UDACITY

Return to Classroom

DISCUSS ON STUDENT HUB

Trading with Momentum

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Congratulations

Well Done! You have done a good job on this project! You understand all parts of this project well.

Here are some more links which you might want to checkout:

- Hypothesis Testing (P-Value Approach) : This will give idea about Hypothesis Testing with an example.
- How to Correctly Interpret P Values : This will help you understand p-values in a more human way.
- What a p-Value Tells You about Statistical Data : There is a very good example using p-value.
- Why Shrewd Experts "Fail to Reject the Null" Every Time : This article explain Null Hypothesis and *Fail to Reject*

More Resources but they will need more focussed attention :

1. Guide_to_Misinterpretations_of_Pvalues.pdf
2. The ASA s Statement on p Values Context Process and Purpose.pdf

Disclaimer

Please note that all the required changes / suggestions given in your project is given with the intent to improve your

skills. It is not meant to demotivate you or insult your work!

PS - You can always attach a picture of your error (if there is any) in the future in a folder and then mention it in the notes to the reviewer section while submitting it. That way we can help you in a better way. Take care of that in the future.

Market Data

The function `resample_prices` computes the monthly prices.

Well done!

Great job! You correctly used the Pandas resample method to resample prices to their monthly values, and the last method to get the last price in each bucket.

The function `compute_log_returns` computes the log returns from the prices.

Well done!

What you have implemented is correct!
But i am giving you here an wrong implementation conceptually but it will pass the tests too!

```
def compute_log_returns(prices):

    return np.log(prices/prices.shift(1))

project_tests.test_compute_log_returns(compute_log_returns)
```

Reason :
The function will pass because the output values are in the log form. However, the method used within the function to calculate the log_returns is not entirely correct, which yields incorrect outputs.

Retrieving the log variables output is correct. However, the function should yield the final log returns. Remember yielding returns are the difference between two variables, so in this case, it will be between the log prices and log shifted prices. Refer back to Lesson-7-Stock-Returns-Quize-Log-Returns for more information.

Also checkout `diff()` for an alternative implementation.

The function `shift_returns` computes the shifted returns.

Well done! this is done correctly using shift.

👏
🌟🌟🌟🌟🌟

Portfolio

The function `get_top_n` selects the `top_n` number of the top performing stocks.

You are properly calculating the `top_n` tickers based on the previous returns!

In case you are interested, here's an implementation that uses vectorization in a single line:

```
return (prev_returns.rank(axis=1, ascending=False)<=top_n).astype(int)
```

. . . And here's how it works . . .

- The Pandas rank function gives us the numerical rank in each row (when we specify axis=1)

```
print(previous_returns)
print('-' * 80)
print(previous_returns.rank(axis=1))
```

```
PZII        WVZR         CCG         FER        PRJT
2008-08-31         nan         nan         nan         nan         nan
2008-09-30         nan         nan         nan         nan         nan
2008-10-31  3.13172138  0.72709204  5.76874778  1.77557845  0.04098317
2008-11-30 -3.78816218 -0.67583590 -4.95433863 -1.67093250 -0.24929051
--------------------------------------------------------------------------------
                PZII       WVZR        CCG        FER       PRJT
2008-08-31         nan        nan        nan        nan        nan
2008-09-30         nan        nan        nan        nan        nan
2008-10-31 4.00000000 2.00000000 5.00000000 3.00000000 1.00000000
2008-11-30 2.00000000 4.00000000 1.00000000 3.00000000 5.00000000
```

- We can see that the highest item in each row has rank 5 . . .
- If we reverse the order, we should get the highest item in each row having rank 1

```
print(previous_returns)
print('-' * 80)
print(previous_returns.rank(axis=1, ascending=False))
```

```
PZII        WVZR         CCG         FER        PRJT
2008-08-31         nan         nan         nan         nan         nan
2008-09-30         nan         nan         nan         nan         nan
2008-10-31  3.13172138  0.72709204  5.76874778  1.77557845  0.04098317
2008-11-30 -3.78816218 -0.67583590 -4.95433863 -1.67093250 -0.24929051
--------------------------------------------------------------------------------
                PZII       WVZR        CCG        FER       PRJT
2008-08-31         nan        nan        nan        nan        nan
2008-09-30         nan        nan        nan        nan        nan
2008-10-31 2.00000000 4.00000000 1.00000000 3.00000000 5.00000000
2008-11-30 4.00000000 2.00000000 5.00000000 3.00000000 1.00000000
```

