UDACITY

‹ Return to Classroom

Trading with Momentum

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Brilliant Udacity Learner ,

Congratulations!🎉
You nailed it by one shot! This is an excellent submission. You have demonstrated and understood the concept of momentum, the design of the strategy, and the evaluation of the strategy using the t-test correctly.You are off to a great start with the program. Keep up the hard work and good luck with the next project. 🏆

Extra resources 📚

Below are some additional resources related to p-value analysis:

- P-value
- P-values Explained By Data Scientist
- How to Conduct a Two Sample T-Test
- How to Conduct a Paired Samples T-Test

Market Data

The function `resample_prices` computes the monthly prices.

The `resample_prices` function required in this section is perfectly implemented. It helps to resample `close_prices` at the sampling frequency of `freq` . 👌

Pro Tips

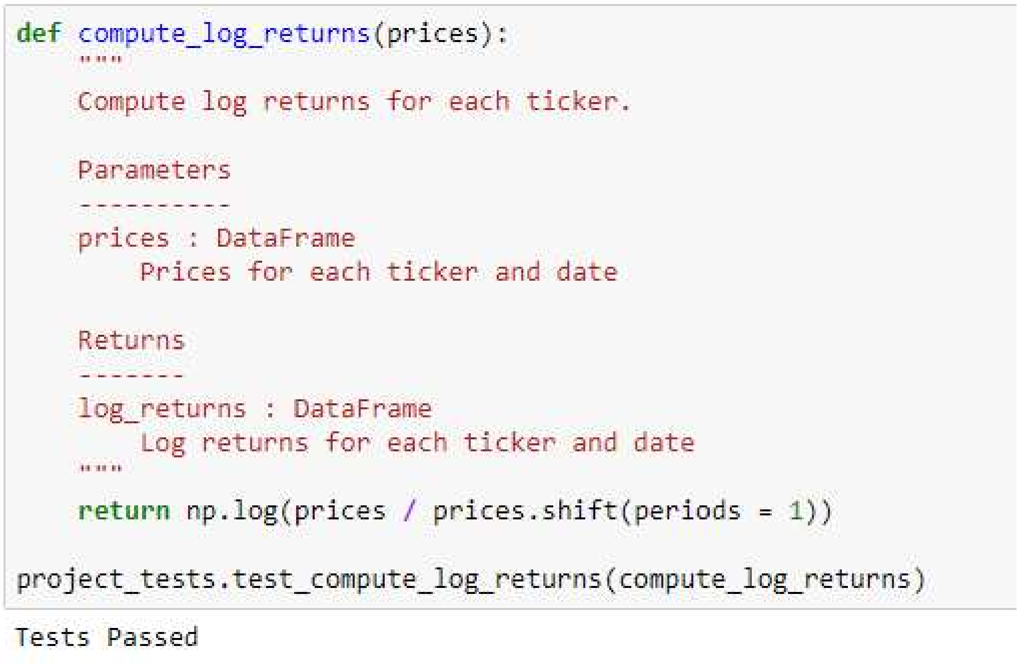
Resampling involves changing the frequency of your time series observations.
Two types of resampling are:

- Upsampling: Where you increase the frequency of the samples, such as from minutes to seconds.
- Downsampling: Where you decrease the frequency of the samples, such as from days to months.

There are perhaps two main reasons why you may be interested in resampling your time series data:

- **Problem Framing**: Resampling may be required if your data is available at the same frequency that you - want to make predictions.
- **Feature Engineering**: Resampling can also be used to provide additional structure or insight into the learning problem for supervised learning models.
  There is a lot of overlap between these two cases. For example, you may have daily data and want to predict a monthly problem. You could use the daily data directly or you could downsample it to monthly data and develop your model.
  A feature engineering perspective may use observations and summaries of observations from both time scales and more in developing a model.

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



```
def compute_log_returns(prices):
    """
    Compute log returns for each ticker.

    Parameters
    ----------
    prices : DataFrame
        Prices for each ticker and date

    Returns
    -------
    log_returns : DataFrame
        Log returns for each ticker and date
    """
    return np.log(prices / prices.shift(periods = 1))

project_tests.test_compute_log_returns(compute_log_returns)
```

```
Tests Passed
```

Good work using the `log` function to implement the `compute_log_returns` function.✔

