Statistical analysis on parameters regarding Little_Shark v1.0 on Binance

Q1: Strategy Verification: How long does the integration feature last?

Let's say if a certain pair has an estimated return >0 for the 499 intervals, would it be profitable for the next 100/100-200/200-300/300-400/400-499 intervals?

Step 1: get all the tradable symbols from binance

```
"""Process of getting target trading symbols"""
from config_logger import logger
from func_get_traget_symbols import get_tradeable_symbols_dynamic,
store_price_history_static, get_cointegrated_pairs
import json


# STEP 1: Get all the tradable symbols
logger.info("Getting tradable symbols from Binance.")
tradeable_symbols = get_tradeable_symbols_dynamic()

tradeable_symbols
```

```
2023-07-11 23:06:04,961 - Rovers_3.0 - INFO - Getting tradable symbols
from Binance.
2023-07-11 23:06:56,060 - Rovers_3.0 - INFO - 46 pairs found
```

```
['BTCUSDT',
 'ETHUSDT',
 'BCHUSDT',
 'XRPUSDT',
 'EOSUSDT',
 'LTCUSDT',
 'ETCUSDT',
 'LINKUSDT',
 'ADAUSDT',
 'BNBUSDT',
 'ATOMUSDT',
 'COMPUSDT',
 'DOGEUSDT',
 'KAVAUSDT',
 'WAVESUSDT',
 'MKRUSDT',
```

```
 'DOTUSDT',
 'CRVUSDT',
 'SOLUSDT',
 'STORJUSDT',
 'AVAXUSDT',
 'FTMUSDT',
 'FLMUSDT',
 'TOMOUSDT',
 'NEARUSDT',
 'AAVEUSDT',
 'FILUSDT',
 'MATICUSDT',
 'OCEANUSDT',
 'SANDUSDT',
 'ANKRUSDT',
 'LINAUSDT',
 'STMXUSDT',
 'MTLUSDT',
 '1000SHIBUSDT',
 'MASKUSDT',
 'DYDXUSDT',
 '1000XECUSDT',
 'GALAUSDT',
 'CTSIUSDT',
 'APEUSDT',
 'OPUSDT',
 'INJUSDT',
 'STGUSDT',
 'LDOUSDT',
 'APTUSDT']
```

Step 2: derive all the prices from binance for 1000 intervals

```
# Get prices and store in DataFrame
from binance_market_observer import binance_get_recent_close_price
import pandas as pd

interval = "15m"
num_interval_limit = 1000

counts = 0
price_history_dict = {}
for sym in tradeable_symbols:
    price_history = binance_get_recent_close_price(sym,
interval=interval, limit=num_interval_limit)
    if len(price_history) == num_interval_limit: # make sure that each
symbol has the same amount of data
        price_history_dict[sym] = price_history
        counts += 1
logger.info (f"{counts} items stored, {len(tradeable_symbols)-
```

```
counts}items not stored")

# Output prices to JSON
if len(price_history_dict) > 0:
    filename = f"experiment_price_list.json"
    with open(filename, "w") as fp:
        json.dump(price_history_dict, fp, indent=4)
    logger.info("Prices saved successfully.")

price_history_pandas = pd.DataFrame(price_history_dict)
price_history_pandas
```

```
2023-07-11 23:07:35,281 - Rovers_3.0 - INFO - 46 items stored, 0items
not stored
2023-07-11 23:07:35,408 - Rovers_3.0 - INFO - Prices saved
successfully.
```

```
                      BTCUSDT                ETHUSDT               BCHUSDT
\
0    30385.500000000000000  1915.740000000000009  284.720000000000027

1    30394.299999999999272  1916.250000000000000  284.829999999999984

2    30415.200000000000728  1918.480000000000018  285.829999999999984

3    30424.900000000001455  1918.910000000000082  288.920000000000016

4    30413.599999999998545  1918.009999999999991  289.129999999999995

..                     ...                   ...                   ...

995  30384.799999999999272  1865.039999999999964  272.259999999999991

996  30493.299999999999272  1869.920000000000073  273.670000000000016

997  30518.099999999998545  1872.750000000000000  274.160000000000025

998  30515.000000000000000  1872.839999999999918  273.839999999999975

999  30524.900000000001455  1873.769999999999982  274.000000000000000


     XRPUSDT  EOSUSDT              LTCUSDT             ETCUSDT
LINKUSDT  \
0     0.4673    0.740  106.680000000000007  21.286000000000001
6.177
1     0.4671    0.744  107.260000000000005  21.311000000000000
6.183
2     0.4683    0.745  106.859999999999999  21.411000000000001
6.202
3     0.4673    0.744  107.400000000000006  21.672999999999998
```

```
6.215
4     0.4682    0.742  107.540000000000006  21.765999999999998
6.199
..       ...      ...                  ...                 ...        .
..
995   0.4723    0.719   97.010000000000005  18.648000000000000
6.139
996   0.4735    0.722   97.299999999999997  18.718000000000000
6.165
997   0.4739    0.726   97.359999999999999  18.771000000000001
6.173
998   0.4734    0.725   97.359999999999999  18.779000000000000
6.179
999   0.4739    0.726   97.140000000000001  18.818000000000001
6.182

     ADAUSDT              BNBUSDT  ...  DYDXUSDT  1000XECUSDT
GALAUSDT  \
0     0.2837  240.979999999999990  ...     2.018      0.04329
0.02413
1     0.2841  241.060000000000002  ...     2.024      0.04373
0.02423
2     0.2845  240.990000000000009  ...     2.029      0.04410
0.02428
3     0.2853  241.539999999999992  ...     2.030      0.04433
0.02434
4     0.2847  241.199999999999989  ...     2.026      0.04421
0.02434
..       ...                  ...  ...       ...          ...        ..
.
995   0.2902  246.900000000000006  ...     1.857      0.03287
0.02444
996   0.2915  246.870000000000005  ...     1.865      0.03313
0.02458
997   0.2918  246.919999999999987  ...     1.887      0.03311
0.02469
998   0.2915  246.889999999999986  ...     1.895      0.03304
0.02463
999   0.2918  246.960000000000008  ...     1.883      0.03307
0.02462

     CTSIUSDT  APEUSDT  OPUSDT            INJUSDT  STGUSDT  LDOUSDT
APTUSDT
0      0.1520    2.189  1.3083  8.345000000000001   0.5709   2.0249
7.086
1      0.1521    2.193  1.3123  8.398999999999999   0.5730   2.0328
7.122
2      0.1526    2.205  1.3148  8.401000000000000   0.5759   2.0443
7.149
```

```
3       0.1525    2.211  1.3201  8.417000000000000   0.5772   2.0536
7.165
4       0.1524    2.208  1.3210  8.459000000000000   0.5767   2.0487
7.168
..         ...      ...     ...                ...      ...      ...
...
995     0.1652    1.899  1.2016  8.170000000000000   0.6541   1.9090
6.919
996     0.1645    1.911  1.2065  8.202999999999999   0.6547   1.9171
6.958
997     0.1656    1.918  1.2103  8.199999999999999   0.6522   1.9205
6.980
998     0.1631    1.920  1.2124  8.175000000000001   0.6496   1.9202
6.978
999     0.1636    1.921  1.2110  8.172000000000001   0.6494   1.9189
6.968

[1000 rows x 46 columns]
```

Step 3: Check co-integrated pairs based on the 15m intervals

```
from statsmodels.tsa.stattools import coint
import statsmodels.api as sm
import scipy.stats as stats
import pandas as pd
import numpy as np

# Calculate co-integration
def calculate_cointegration_static(series_1, series_2):
    """
    Calculate the cointegration between two series and return
cointegration flag,
    hedge ratio, and initial intercept.

    Args:
        series_1 (np.ndarray): First series for cointegration
analysis.
        series_2 (np.ndarray): Second series for cointegration
analysis.

    Returns:
        tuple: A tuple containing cointegration flag, hedge ratio, and
initial intercept.

    Notes:
        - The series should have the same length.
        - Cointegration tests the long-term relationship between two
time series.
        - The cointegration flag indicates if the two series are
cointegrated.
```

```
        - The hedge ratio represents the relationship between the two
series.
        - The initial intercept is the intercept of the linear
regression model.

    Raises:
        ValueError: If the input series have different lengths.

    """

    coint_flag = 0
    coint_res = coint(series_1, series_2)
    coint_t = coint_res[0]
    p_value = coint_res[1]
    critical_value = coint_res[2][1]


    # get initial intercept and hedge_ration of the model
    series_2 = sm.add_constant(series_2)
    model = sm.OLS(series_1, series_2).fit()
    initial_intercept = model.params[0]
    hedge_ratio = model.params[1]

    if p_value < 0.5 and coint_t < critical_value:
        coint_flag = 1
    return coint_flag, p_value, hedge_ratio, initial_intercept

# Calculate spread
def calculate_spread_static(series_1: list, series_2: list,
hedge_ratio):
    """
    Calculates the spread between two series using a given hedge
ratio.

    Args:
        series_1 (list): A list of values representing the first
series.
        series_2 (list): A list of values representing the second
series.
        hedge_ratio (float): The hedge ratio to be applied.

    Returns:
        list: A list containing the calculated spread.
    """

    spread = pd.Series(series_1) - (pd.Series(series_2) * hedge_ratio)
    return spread.tolist()

# Calculate Z-Score
def calculate_zscore_static(spread: list) -> list:
```

```
    """
    Calculates the Z-Score of a given spread.

    Args:
        spread (list): A list of values representing the spread.

    Returns:
        list: A list containing the Z-Score values.
    """
    data = np.array(spread)
    return stats.zscore(data)

def check_differnet_signal(a,b):
    return a + b != abs(a) + abs(b)

def calculate_trading_estimated_oppotunities_return(series_1,
series_2, hedge_ratio: float,
                                                    initial_intercept:
float, threshod: float, trading_times_threshod = 5,
investable_capital_each_time = 100, estimated_trading_fee_rate =
0.0004,
                                                    func_z_score =
calculate_zscore_static, num_window = 0) -> int:
    enter_market_signal = False
    spread = calculate_spread_static(series_1, series_2, hedge_ratio)

    if num_window == 0:
        zscore_series = func_z_score(spread)
    elif num_window > 0:
        zscore_series = func_z_score(spread, num_window)

    trade_oppotunities = 0
    last_value = 0.01

    cumulative_return = 0
    cumulative_trading_qty = 0
    count_entering_time = 0

    open_long_price_list = []
    open_short_price_list = []

    for index, value in enumerate(zscore_series):
        if abs(value) >= abs(threshod) and not
check_differnet_signal(value, last_value):
            enter_market_signal = True

            if value >= threshod:
                direction = "sell"
            elif value <= -threshod:
                direction = "buy"
```

```
            if count_entering_time < trading_times_threshod:
                cumulative_trading_qty +=
(investable_capital_each_time / (series_1[index] + hedge_ratio *
series_2[index]))  # qty for symbol 1
                if direction == "buy":
                    open_long_price_list.append(series_1[index])
                    open_short_price_list.append(series_2[index])
                elif direction == "sell":
                    open_short_price_list.append(series_1[index])
                    open_long_price_list.append(series_2[index])

                count_entering_time += 1

        elif enter_market_signal and check_differnet_signal(value,
last_value):
            trade_oppotunities += 1

            # calculate the exiting_revenue of the symbols
            if direction == "buy":
                buy_side_profit = (series_1[index] -
sum(open_long_price_list)/len(open_long_price_list)) *
cumulative_trading_qty
                sell_side_profit =
(sum(open_short_price_list)/len(open_short_price_list) -
series_2[index]) * cumulative_trading_qty * hedge_ratio
            elif direction == "sell":
                buy_side_profit = (series_2[index] -
sum(open_long_price_list)/len(open_long_price_list)) *
cumulative_trading_qty * hedge_ratio
                sell_side_profit =
(sum(open_short_price_list)/len(open_short_price_list) -
series_1[index]) * cumulative_trading_qty

            exiting_profit = buy_side_profit + sell_side_profit -
investable_capital_each_time * count_entering_time *
estimated_trading_fee_rate # revenue for all symbols


            # Cumulate the return
            cumulative_return += exiting_profit

            # Reset
            enter_market_signal = False
            cumulative_trading_qty = 0
            count_entering_time = 0
            direction = ""
            open_long_price_list = []
            open_short_price_list = []
```

```python
        last_value = value


    return trade_oppotunities, cumulative_return

def get_cointegrated_pairs_experiment(prices, num_wave=0,
trigger_z_score_threshod = 1.0) -> str:
    # Loop through coins and check for co-integration
    coint_pair_list = []

    loop_count = 0
    for sym_1 in tradeable_symbols:
        loop_count += 1
        # Check each coin against the first (sym_1)
        for sym_2 in tradeable_symbols[loop_count:]:

            # Get close prices
            series_1 = price_history_dict[sym_1][:498]
            series_2 = price_history_dict[sym_2][:498]

            # Check for cointegration and add cointegrated pair
            coint_flag, p_value, hedge_ratio, initial_intercept =
calculate_cointegration_static(series_1, series_2)

            trading_oppotunities_first_499intervals,
estimated_return_first_499intervals =
calculate_trading_estimated_oppotunities_return(series_1, series_2,
hedge_ratio, initial_intercept, trigger_z_score_threshod)
            if coint_flag == 1:
                time_period_list = [100, 200, 300, 400, 499]
                trading_oppotunities_list = []
                estimated_return_list = []
                for time_period in time_period_list:

                    series_1_after = price_history_dict[sym_1][499 +
time_period - 100:(499 + time_period)]
                    series_2_after = price_history_dict[sym_2][499 +
time_period - 100:(499 + time_period)]

                    coint_flag, p_value, hedge_ratio_new,
initial_intercept = calculate_cointegration_static(series_1_after,
series_2_after)

                    trading_oppotunities, estimated_return =
calculate_trading_estimated_oppotunities_return(series_1_after,
series_2_after, hedge_ratio_new, initial_intercept,
trigger_z_score_threshod)
```

```
trading_oppotunities_list.append(trading_oppotunities)
                    estimated_return_list.append(estimated_return)

                coint_pair_list.append({
                    "sym_1": sym_1,
                    "sym_2": sym_2,
                    "hedge_ratio": hedge_ratio,
                    "initial_intercept": initial_intercept,
                    "trading_oppotunities_first_499intervals":
trading_oppotunities_first_499intervals,
                    "estimated_return_first_499intervals":
estimated_return_first_499intervals,
                    "estimated_return_next_100intervals":
estimated_return_list[0],
                    "estimated_return_next_100-200intervals":
estimated_return_list[1],
                    "estimated_return_next_200-300intervals":
estimated_return_list[2],
                    "estimated_return_next_300-400intervals":
estimated_return_list[3],
                    "estimated_return_next_400-500intervals":
estimated_return_list[4],
                })

    # Output results and rank all the trading pairs
    df_coint = pd.DataFrame(coint_pair_list)
    # add the total score column
    df_coint["total_score"] =
df_coint["estimated_return_first_499intervals"]
    df_coint =
df_coint.sort_values("estimated_return_first_499intervals",
ascending=False)
    filename =
f"experiment_15mintervals_different_period_cointegrated_pairs.csv"
    df_coint.to_csv(filename)

    return df_coint

df_coint = get_cointegrated_pairs_experiment(price_history_dict, 0,
0.8)

df_coint
```

```
          sym_1      sym_2          hedge_ratio  initial_intercept  \
61      MTLUSDT   DYDXUSDT  -0.610901279785415  0.985359636458149
63  1000XECUSDT   GALAUSDT  -0.066747042132137  0.078454712345471
67  1000XECUSDT    LDOUSDT   0.000618958598972  0.006849997951645
59     ANKRUSDT   DYDXUSDT  -0.005792564361732 -0.009795688895947
66  1000XECUSDT    INJUSDT   0.002108501980313  0.074419658193398
..          ...        ...                 ...                ...
```

```
19      ADAUSDT   AVAXUSDT   0.018143061708651  0.206598788250214
54     SANDUSDT   LINAUSDT  16.693308247832295  0.302705371179136
35    WAVESUSDT    FILUSDT   0.040594084039041  1.898014060652207
36    WAVESUSDT  OCEANUSDT   2.223270695160325 -0.472757519638569
37    WAVESUSDT    MTLUSDT   0.222955361309421  2.558582914709757

    trading_oppotunities_first_499intervals  \
61                                         3
63                                         2
67                                         2
59                                         3
66                                         2
..                                       ...
19                                         3
54                                         2
35                                         2
36                                         1
37                                         1

    estimated_return_first_499intervals
estimated_return_next_100intervals  \
61                  803.012713940487515
2.052066190265482
63                  140.457506733074581
38.149985307449811
67                  126.019319375777997
48.046765868937854
59                   90.271139760920065
5.183067122878750
66                   86.633684060121325
38.633330058860764
..                                  ...
...
19                    7.276977563202547
3.707577643396315
54                    5.094259171437786
5.464718463100875
35                    4.390154982411246
14.466761494262723
36                    3.417390829191746
36.914305626008201
37                    2.230519926799618
28.950058978190260

    estimated_return_next_100-200intervals  \
61                       5.938366858710729
63                     -46.808818939909244
67                     -75.602007367077476
59                       3.918642816405311
66                       6.582698676120422
```

```
..                                ...
19                  4.450355358203478
54                  4.256900734630615
35                  0.000000000000000
36                  8.733402498864645
37                 12.564374413623348

    estimated_return_next_200-300intervals  \
61                  6.561709790373469
63                -67.021582090134899
67                 12.617507531177731
59                  0.000000000000000
66               -549.202082302996359
..                                ...
19                  8.504398834365446
54                  3.501341976939190
35                 16.360506772172258
36                 40.274726823515792
37                 18.129540527758245

    estimated_return_next_300-400intervals  \
61                  0.000000000000000
63                 28.488714399423362
67                 60.658250194643173
59                  1.162074992561151
66                 50.298849543817788
..                                ...
19                  3.030187245955257
54                  3.787071864426335
35                -18.261925475433817
36                -93.388791461744503
37                 37.302128184076942

    estimated_return_next_400-500intervals         total_score
61                 19.095028497191645  803.012713940487515
63                -78.696406365079397  140.457506733074581
67                  9.943805627148880  126.019319375777997
59                  7.770679064567187   90.271139760920065
66               -169.703788150292013   86.633684060121325
..                                ...                  ...
19                  0.000000000000000    7.276977563202547
54                  6.860873587350145    5.094259171437786
35                 11.853304917305040    4.390154982411246
36                  8.660399796303766    3.417390829191746
37                 14.524769351354600    2.230519926799618

[69 rows x 12 columns]
```

