MATHEMATICAL TRADING STRATEGIES

Final Assignment Submission

Introduction

In **Assignment-4**, I implemented and tested a trading strategy using the indicator **Keltner Channel** and the chart pattern - **Head and Shoulders** using the metrics. I used **Infosys** data to optimise the strategy and then tested it on **Wipro**, a company in roughly the same industrial sector and market segment.

Results and Metrics

I optimised the parameters three times, using o**nly Keltner Channel**, using **only Head and Shoulder pattern** and then **combined** them for signal generation. The results are given in the table below -

	Combined	**Keltner Channel**	**Head and Shoulders**
Cumulative returns	0.856494292	1.96013546	1.00959
Sharpe Ratio	0.000377	0.006543	0.000083
Max Drawdown	-0.585276	-0.31272	-0.030940

I found it better to use **Keltner Channel** only.

Procedure and Logic Involved

Keltner Channel

- I implemented Keltner channel with a variable period and a fixed multiplier 2 with the function `calculate_keltner_channel(df, period=20, multiplier=2`.
- I then created function `generate_kc_signals(df, period=20)` to generate Buy/Sell Signals for the data.

Logic for signal generation :

- The code iterates through each row in the DataFrame and compares the closing price `df['Close'][i]` with the upper and lower bands of the Keltner Channel.
- If the closing price is above the upper band, it sets the `kc_signal` to `-1`, indicating a sell signal.
- If the closing price is below the lower band, it sets the `kc_signal` to `1`, indicating a buy signal.
- Otherwise, if the closing price is within the bounds of the Keltner Channel, it sets the `kc_signal` to `0`, indicating no signal.

Logic for optimisation:

- After creating a function to generate signals I tried to optimise the parameter `period` for keltner channel by **plotting the metrics separately** for all possible values and trying to pick manually the parameter that optimises all of the metrics simultaneously.

Head and Shoulders

- I created a function `generate_hns_signals(df, shoulderperiod=5, headperiod=10)` to generate Buy/Sell Signals for the data.

Logic for signal generation :

- The function calculates three SMAs: `left_shoulder` using the `High` prices with a period of `shoulderperiod`, `head` using the `Low` prices with a period of `headperiod`, and `right_shoulder` using the `High` prices with a period of the same `shoulderperiod`.

- Next, it creates a Boolean Series called `pattern` by checking if the left and right shoulder SMAs are both less than the head SMA.

- In the subsequent loop, it iterates through each row in the `pattern` and checks if the pattern is present and has completed (i.e., `pattern[i]` is `True` and `i` is greater than or equal to `2`).

- If the conditions are met, it further checks the closing prices at the current and previous two time steps. If the conditions for a head and shoulders pattern are satisfied (rising prices followed by lower prices and vice versa), it assigns a value of `1` for a buy signal or `-1` for a sell signal to the `'hns_signal'` column. If none of the conditions are met, it assigns a value of `0` for no signal.

- Finally, if the pattern is not present or has not completed, it assigns a value of `0` to the `'hns_signal'` column.

Logic for optimisation:

- In the loop, function iterates through each row in the DataFrame and checks two conditions:
-
    1. If the closing price is above the upper band of the Keltner Channel `df['Close'][i] > upper_kc[i]`
-
    2. If the pattern is present and has completed (i.e., `pattern[i]` is True and `i` is greater than or equal to `2`).
- If either of these conditions is met, it assigns a value of `-1` for a sell signal to the `'combined_signal'` column.
- If none of the conditions are met, it assigns a value of `0` for no signal.

Logic for optimisation:

- The code utilizes a nested loop structure to iterate over the values of k`eltner_channel_period`, `head_period`, and `shoulder_period`. For each combination of these parameters, the code performs the following steps:

-

1. Calls the `generate_combined_signals` function with the current values of `kcp`, `hp`, and `sp` to generate trade signals for the given parameter combination.

2. Calculates the **strategy returns** by multiplying the `'combined_signal'` column with the percentage change in the `'Close'` price shifted by one day.

3. Computes the **cumulative returns** by taking the cumulative product of (1 + strategy returns).

4. Calculates the **maximum drawdown** by finding the minimum value of ((cumulative returns / cumulative maximum returns) - 1).

5. Computes the **Sharpe ratio** by taking the mean of strategy returns divided by the standard deviation of the percentage change in 'Close' price.

6. Compares the strategy returns with the previous best returns stored in `best_metrics['returns']`. If the current returns are higher, it updates the `best_metrics` dictionary with the current returns, cumulative returns, maximum drawdown, and Sharpe ratio. It also updates the `best_params` dictionary with the current parameter values.

Limitations and Possible Enhancements

Limitations

- The optimization process may lead to overfitting the data if the parameter values are selected solely based on the historical performance.
- For the above reasons, The optimized parameter values may not remain optimal in different market conditions or time periods.
- As the optimisation used infosys data only, the results of parameter optimization are sensitive to the input data used.

Possible Enhancements

- I have a relatively simple signal generation and optimistaion procedure. I could explore and incorporate more advanced signal generation techniques beyond simple moving averages, such as other technical indicators, machine learning algorithms, or pattern recognition algorithms.
- I could perform sensitivity analysis on the optimized parameters to evaluate their robustness and stability and assess how variations in the parameter values affect the strategy's performance under different market conditions. This can help identify optimal parameter ranges that are less sensitive to market fluctuations.