Pro Tips

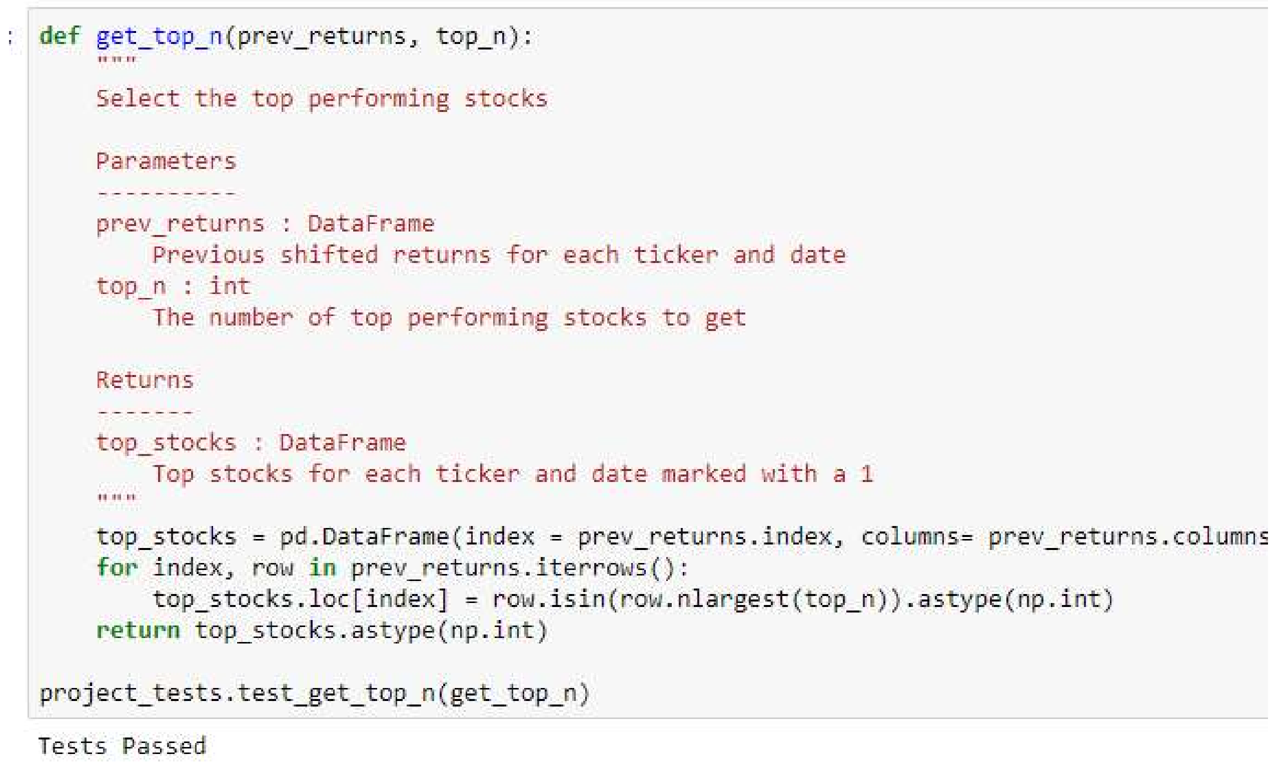
The simple **return** of a portfolio is the weighted sum of the simple **returns** of the constituents of the portfolio. The **log return** for a time period is the sum of the log returns of partitions of the time period. For example the **log return** for a year is the sum of the **log returns** of the days within the year. Check out Lognormal and normal distribution for more insight.

The function `shift_returns` computes the shifted returns.

Good implementation! This section passed as the `shift_returns` function computes the shifted returns as expected. 🎉

Portfolio

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



```
: def get_top_n(prev_returns, top_n):
      """
      Select the top performing stocks

      Parameters
      ----------
      prev_returns : DataFrame
          Previous shifted returns for each ticker and date
      top_n : int
          The number of top performing stocks to get

      Returns
      -------
      top_stocks : DataFrame
          Top stocks for each ticker and date marked with a 1
      """
      top_stocks = pd.DataFrame(index = prev_returns.index, columns= prev_returns.columns
      for index, row in prev_returns.iterrows():
          top_stocks.loc[index] = row.isin(row.nlargest(top_n)).astype(np.int)
      return top_stocks.astype(np.int)

  project_tests.test_get_top_n(get_top_n)

  Tests Passed
```

The function successfully got the top performing stock for each month, using top performing stocks from `prev_returns`. Good job! 👌

Faster and simpler implementation:

```
nlar_float = prev_returns.apply(lambda x:x.nlargest(top_n), axis = 1)
   return nlar_float.notnull().astype('int64')
```

The function `portfolio_returns` calculates the projected returns.

Good work here, using `n_stocks`. The `portfolio_returns` function correctly computed the expected portfolio returns. Excellent! 🎉

Pro Tips

Portfolio return refers to the gain or loss realized by an investment portfolio containing several types of investments. Portfolios aim to deliver returns based on the stated objectives of the investment strategy, as well as the risk tolerance of the type of investors targeted by the portfolio. Check out Portfolio Return - Investopedia for more insight.

Statistical Tests

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

Good job on correctly obtaining the `t-value` and `p-value` using the `analyze_alpha` function. Perfect! ✌️

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

```
With the Alpha analysis, the observed p-value is 0.073359, which is higher than the given threshold of 0.05 and we cannot reject the null hypothesis, the actual population mean return from the signal is zero.

Conclusion: Our null hypothesis is correct and the signal might not always return positive results.
```

The justification on how the p-value compared to alpha value is well explained. I like the reasoning here. The p-value (`0.073359`) being greater than alpha (`0.05`) indicates that the null hypothesis cannot be ignored and therefore a better strategy may be required. 🎉 🎉

Suggestions

- WHAT A P-VALUE TELLS YOU ABOUT STATISTICAL DATA
- How to Correctly Interpret P Values

DOWNLOAD PROJECT

RETURN TO PATH