DIFFUSION LABS

BACKTESTING ASSESMENT REPORT

Presented To

CHARANDEEP KAPOOR

Presented By

PRIYANSHU CHAURASIYA

Abstract

If you’re into technology, you’ve probably heard of DeFi, blockchain, and cryptocurrency. A new concept in finance—liquidity pool trading—offers an alternative to traditional trading system. Uniswap, a leading decentralized exchange, allows users to provide liquidity and earn a share of trading fees. Each trade incurs a small fee, which is distributed among liquidity providers based on their pool share. While this can be profitable, risks like impermanent loss must be managed. This report outlines a basic strategy for liquidity providers trading in the ETH/USDC pool on Uniswap, focusing on maximizing returns while mitigating risks.

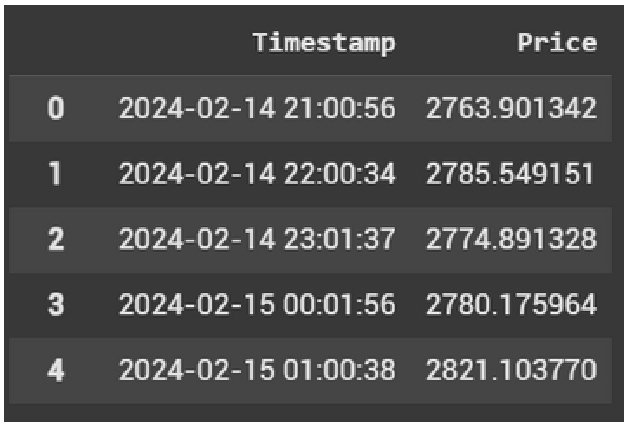
Problem Statement

Develop and evaluate a Python-based backtesting framework for Uniswap v3 liquidity provision in the ETH/USDC pool. The framework will simulate an initial liquidity position with 100 ETH on past one year data.
It will implement a dynamic rebalancing strategy, where liquidity positions are adjusted when market prices exit predefined ranges. Range will be derived from a parameter x = [0. 01, 0.05, 0.10, ...] which needs optimisation to balance between risk and return. Performance will be assessed against a HODL (buy-and-hold) strategy using key financial metrics such as total returns, Sharpe ratio, maximum drawdown, and impermanent loss.

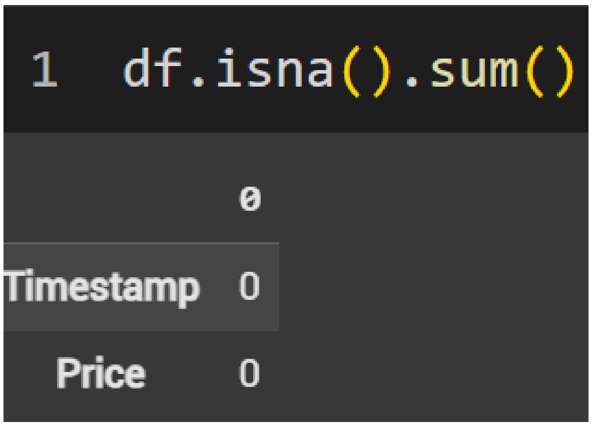
******

Solution

Well to start with, it was challenging to get one year hourly historical data with a free account, however with extensive search i found the coingecko which gives 90 days data, so i extracted the data in chunks and mereged it. This is how it looks like Fortunately data was clear with no missing value



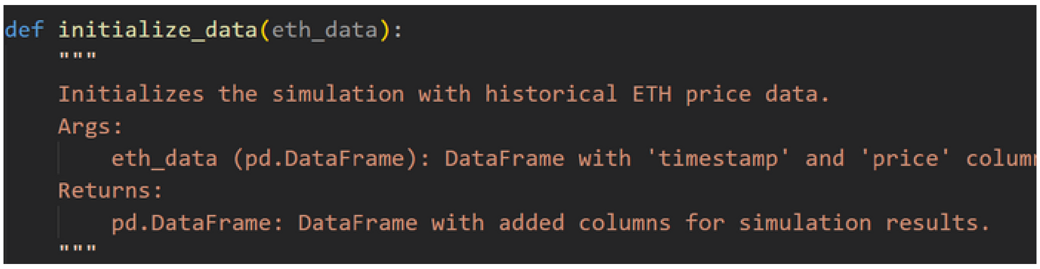
	Timestamp	Price
0	2024-02-14 21:00:56	2763.901342
1	2024-02-14 22:00:34	2785.549151
2	2024-02-14 23:01:37	2774.891328
3	2024-02-15 00:01:56	2780.175964
4	2024-02-15 01:00:38	2821.103770



```
df.isna().sum()
```

	0
Timestamp	0
Price	0

Added the column lp_value, hodl_value, etc by creating a function initialize_data

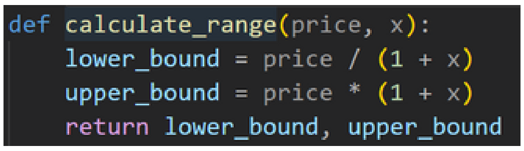


```python
def initialize_data(eth_data):
    """
    Initializes the simulation with historical ETH price data.
    Args:
        eth_data (pd.DataFrame): DataFrame with 'timestamp' and 'price' colum
    Returns:
        pd.DataFrame: DataFrame with added columns for simulation results.
    """
```

The idea is we will start with 100 ETH from day 1, we will put 50ETH and USDC worth equivalent to 50ETH (50/50 distribution) in the Liquidity pool. Now the bechmanrk strategy is HODL strategy i,e. we are going to hold this asset as it for 1year, so at any time the worth of this portfolio will be calculated as below