Let's summarize the result so that we can better analyse it.

```
num_overall = df_coint.shape[0]
profitable_first499 =
df_coint[df_coint["estimated_return_first_499intervals"] > 0]
num_profitable_first499 = profitable_first499.count()[0]

profitable_499_599 =
profitable_first499[profitable_first499["estimated_return_next_100inte
rvals"] > 0]
num_profitable_499_599 = profitable_499_599.count()[0]

profitable_599_699 =
profitable_499_599[profitable_499_599["estimated_return_next_100-
200intervals"] > 0]
num_profitable_599_699 = profitable_599_699.count()[0]

profitable_699_799 =
profitable_599_699[profitable_599_699["estimated_return_next_200-
300intervals"] > 0]
num_profitable_699_799 = profitable_699_799.count()[0]

profitable_799_899 =
profitable_699_799[profitable_699_799["estimated_return_next_300-
400intervals"] > 0]
num_profitable_799_899 = profitable_799_899.count()[0]

profitable_899_999 =
profitable_799_899[profitable_799_899["estimated_return_next_400-
500intervals"] > 0]
num_profitable_899_999 = profitable_899_999.count()[0]

print(f"overall number is {num_overall}, among them,
{num_profitable_first499} pairs have positive return, with the ratio
of {round(num_profitable_first499/num_overall * 100, 2)}%")
num_profitable_list = [num_profitable_499_599, num_profitable_599_699,
num_profitable_699_799, num_profitable_799_899,
num_profitable_899_999]
for i in range(5):
    print(f"For all pairs that meet the condition above,
{num_profitable_list[i]} pairs have positive return in the next 100
intervals, with the ratio of {round(num_profitable_list[i]/num_overall
* 100, 2)}%")
```

```
overall number is 69, among them, 69 pairs have positive return, with
the ratio of 100.0%
For all pairs that meet the condition above, 66 pairs have positive
return in the next 100 intervals, with the ratio of 95.65%
For all pairs that meet the condition above, 59 pairs have positive
return in the next 100 intervals, with the ratio of 85.51%
For all pairs that meet the condition above, 50 pairs have positive
return in the next 100 intervals, with the ratio of 72.46%
```

```
For all pairs that meet the condition above, 41 pairs have positive
return in the next 100 intervals, with the ratio of 59.42%
For all pairs that meet the condition above, 34 pairs have positive
return in the next 100 intervals, with the ratio of 49.28%
```

Summary

Now we can draw some interesting observations from the results above:

- It's quite intuitive that with the time passing by, the profitable pairs are no longer profitable. But luckily, we can still have a sweet spot of win rate for the next 100 15m intervals, whcih is a quite optimistic result compared to my hypothesis.
- for the trading period, the next 100 - 200 intervals has a win rate lower than 50%, which indicates that it is not a good choice for getting stable return compared to the next 0-100 intervals. Therefore, next 0-100 intervals should be the trading period that I'm focusing on.

Answer to Q1: Yes, the cointegration feature would last for the next 100 intervals with the win rate over 100% under current circumstances.

Next step

- I want to play around with the trading oppotunities and estimated return in the first 499 intervals, to see if I can find some interesting relations of these two parameter with the win rate.
- Next I want to check the ways of calculating the z-score, such as by adding a scrolling window so that the calculation of z-score should not be static, to see if I can make the return looks much better.
- Finally, I should try other intervals like 30m, 10m, 60m, 90m, 120m, to see which one can bring me a better return by comparing them.

Q2: Trading oppotunities, estimated returns in the past, or anything else.....? What should be the best signal to judge on?

Should I use z-score window to calculate dynamic z-score instead of the static one?

Step 1: Optimize the selection of pairs by filtering the trading oppotunities in the first 499 intervals.

```
# Discover the relationship of trading oppotunities and winrate
# df_coint.nlargest(50, "trading_oppotunities_first_499intervals")
for i in [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]:
    top_oppotunities_positive_return_first499 =
profitable_first499.nlargest(i,
"trading_oppotunities_first_499intervals")
    top_oppotunities_next100_positive_return =
top_oppotunities_positive_return_first499[top_oppotunities_positive_re
turn_first499["estimated_return_next_100intervals"] > 0]
    num_top_oppotunities_next100_positive_return =
top_oppotunities_next100_positive_return.count()[0]
    print(f"For the pairs with the top {i} number of trading
oppotunities, the win rate for next 100 intervals is
{round((num_top_oppotunities_next100_positive_return / i) * 100, 2)}")
```

```
For the pairs with the top 10 number of trading oppotunities, the win
rate for next 100 intervals is 100.0
For the pairs with the top 20 number of trading oppotunities, the win
rate for next 100 intervals is 100.0
For the pairs with the top 30 number of trading oppotunities, the win
rate for next 100 intervals is 100.0
For the pairs with the top 40 number of trading oppotunities, the win
rate for next 100 intervals is 97.5
For the pairs with the top 50 number of trading oppotunities, the win
rate for next 100 intervals is 94.0
For the pairs with the top 60 number of trading oppotunities, the win
rate for next 100 intervals is 95.0
For the pairs with the top 70 number of trading oppotunities, the win
rate for next 100 intervals is 94.29
For the pairs with the top 80 number of trading oppotunities, the win
rate for next 100 intervals is 82.5
For the pairs with the top 90 number of trading oppotunities, the win
rate for next 100 intervals is 73.33
```

```
For the pairs with the top 100 number of trading oppotunities, the win rate for next 100 intervals is 66.0
```

Summary for trading oppotunities

- From the results above, it is noted that the win rate has a descending trend when the number of top trading oppotunities are in [10, 60], but the trend becomes ambiguious after the number reach 60.

Conclusion

- We can draw a conclusion that we can get a better win rate when we choose a trading pair with the higher number of trading oppotunities.

Step 2: Discover the relationship between win rate and estimated return

```python
# Discover the relationship between win rate and estimated return
for i in [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]:
    top_return_positive_return_first499 = profitable_first499.nlargest(i, "estimated_return_first_499intervals")
    top_return_next100_positive_return = top_return_positive_return_first499[top_return_positive_return_first499["estimated_return_next_100intervals"] > 0]
    num_top_return_next100_positive_return = top_return_next100_positive_return.count()[0]
    print(f"For the pairs with the top {i} returns, the win rate for next 100 intervals is {round((num_top_return_next100_positive_return / i) * 100, 2)}")

profitable_first499.nlargest(10, "estimated_return_first_499intervals")
```

```
For the pairs with the top 10 returns, the win rate for next 100 intervals is 90.0
For the pairs with the top 20 returns, the win rate for next 100 intervals is 95.0
For the pairs with the top 30 returns, the win rate for next 100 intervals is 96.67
For the pairs with the top 40 returns, the win rate for next 100 intervals is 97.5
For the pairs with the top 50 returns, the win rate for next 100 intervals is 96.0
For the pairs with the top 60 returns, the win rate for next 100 intervals is 96.67
For the pairs with the top 70 returns, the win rate for next 100 intervals is 94.29
For the pairs with the top 80 returns, the win rate for next 100 intervals is 82.5
For the pairs with the top 90 returns, the win rate for next 100
```

```
intervals is 73.33
For the pairs with the top 100 returns, the win rate for next 100
intervals is 66.0

           sym_1     sym_2          hedge_ratio    initial_intercept  \
61       MTLUSDT  DYDXUSDT   -0.610901279785415    0.985359636458149
63   1000XECUSDT  GALAUSDT   -0.066747042132137    0.078454712345471
67   1000XECUSDT   LDOUSDT    0.000618958598972    0.006849997951645
59      ANKRUSDT  DYDXUSDT   -0.005792564361732   -0.009795688895947
66   1000XECUSDT   INJUSDT    0.002108501980313    0.074419658193398
60      STMXUSDT  DYDXUSDT   -0.000928804049622   -0.010792129419748
51     OCEANUSDT  DYDXUSDT   -0.043225085404783    0.496935599093314
39       MKRUSDT   MTLUSDT  975.300726842769109  797.385812863650244
34     WAVESUSDT  NEARUSDT   -0.457068107952115    2.325860339982962
65   1000XECUSDT    OPUSDT    0.043546571495451    0.059702949611561

    trading_oppotunities_first_499intervals  \
61                                        3
63                                        2
67                                        2
59                                        3
66                                        2
60                                        4
51                                        3
39                                        6
34                                        3
65                                        1

    estimated_return_first_499intervals
estimated_return_next_100intervals  \
61                   803.012713940487515
2.052066190265482
63                   140.457506733074581
38.149985307449811
67                   126.019319375777997
48.046765868937854
59                    90.271139760920065
5.183067122878750
66                    86.633684060121325
38.633330058860764
60                    82.965099947630478
2.182193486010883
51                    59.665714029937888
0.000000000000000
39                    50.429893747983201
21.647464329672161
34                    48.953341568449744
25.289854153473300
65                    45.950352097147274
29.844819189508286
```

```
    estimated_return_next_100-200intervals  \
61                       5.938366858710729
63                     -46.808818939909244
67                     -75.602007367077476
59                       3.918642816405311
66                       6.582698676120422
60                       3.191839646365737
51                       4.810371883817773
39                      21.652123399082249
34                       8.740847321395497
65                     -50.657869671365432

    estimated_return_next_200-300intervals  \
61                       6.561709790373469
63                     -67.021582090134899
67                      12.617507531177731
59                       0.000000000000000
66                    -549.202082302996359
60                       0.000000000000000
51                       0.000000000000000
39                       4.756814976628632
34                      13.603318989761492
65                      84.456937604009894

    estimated_return_next_300-400intervals  \
61                       0.000000000000000
63                      28.488714399423362
67                      60.658250194643173
59                       1.162074992561151
66                      50.298849543817788
60                       6.614754854906669
51                      13.639263734033491
39                      40.717033530976693
34                     -13.447373166982867
65                      84.036365387204427

    estimated_return_next_400-500intervals          total_score
61                      19.095028497191645  803.012713940487515
63                     -78.696406365079397  140.457506733074581
67                       9.943805627148880  126.019319375777997
59                       7.770679064567187   90.271139760920065
66                    -169.703788150292013   86.633684060121325
60                      11.727581122014353   82.965099947630478
51                      29.183200701443173   59.665714029937888
39                       6.686921487697806   50.429893747983201
34                      13.058886503173685   48.953341568449744
65                      81.238394508427007   45.950352097147274
```

