAVE','LDO','SUI','WLD','GOAT','EIGEN','AVAX','ENS']
    weights = self.oracle.fetch_portfolio_weights(
        timestamp,
        tradable_universe
    )

    if not self.oracle.validate_weights(weights):
        raise ValueError("Invalid portfolio weights received from Oracle.")

    print(f"[PortfolioManager] Portfolio weights: {weights}")
```

```
dex.set_portfolio_weights( weights )
```

[

Modular Design Schematic

1220×616 66.2 KB

](https://forum.numer.ai/uploads/default/original/2X/f/f2bf03ccab9f743af8ee8a85e78a140092c5cdad.png)

Given this generic design, making anything happen in the markets requires overriding the generic methods for concrete cases. I've done this in the example notebook.

Some Takeaways

You'll see in the code that most of the work so far has been implementing the DEX interface with HyperLiquid. It certainly is one of the three main parts here. However the process is straight-forward. As long as you implement the desire methods with the same inputs and outputs, you can make a DEX interface to any DEX.

I learned that Coinbase and Dydx technology for the perpetual futures markets is restricted and not available for people living in North America. Hyperliquid worked for me from my location. (This is not an endorsement of the exchange, I really don't know much about it yet. And every geographic location has its own laws and restrictions, so please find out based on your situation.) So far, it seems like a quality project. More amazingly is that all you have to do is connect a local wallet, like meta mask or even coinbase wallet, to the hyperliquid exchange. You have to make sure it is funded on the Arbitrum network with USDC and some ETH for gas. And for test net

, which is what I am currently using for this testing, you can get free mock USDC from their faucet. The whole thing is a cool process, and reminds me the utility that actually exists here in the crypto space. It seems to me that Hyperliquid is trying to make it as easy as it needs to be to get access to their perpetual futures market. The api works well. Hopefully my code will work for you out-of-the-box.

Next Steps

Hopefully before long I will make available a simple backtester which should integrate into the framework. The oracles receive a timestamp argument. The idea is that if we supply a historical timestamp, the orcale can give portfolios for those historical days as well. I think in the current version it will do that. Historical data should be available directly from the DEX but also I will check the coverage in yahoo data.

Disclaimer

I'm working on this project as a concept to try to encourage the community to realize that it seems all the tools are at our fingertips already to trade the AI-generated meta model from Numerai. Be careful with your money and your wallet credentials! The code I've posted is clearly explained, use it at your own risk!