HODL current value = 50 * current eth price + initial USDC amount

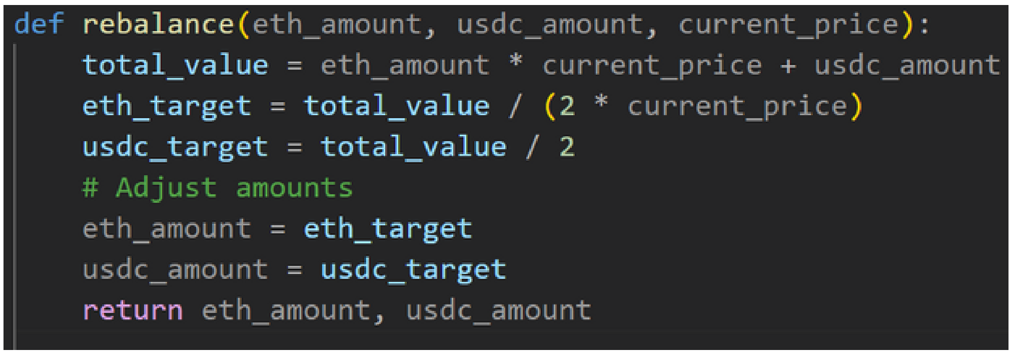
In lp strategy we will first define a band based on current ETH Price and a parameter x which is as below:



```python
def calculate_range(price, x):
    lower_bound = price / (1 + x)
    upper_bound = price * (1 + x)
    return lower_bound, upper_bound
```

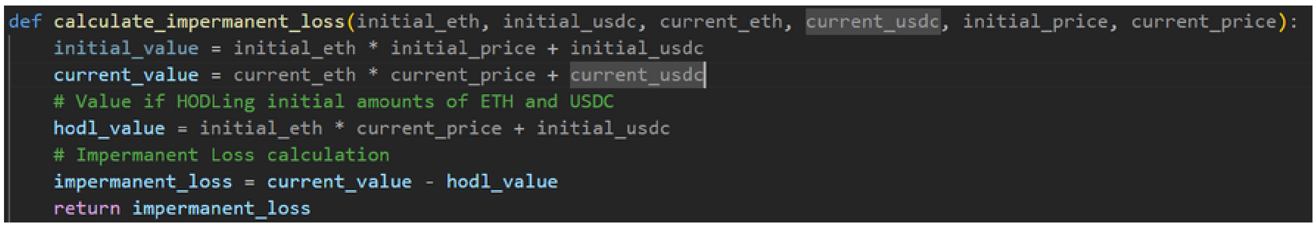
Now as a LP provider you will earn from swap fee (few % depend on pool) as long as price will be in this range. However, as soon as the price go out of this range you will get no return on your invested money, so the idea is to dynamically monitor the price and keep changing the band to optimise the profit with adjusted risk.

In implementation i made a function rebalance, at any time price crosses the range it will again divide ETH and USDC in 50/50 proportion and change the liquidity pool range as per the new price.



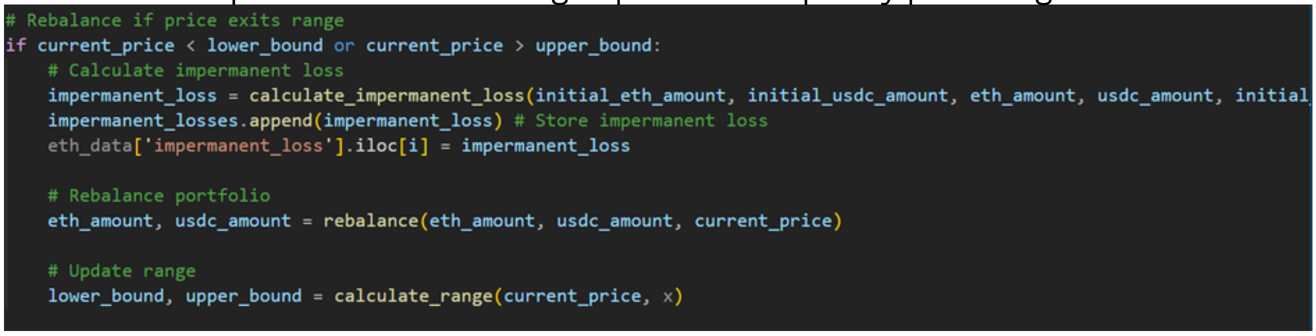
```python
def rebalance(eth_amount, usdc_amount, current_price):
    total_value = eth_amount * current_price + usdc_amount
    eth_target = total_value / (2 * current_price)
    usdc_target = total_value / 2
    # Adjust amounts
    eth_amount = eth_target
    usdc_amount = usdc_target
    return eth_amount, usdc_amount
```

Another important thing is to calculate the impermanent loss which occurs during rebalancing, implementing exactly as per uniswap v3 is not that easy cause it’s too sophosticated, which requires many parametres not given in the assesment such as tick, exact pool , swap fee, etc. So i implemented an easier version of it. You may refere to this paper for more clear understanding.
https://milkroad.com/guide/impermanent-loss/



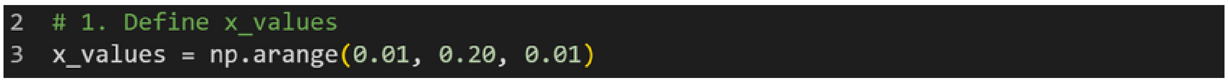
```
def calculate_impermanent_loss(initial_eth, initial_usdc, current_eth, current_usdc, initial_price, current_price):
    initial_value = initial_eth * initial_price + initial_usdc
    current_value = current_eth * current_price + current_usdc
    # Value if HODLing initial amounts of ETH and USDC
    hodl_value = initial_eth * current_price + initial_usdc
    # Impermanent Loss calculation
    impermanent_loss = current_value - hodl_value
    return impermanent_loss
```