Summary

- From the result above, we can see that for the best trading pairs with the most returns in the first 499 intervals, their performance for the next 100 intervals can not be guaranteed.
- But for the top 80 - 100 returns, we can observe a clear descending value for the win rate, which means that we should not select the trading pair with a bad performance in the trainning period.
  - How about picking the trading pairs that perform the worst. Would they perform well in the next several intervals?
  - How about change the number of intervals to choose my trading pairs, instead of 499, let's say 200, 250, 300, 350, 400, 450?

```
# Discover the relationship between win rate and estimated return
for i in [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]:
    least_return_positive_return_first499 = df_coint.nsmallest(i, 
"estimated_return_first_499intervals")
    least_return_next100_positive_return = 
least_return_positive_return_first499[least_return_positive_return_fir
st499["estimated_return_next_100intervals"] > 0]
    num_least_return_next100_positive_return = 
least_return_next100_positive_return.count()[0]
    print(f"For the pairs with the least {i} returns, the win rate for 
next 100 intervals is {round((num_least_return_next100_positive_return 
/ i) * 100, 2)}")

df_coint.nsmallest(10, "estimated_return_first_499intervals")
```

```
For the pairs with the least 10 returns, the win rate for next 100 
intervals is 90.0
For the pairs with the least 20 returns, the win rate for next 100 
intervals is 95.0
For the pairs with the least 30 returns, the win rate for next 100 
intervals is 93.33
For the pairs with the least 40 returns, the win rate for next 100 
intervals is 95.0
For the pairs with the least 50 returns, the win rate for next 100 
intervals is 96.0
For the pairs with the least 60 returns, the win rate for next 100 
intervals is 96.67
For the pairs with the least 70 returns, the win rate for next 100 
intervals is 94.29
For the pairs with the least 80 returns, the win rate for next 100 
intervals is 82.5
For the pairs with the least 90 returns, the win rate for next 100 
intervals is 73.33
For the pairs with the least 100 returns, the win rate for next 100 
intervals is 66.0
```

```
         sym_1        sym_2           hedge_ratio    initial_intercept
\
37  WAVESUSDT      MTLUSDT     0.222955361309421    2.558582914709757

36  WAVESUSDT    OCEANUSDT     2.223270695160325   -0.472757519638569

35  WAVESUSDT      FILUSDT     0.040594084039041    1.898014060652207

54   SANDUSDT     LINAUSDT    16.693308247832295    0.302705371179136

19    ADAUSDT     AVAXUSDT     0.018143061708651    0.206598788250214

24    BNBUSDT      FLMUSDT  1469.784523906476807  227.318620799207849

42   AVAXUSDT     GALAUSDT   345.060535151595559   10.368362410874246

40    DOTUSDT     AVAXUSDT     0.444361488352624    3.640650226981381

52  OCEANUSDT     CTSIUSDT     1.504197380882251    0.377678397072173

55   SANDUSDT  1000SHIBUSDT    69.163344413476509    0.015829230781767


    trading_oppotunities_first_499intervals  \
37                                        1
36                                        1
35                                        2
54                                        2
19                                        3
24                                        5
42                                        3
40                                        6
52                                        5
55                                        3

    estimated_return_first_499intervals
estimated_return_next_100intervals  \
37                    2.230519926799618
28.950058978190260
36                    3.417390829191746
36.914305626008201
35                    4.390154982411246
14.466761494262723
54                    5.094259171437786
5.464718463100875
19                    7.276977563202547
3.707577643396315
24                    7.310643651029071
3.224886620051587
42                    9.355264396991668
3.267453320665680
```

```
40                    9.356951073219955
0.981547057697468
52                   10.065943300235526
0.000000000000000
55                   10.225310279691254
3.030419424226046

    estimated_return_next_100-200intervals  \
37                        12.564374413623348
36                         8.733402498864645
35                         0.000000000000000
54                         4.256900734630615
19                         4.450355358203478
24                         2.083890389299581
42                         0.000000000000000
40                         7.782790428814779
52                         5.355710414580503
55                         3.320335969321573

    estimated_return_next_200-300intervals  \
37                        18.129540527758245
36                        40.274726823515792
35                        16.360506772172258
54                         3.501341976939190
19                         8.504398834365446
24                         4.821234246306526
42                         6.521569852971095
40                         6.021518136218837
52                         0.000000000000000
55                         4.872629836747552

    estimated_return_next_300-400intervals  \
37                        37.302128184076942
36                       -93.388791461744503
35                       -18.261925475433817
54                         3.787071864426335
19                         3.030187245955257
24                         0.000000000000000
42                         4.992460932369570
40                         2.796696216802659
52                        16.020362512893236
55                         2.892534945181081

    estimated_return_next_400-500intervals         total_score
37                        14.524769351354600   2.230519926799618
36                         8.660399796303766   3.417390829191746
35                        11.853304917305040   4.390154982411246
54                         6.860873587350145   5.094259171437786
19                         0.000000000000000   7.276977563202547
24                         2.365102986166917   7.310643651029071
```

```
42                        13.292877734038122   9.355264396991668
40                         3.515543103873365   9.356951073219955
52                        18.471103380525975  10.065943300235526
55                         0.000000000000000  10.225310279691254

# Discover the relationship between win rate and estimated return
for i in [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]:
    least_return_positive_return_first499 =
profitable_first499.nsmallest(i,
"estimated_return_first_499intervals")
    least_return_next100_positive_return =
least_return_positive_return_first499[least_return_positive_return_fir
st499["estimated_return_next_100intervals"] > 0]
    num_least_return_next100_positive_return =
least_return_next100_positive_return.count()[0]
    print(f"For the pairs with the least {i} returns, the win rate for
next 100 intervals is {round((num_least_return_next100_positive_return
/ i) * 100, 2)}")
profitable_first499.nsmallest(10,
"estimated_return_first_499intervals")

For the pairs with the least 10 returns, the win rate for next 100
intervals is 90.0
For the pairs with the least 20 returns, the win rate for next 100
intervals is 95.0
For the pairs with the least 30 returns, the win rate for next 100
intervals is 93.33
For the pairs with the least 40 returns, the win rate for next 100
intervals is 95.0
For the pairs with the least 50 returns, the win rate for next 100
intervals is 96.0
For the pairs with the least 60 returns, the win rate for next 100
intervals is 96.67
For the pairs with the least 70 returns, the win rate for next 100
intervals is 94.29
For the pairs with the least 80 returns, the win rate for next 100
intervals is 82.5
For the pairs with the least 90 returns, the win rate for next 100
intervals is 73.33
For the pairs with the least 100 returns, the win rate for next 100
intervals is 66.0

        sym_1         sym_2            hedge_ratio    initial_intercept
\
37  WAVESUSDT       MTLUSDT     0.222955361309421    2.558582914709757

36  WAVESUSDT     OCEANUSDT     2.223270695160325   -0.472757519638569

35  WAVESUSDT       FILUSDT     0.040594084039041    1.898014060652207
```

```
54   SANDUSDT      LINAUSDT    16.693308247832295    0.302705371179136

19    ADAUSDT      AVAXUSDT     0.018143061708651    0.206598788250214

24    BNBUSDT       FLMUSDT  1469.784523906476807  227.318620799207849

42   AVAXUSDT      GALAUSDT   345.060535151595559   10.368362410874246

40    DOTUSDT      AVAXUSDT     0.444361488352624    3.640650226981381

52  OCEANUSDT      CTSIUSDT     1.504197380882251    0.377678397072173

55   SANDUSDT  1000SHIBUSDT    69.163344413476509    0.015829230781767


    trading_oppotunities_first_499intervals  \
37                                        1
36                                        1
35                                        2
54                                        2
19                                        3
24                                        5
42                                        3
40                                        6
52                                        5
55                                        3

    estimated_return_first_499intervals
estimated_return_next_100intervals  \
37                    2.230519926799618
28.950058978190260
36                    3.417390829191746
36.914305626008201
35                    4.390154982411246
14.466761494262723
54                    5.094259171437786
5.464718463100875
19                    7.276977563202547
3.707577643396315
24                    7.310643651029071
3.224886620051587
42                    9.355264396991668
3.267453320665680
40                    9.356951073219955
0.981547057697468
52                   10.065943300235526
0.000000000000000
55                   10.225310279691254
3.030419424226046
```

```
    estimated_return_next_100-200intervals  \
37                      12.564374413623348
36                       8.733402498864645
35                       0.000000000000000
54                       4.256900734630615
19                       4.450355358203478
24                       2.083890389299581
42                       0.000000000000000
40                       7.782790428814779
52                       5.355710414580503
55                       3.320335969321573

    estimated_return_next_200-300intervals  \
37                      18.129540527758245
36                      40.274726823515792
35                      16.360506772172258
54                       3.501341976939190
19                       8.504398834365446
24                       4.821234246306526
42                       6.521569852971095
40                       6.021518136218837
52                       0.000000000000000
55                       4.872629836747552

    estimated_return_next_300-400intervals  \
37                      37.302128184076942
36                     -93.388791461744503
35                     -18.261925475433817
54                       3.787071864426335
19                       3.030187245955257
24                       0.000000000000000
42                       4.992460932369570
40                       2.796696216802659
52                      16.020362512893236
55                       2.892534945181081

    estimated_return_next_400-500intervals         total_score
37                      14.524769351354600   2.230519926799618
36                       8.660399796303766   3.417390829191746
35                      11.853304917305040   4.390154982411246
54                       6.860873587350145   5.094259171437786
19                       0.000000000000000   7.276977563202547
24                       2.365102986166917   7.310643651029071
42                      13.292877734038122   9.355264396991668
40                       3.515543103873365   9.356951073219955
52                      18.471103380525975  10.065943300235526
55                       0.000000000000000  10.225310279691254
```

Summary on least return trading pairs for the first 499 intervals

- From the results above, there is no clue for a relationship to be shown.
- **But I don't think this is a good trading signal to choose trading pair, as you can see, even when they can earn money in the following terms. The profits of them are not attractive.**

Answer to Q2:

