UDACITY

‹ Return to Classroom

DISCUSS ON STUDENT HUB

Breakout Strategy

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Dear Student,

Great job passing all the specifications for the project, you have now successfully implemented a Breakout Strategy using a Notebook. The following are a few resources for continued learning on the topic:
Here is a blog with more information on high low breakout strategy
Here is some more information on how to determine if histograms are normal or not and what causes the skewness.
Here is some more info in ks test You can learn to apply lambda functions from here.
Here is some more info on breakout strategy
Here are some good reads:
List of code, papers, and resources for AI/deep learning/machine learning/neural networks applied to algorithmic trading
Machine Trading: Deploying Computer Algorithms to Conquer The Markets
Quantitative Trading: How to Build Your Own Algorithmic Trading Business

The project meets all specifications, well done on completing the project.

Thanks!

Generate Signal

The function `get_high_lows_lookback` computes the maximum and minimum of the closing prices over a window of days.

The maximum and minimum of the closing prices over a window of days have been correctly calculated. Function implementation properly applies the `shift` and `rolling` functions. 👍

```
Tests Passed
```

Congratulations!

Below are a few resources to understand `shift` and `rolling` functions better:
What is meant by shift in dataframe
Shifting and lagging time-series data
Pandas DataFrame Rolling
Using shift and rolling in pandas with groupBy

The function `get_long_short` computes long and short signals using a breakout strategy.

Long and short signals using a breakout strategy are correctly calculated. Nice use of data type using the astype function to implement `get_long_short()` correctly 👌

```
Tests Passed
```

Congratulations!

SUGGESTION

An alternative way to compute long and short signals using pandas would be as follows :

```
    long_short = pd.DataFrame(0,index = close.index, columns = close.columns)
    long_short[lookback_low > close] = -1
    long_short[lookback_high < close] = 1
    return long_short
```

ADDITIONAL RESOURCES

Below are a few resources to help understand some of the `pandas` and `numpy` functionality:

Pandas DataFrame astype

Change data type of columns in Pandas

Numpy Array types and conversions between types

Difference between np.int, np.int_, int, and np.int_t in cython

The function **`filter_signals`** filters out repeated long or short signals.

`filter_signals` has been implemented correctly 👏

```
Tests Passed
```

Congratulations!

SUGGESTION

You may also use iterrows over each column as recommended. `iterrows()` method is optimized to work with Pandas dataframes, hence a significant improvement over crude looping.

`filter_signals` can also be implemented using lambda function like this:

```
    pos_signal = signal[signal == 1].fillna(0)
    neg_signal = signal[signal == -1].fillna(0)

    pos_signal = pos_signal.apply(lambda signals: clear_signals(signals, lookahead_days))
    neg_signal = neg_signal.apply(lambda signals: clear_signals(signals, lookahead_days))

    return pos_signal + neg_signal
```

`filter_signals` can be implemented in one line as follows as well:

```
return signal.replace(-1, 0).apply(lambda x: clear_signals(x, lookahead_days), axis=0) + signal.replace(1, 0).apply(lambda x: clear_signals(x, lookahead_days), ax
```

`filter_signals` can also be implemented without lambda function as follows:

```
return (signal == 1).replace({True: 1, False: 0}).apply(clear_signals, args=(lookahead_days,)) + (signal == -1).replace({True: -1, False: 0}).apply(clear_signals,
```

ADDITIONAL RESOURCES

You may refer on the following links below to deepen your understanding on how to access a group of rows and columns by label(s) or a boolean array using Pandas Dataframe functions:

Pandas DataFrame .loc

Using iloc, loc, & ix to select rows and columns in Pandas DataFrames

Selection with .loc in python

The difference between iloc and loc in Pandas

Looping with iterrows

The function **`get_lookahead_prices`** gets the close price days ahead in time.

Good job implementing `get_lookahead_prices` to get the close price days ahead in time 👍

```
Tests Passed
```

Congratulations!