As soon as the price crosses the range update the liquidity pool range



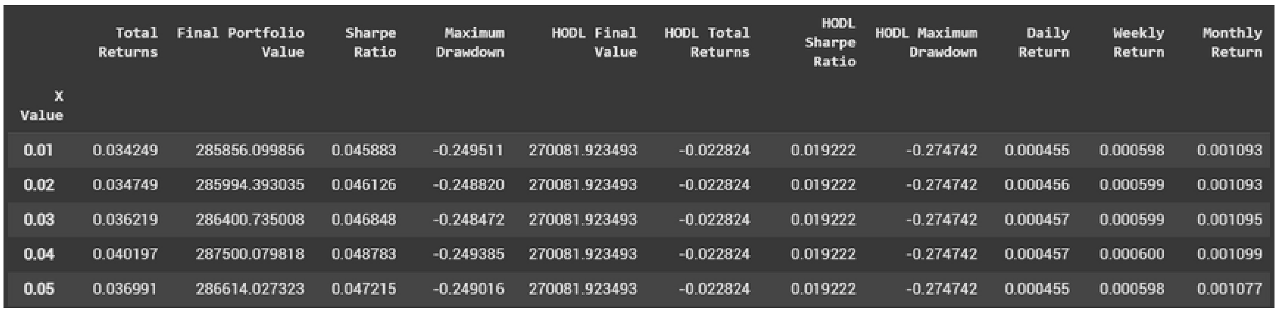
```
# Rebalance if price exits range
if current_price < lower_bound or current_price > upper_bound:
    # Calculate impermanent loss
    impermanent_loss = calculate_impermanent_loss(initial_eth_amount, initial_usdc_amount, eth_amount, usdc_amount, initial
    impermanent_losses.append(impermanent_loss) # Store impermanent loss
    eth_data['impermanent_loss'].iloc[i] = impermanent_loss

    # Rebalance portfolio
    eth_amount, usdc_amount = rebalance(eth_amount, usdc_amount, current_price)

    # Update range
    lower_bound, upper_bound = calculate_range(current_price, x)
```

Now the most critical factor for lp_strategy is value of x. x can vary from 0 to 1, But should we try to work on over whole range. No, we know that asset (ETH) price will not deviate 50% or 100% in a day or in hour so why should we spend our money for a wide range, to increase capital efficiency i only taken the x range from 0 to 20% to find the optimal x_value.



```
# 1. Define x_values
x_values = np.arange(0.01, 0.20, 0.01)
```

The final dataframe i got for these x looks like this



x Value	Total Returns	Final Portfolio Value	Sharpe Ratio	Maximum Drawdown	HODL Final Value	HODL Total Returns	HODL Sharpe Ratio	HODL Maximum Drawdown	Daily Return	Weekly Return	Monthly Return
0.01	0.034249	285856.099856	0.045883	-0.249511	270081.923493	-0.022824	0.019222	-0.274742	0.000455	0.000598	0.001093
0.02	0.034749	285994.393035	0.046126	-0.248820	270081.923493	-0.022824	0.019222	-0.274742	0.000456	0.000599	0.001093
0.03	0.036219	286400.735008	0.046848	-0.248472	270081.923493	-0.022824	0.019222	-0.274742	0.000457	0.000599	0.001095
0.04	0.040197	287500.079818	0.048783	-0.249385	270081.923493	-0.022824	0.019222	-0.274742	0.000457	0.000600	0.001099
0.05	0.036991	286614.027323	0.047215	-0.249016	270081.923493	-0.022824	0.019222	-0.274742	0.000455	0.000598	0.001077

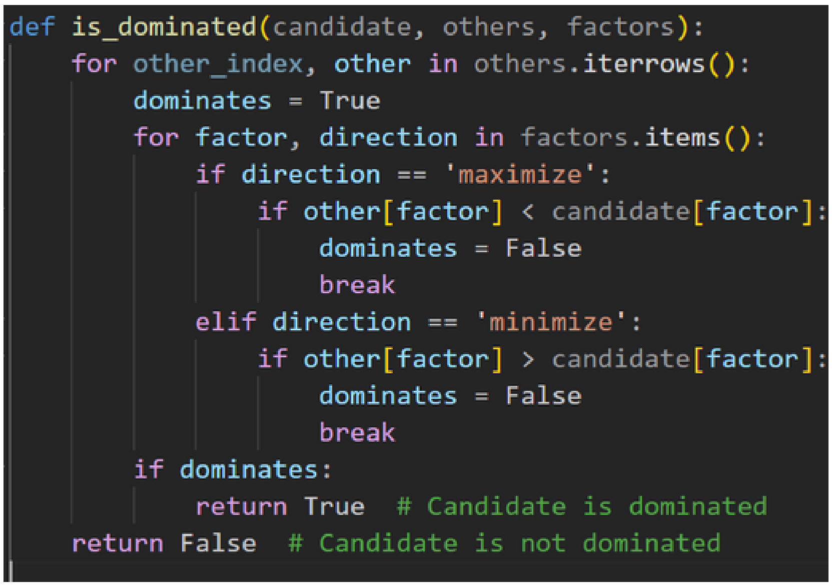
Perfromance metrics i used as given in the assesment which are standard and am not explaining that part.
Now we have 3 factor *sharp ratio, total return and max drawdown* for each x and need to decide optimal x value. If we only consider the total return, we are losing information of sharp ratio and Max_drawdown. A better way would be to try a weighted approach something like calculate a score and find the best

score = 0.33 * sharp_ratio + 0.33* total_return - 0.33*max_drawdown

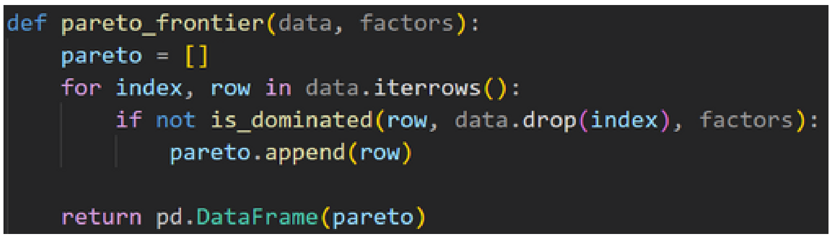
But in this approach i don't know what weight should i choose for each factor.
I tried a more sophosticated approach to tacle this problem which is ***Pareto Frontier (Non-Dominated Sorting)***

Pareto Frontier Algorithm says

- Select assets that **excel in at least one metric without being dominated in others**.
- Example: An asset with high Sharpe Ratio and Total Returns but moderate Drawdown might still be selected over another with **only** high Total Returns but excessive Drawdown.