1. The more trading oppotunities, the better

2. The trading pair with bad performance on estimated return in the trainning period should not be selected

3. Use the investable_value/(price_1 + price_2 * hedge_ratio) to denote the total revenue

Q3: How should I calculate z-score?

```
# Encapsulate the process

# Calculate Z-Score
def calculate_zscore_window(spread: list, window) -> list:
    """
    Calculates the Z-Score of a given spread.

    Args:
        spread (list): A list of values representing the spread.

    Returns:
        list: A list containing the Z-Score values.
    """
    data = pd.DataFrame(spread)
    rolling = data.rolling(window=window)
    m = rolling.mean()
    s = rolling.std()
    z_score = (data - m) / s
    z_score[0][:(window-1)] = 0
    return z_score[0].tolist()

#
test_price = price_history_dict["BTCUSDT"][:30]
calculate_zscore_window(test_price, 21)
```

```
[0.0,
 0.0,
```

```
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 0.0,
 2.3389074206831677,
 1.9114332688494051,
 1.0352946186501941,
 2.1639111487659406,
 1.4189848092712096,
 2.884808689508618,
 1.8280986880707872,
 2.487587642840475,
 2.4052927849538546,
 2.346054928038812]
```

```
experiment_different_z_score_calculation =
profitable_first499[["sym_1", "sym_2",
"hedge_ratio","initial_intercept","trading_oppotunities_first_499inter
vals", "estimated_return_first_499intervals",
"estimated_return_next_100intervals"]].copy()
windows = [15, 21, 26, 30, 35, 40, 45, 50, 60, 70, 80, 90, 100, 150,
200]
for window in windows:


    estimated_return_list = []
    estimated_oppotunities_list = []

    for i in range(69):
        symbol_1 = experiment_different_z_score_calculation["sym_1"]
[i]
        symbol_2 = experiment_different_z_score_calculation["sym_2"]
[i]

        symbol_1_series = price_history_dict[symbol_1][499 - window:
599 - window]
```

```
        symbol_2_series = price_history_dict[symbol_2][499 - window:
599 - window]

        hedge_ratio_new =
experiment_different_z_score_calculation["hedge_ratio"][i]
        initial_intercept =
experiment_different_z_score_calculation["initial_intercept"][i]

        trading_oppotunities, estimated_return =
calculate_trading_estimated_oppotunities_return(symbol_1_series,
symbol_2_series, hedge_ratio_new ,initial_intercept,

0.8, func_z_score=calculate_zscore_window, num_window=window)
        estimated_return_list.append(estimated_return)
        estimated_oppotunities_list.append(trading_oppotunities)


experiment_different_z_score_calculation[f"window_{window}_estimated_r
eturn"] = estimated_return_list

experiment_different_z_score_calculation[f"window_{window}_trading_opp
otunities"] = estimated_oppotunities_list


experiment_different_z_score_calculation

          sym_1      sym_2          hedge_ratio  initial_intercept  \
61      MTLUSDT   DYDXUSDT  -0.610901279785415  0.985359636458149
63  1000XECUSDT   GALAUSDT  -0.066747042132137  0.078454712345471
67  1000XECUSDT    LDOUSDT   0.000618958598972  0.006849997951645
59     ANKRUSDT   DYDXUSDT  -0.005792564361732 -0.009795688895947
66  1000XECUSDT    INJUSDT   0.002108501980313  0.074419658193398
..          ...        ...                 ...                ...
19      ADAUSDT   AVAXUSDT   0.018143061708651  0.206598788250214
54     SANDUSDT   LINAUSDT  16.693308247832295  0.302705371179136
35    WAVESUSDT    FILUSDT   0.040594084039041  1.898014060652207
36    WAVESUSDT  OCEANUSDT   2.223270695160325 -0.472757519638569
37    WAVESUSDT    MTLUSDT   0.222955361309421  2.558582914709757

    trading_oppotunities_first_499intervals  \
61                                        3
63                                        2
67                                        2
59                                        3
66                                        2
..                                      ...
19                                        3
54                                        2
35                                        2
36                                        1
```

```
37                                        1

    estimated_return_first_499intervals
estimated_return_next_100intervals  \
61                   803.012713940487515
2.052066190265482
63                   140.457506733074581
38.149985307449811
67                   126.019319375777997
48.046765868937854
59                    90.271139760920065
5.183067122878750
66                    86.633684060121325
38.633330058860764
..                                   ...
...
19                     7.276977563202547
3.707577643396315
54                     5.094259171437786
5.464718463100875
35                     4.390154982411246
14.466761494262723
36                     3.417390829191746
36.914305626008201
37                     2.230519926799618
28.950058978190260

    window_15_estimated_return  window_15_trading_oppotunities  \
61           0.532838496052880                               2
63           1.158899803927632                               3
67           3.496084760209859                               4
59           0.597639331555339                               3
66           1.689151889445184                               4
..                         ...                             ...
19         -27.892181922539205                               2
54           3.013334911360105                               4
35          11.062701411348730                               4
36          16.179180597437764                               5
37          -1.788540598883266                               3

    window_21_estimated_return  ...  window_80_estimated_return  \
61           2.312914354921240  ...           2.312914354921240
63           0.321226084659823  ...           2.226107201071540
67          -0.727222391028967  ...           5.729788238192381
59          -0.037875617071305  ...           0.000000000000000
66           2.033012064586179  ...           0.000000000000000
..                         ...  ...                         ...
19           0.381730531082838  ...           0.000000000000000
54           8.219621042873268  ...           0.000000000000000
35          10.876457448649937  ...           0.000000000000000
```

```
36         25.252310267144686  ...          0.000000000000000
37          0.784407998743935  ...          0.000000000000000

    window_80_trading_oppotunities  window_90_estimated_return  \
61                               1            2.31291435492124
63                               1            2.22610720107154
67                               1            0.00000000000000
59                               0            0.00000000000000
66                               0            0.00000000000000
..                             ...                         ...
19                               0            0.00000000000000
54                               0            0.00000000000000
35                               0            0.00000000000000
36                               0            0.00000000000000
37                               0            0.00000000000000

    window_90_trading_oppotunities  window_100_estimated_return  \
61                               1                            0
63                               1                            0
67                               0                            0
59                               0                            0
66                               0                            0
..                             ...                          ...
19                               0                            0
54                               0                            0
35                               0                            0
36                               0                            0
37                               0                            0

    window_100_trading_oppotunities  window_150_estimated_return  \
61                                0                            0
63                                0                            0
67                                0                            0
59                                0                            0
66                                0                            0
..                              ...                          ...
19                                0                            0
54                                0                            0
35                                0                            0
36                                0                            0
37                                0                            0

    window_150_trading_oppotunities  window_200_estimated_return  \
61                                0                            0
63                                0                            0
67                                0                            0
59                                0                            0
66                                0                            0
..                              ...                          ...
19                                0                            0
```

```
54                                 0                              0
35                                 0                              0
36                                 0                              0
37                                 0                              0

    window_200_trading_oppotunities
61                                0
63                                0
67                                0
59                                0
66                                0
..                              ...
19                                0
54                                0
35                                0
36                                0
37                                0

[69 rows x 37 columns]
```

```
experiment_different_z_score_calculation =
profitable_first499[["sym_1", "sym_2",
"hedge_ratio","initial_intercept","trading_oppotunities_first_499inter
vals", "estimated_return_first_499intervals",
"estimated_return_next_100intervals"]].copy()
windows = [15, 21, 26, 30, 35, 40, 45, 50, 60, 70, 80, 90, 100, 150,
200, 350, 400, 450]
for window in windows:


    estimated_return_list = []
    estimated_oppotunities_list = []

    for i in range(69):
        symbol_1 = experiment_different_z_score_calculation["sym_1"]
[i]
        symbol_2 = experiment_different_z_score_calculation["sym_2"]
[i]

        symbol_1_series = price_history_dict[symbol_1][499 - window:
599]
        symbol_2_series = price_history_dict[symbol_2][499 - window:
599]

        hedge_ratio_new =
experiment_different_z_score_calculation["hedge_ratio"][i]
        initial_intercept =
experiment_different_z_score_calculation["initial_intercept"][i]

        trading_oppotunities, estimated_return =
```

```
calculate_trading_estimated_oppotunities_return(symbol_1_series,
symbol_2_series, hedge_ratio_new ,initial_intercept,

0.8, func_z_score=calculate_zscore_window, num_window=window)
        estimated_return_list.append(estimated_return)
        estimated_oppotunities_list.append(trading_oppotunities)


experiment_different_z_score_calculation[f"window_{window}_estimated_r
eturn"] = estimated_return_list

experiment_different_z_score_calculation[f"window_{window}_trading_opp
otunities"] = estimated_oppotunities_list


experiment_different_z_score_calculation

          sym_1      sym_2         hedge_ratio  initial_intercept  \
61      MTLUSDT   DYDXUSDT  -0.610901279785415  0.985359636458149
63  1000XECUSDT   GALAUSDT  -0.066747042132137  0.078454712345471
67  1000XECUSDT    LDOUSDT   0.000618958598972  0.006849997951645
59     ANKRUSDT   DYDXUSDT  -0.005792564361732 -0.009795688895947
66  1000XECUSDT    INJUSDT   0.002108501980313  0.074419658193398
..          ...        ...                 ...                ...
19      ADAUSDT   AVAXUSDT   0.018143061708651  0.206598788250214
54     SANDUSDT   LINAUSDT  16.693308247832295  0.302705371179136
35    WAVESUSDT    FILUSDT   0.040594084039041  1.898014060652207
36    WAVESUSDT  OCEANUSDT   2.223270695160325 -0.472757519638569
37    WAVESUSDT    MTLUSDT   0.222955361309421  2.558582914709757

    trading_oppotunities_first_499intervals  \
61                                        3
63                                        2
67                                        2
59                                        3
66                                        2
..                                      ...
19                                        3
54                                        2
35                                        2
36                                        1
37                                        1

    estimated_return_first_499intervals
estimated_return_next_100intervals  \
61                    803.012713940487515
2.052066190265482
63                    140.457506733074581
38.149985307449811
67                    126.019319375777997
```

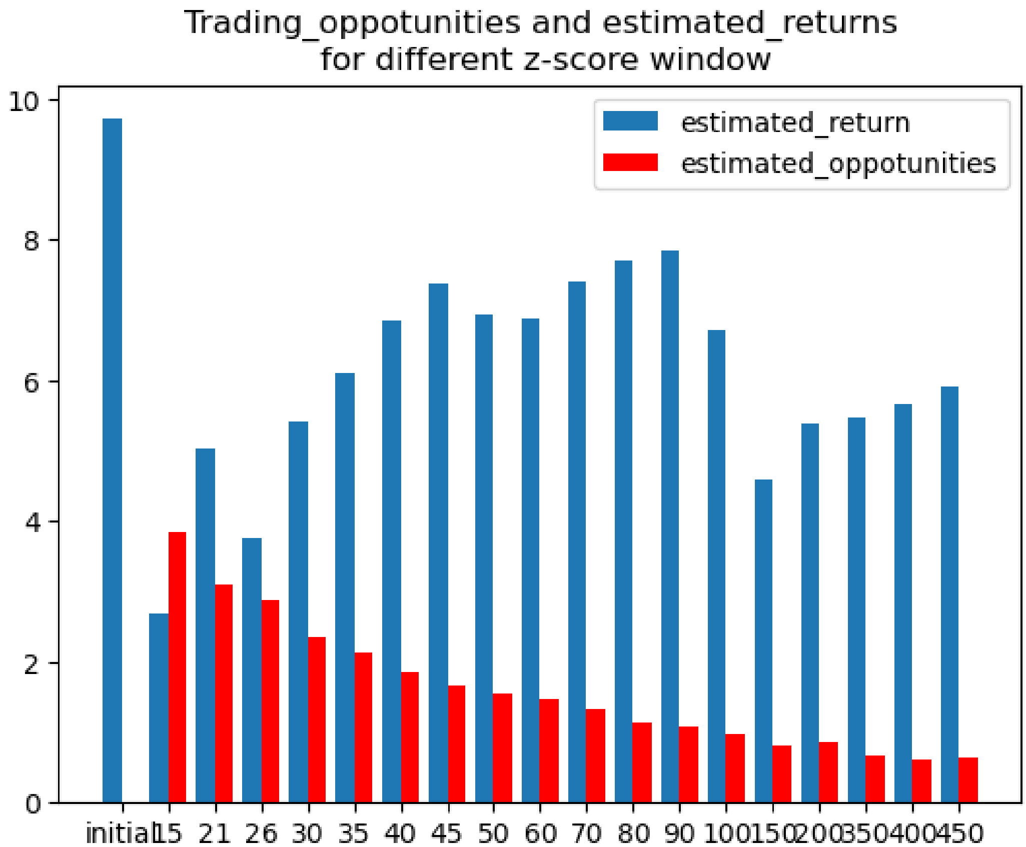
```
48.046765868937854
59                   90.271139760920065
5.183067122878750
66                   86.633684060121325
38.633330058860764
..                                  ...
...
19                    7.276977563202547
3.707577643396315
54                    5.094259171437786
5.464718463100875
35                    4.390154982411246
14.466761494262723
36                    3.417390829191746
36.914305626008201
37                    2.230519926799618
28.950058978190260

    window_15_estimated_return  window_15_trading_oppotunities  \
61           0.532838496052880                               2
63           1.585292175455999                               5
67           3.496084760209859                               4
59           0.597639331555339                               3
66           1.689151889445184                               4
..                         ...                             ...
19         -27.892181922539205                               2
54           3.013334911360105                               4
35          11.062701411348730                               4
36          16.179180597437764                               5
37          -3.063226709773232                               4

    window_21_estimated_return  ...  window_150_estimated_return  \
61           2.924742394087747  ...            3.754379708585649
63           1.232106538499898  ...            2.586601107479011
67          -0.727222391028967  ...            2.732359589136839
59          -0.024988933930017  ...            0.000000000000000
66           3.260125218237876  ...            0.000000000000000
..                         ...  ...                          ...
19         -26.217266368838764  ...            0.000000000000000
54           8.219621042873268  ...            0.000000000000000
35          10.876457448649937  ...            0.000000000000000
36          25.252310267144686  ...            0.000000000000000
37           0.784407998743935  ...            2.449427351112031

    window_150_trading_oppotunities  window_200_estimated_return  \
61                                1            3.008856686775997
63                                1            2.226107201071540
67                                1            5.204158855574263
59                                0            0.000000000000000
66                                0            0.000000000000000
```

```
..                              ...                          ...
19                                0             0.000000000000000
54                                0             0.000000000000000
35                                0             0.000000000000000
36                                0             0.000000000000000
37                                1             2.449427351112031

    window_200_trading_oppotunities  window_350_estimated_return  \
61                                1             2.312914354921240
63                                1             2.226107201071540
67                                1             0.000000000000000
59                                0             0.000000000000000
66                                0             0.000000000000000
..                              ...                          ...
19                                0             0.000000000000000
54                                0             0.000000000000000
35                                0             0.000000000000000
36                                0             0.000000000000000
37                                1             2.995613822852811

    window_350_trading_oppotunities  window_400_estimated_return  \
61                                1             2.312914354921240
63                                1             2.226107201071540
67                                0             0.000000000000000
59                                0             0.000000000000000
66                                0             0.000000000000000
..                              ...                          ...
19                                0             0.000000000000000
54                                0             0.000000000000000
35                                0             0.000000000000000
36                                0             0.000000000000000
37                                1             2.995613822852811

    window_400_trading_oppotunities  window_450_estimated_return  \
61                                1             2.312914354921240
63                                1             2.586601107479011
67                                0             0.000000000000000
59                                0             0.000000000000000
66                                0             0.000000000000000
..                              ...                          ...
19                                0             0.000000000000000
54                                0             0.000000000000000
35                                0             0.000000000000000
36                                0             0.000000000000000
37                                1             2.995613822852811

    window_450_trading_oppotunities
61                                1
63                                1
67                                0
```

