UDACITY

< Back to AI for Trading

Breakout Strategy

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Here is a blog with more information on high low break out 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:

1. Machine Trading: Deploying Computer Algorithms to Conquer The Markets (Ernest Chan 2017)
2. Quantitative Trading: How to Build Your Own Algorithmic Trading Business
3. David Byrd's slides on how to vectorize technical analysis methods

Great job on completing the project successfully, hope you had fun implementing the project 👍

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!

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!

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

The repeated long and short signal are filtered out correctly!

You can implement the filter signals in an efficient way using iterrows as follows:

```
    filtered_signal = signal.copy()
    for ticker, ticker_signals in signal.T.iterrows():
        long_signals = ticker_signals.copy()
        long_signals[long_signals == -1] = 0
        short_signals = ticker_signals.copy()
        short_signals[long_signals == 1] = 0
        filtered_signal[ticker] = clear_signals(long_signals, lookahead_days) + clear_signals(short_signals, lookahead_d
ays)

    return filtered_signal
```

You can implement the filter signals in an efficient way using lamba function like this:

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

    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*-1
```

You can implement this in one line as follows:

```
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
```

You can implement this 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_sig
```

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

The close price days ahead in time are returned correctly!

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 price is correctly calculated!

The function `get_signal_return` generates the signal returns.

✅

Evaluate Signal

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

You have correctly observed that the distributions do not look normal, and that there appear to be outliers in the right tails.

Outliers

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

ks and p values are calculated correctly!
An efficient implementation for this method can be as follows:

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

    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']
```

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

The list of outlier symbols are correctly returned!
You can implement this in one line as follows:

```
return set(ks_values[ks_values > ks_threshold].index).intersection(p_values[p_values < pvalue_threshold].index)
```

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review