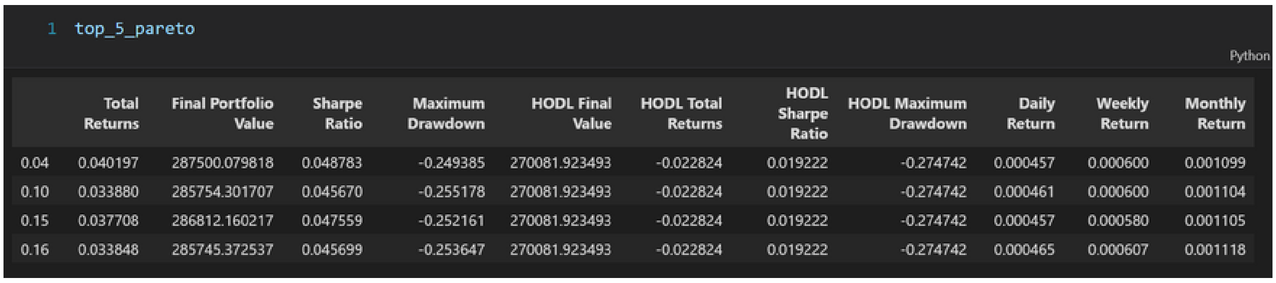


```python
def is_dominated(candidate, others, factors):
    for other_index, other in others.iterrows():
        dominates = True
        for factor, direction in factors.items():
            if direction == 'maximize':
                if other[factor] < candidate[factor]:
                    dominates = False
                    break
            elif direction == 'minimize':
                if other[factor] > candidate[factor]:
                    dominates = False
                    break
        if dominates:
            return True  # Candidate is dominated
    return False  # Candidate is not dominated
```



```python
def pareto_frontier(data, factors):
    pareto = []
    for index, row in data.iterrows():
        if not is_dominated(row, data.drop(index), factors):
            pareto.append(row)

    return pd.DataFrame(pareto)
```

Using the above algorithm i found out the top 5 x _values which are as below

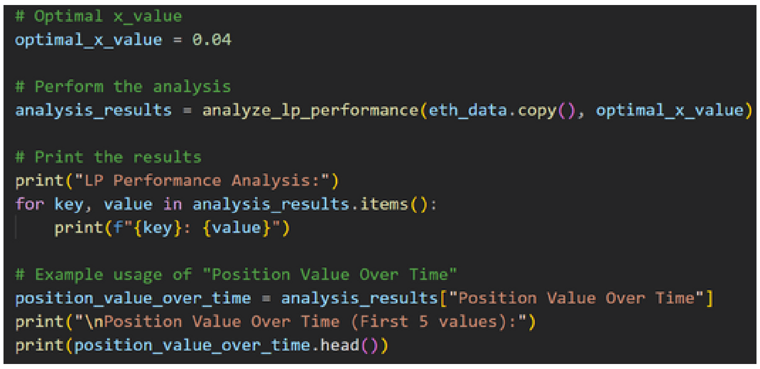


```python
top_5_pareto
```

Python

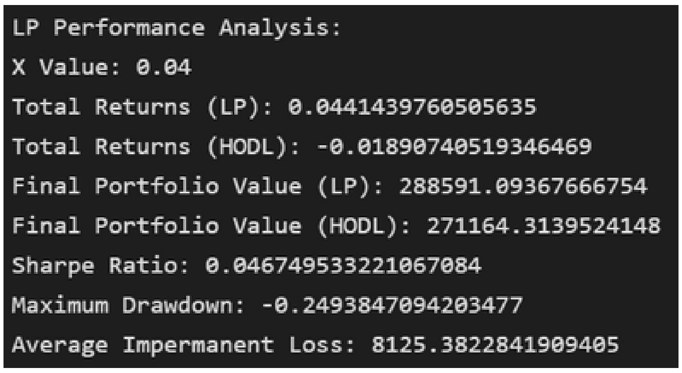
	Total Returns	Final Portfolio Value	Sharpe Ratio	Maximum Drawdown	HODL Final Value	HODL Total Returns	HODL Sharpe Ratio	HODL Maximum Drawdown	Daily Return	Weekly Return	Monthly Return
0.04	0.040197	287500.079818	0.048783	-0.249385	270081.923493	-0.022824	0.019222	-0.274742	0.000457	0.000600	0.001099
0.10	0.033880	285754.301707	0.045670	-0.255178	270081.923493	-0.022824	0.019222	-0.274742	0.000461	0.000600	0.001104
0.15	0.037708	286812.160217	0.047559	-0.252161	270081.923493	-0.022824	0.019222	-0.274742	0.000457	0.000580	0.001105
0.16	0.033848	285745.372537	0.045699	-0.253647	270081.923493	-0.022824	0.019222	-0.274742	0.000465	0.000607	0.001118

Out of 5 values i chooses x_optimal_value = 0.04 as per max final portfolio value.



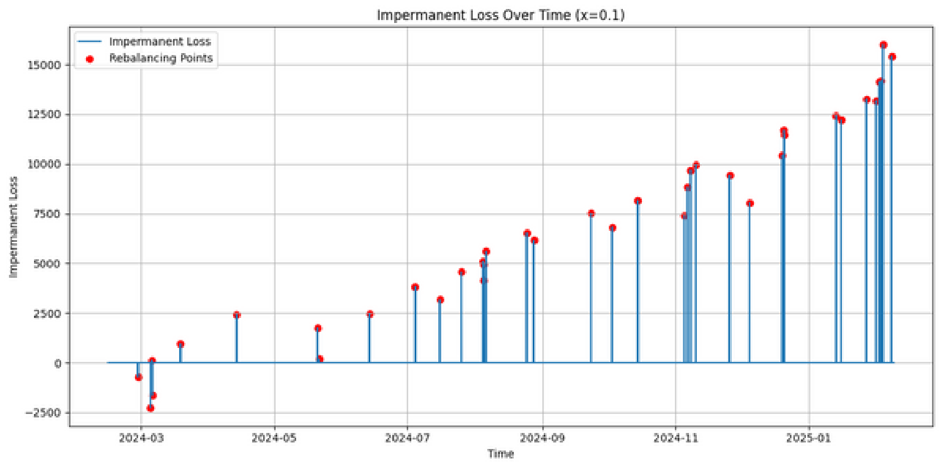
```python
# Optimal x_value
optimal_x_value = 0.04

# Perform the analysis
analysis_results = analyze_lp_performance(eth_data.copy(), optimal_x_value)

# Print the results
print("LP Performance Analysis:")
for key, value in analysis_results.items():
    print(f"{key}: {value}")

# Example usage of "Position Value Over Time"
position_value_over_time = analysis_results["Position Value Over Time"]
print("\nPosition Value Over Time (First 5 values):")
print(position_value_over_time.head())
```