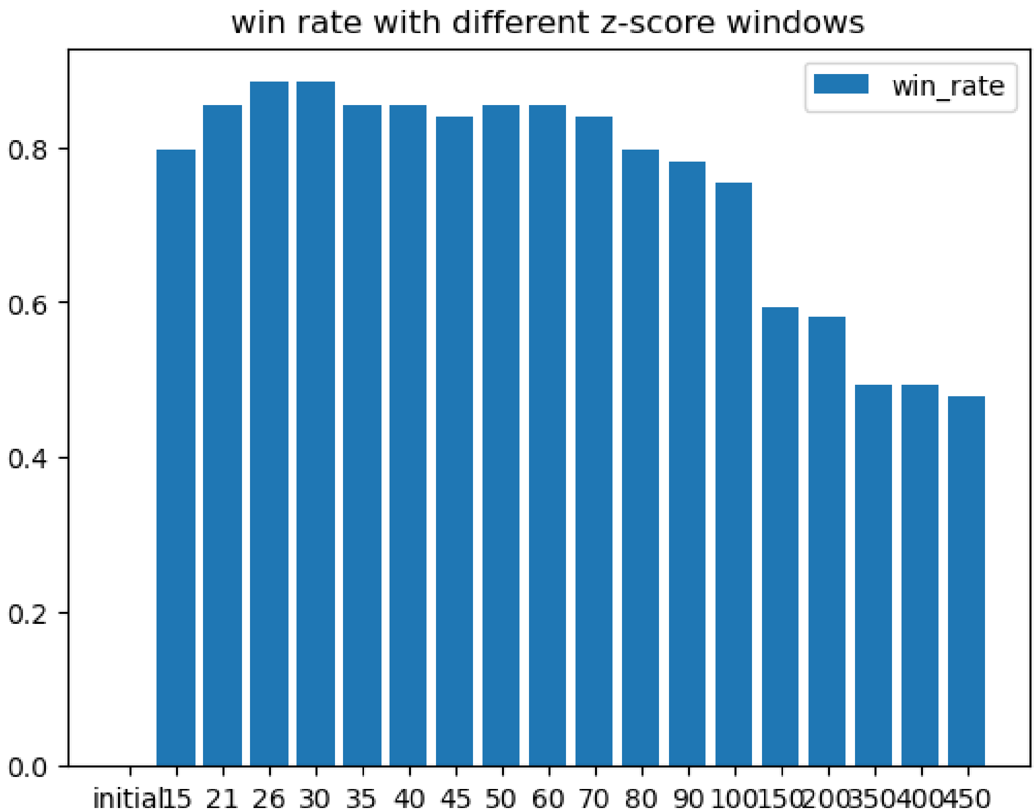
```
59                                  0
66                                  0
..                                ...
19                                  0
54                                  0
35                                  0
36                                  0
37                                  1

[69 rows x 43 columns]
```

Now we can start to analyse it a little bit.

```
import matplotlib.pyplot as plt

x = ["initial"]
y1 =
[experiment_different_z_score_calculation["estimated_return_next_100in
tervals"].mean()]
y2 = [0]



for window in windows:
    x.append(f"{window}")

y1.append(experiment_different_z_score_calculation[f"window_{window}_e
stimated_return"].mean())

y2.append(experiment_different_z_score_calculation[f"window_{window}_t
rading_oppotunities"].mean())

width = 0.4

position = np.arange(len(x))
fig, ax = plt.subplots()
p1 = ax.bar(position - width/2, y1, width=width, label =
"estimated_return")
p2 = ax.bar(position + width/2, y2, color = "r", width=width, label =
"estimated_oppotunities")
ax.set_xticks(position)
ax.set_xticklabels(x)
ax.legend(("estimated_return", "estimated_oppotunities"))
ax.set_title("Trading_oppotunities and estimated_returns\n for
different z-score window")

plt.show()
```




```
y_win_rate_list = [0]
for window in windows:

y_win_rate_list.append(experiment_different_z_score_calculation[experi
ment_different_z_score_calculation[f"window_{window}_estimated_return"
] > 0].shape[0] / experiment_different_z_score_calculation.shape[0])

# fig, ax = plt.subplots()
# p3 = ax.bar(y_win_rate_list, width=width, label = "win_rate")

plt.bar(x, y_win_rate_list, label = "win_rate")
plt.title("win rate with different z-score windows")
plt.legend()
plt.show()

experiment_different_z_score_calculation[experiment_different_z_score_
calculation[f"window_{45}_estimated_return"] > 0].shape[0] /
experiment_different_z_score_calculation.shape[0]
```




```
0.8405797101449275
```

Summary

- Considering win rate, estimated returns and trading oppotunities, z-score window of 45 should be the best choice.

Answer to Q3:

Use scrolling z-score window with the size of 45 should be the best choice. I like the number of ***46***, let's use this one. Haha

Q4: What is the best z-score threshod?

```
experiment_different_threshod_calculation = 
profitable_first499[["sym_1", "sym_2",
"hedge_ratio","initial_intercept","trading_oppotunities_first_499inter
vals", "estimated_return_first_499intervals",
"estimated_return_next_100intervals"]].copy()
z_score_threshod_list = [0.01 ,0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9,
1.0, 1.1, 1.2, 1.3]
for z_score_threshod in z_score_threshod_list:
```

```
    window = 46
    estimated_return_list = []
    estimated_oppotunities_list = []

    for i in range(69):
        symbol_1 = experiment_different_threshod_calculation["sym_1"]
[i]
        symbol_2 = experiment_different_threshod_calculation["sym_2"]
[i]

        symbol_1_series = price_history_dict[symbol_1][499 - window:
599]
        symbol_2_series = price_history_dict[symbol_2][499 - window:
599]

        hedge_ratio_new =
experiment_different_threshod_calculation["hedge_ratio"][i]
        initial_intercept =
experiment_different_threshod_calculation["initial_intercept"][i]

        trading_oppotunities, estimated_return =
calculate_trading_estimated_oppotunities_return(symbol_1_series,
symbol_2_series, hedge_ratio_new ,initial_intercept,

threshod=z_score_threshod, func_z_score=calculate_zscore_window,
num_window=window)
        estimated_return_list.append(estimated_return)
        estimated_oppotunities_list.append(trading_oppotunities)


experiment_different_threshod_calculation[f"threshod_{z_score_threshod
}_estimated_return"] = estimated_return_list

experiment_different_threshod_calculation[f"threshod_{z_score_threshod
}_trading_oppotunities"] = estimated_oppotunities_list


experiment_different_threshod_calculation
```

```
          sym_1      sym_2          hedge_ratio  initial_intercept  \
61      MTLUSDT   DYDXUSDT  -0.610901279785415  0.985359636458149
63  1000XECUSDT   GALAUSDT  -0.066747042132137  0.078454712345471
67  1000XECUSDT    LDOUSDT   0.000618958598972  0.006849997951645
59     ANKRUSDT   DYDXUSDT  -0.005792564361732 -0.009795688895947
66  1000XECUSDT    INJUSDT   0.002108501980313  0.074419658193398
..          ...        ...                 ...                ...
19      ADAUSDT   AVAXUSDT   0.018143061708651  0.206598788250214
54     SANDUSDT   LINAUSDT  16.693308247832295  0.302705371179136
35    WAVESUSDT    FILUSDT   0.040594084039041  1.898014060652207
```

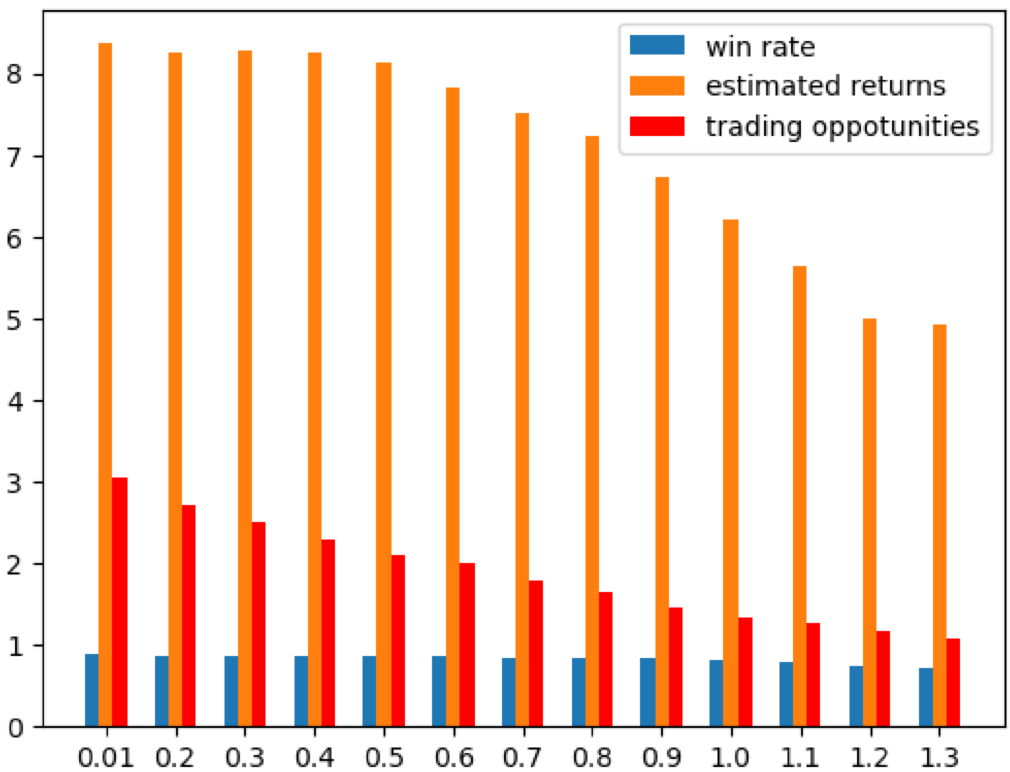
```
36    WAVESUSDT  OCEANUSDT   2.223270695160325 -0.472757519638569
37    WAVESUSDT    MTLUSDT   0.222955361309421  2.558582914709757

    trading_oppotunities_first_499intervals  \
61                                        3
63                                        2
67                                        2
59                                        3
66                                        2
..                                      ...
19                                        3
54                                        2
35                                        2
36                                        1
37                                        1

    estimated_return_first_499intervals
estimated_return_next_100intervals  \
61                    803.012713940487515
2.052066190265482
63                    140.457506733074581
38.149985307449811
67                    126.019319375777997
48.046765868937854
59                     90.271139760920065
5.183067122878750
66                     86.633684060121325
38.633330058860764
..                                    ...
...
19                      7.276977563202547
3.707577643396315
54                      5.094259171437786
5.464718463100875
35                      4.390154982411246
14.466761494262723
36                      3.417390829191746
36.914305626008201
37                      2.230519926799618
28.950058978190260

    threshod_0.01_estimated_return  threshod_0.01_trading_oppotunities
\
61               2.312914354921240                                   1

63               2.513367595607368                                   2

67               7.804333929648690                                   5

59               2.481354496495377                                   3
```

```
66                3.600197142249352                                      4
..                              ...                                    ...
19                0.000000000000000                                      0
54              -10.550736152636496                                      1
35               11.701727572090860                                      4
36               68.230843839763580                                      6
37                0.000000000000000                                      0

    threshod_0.2_estimated_return  ...  threshod_0.9_estimated_return  \
61               2.312914354921240  ...              2.312914354921240
63               2.500583100391401  ...              1.140576534327359
67               7.714753360192241  ...              4.629249315290621
59               2.180625830110339  ...              1.563149308935928
66               4.379693820968741  ...              1.918271783135784
..                             ...  ...                            ...
19               0.000000000000000  ...              0.000000000000000
54             -10.550736152636496  ...            -10.550736152636496
35              14.220749892852425  ...             30.856991292126118
36              66.462461103419585  ...             57.809128481627525
37               0.000000000000000  ...              0.000000000000000

    threshod_0.9_trading_oppotunities
threshod_1.0_estimated_return  \
61                                    1              2.312914354921240
63                                    1              1.140576534327359
67                                    1              5.007800529241218
59                                    2              1.563149308935928
66                                    2              1.399315265789089
```

```
..                                 ...                ...
19                                   0   0.000000000000000
54                                   1 -10.550736152636496
35                                   3  30.856991292126118
36                                   3  55.399518191316645
37                                   0   0.000000000000000

    threshod_1.0_trading_oppotunities
threshod_1.1_estimated_return  \
61                                   1   2.312914354921240
63                                   1   1.140576534327359
67                                   1   5.254932462055988
59                                   2   1.075065886089406
66                                   2   0.710499035116340
..                                 ...                ...
19                                   0   0.000000000000000
54                                   1 -10.550736152636496
35                                   3  34.258869718222755
36                                   3  52.158170245402985
37                                   0   0.000000000000000

    threshod_1.1_trading_oppotunities
threshod_1.2_estimated_return  \
61                                   1   2.312914354921240
63                                   1   1.140576534327359
67                                   1   5.254932462055988
59                                   2   1.139031968729213
66                                   1   0.420986038769043
..                                 ...                ...
```

```
19                                   0                  0.000000000000000
54                                   1                -10.550736152636496
35                                   3                 34.258869718222755
36                                   2                 52.158170245402985
37                                   0                  0.000000000000000

    threshod_1.2_trading_oppotunities
threshod_1.3_estimated_return  \
61                                   1                  2.312914354921240
63                                   1                  1.140576534327359
67                                   1                  5.254932462055988
59                                   2                  1.139031968729213
66                                   1                  0.420986038769043
..                                 ...                                ...
19                                   0                  0.000000000000000
54                                   1                -10.550736152636496
35                                   3                 37.153947460738770
36                                   2                 52.158170245402985
37                                   0                  0.000000000000000

    threshod_1.3_trading_oppotunities
61                                   1
63                                   1
67                                   1
59                                   2
66                                   1
..                                 ...
19                                   0
54                                   1
35                                   3
36                                   2
37                                   0

[69 rows x 33 columns]
```

```
x_threshod = []
y_winrate_threshod = []
y_estimated_returns_threshod = []
y_trading_oppotunities_threshod = []

for z_score_threshod in z_score_threshod_list:
    x_threshod.append(f"{z_score_threshod}")

y_winrate_threshod.append(experiment_different_threshod_calculation[ex
periment_different_threshod_calculation[f"threshod_{z_score_threshod}_
estimated_return"] > 0].shape[0] /
experiment_different_threshod_calculation.shape[0])

y_estimated_returns_threshod.append(experiment_different_threshod_calc
ulation[f"threshod_{z_score_threshod}_estimated_return"].mean())

y_trading_oppotunities_threshod.append(experiment_different_threshod_c
alculation[f"threshod_{z_score_threshod}_trading_oppotunities"].mean()
)

width = 0.2


position = np.arange(len(x_threshod))

plt.subplot()
plt.bar(position - width, y_winrate_threshod, width=width, label =
"win rate")
plt.bar(position, y_estimated_returns_threshod, width=width, label =
"estimated returns")
plt.bar(position + width, y_trading_oppotunities_threshod, color =
"r", width=width, label = "trading oppotunities")

plt.xticks(position, x_threshod)

plt.legend()
ax.set_title("Win rate, estimated_returns and trading_oppotunities \n
for different z-score threshod")

plt.show()
```


