UDACITY

Return to Classroom

DISCUSS ON STUDENT HUB

Backtesting

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Great job, you are ready to go! 💯 Clearly, you have acquired all the important concepts from this project. Congrats on completing this course! Hope you find all the previous sharing resources helpful.

Shift Daily Returns Data

The variable `frames` contains the data from `daily_return` and `data` correctly shifted.

Build Universe Based on Filters

The function `get_universe` correctly creates a stock universe by selecting only those companies that have a market capitalization of at least 1 billion dollars (1e9) OR that are in the previous day's holdings, even if on the current day, the company no longer meets the 1 billion dollar criteria.

They should use the `.copy()` attribute to create a copy of the data and they should drop the column containing the daily return from the universe dataframe.

The returned `universe` dataframe should have a shape of (2265, 93)

Factor covariance matrix

The function `diagonal_factor_cov` correctly creates the factor covariance matrix. The factor matrix must be scaled by (0.01**2). They must use the given `colnames` function to get the column names from `X` and use the statement 'covariance[date]' to get the covariances for the given `date`.

The returned factor covariance matrix should have shape (77, 77)

Alpha Combination

The function `get_B_alpha` correctly creates a matrix of alpha factors. They must use the given `get_formula` and `model_matrix` functions.

The returned `B_alpha` should be of type patsy.design_info.DesignMatrix and it should have shape (2265, 4). The 4 columns of this matrix should correspond to the 4 alpha factors chosen at the beginning, namely:

"USFASTD_1DREVRSL"
"USFASTD_EARNYILD"

"USFASTD_VALUE"
"USFASTD_SENTMT"

The function `get_alpha_vec` correctly creates a vector of alpha factors. To do this, they must add the rows of the Matrix of Alpha Factors and multiply the result by 1e-4.

The returned `alpha_vac` should have shape (2265,)

Objective function

The `obj_func(h)` function correctly implements the objective function. The equation of the objective function is given in the notebook.

Gradient

The `grad_func(h)` function correctly implements the gradient of the objective function. The equation of the gradient of the objective function is given in the notebook.

Optimize

The function `get_h_star` correctly optimizes the objective function using the following functions `obj_func`, `grad_func`, and `scipy.optimize.fmin_l_bfgs_b`.

The returned `h_star` should have shape (2265,)

Risk Exposures

The function `get_risk_exposures` correctly calculates the portfolio's risk exposures

The returned `risk_exposures` Pandas series should have shape (77,). The index of this Pandas Series should correspond to the risk factors such as 'USFASTD_AERODEF', 'USFASTD_AIRLINES', 'USFASTD_ALUMSTEL', ...

The function `get_portfolio_alpha_exposure` correctly calculates the portfolio's alpha exposures.

The returned `portfolio_alpha_exposure` Pandas series should have shape (4,). The index of this Pandas Series should should correspond to the 4 alpha factors chosen at the beginning, namely:

"USFASTD_1DREVRSL"
"USFASTD_EARNYILD"
"USFASTD_VALUE"
"USFASTD_SENTMT"

Transaction Costs

The function `get_total_transaction_costs` correctly calculates the total transaction costs according to the equation given in the notebook.

Profit-and-Loss (PnL) attribution

Correctly calculate the PnL attributed to the alpha factors, the PnL attributed to the risk factors, and attribution to cost.

To calculate the alpha and risk exposures they must use the provided `partial_dot_product` function

Good job! You correctly calculate all the related PnL.💯

```
df.at[dt,"attribution.alpha.pnl"] = partial_dot_product(fr, p["alpha.exposures"])
df.at[dt,"attribution.risk.pnl"] = partial_dot_product(fr, p["risk.exposures"])
df.at[dt,"attribution.cost"] = p["total.cost"]
```

Build portfolio characteristics

Correctly calculates the sum of long positions, short positions, net positions, gross market value, and amount of dollars traded.

Good job! You correctly calculate all the numbers👍

```
long = np.sum(h[h>0])
short = np.sum(h[h<0])
df.at[dt,"long"] = long
df.at[dt,"short"] = short
df.at[dt,"net"] = long + short # np.sum(h)
df.at[dt,"gmv"] = np.abs(long) + np.abs(short) # np.sum(np.abs(h))
df.at[dt,"traded"] = np.sum(np.abs(tradelist['h.opt'] - tradelist['h.opt.previous']))
```

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review

START