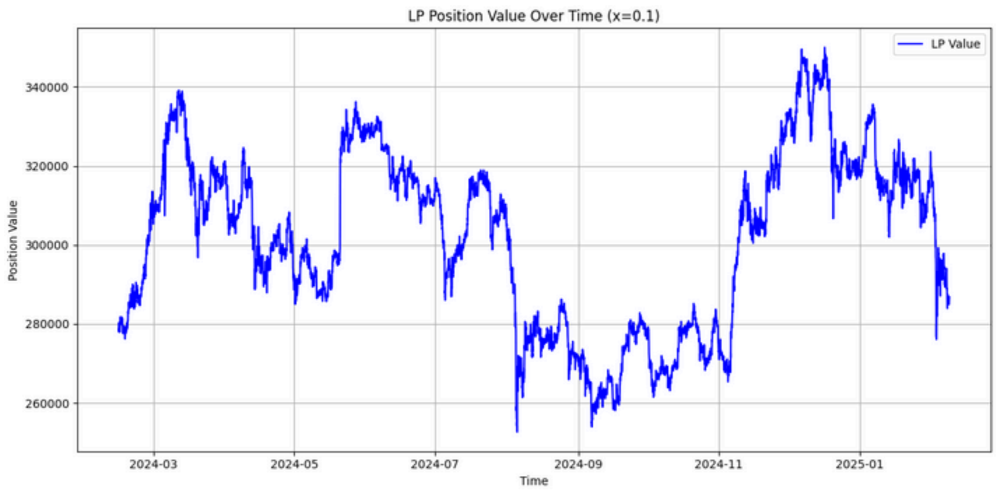
Finally for this x, igot the below results



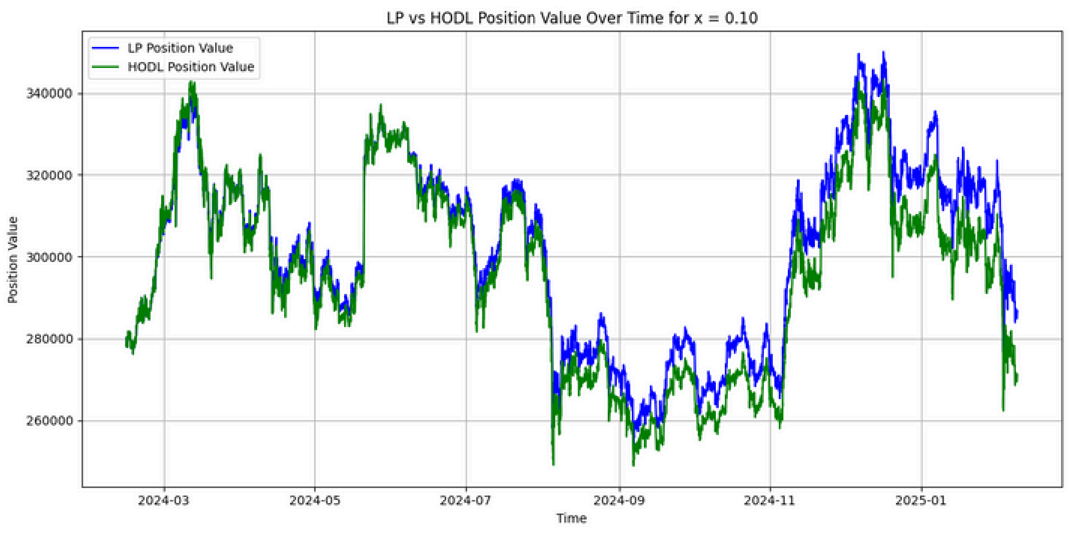
```
LP Performance Analysis:
X Value: 0.04
Total Returns (LP): 0.0441439760505635
Total Returns (HODL): -0.01890740519346469
Final Portfolio Value (LP): 288591.09367666754
Final Portfolio Value (HODL): 271164.3139524148
Sharpe Ratio: 0.046749533221067084
Maximum Drawdown: -0.2493847094203477
Average Impermanent Loss: 8125.3822841909405
```

The lp strategy outperform than simple hodl strategy with a postive return of 4.41%, while hodl gives the negative return of -1.8%. Maximum Drawdwon in the lp strategy is nearly 25% which would might wipe out one-fourth capital, which is not a good practice in trading. This suggest having a stop loss in strategy would be better.

