‹ Return to "AI for Trading" in the classroom

DISCUSS ON STUDENT HUB ›

Smart Beta Portfolio and Portfolio Optimization

REVIEW | CODE REVIEW | HISTORY

Meets Specifications

Great job submitting a well-implemented project👍

If you want to get an overview of some real-world ETFs and dig in deeper in their statistics, you can find them here.
I left you comments below.

Congratulations and all the best for the rest of your learning journey!

Rate this project
☆☆☆☆☆

Part 1: Smart Beta Portfolio

✓ **The function `generate_dollar_volume_weights` computes dollar volume weights.**

Each value is appropriately normalized by the sum total dollar-volume traded for these stocks each day and the dollar-volume is correctly calculated. Great!

✓ **The function `calculate_dividend_weights` computes dividend weights.**

The function correctly calculates the weights for the dividend ETF based on cumulative dividends. Well done, this is a tricky part to implement!

✓ **The function `generate_returns` computes returns.**

Nice job calculating the returns.

✓ **The function `generate_weighted_returns` computes weighted returns.**

Well done!

✓ **The function `calculate_cumulative_returns` computes cumulative returns.**

Nice job calculating the cumulative returns.

✓ **The function `tracking_error` computes tracking error.**

Great work calculating annualized tracking error between the ETF and benchmark in tracking_error function. This completes your portfolio, now it is time to optimize it

Part 2: Portfolio Optimization

✓ **The function `get_covariance_returns` computes covariance of the returns.**

Excellent work calculating the covariance and replacing the `NaN` values correctly with zero!

✓ **The function `get_optimal_weights` computes optimal weights.**

Awesome

Correct optimization of weights here using `cvxpy`. You correctly set the variable, the objective and the constraints for the optimization problem to solve it with `cvx.Problem(objective, constraints).solve()` and returning the `x.value`

✓ **The function `rebalance_portfolio` computes weights for each rebalancing of the portfolio.**

In this step, you correctly re-balanced the portfolio over time instead of using the same weights for the entire history. In the `rebalance_portfolio` function you made good use of your previous `get_optimal_weights` and `get_covariance_returns function`s after slicing your returns based on the `chunk_size` and the `shift_size`!

✓ **The function `get_portfolio_turnover` computes cost of all the rebalancing.**

Good work computing the cost of the rebalancing! Your portfolio turnover is *16.594080020340048*, which is the correct value and shows that you implemented the Smart Beta Portfolio correctly. This is a very impressive turnover, but remember that this is a simulated market cap weighted index with large dollar volume stocks only and cannot be compared 1:1 to the real-world ETF and portfolio performances.

DOWNLOAD PROJECT

RETURN TO PATH