ADDITIONAL RESOURCES

Please find a few resources below on shift function in a pandas DataFrame:

How to shift several rows in a pandas DataFrame

Shifting or lagging values in a dataframe
Shift Pandas DataFrame with a multiindex

The function `get_return_lookahead` generates the log price return between the closing price and the lookahead price.

The log price return between the closing price and the lookahead prices are correctly calculated for 5, 10 and 20 days. Good use of Natural logarithm or np.log to implement this function 👏

```
Tests Passed
```

Congratulations!

ADDITIONAL RESOURCES

Please check the following links to know more about natural logarithm:
NumPy: Logarithm with base n
numpy.log() in Python
log(x) vs ln(x):The curse of scientific computing

The function `get_signal_return` generates the signal returns.

`get_signal_return` has been implemented correctly 👌

```
Tests Passed
```

Congratulations!

Evaluate Signal

Correctly answers the question “What do the histograms tell you about the signal returns?"

Here is a detailed explanation for each histogram:

```
•    5 Days
–    Resembles normal distribution that is slightly fatter tailed.
•    10 days
–    Somewhat resembles a log–normal distribution.
–    There is a peak of outliers visible close to the right tail .
•    20 days
–    Resembles a normal distribution with fatter tails, plus a peak of outliers visible on the right edge that breaks the sy
mmetricity.
–    Has a higher spread than the previous two distributions.
```

The following are a few documents one can read to understand histograms better :
https://www.researchgate.net/publication/228315820_The_Impact_of_Skewness_and_Fat_Tails_on_the_Asset_Allocation_Decision
https://www.investopedia.com/terms/s/skewness.asp
Typical Histogram Shapes and What They Mean
Common shapes of distributions
Data Visualization in Python—Histogram in Matplotlib
How to Analyze a Histogram
Interpreting a Histogram
3 Things a Histogram Can Tell You
How to interpret the shape of statistical data in a Histogram

Outliers

The function `calculate_kstest` calculates the ks and p values.

`calculate_kstest` has been implemented correctly 👍

```
Tests Passed
```

Congratulations!
An alternative implementation for this method can be as follows:

```
    g_mu,g_std = long_short_signal_returns.mean(), long_short_signal_returns.std(ddof=0)

     grp = pd.DataFrame(long_short_signal_returns.groupby('ticker')['signal_return'].apply(list))
     rzlt = pd.DataFrame(grp['signal_return'].map(lambda x: kstest(x, 'norm', args=(g_mu,g_std))))
     rzlt['k'] = rzlt['signal_return'].map(lambda x: x[0])
     rzlt['p'] = rzlt['signal_return'].map(lambda x: x[1])

     return  rzlt['k'], rzlt['p']
```

ADDITIONAL RESOURCES
What is a Kolmogorov-Smirnov normality test?
How to use a proper normalization to have the right p_values and ks_values from Kolmogorov-Smirnov test (KS test)?
numpy.mean can be used to compute the arithmetic mean along the specified axis.
numpy.std can be used to compute the standard deviation along the specified axis.
What are the differences between np.mean and np.average?
Summarising, Aggregating, and Grouping data in Python Pandas

The function `find_outliers` returns the list of outlying symbols.

The function find_outliers implemented correctly to return the list of outlying symbols.

```
Tests Passed
```

Congratulations!
`find_outliers` function correctly returns 24 outliers 👍
`find_outliers` can also be implemented in one line as follows:
`return set(ks_values[ks_values > ks_threshold].index).intersection(p_values[p_values < pvalue_threshold].index)`
`return set(ks_values[ks_values > ks_threshold][p_values < pvalue_threshold].index.values)`

ADDITIONAL RESOURCES

Python Set intersection
You may use numpy.logical_and to compute the truth value of x1 AND x2 element-wise.
numpy.logical_and Parameters
Difference between numpy.logical_and and &
Logical operations on boolean arrays

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review

START