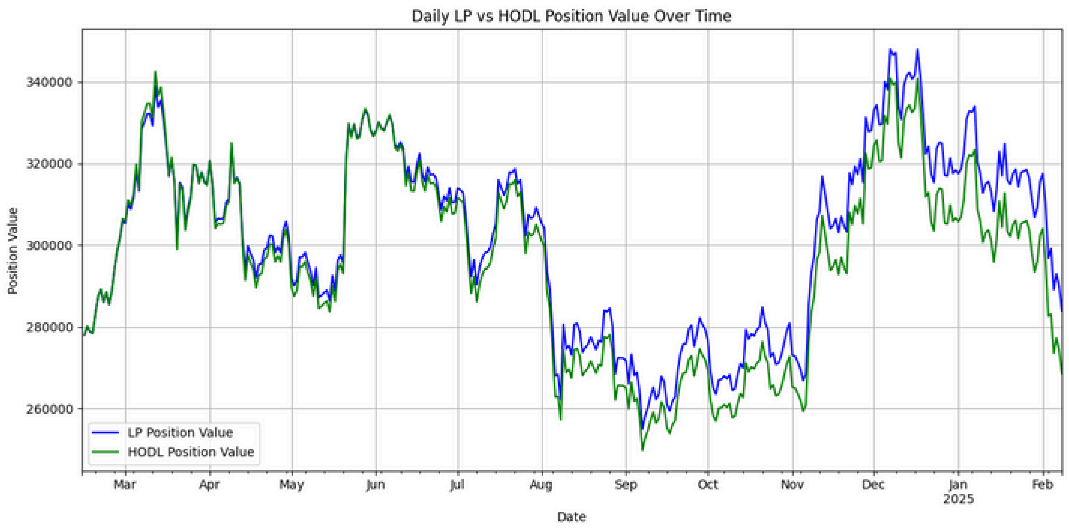


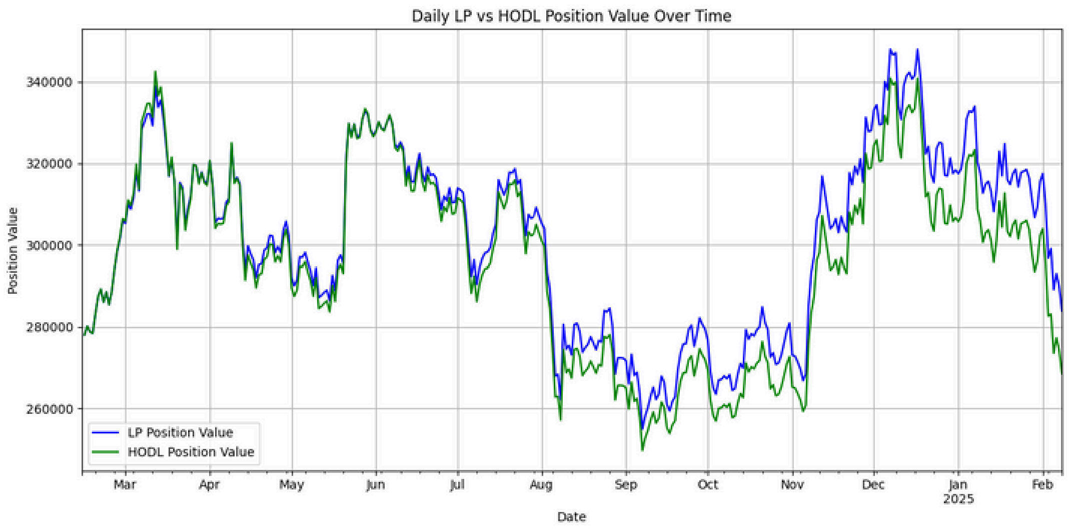


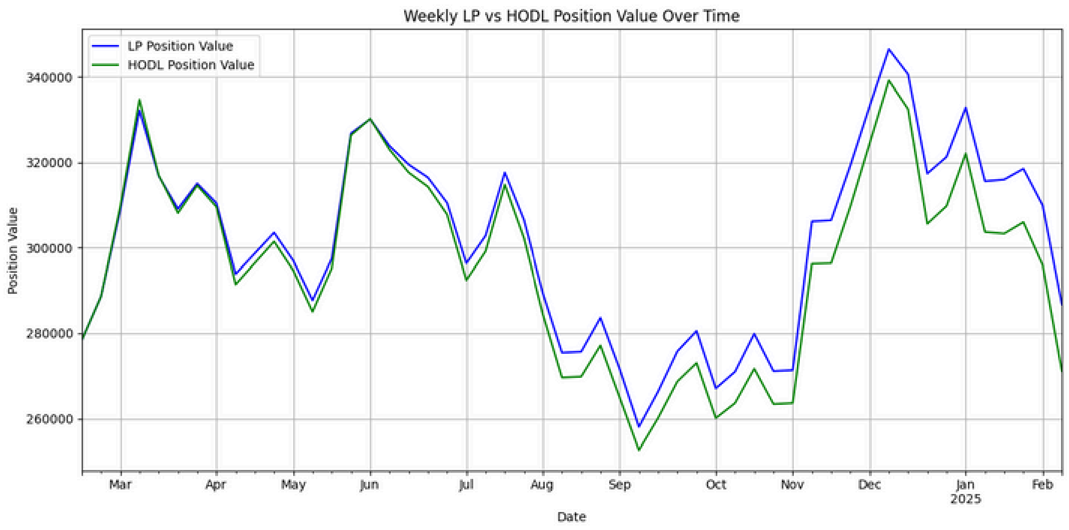
The final lp postion value comes nearly at it's initial level and hence we got just a total return of 4.4% for one year, not an impressive return . However if we look the graph peaks three times at nearly it's maximum. If we have some strategy to book a profit above certain threshold return, we might would get a better return than lp_strategy.
Looking at impermanent loss graph most of the values are positive indicates the strategy incurs less loss during rebalancing the portfolio, which also inline with the postive return as compared to hodl return which is negative -1.8%.