win rate
estimated returns
trading oppotunities
0
1
2
3
4
5
6
7
8
0.01
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
1.1
1.2
1.3

Answer to Q4:

[0.2, 0.5] is the best range from the graph. For the sake of safety and my intuition, I'd choose ***0.8***.

Q5: What is the best trainning period? (200, 250, 300, 350, 400, 450, 499)

Q6: What is the best trading intervals? (10m, 15m, 30m, 60m, 120m)

parameters: z-score window: 46, z-score threshod:0.8

critical factors: total win rate, average_returns_every_24h(total), average_trading_oppotunities_every_24h(total)

```
# first loop through intervals
# then get the trainning period
import time
test_intervals = ["1m", "3m", "5m", "15m", "30m", "1h", "2h", "4h",
"6h"]


test_price_history_dict = []

for test_interval in test_intervals:
    counts = 0
    price_history_dict = {}
    for sym in tradeable_symbols:
        price_history = binance_get_recent_close_price(sym,
interval=test_interval, limit=num_interval_limit)
        if len(price_history) == num_interval_limit: # make sure that
each symbol has the same amount of data
            price_history_dict[sym] = price_history
            counts += 1

    test_price_history_dict.append(price_history_dict)
    logger.info (f"{counts} items stored, {len(tradeable_symbols)-
counts}items not stored")
    time.sleep(5)
```

```
2023-07-12 05:34:45,206 - Rovers_3.0 - INFO - 46 items stored, 0items
not stored
2023-07-12 05:35:00,596 - Rovers_3.0 - INFO - 46 items stored, 0items
not stored
2023-07-12 05:35:17,096 - Rovers_3.0 - INFO - 46 items stored, 0items
not stored
2023-07-12 05:35:32,673 - Rovers_3.0 - INFO - 46 items stored, 0items
not stored
2023-07-12 05:35:48,355 - Rovers_3.0 - INFO - 46 items stored, 0items
not stored
2023-07-12 05:36:04,243 - Rovers_3.0 - INFO - 46 items stored, 0items
not stored
2023-07-12 05:36:20,025 - Rovers_3.0 - INFO - 46 items stored, 0items
not stored
2023-07-12 05:36:35,660 - Rovers_3.0 - INFO - 46 items stored, 0items
not stored
2023-07-12 05:36:50,949 - Rovers_3.0 - INFO - 46 items stored, 0items
not stored
```

```
for index, i in enumerate(test_intervals):
    filename = f"{i}_experiment_price_list.json"
    with open(filename, "w") as fp:
        json.dump(test_price_history_dict[index], fp, indent=4)
    logger.info(f"Prices saved successfully for
{test_intervals[index]}.")
```

```
2023-07-12 05:41:10,448 - Rovers_3.0 - INFO - Prices saved
successfully for 1m.
2023-07-12 05:41:10,548 - Rovers_3.0 - INFO - Prices saved
successfully for 3m.
2023-07-12 05:41:10,647 - Rovers_3.0 - INFO - Prices saved
successfully for 5m.
2023-07-12 05:41:10,745 - Rovers_3.0 - INFO - Prices saved
successfully for 15m.
2023-07-12 05:41:10,842 - Rovers_3.0 - INFO - Prices saved
successfully for 30m.
2023-07-12 05:41:10,950 - Rovers_3.0 - INFO - Prices saved
successfully for 1h.
2023-07-12 05:41:11,054 - Rovers_3.0 - INFO - Prices saved
successfully for 2h.
2023-07-12 05:41:11,263 - Rovers_3.0 - INFO - Prices saved
successfully for 4h.
2023-07-12 05:41:11,362 - Rovers_3.0 - INFO - Prices saved
successfully for 6h.
```

```
def get_profitable_cointegrated_pairs_test_trainning_period(prices,
trainning_period,  trigger_z_score_threshold = 0.8) -> str:
    # Loop through coins and check for co-integration
    coint_pair_list = []
```

```
    loop_count = 0
    for sym_1 in tradeable_symbols:
        loop_count += 1
        # Check each coin against the first (sym_1)
        for sym_2 in tradeable_symbols[loop_count:]:

            # Get close prices
            series_1_trainning = prices[sym_1][999 - 100 -
trainning_period - 26:(999-100)]
            series_2_trainning = prices[sym_2][999 - 100 -
trainning_period - 26:(999-100)]

            series_1_next_100 = prices[sym_1][899:]
            series_2_next_100 = prices[sym_2][899:]

            # Check for cointegration and add cointegrated pair
            coint_flag, p_value, hedge_ratio, initial_intercept =
calculate_cointegration_static(series_1_trainning, series_2_trainning)

            if coint_flag == 1:

                trading_oppotunities_trainning,
estimated_return_trainning =
calculate_trading_estimated_oppotunities_return(series_1_trainning,
series_2_trainning, hedge_ratio ,initial_intercept,

trigger_z_score_threshod, func_z_score=calculate_zscore_window,
num_window=26)

                trading_oppotunities_next_100,
estimated_return_next_100 =
calculate_trading_estimated_oppotunities_return(series_1_next_100,
series_2_next_100, hedge_ratio ,initial_intercept,

trigger_z_score_threshod, func_z_score=calculate_zscore_window,
num_window=26)


                coint_pair_list.append({
                    "sym_1": sym_1,
                    "sym_2": sym_2,
                    "hedge_ratio": hedge_ratio,
                    "initial_intercept": initial_intercept,
                    "trading_oppotunities_trainning":
trading_oppotunities_trainning,
                    "estimated_returns_trainning":
estimated_return_trainning,
                    "trading_oppotunities_next_100":
trading_oppotunities_next_100,
                    "estimated_returns_next_100":
```

```
estimated_return_next_100,
                })

    # Output results and rank all the trading pairs
    df_coint = pd.DataFrame(coint_pair_list)
    # add the total score column
    df_coint = df_coint[df_coint["estimated_returns_trainning"] > 0]
    df_coint = df_coint.sort_values("estimated_returns_trainning",
ascending=False)

    return df_coint

# test
get_profitable_cointegrated_pairs_test_trainning_period(test_price_his
tory_dict[3], 200)
```

```
           sym_1        sym_2            hedge_ratio
initial_intercept  \
14     WAVESUSDT     AAVEUSDT     -0.027144901334292
4.091041378327500
16     WAVESUSDT    OCEANUSDT     -4.654019339688396
3.900264795768598
35  1000SHIBUSDT  1000XECUSDT     -0.080285767781429
0.010345076106306
18     WAVESUSDT      MTLUSDT      1.127501334723919
0.525638807516285
12      KAVAUSDT      MTLUSDT      0.281839174963871
0.547297810808951
24     STORJUSDT      APTUSDT      0.028542654194872
0.137170339492452
26       FLMUSDT     ANKRUSDT      2.011716206915794
0.018978283746582
34      LINAUSDT      INJUSDT      0.001040947896923
0.004909295612715
22       SOLUSDT     NEARUSDT     11.656486530138055
5.761557391858250
29      AAVEUSDT      APEUSDT     29.214890734935377
14.813881631968728
8        BNBUSDT     KAVAUSDT    204.029760227759283
42.773226750390542
10      DOGEUSDT     AVAXUSDT      0.001766665799900
0.041164070086136
5       LINKUSDT     ANKRUSDT    155.590938719921837
2.574216575091202
9        BNBUSDT      STGUSDT    218.380828160309278
104.845863639781484
25       FTMUSDT     MASKUSDT      0.083692106758890      -
0.016312267398498
32      ANKRUSDT     STMXUSDT      3.142018361722969
0.010334898802429
```

```
36      DYDXUSDT      LDOUSDT      0.755677991485296
0.377028760699363
4        XRPUSDT      STGUSDT      0.119318539088822
0.397501677706535
27       FLMUSDT      APTUSDT      0.006006222787884
0.022902157419484
33      ANKRUSDT      APTUSDT      0.002711772016637
0.003881470418298
1        ETHUSDT    MATICUSDT    437.779742641372820
1561.992200134429140
3        XRPUSDT     CTSIUSDT      0.249525195980046
0.431540474637795
30       FILUSDT      LDOUSDT      1.943075702307259
0.560670526178752
31      SANDUSDT      LDOUSDT      0.165940360645855
0.092357691426693
7       LINKUSDT      LDOUSDT      1.489124651410677
3.280981797584043
6       LINKUSDT     DYDXUSDT      1.893080797202820
2.680196987195404
0        BTCUSDT     CTSIUSDT  13809.483280618889694
28127.450150351225602
11      DOGEUSDT      FILUSDT      0.007359452817406
0.033561077887766
2        ETHUSDT     CTSIUSDT    874.760303279824029
1730.691515036116698
23       SOLUSDT     MASKUSDT      5.571804000955281
2.336384546379955

    trading_oppotunities_trainning  estimated_returns_trainning  \
14                               5           500.395351379670728
16                               4           193.027566777360192
35                              11            46.655598641930347
18                               6            32.315512997821514
12                               7            18.529504296218764
24                               4            12.561956685356050
26                               8            12.229388390285154
34                               5             9.513641873308352
22                               7             9.400594536976328
29                               8             9.156370341923209
8                                5             8.902114939441194
10                               8             8.553311742753422
5                                8             8.148557084725327
9                                5             8.125748529971521
25                               8             7.045521819005574
32                               7             6.672773313348008
36                              10             6.404290549068461
4                               11             5.992604550689332
27                               9             4.587724034861558
```

```
33                                7          4.365505569841560
1                                 6          4.161426963015969
3                                 9          3.113527993448878
30                                6          2.971163024749253
31                                6          2.849406031167870
7                                 9          2.808868159408530
6                                 9          2.805809692516683
0                                 6          2.315599561753519
11                                7          2.040826078433121
2                                 8          1.780869293059517
23                                7          1.540129310864035
```

```
    trading_oppotunities_next_100  estimated_returns_next_100
14                              3         223.049331862096864
16                              2          30.577236904217809
35                              2           1.215410614924124
18                              2           4.407838145107315
12                              2           1.076115459648221
24                              2           5.660096502816888
26                              1          -9.095985200907656
34                              2           1.189803087991315
22                              3          -3.182388901155567
29                              2          -6.138350990232068
8                               2          -0.240463195133254
10                              1           0.660908990276920
5                               3           2.711922752182168
9                               2           2.245603011003527
25                              3           1.020036214411710
32                              1           0.409073314875396
36                              2          -1.898134020581155
4                               2           1.288034709728973
27                              2         -12.617817866260022
33                              2           0.639970459407351
1                               3           1.473554042612010
3                               2           2.294536919169141
30                              4           1.784308279231703
31                              4           0.377162343977116
7                               3          -0.894044976182447
6                               4           2.262032315977230
0                               3           2.101607752513387
11                              2           0.668462395707433
2                               2           3.063100674783619
23                              1          -4.997894910672692
```