- Now, if we wanted to get the top 2 items in each row, we could ask which ranks (in descending order) are less than or equal to 2 . . .

```
print(previous_returns)
print('-' * 80)
print(previous_returns.rank(axis=1, ascending=False) <= 2)
```

```
PZII        WVZR         CCG         FER        PRJT
2008-08-31         nan         nan         nan         nan         nan
2008-09-30         nan         nan         nan         nan         nan
2008-10-31  3.13172138  0.72709204  5.76874778  1.77557845  0.04098317
2008-11-30 -3.78816218 -0.67583590 -4.95433863 -1.67093250 -0.24929051
--------------------------------------------------------------------------------
             PZII   WVZR    CCG    FER   PRJT
2008-08-31  False  False  False  False  False
2008-09-30  False  False  False  False  False
2008-10-31   True  False   True  False  False
2008-11-30  False   True  False  False   True
```

- We now see that we have True and False values - any values of True indicate items that are top 2 in their row . . .
- All that's left to do is cast to int (which will turn True values into 1 and False values into 0) . . .

```
print(previous_returns)
print('-' * 80)
print((previous_returns.rank(axis=1, ascending=False) <= 2).astype(int))
```

```
PZII        WVZR         CCG         FER         PRJT
2008-08-31         nan         nan         nan         nan         nan
2008-09-30         nan         nan         nan         nan         nan

2008-10-31  3.13172138  0.72709204  5.76874778  1.77557845  0.04098317
2008-11-30 -3.78816218 -0.67583590 -4.95433863 -1.67093250 -0.24929051
--------------------------------------------------------------------------------
            PZII  WVZR  CCG  FER  PRJT
2008-08-31     0     0    0    0     0
2008-09-30     0     0    0    0     0
2008-10-31     1     0    1    0     0
2008-11-30     0     1    0    0     1
```

The function `portfolio_returns` calculates the projected returns.

Well done! An optimization trick: try to minimize the expensive floating point computations (integers are far easier for our computers to process, and that's why we have GPU for floating point computations) like:

```
return (df_long-df_short)*lookahead_returns/n_stocks
```

Also checkout this article : A Beginner's Guide to Optimizing Pandas Code for Speed

Statistical Tests

The function `analyze_alpha` calculates the t-value and p-value.

Well done!

Excellent! The function correctly calculates the t-statistic and p-value.

Alternative Implementation

The below implementation will help have a deeper insight about how things are going on rather than directly using the the method `stats.ttest_1samp` from the pre-defined library

```
def analyze_alpha(expected_portfolio_returns_by_date):
    """
```

```
    Perform a t-test with the null hypothesis being that the expected mean return is
zero.

    Parameters
    ----------
    expected_portfolio_returns_by_date : Pandas Series
        Expected portfolio returns for each date

    Returns
    -------
    t_value
        T-statistic from t-test
    p_value
        Corresponding p-value
    """
    # TODO: Implement Function
    sample_size = len(expected_portfolio_returns_by_date)
    sample_mean = np.mean(expected_portfolio_returns_by_date)
    sample_error = stats.tstd(expected_portfolio_returns_by_date)

    t_value = 2 * (sample_mean / sample_error)
    p_value = stats.t.sf(np.abs(t_value), sample_size - 1)

    return t_value, p_value
```

The student correctly identifies the p-value they got. The student indicates what the p-value indicates about their signal.

Nicely Interpreted !

You're right, since the pvalue is greater than alpha our threshold for significance, we fail to reject the null hypothesis. It means there's a chance this strategy will yield mean return of zero. For us to ascertain this we'll need to conduct further analysis on the strategy.

What does this mean for our null hypothesis and the expected return?

A little explanation regarding the p-value:

Many students mistake the p-value as a number to calculate the expected profit, but it is rather a statistical measure of confidence about the significance of your results regarding proof of the null hypothesis (not making a profit). Essentially, it's calculating the probability that your trading results could have just been due to random chance.

A p-value of 0.05, for example, indicates that you would have only a 5% chance of drawing the sample being

tested if the null hypothesis was actually true (= 5% chance that the strategy would not yield positive returns in reality, even when our tests did calculate returns bigger than zero).

In the majority of analyses, an alpha of 0.05 is used as the cutoff for significance. If the p-value is less than 0.05, we reject the null hypothesis (to prove that the strategy is expected to yield returns greater than zero) and

conclude that a significant difference does exist between the null hypothesis (zero returns) and the calculated returns (profits expected) ... Below 0.05, significant. Over 0.05, not significant.

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review

START