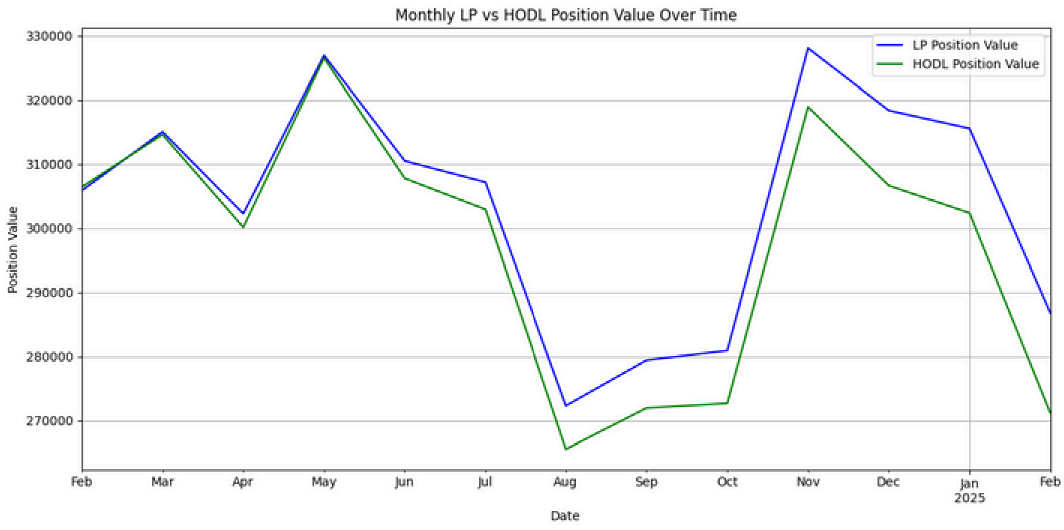


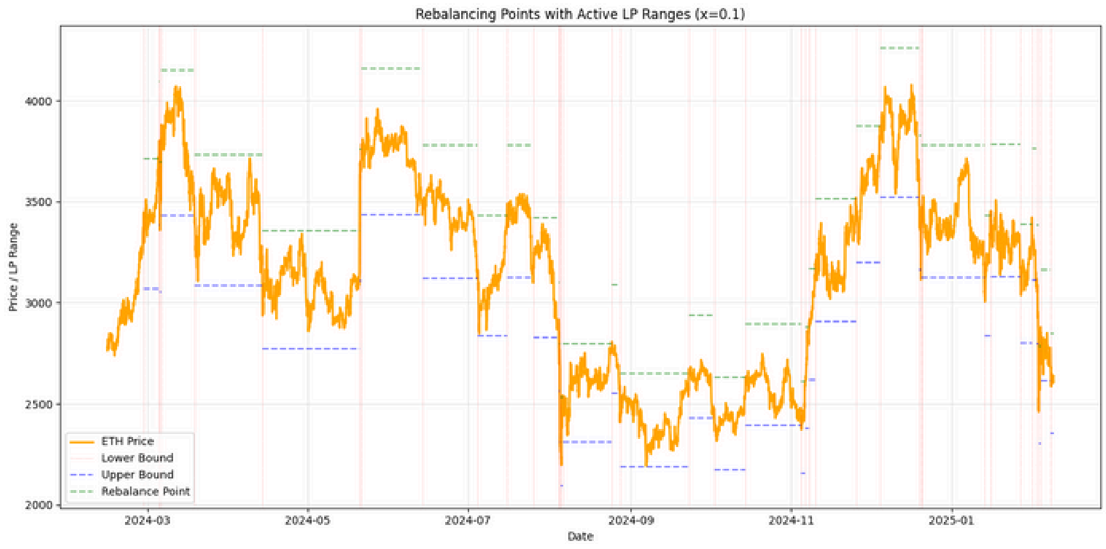
From the above graph i infer that almost all the time lp_strategy outperform than the hodl strategy.



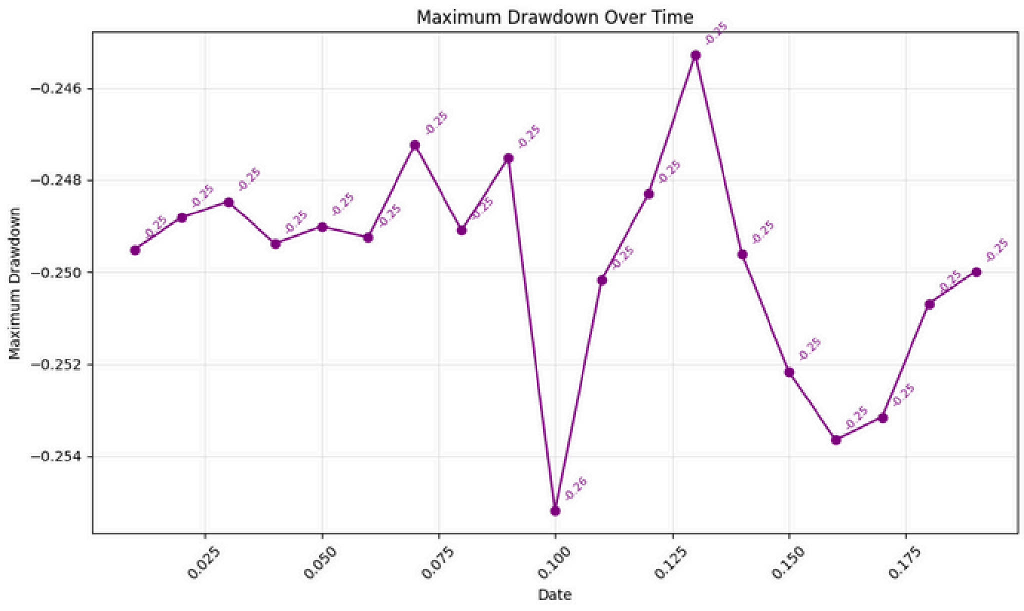



Weekly LP vs HODL Position Value Over Time
LP Position Value
HODL Position Value
Position Value
Date
2025

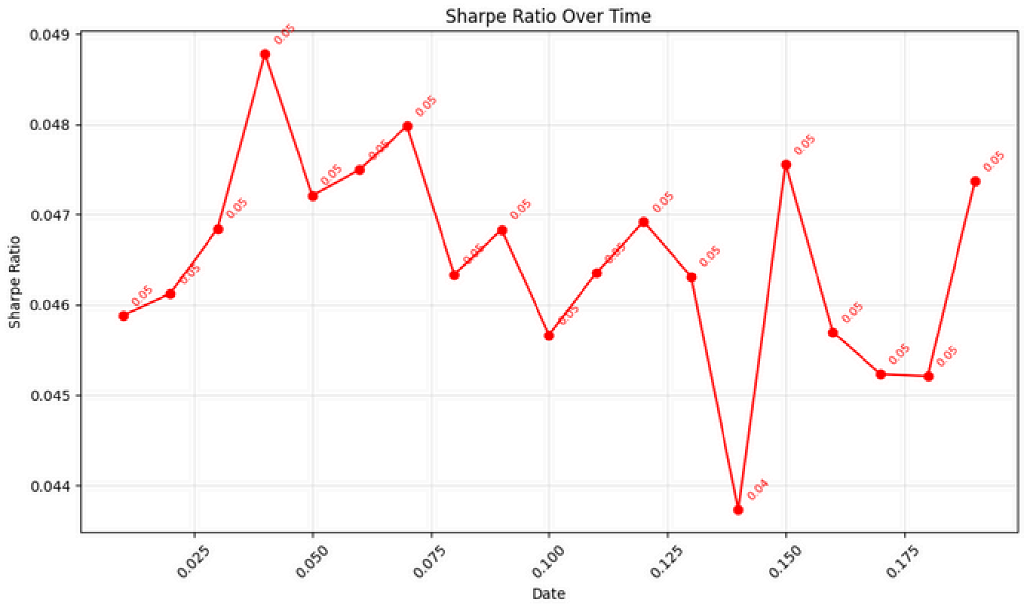

Monthly LP vs HODL Position Value Over Time
LP Position Value
HODL Position Value
Position Value
Date
2025