```python
def
profitable_cointegrated_pairs_test_intervals_trainning_period_result(i
nterval, trainning_period, data: pd.DataFrame):
    logger.info(f"Deriving data for{interval}_{trainning_period}")
    ave_trading_oppotunities_next_100 =
data["trading_oppotunities_next_100"].mean()
```

```
    ave_returns_next_100 = data["estimated_returns_next_100"].mean()
    win_rate_next_100 = data[data["estimated_returns_next_100"] >
0].shape[0] /data.shape[0]
    return {f"{interval}_{trainning_period}":
[ave_trading_oppotunities_next_100, ave_returns_next_100,
win_rate_next_100]}

# test
# data_1 =
profitable_cointegrated_pairs_test_intervals_trainning_period_result("
1m",200,get_profitable_cointegrated_pairs_test_trainning_period(test_p
rice_history_dict[1], 400))
# data_2 =
profitable_cointegrated_pairs_test_intervals_trainning_period_result("
1h",200,get_profitable_cointegrated_pairs_test_trainning_period(test_p
rice_history_dict[2], 400))
# data_4 =
profitable_cointegrated_pairs_test_intervals_trainning_period_result("
30m",200,get_profitable_cointegrated_pairs_test_trainning_period(test_
price_history_dict[3], 400))
# data_3 =
profitable_cointegrated_pairs_test_intervals_trainning_period_result("
6h",200,get_profitable_cointegrated_pairs_test_trainning_period(test_p
rice_history_dict[4], 400))

# data_1, data_2, data_3, data_4
# test_intervals
```

```
2023-07-12 07:14:50,574 - Rovers_3.0 - INFO - Deriving data for1m_200
2023-07-12 07:14:55,719 - Rovers_3.0 - INFO - Deriving data for1h_200
2023-07-12 07:15:01,076 - Rovers_3.0 - INFO - Deriving data for30m_200
2023-07-12 07:15:06,231 - Rovers_3.0 - INFO - Deriving data for6h_200
```

```
({'1m_200': [2.642857142857143, 0.8576456412480626,
0.7678571428571429]},
 {'1h_200': [2.659340659340659, 0.9683367389845042,
0.8241758241758241]},
 {'6h_200': [2.2653061224489797, -1.4275832603143481,
0.6632653061224489]},
 {'30m_200': [2.2319391634980987, 2.1962256405619636,
0.8593155893536122]})
```

```
# next, loop through trainning period, compare the results in recent
100 intervals
test_trainning_period = [200, 250, 300, 350, 400, 450, 499]
test_intervals = ["1m", "3m", "5m", "15m", "30m", "1h", "2h", "4h",
"6h"]

result_list = []
for index, interval in enumerate(test_intervals):
```

```
    for period in test_trainning_period:
        data =
get_profitable_cointegrated_pairs_test_trainning_period(test_price_his
tory_dict[index], period)

result_list.append(profitable_cointegrated_pairs_test_intervals_trainn
ing_period_result(interval, period, data))

result_list
```

```
2023-07-12 07:17:41,389 - Rovers_3.0 - INFO - Deriving data for1m_200
2023-07-12 07:17:45,101 - Rovers_3.0 - INFO - Deriving data for1m_250
2023-07-12 07:17:49,238 - Rovers_3.0 - INFO - Deriving data for1m_300
2023-07-12 07:17:53,737 - Rovers_3.0 - INFO - Deriving data for1m_350
2023-07-12 07:17:58,954 - Rovers_3.0 - INFO - Deriving data for1m_400
2023-07-12 07:18:06,750 - Rovers_3.0 - INFO - Deriving data for1m_450
2023-07-12 07:18:27,333 - Rovers_3.0 - INFO - Deriving data for1m_499
2023-07-12 07:18:30,541 - Rovers_3.0 - INFO - Deriving data for3m_200
2023-07-12 07:18:34,068 - Rovers_3.0 - INFO - Deriving data for3m_250
2023-07-12 07:18:38,336 - Rovers_3.0 - INFO - Deriving data for3m_300
2023-07-12 07:18:43,007 - Rovers_3.0 - INFO - Deriving data for3m_350
2023-07-12 07:18:48,207 - Rovers_3.0 - INFO - Deriving data for3m_400
2023-07-12 07:18:55,924 - Rovers_3.0 - INFO - Deriving data for3m_450
2023-07-12 07:19:18,774 - Rovers_3.0 - INFO - Deriving data for3m_499
2023-07-12 07:19:21,992 - Rovers_3.0 - INFO - Deriving data for5m_200
2023-07-12 07:19:25,532 - Rovers_3.0 - INFO - Deriving data for5m_250
2023-07-12 07:19:29,623 - Rovers_3.0 - INFO - Deriving data for5m_300
2023-07-12 07:19:34,102 - Rovers_3.0 - INFO - Deriving data for5m_350
2023-07-12 07:19:39,281 - Rovers_3.0 - INFO - Deriving data for5m_400
2023-07-12 07:19:47,140 - Rovers_3.0 - INFO - Deriving data for5m_450
2023-07-12 07:20:09,362 - Rovers_3.0 - INFO - Deriving data for5m_499
2023-07-12 07:20:12,470 - Rovers_3.0 - INFO - Deriving data for15m_200
2023-07-12 07:20:15,988 - Rovers_3.0 - INFO - Deriving data for15m_250
2023-07-12 07:20:20,157 - Rovers_3.0 - INFO - Deriving data for15m_300
2023-07-12 07:20:24,533 - Rovers_3.0 - INFO - Deriving data for15m_350
2023-07-12 07:20:29,889 - Rovers_3.0 - INFO - Deriving data for15m_400
2023-07-12 07:20:37,529 - Rovers_3.0 - INFO - Deriving data for15m_450
2023-07-12 07:20:58,779 - Rovers_3.0 - INFO - Deriving data for15m_499
2023-07-12 07:21:01,976 - Rovers_3.0 - INFO - Deriving data for30m_200
2023-07-12 07:21:05,604 - Rovers_3.0 - INFO - Deriving data for30m_250
2023-07-12 07:21:09,856 - Rovers_3.0 - INFO - Deriving data for30m_300
2023-07-12 07:21:14,348 - Rovers_3.0 - INFO - Deriving data for30m_350
2023-07-12 07:21:19,501 - Rovers_3.0 - INFO - Deriving data for30m_400
2023-07-12 07:21:27,290 - Rovers_3.0 - INFO - Deriving data for30m_450
2023-07-12 07:21:47,150 - Rovers_3.0 - INFO - Deriving data for30m_499
2023-07-12 07:21:50,386 - Rovers_3.0 - INFO - Deriving data for1h_200
2023-07-12 07:21:53,835 - Rovers_3.0 - INFO - Deriving data for1h_250
2023-07-12 07:21:57,848 - Rovers_3.0 - INFO - Deriving data for1h_300
2023-07-12 07:22:02,167 - Rovers_3.0 - INFO - Deriving data for1h_350
2023-07-12 07:22:07,093 - Rovers_3.0 - INFO - Deriving data for1h_400
```

```
2023-07-12 07:22:13,732 - Rovers_3.0 - INFO - Deriving data for1h_450
2023-07-12 07:22:24,356 - Rovers_3.0 - INFO - Deriving data for1h_499
2023-07-12 07:22:27,378 - Rovers_3.0 - INFO - Deriving data for2h_200
2023-07-12 07:22:30,805 - Rovers_3.0 - INFO - Deriving data for2h_250
2023-07-12 07:22:34,787 - Rovers_3.0 - INFO - Deriving data for2h_300
2023-07-12 07:22:39,023 - Rovers_3.0 - INFO - Deriving data for2h_350
2023-07-12 07:22:43,984 - Rovers_3.0 - INFO - Deriving data for2h_400
2023-07-12 07:22:50,657 - Rovers_3.0 - INFO - Deriving data for2h_450
2023-07-12 07:23:04,428 - Rovers_3.0 - INFO - Deriving data for2h_499
2023-07-12 07:23:07,509 - Rovers_3.0 - INFO - Deriving data for4h_200
2023-07-12 07:23:10,970 - Rovers_3.0 - INFO - Deriving data for4h_250
2023-07-12 07:23:15,059 - Rovers_3.0 - INFO - Deriving data for4h_300
2023-07-12 07:23:19,437 - Rovers_3.0 - INFO - Deriving data for4h_350
2023-07-12 07:23:24,578 - Rovers_3.0 - INFO - Deriving data for4h_400
2023-07-12 07:23:31,981 - Rovers_3.0 - INFO - Deriving data for4h_450
2023-07-12 07:23:48,976 - Rovers_3.0 - INFO - Deriving data for4h_499
2023-07-12 07:23:52,176 - Rovers_3.0 - INFO - Deriving data for6h_200
2023-07-12 07:23:55,736 - Rovers_3.0 - INFO - Deriving data for6h_250
2023-07-12 07:23:59,872 - Rovers_3.0 - INFO - Deriving data for6h_300
2023-07-12 07:24:04,455 - Rovers_3.0 - INFO - Deriving data for6h_350
2023-07-12 07:24:09,557 - Rovers_3.0 - INFO - Deriving data for6h_400
2023-07-12 07:24:16,874 - Rovers_3.0 - INFO - Deriving data for6h_450
2023-07-12 07:24:33,873 - Rovers_3.0 - INFO - Deriving data for6h_499
```


```
[{'1m_200': [2.4823529411764707, 0.3277860865370313,
0.8235294117647058]},
 {'1m_250': [2.338345864661654, 0.41213204970910494,
0.8045112781954887]},
 {'1m_300': [2.4285714285714284, 0.12093554841316073,
0.7285714285714285]},
 {'1m_350': [2.159090909090909, 0.017965097561346812,
0.5681818181818182]},
 {'1m_400': [2.1739130434782608, 0.3600826354098599,
0.8260869565217391]},
 {'1m_450': [2.328358208955224, 0.5090450015800227,
0.8656716417910447]},
 {'1m_499': [2.367816091954023, 0.4815006628788652,
0.8390804597701149]},
 {'3m_200': [1.8264462809917354, 0.39827780332954776,
0.6611570247933884]},
 {'3m_250': [1.8888888888888888, 0.3446420435619983,
0.5873015873015873]},
 {'3m_300': [1.7588235294117647, 0.24352994849647636,
0.5411764705882353]},
 {'3m_350': [2.3148936170212764, 0.5186668928433399,
0.6510638297872341]},
 {'3m_400': [2.642857142857143, 0.8576456412480626,
0.7678571428571429]},
 {'3m_450': [2.689655172413793, 0.8103425420544295,
0.7586206896551724]},
```

```
 {'3m_499': [3.185567010309278, 1.0404504620787047,
0.8556701030927835]},
 {'5m_200': [2.293103448275862, 1.0071824022888316,
0.8103448275862069]},
 {'5m_250': [2.826923076923077, 2.533028683743895,
0.8461538461538461]},
 {'5m_300': [2.5555555555555554, 4.325450563812059,
0.7777777777777778]},
 {'5m_350': [2.3125, 1.329673975977576, 0.8541666666666666]},
 {'5m_400': [2.659340659340659, 0.9683367389845042,
0.8241758241758241]},
 {'5m_450': [2.5234375, 2.2809711538733515, 0.90625]},
 {'5m_499': [2.7567567567567566, 1.7567349393957563,
0.9054054054054054]},
 {'15m_200': [2.3, 8.370368889717843, 0.7333333333333333]},
 {'15m_250': [2.096774193548387, 0.229339342908743,
0.8387096774193549]},
 {'15m_300': [2.5813953488372094, 1.6365610354330675,
0.7441860465116279]},
 {'15m_350': [2.263157894736842, 1.960218724125733,
0.8947368421052632]},
 {'15m_400': [2.2319391634980987, 2.1962256405619636,
0.8593155893536122]},
 {'15m_450': [2.369565217391304, 2.9286928472382434,
0.782608695652174]},
 {'15m_499': [2.1640625, 2.607491293295336, 0.8046875]},
 {'30m_200': [2.4065934065934065, -0.5171148859381978,
0.7252747252747253]},
 {'30m_250': [2.0, 1.1798015580921883, 0.631578947368421]},
 {'30m_300': [1.9452054794520548, -12.858470901227363,
0.684931506849315]},
 {'30m_350': [2.3222222222222224, -0.6816910403899489,
0.6888888888888889]},
 {'30m_400': [2.2653061224489797, -1.4275832603143481,
0.6632653061224489]},
 {'30m_450': [2.1757575757575756, -1.3538205693889886,
0.5636363636363636]},
 {'30m_499': [2.239130434782609, 0.30193807365617426,
0.5217391304347826]},
 {'1h_200': [2.4214876033057853, 0.9382472018874632,
0.5950413223140496]},
 {'1h_250': [2.6534653465346536, 2.5804673199687893,
0.693069306930693]},
 {'1h_300': [2.506172839506173, 2.9640391776161312,
0.7160493827160493]},
 {'1h_350': [2.8877551020408165, 3.231997395185808,
0.7551020408163265]},
 {'1h_400': [2.3783783783783785, 2.9892874421038513,
0.7837837837837838]},
```

```
 {'1h_450': [2.08, 2.7737059431663584, 0.76]},
 {'1h_499': [1.9565217391304348, 5.267007468289501,
0.8260869565217391]},
 {'2h_200': [1.671641791044776, 1.4819216540054834,
0.6567164179104478]},
 {'2h_250': [1.943661971830986, 1.2234553269568897,
0.6619718309859155]},
 {'2h_300': [1.8955223880597014, 3.33141080911733,
0.746268656716418]},
 {'2h_350': [1.763157894736842, 2.383455397937404,
0.7631578947368421]},
 {'2h_400': [1.8666666666666667, 4.538595130836395,
0.8666666666666667]},
 {'2h_450': [2.0, 4.731703566848906, 0.8095238095238095]},
 {'2h_499': [1.9428571428571428, 4.324079657569119, 0.8]},
 {'4h_200': [1.8666666666666667, -6.421630478440533,
0.4888888888888889]},
 {'4h_250': [1.8780487804878048, -13.067979524635048,
0.43902439024390244]},
 {'4h_300': [1.6538461538461537, -13.94180158931624,
0.38461538461538464]},
 {'4h_350': [1.6285714285714286, -11.135747807648327,
0.5142857142857142]},
 {'4h_400': [1.798165137614679, -7.412681046438285,
0.5779816513761468]},
 {'4h_450': [1.7179487179487178, -12.476287836025614, 0.5]},
 {'4h_499': [1.8487394957983194, -0.5346673438441002,
0.6050420168067226]},
 {'6h_200': [1.7168141592920354, -67.68130397768157,
0.4247787610619469]},
 {'6h_250': [1.7769230769230768, -31.491594953011756,
0.5461538461538461]},
 {'6h_300': [1.7972027972027973, -67.06982542478674,
0.5524475524475524]},
 {'6h_350': [2.067873303167421, -14.90467080237557,
0.6108597285067874]},
 {'6h_400': [1.8376068376068375, -52.808842885879024,
0.47863247863247865]},
 {'6h_450': [1.8914285714285715, -31.705343496972464,
0.5771428571428572]},
 {'6h_499': [1.768421052631579, -31.284744054826863,
0.5421052631578948]}]
```


```python
with
open("profitable_cointegrated_pairs_test_intervals_trainning_period_re
sult", "w") as fp:
        json.dump(result_list, fp, indent=4)

pip install seaborn
```