Rebalancing Points with Active LP Ranges (x=0.1)
ETH Price
Lower Bound
Upper Bound
Rebalance Point
Price / LP Range
Date

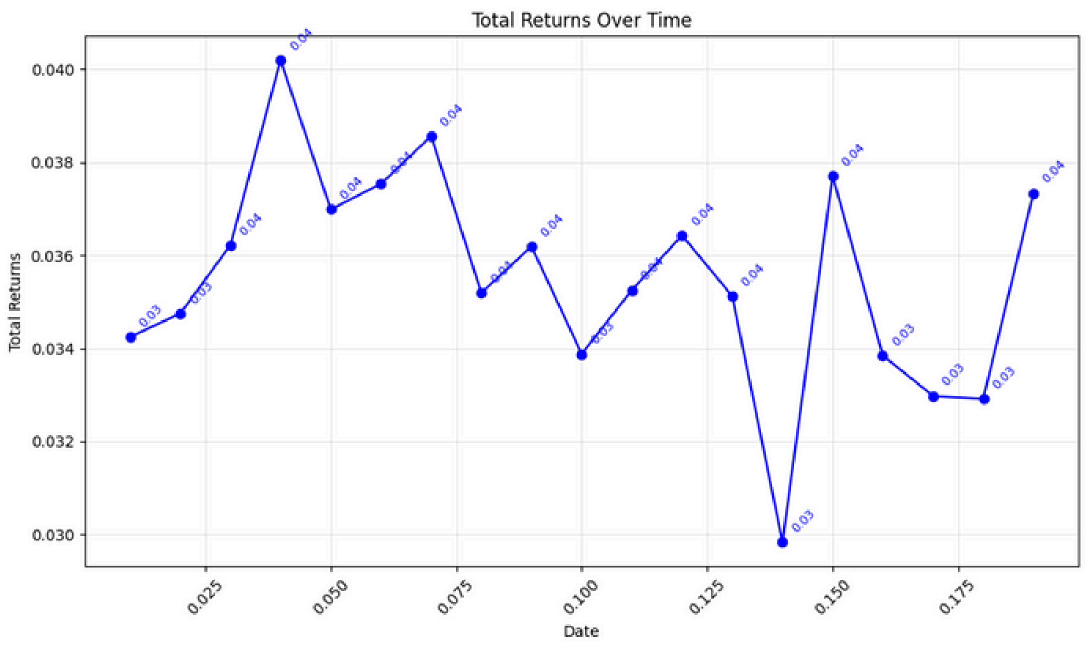
Visulization of trends in performance metrics over range of x_values

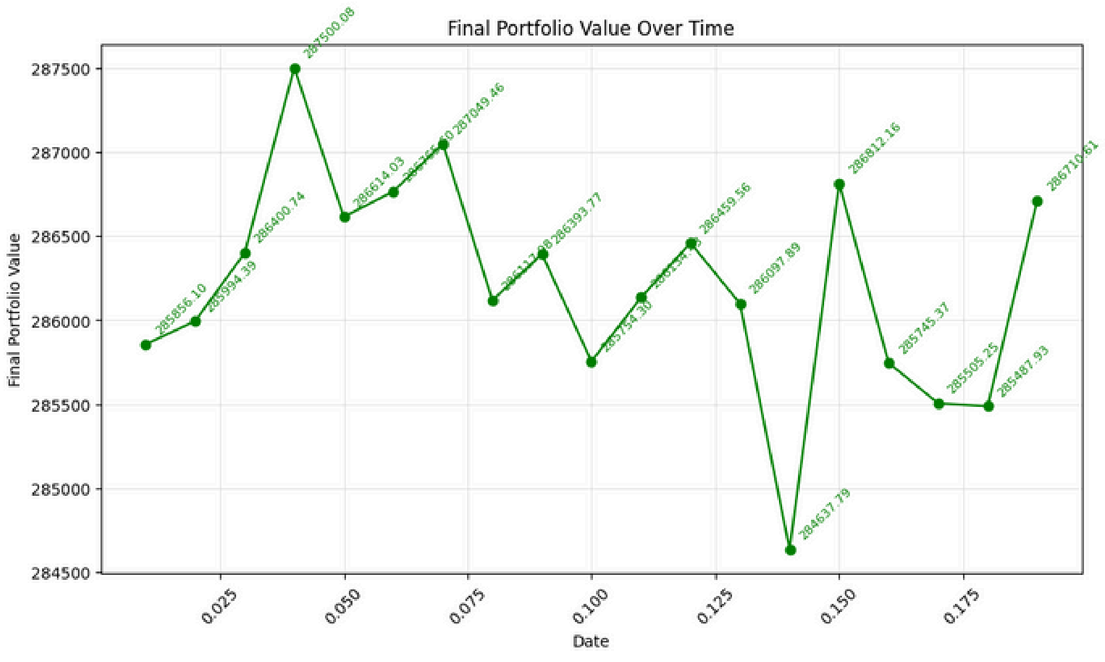


From the above graph i infer, for all the x in [0, 0.20] have a max drawdown of 25%.



Above Graph indicates for all the x in [0, 0.20] has a sharp ratio in a narrow range of 4.5 to 5%


Total Returns Over Time
Total Returns
Date
0.030
0.032
0.034
0.036
0.038
0.040
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.03
0.03
0.04
0.04
0.04
0.04
0.04
0.04
0.04
0.03
0.04
0.04
0.04
0.03
0.04
0.03
0.03
0.03
0.04


Final Portfolio Value Over Time
Final Portfolio Value
Date
284500
285000
285500
286000
286500
287000
287500
0.025
0.050
0.075
0.100
0.125
0.150
0.175
285856.10
285994.39
286400.74
287500.08
286614.03
287049.46
286393.77
285754.30
286459.56
286097.89
284637.79
286812.16
285745.37
285505.25
285487.93
286710.61