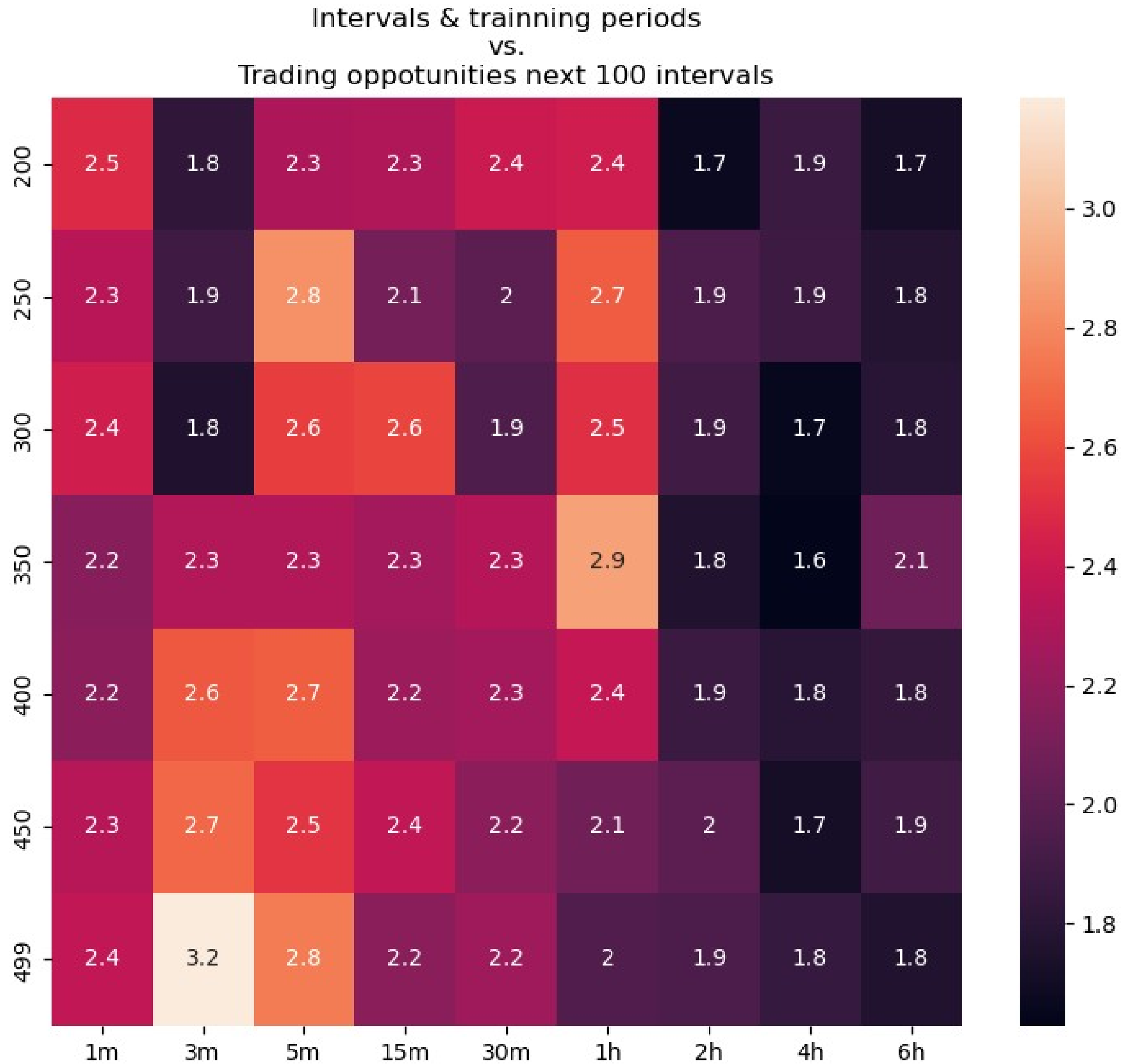
```
Collecting seaborn
  Downloading seaborn-0.12.2-py3-none-any.whl (293 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 293.3/293.3 kB 793.8 kB/s eta
0:00:00a 0:00:01
ent already satisfied: numpy!=1.24.0,>=1.17 in
/Users/haowu/anaconda3/envs/Pybit-trade/lib/python3.11/site-packages
(from seaborn) (1.24.3)
Requirement already satisfied: pandas>=0.25 in
/Users/haowu/anaconda3/envs/Pybit-trade/lib/python3.11/site-packages
(from seaborn) (1.5.3)
Requirement already satisfied: matplotlib!=3.6.1,>=3.1 in
/Users/haowu/anaconda3/envs/Pybit-trade/lib/python3.11/site-packages
(from seaborn) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in
/Users/haowu/anaconda3/envs/Pybit-trade/lib/python3.11/site-packages
(from matplotlib!=3.6.1,>=3.1->seaborn) (1.0.5)
Requirement already satisfied: cycler>=0.10 in
/Users/haowu/anaconda3/envs/Pybit-trade/lib/python3.11/site-packages
(from matplotlib!=3.6.1,>=3.1->seaborn) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in
/Users/haowu/anaconda3/envs/Pybit-trade/lib/python3.11/site-packages
(from matplotlib!=3.6.1,>=3.1->seaborn) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
/Users/haowu/anaconda3/envs/Pybit-trade/lib/python3.11/site-packages
(from matplotlib!=3.6.1,>=3.1->seaborn) (1.4.4)
Requirement already satisfied: packaging>=20.0 in
/Users/haowu/anaconda3/envs/Pybit-trade/lib/python3.11/site-packages
(from matplotlib!=3.6.1,>=3.1->seaborn) (23.0)
Requirement already satisfied: pillow>=6.2.0 in
/Users/haowu/anaconda3/envs/Pybit-trade/lib/python3.11/site-packages
(from matplotlib!=3.6.1,>=3.1->seaborn) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/Users/haowu/anaconda3/envs/Pybit-trade/lib/python3.11/site-packages
(from matplotlib!=3.6.1,>=3.1->seaborn) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in
/Users/haowu/anaconda3/envs/Pybit-trade/lib/python3.11/site-packages
(from matplotlib!=3.6.1,>=3.1->seaborn) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in
/Users/haowu/anaconda3/envs/Pybit-trade/lib/python3.11/site-packages
(from pandas>=0.25->seaborn) (2022.7)
Requirement already satisfied: six>=1.5 in
/Users/haowu/anaconda3/envs/Pybit-trade/lib/python3.11/site-packages
(from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.1->seaborn) (1.16.0)
Installing collected packages: seaborn
Successfully installed seaborn-0.12.2
Note: you may need to restart the kernel to use updated packages.
```

```python
list(result_list[0].values())[0]
```

```
[2.4823529411764707, 0.3277860865370313, 0.8235294117647058]
```

```python
# heatmap for trading oppotunities
import seaborn as sn

test_trainning_period = [200, 250, 300, 350, 400, 450, 499]
test_intervals = ["1m", "3m", "5m", "15m", "30m", "1h", "2h", "4h",
"6h"]
to_hour_coeficient = [60, 20, 12, 4, 2, 1, 1/2, 1/4, 1/6]

# f"{interval}_{trainning_period}":
[ave_trading_oppotunities_next_100, ave_returns_next_100,
win_rate_next_100]}
def turn_result_list_to_pd(parameter: int, result_list = result_list):
    pd_result = pd.DataFrame()
    target_list = []
    for index, interval in enumerate(test_intervals):
        for i, period in  enumerate(test_trainning_period):
            if parameter == 1:
                target_list.append(list(result_list[i + index *
len(test_trainning_period)].values())[0][parameter] *
to_hour_coeficient[index])
            else:
                target_list.append(list(result_list[i + index *
len(test_trainning_period)].values())[0][parameter])
        pd_result[f"{interval}"] = target_list
        target_list = []
    return pd_result

plt.rcParams["figure.figsize"] = [7.5, 20]
plt.rcParams["figure.autolayout"] = True

fig, (ax1, ax2, ax3) = plt.subplots(3,1)


fig.subplots_adjust(wspace=0.01)

# trading oppotunities
pd_ave_trading_oppotunities_next_100 = turn_result_list_to_pd(0,
result_list)
pd_ave_trading_oppotunities_next_100.index = test_trainning_period

# average returns
pd_ave_returns_next_100_each_hr = turn_result_list_to_pd(1,
result_list)
pd_ave_returns_next_100_each_hr.index = test_trainning_period

# winrate
win_rate_next_100 = turn_result_list_to_pd(2, result_list)
win_rate_next_100.index = test_trainning_period

#heatmap
hm_pd_ave_trading_oppotunities_next_100 =
```

```
(sn.heatmap(data=pd_ave_trading_oppotunities_next_100, annot=True,
ax=ax1))
hm_pd_ave_returns_next_100_each_hr =
sn.heatmap(data=pd_ave_returns_next_100_each_hr, annot=True,ax=ax2)
hm_win_rate_next_100 = sn.heatmap(data=win_rate_next_100,
annot=True,ax=ax3)

hm_pd_ave_trading_oppotunities_next_100.set_title("Intervals &
trainning periods\nvs.\nTrading oppotunities next 100 intervals")
hm_pd_ave_returns_next_100_each_hr.set_title("Intervals & trainning
periods\nvs.\nAverage returns next 100 intervals each hour")
hm_win_rate_next_100.set_title("Intervals & trainning periods\nvs.\
nWin rate next 100 intervals")

plt.show()
```


Intervals & trainning periods
vs.
Trading oppotunities next 100 intervals
200
250
300
350
400
450
499
1m
3m
5m
15m
30m
1h
2h
4h
6h
2.5 1.8 2.3 2.3 2.4 2.4 1.7 1.9 1.7
2.3 1.9 2.8 2.1 2 2.7 1.9 1.9 1.8
2.4 1.8 2.6 2.6 1.9 2.5 1.9 1.7 1.8
2.2 2.3 2.3 2.3 2.3 2.9 1.8 1.6 2.1
2.2 2.6 2.7 2.2 2.3 2.4 1.9 1.8 1.8
2.3 2.7 2.5 2.4 2.2 2.1 2 1.7 1.9
2.4 3.2 2.8 2.2 2.2 2 1.9 1.8 1.8
3.0
2.8
2.6
2.4
2.2
2.0
1.8


Intervals & trainning periods
vs.
Average returns next 100 intervals each hour
200
250
300
20 8 12 33 -1 0.94 0.74 -1.6 -11
25 6.9 30 0.92 2.4 2.6 0.61 -3.3 -5.2
7.3 4.9 52 6.5 -26 3 1.7 -3.5 -11
50
40
30
20

Summary

I just want to be short so that I can go to sleep:

- Taking the features shown in the three heatmaps above, we can observe that the sweet spot located in the place where interval should be [5m, 15m], and the trainning period should be [350 400]

Answer to Q5 and Q6:

1. Pick the trainning period of 350

2. Pick the intervals of 15m

Conclusions:

Choosing trading symbols:

-
    a. The feature of cointegration has a strong trend to persist after 100 intervals of the trainning period
-
    a. Pick the trading pair with large number of trading oppotunities during trainning period
-
    a. The trading pair being selected ***must*** be profitable during trainning period
-
    a. Use the investable_value/(price_1 + price_2 * hedge_ratio) to denote the possible revenue

Parameters

- z-score window: 26
- a-score threshod: 0.8
- trainning period: 350
- interval: 15m